Teoria grafurilor

Arbori, parcurgeri și drumuri minime

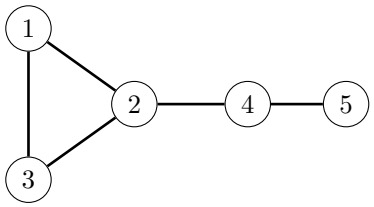
Codul Prüfer · Arbori parțiali de cost minim · BFS și DFS · Punți · Dijkstra · Bellman–Ford · Drumul critic

Notații și rezultate folosite

- Un graf $G = (V, E)$ are $n = |V|$ vârfuri și $m = |E|$ muchii. Pentru un vârf v, $N(v)$ este mulțimea vecinilor lui v, iar $d(v) = |N(v)|$ este gradul lui v.
- Un *drum* este un șir de vârfuri $v_0, v_1, \ldots, v_k$ în care oricare două vârfuri consecutive sunt adiacente. Numărul k este *lungimea* drumului (numărul de muchii). Un *ciclu* este un drum închis ($v_k = v_0$, $k \geq 3$) cu vârfurile $v_0, \ldots, v_{k-1}$ distincte. Graful este *conex* dacă între oricare două vârfuri există un drum. Subgrafurile conexe maximale se numesc *componente conexe*.
- Un *arbore* este un graf conex fără cicluri. Un arbore cu n vârfuri are exact $n - 1$ muchii, iar între oricare două vârfuri ale sale există un drum unic. Un vârf de grad 1 se numește *frunză*. Orice arbore cu cel puțin două vârfuri are cel puțin două frunze.
- Într-un *graf orientat* muchiile sunt înlocuite de *arce* (u, v), parcurse doar de la u la v. Un *circuit* este un drum orientat închis. Un graf orientat fără circuite se numește *aciclic*.
- Într-un *graf ponderat* fiecare muchie uv are o pondere (lungime, cost, timp) $w(uv)$. Lungimea unui drum este suma ponderilor muchiilor sale, iar *distanța* $\mathrm{dist}(u, v)$ este lungimea minimă a unui drum de la u la v.

Memorarea grafurilor

Pentru a aplica un algoritm, graful trebuie memorat într-o structură de date. Vom folosi ca exemplu graful cu vârfurile $1, \ldots, 5$ și muchiile $12,\ 13,\ 23,\ 24,\ 45$, deci cu $n = 5$ și $m = 5$.



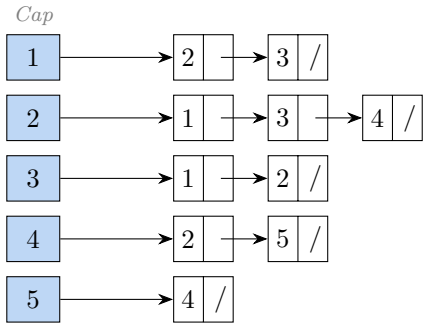
$$A = \begin{pmatrix} 0 & 1 & 1 & 0 & 0 \\ 1 & 0 & 1 & 1 & 0 \\ 1 & 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 & 1 \\ 0 & 0 & 0 & 1 & 0 \end{pmatrix}$$

Matricea de adiacență este matricea $A = (a_{ij})$ de dimensiune $n \times n$ cu $a_{ij} = 1$ dacă ij este muchie și $a_{ij} = 0$ în caz contrar. Pentru un graf neorientat matricea este simetrică, iar suma elementelor de pe linia i este gradul lui i. Matricea ocupă n^2 poziții, indiferent de numărul muchiilor. Testul „sunt i și j adiacente?” se face imediat, citind a_{ij}. În schimb, pentru a găsi vecinii unui vârf trebuie parcursă toată linia lui, adică n poziții.

Listele de adiacență. Pentru fiecare vârf v se memorează lista vecinilor lui, $N(v)$. Concret, structura are două componente:

- Un tablou $Cap[1..n]$, în care $Cap[v]$ indică începutul listei vârfului v.
- Listele propriu-zise, în care fiecare element conține eticheta unui vecin și legătura către elementul următor din aceeași listă (sau marcajul de sfârșit, notat mai jos cu /).

Pentru graful din exemplu, listele (cu vecinii în ordine crescătoare) sunt:



Proprietăți ale listelor de adiacență.

- Într-un graf neorientat, muchia uv apare de două ori: v în lista lui u și u în lista lui v. Lungimea listei lui v este $d(v)$, deci listele au în total $\sum_v d(v) = 2m$ elemente (în exemplu, $2+3+2+2+1 = 10 = 2 \cdot 5$).
- Memoria ocupată este proporțională cu $n+m$, mult mai puțin decât n^2 pentru grafurile rare (cu puține muchii), cum sunt rețelele rutiere sau arborii.
- Vecinii lui v se parcurg în $d(v)$ pași. Parcurgând vecinii tuturor vârfurilor facem $2m$ pași, de aceea BFS și DFS au complexitatea $O(n+m)$.
- Testul „sunt u și v adiacente?" cere parcurgerea listei lui u, deci până la $d(u)$ pași.
- Într-un *graf orientat*, arcul (u, v) apare o singură dată, în lista lui u (lista *succesorilor* lui u), deci listele au în total m elemente.
- Într-un *graf ponderat*, fiecare element al listei memorează perechea (vecin, pondere). De exemplu, dacă muchia 12 ar avea lungimea 4, lista lui 1 ar începe cu perechea $(2, 4)$.

Varianta cu doi vectori. Când graful nu se modifică, listele se pot memora compact, una după alta, într-un vector L de lungime $2m$, împreună cu un vector P de lungime $n+1$, unde $P[v]$ este poziția din L la care începe lista lui v și $P[n+1] = 2m+1$. Vecinii lui v sunt atunci $L[P[v]], \ldots, L[P[v+1]-1]$. Pentru exemplul nostru:

Poziție	1	2	3	4	5	6	7	8	9	10
L	2	3	1	3	4	1	2	2	5	4

v	1	2	3	4	5	6
P	1	3	6	8	10	11

De exemplu, vecinii lui 2 sunt $L[3], L[4], L[5]$, adică $1, 3, 4$.

Comparație.

	Matrice de adiacență	Liste de adiacență
Memorie	n^2	proporțională cu $n+m$
Test „uv este muchie?"	un pas	până la $d(u)$ pași
Parcurgerea vecinilor lui v	n pași	$d(v)$ pași

Algoritmii din acest seminar (BFS, DFS, Dijkstra) parcurg în mod repetat vecinii vârfurilor, de aceea folosesc listele de adiacență. În toate aplicațiile, vecinii unui vârf sunt examinați în ordine crescătoare (respectiv alfabetică), adică în ordinea din listă.

Codul Prüfer al unui arbore

Fie T un arbore cu vârfurile etichetate $1, 2, \ldots, n$, unde $n \geq 3$. Codul Prüfer asociază lui T un șir de $n-2$ numere din $\{1, \ldots, n\}$, din care arborele poate fi reconstituit complet.

Codificarea (arbore → cod)

Se repetă de $n-2$ ori:

1. Se alege *frunza cu cea mai mică etichetă*.
2. Se scrie în cod *vecinul* acesteia (unicul ei vecin).
3. Se șterge frunza din arbore, împreună cu muchia ei.

După fiecare ștergere rămâne tot un arbore, deci pasul 1 se poate aplica din nou. La final rămân două vârfuri, unite printr-o muchie. Rezultatul este șirul $(a_1, \ldots, a_{n-2})$.

Proprietatea gradelor. Fiecare vârf v apare în cod de exact $d(v)-1$ ori. Într-adevăr, muchiile lui v dispar pe rând. O muchie vx dispare fie când x este șters ca frunză (și atunci se scrie v), fie când v însuși este șters ca frunză (și atunci se scrie x), fie rămâne ca ultima muchie a arborelui. Doar una dintre muchiile lui v nu produce scrierea lui v: cea pe care o are v în momentul în care este șters, sau ultima muchie, dacă v rămâne printre ultimele două vârfuri. În particular, *frunzele nu apar în cod*.

Decodificarea (cod → arbore)

Fie un cod $(a_1, \ldots, a_{n-2})$. Atunci n este lungimea codului plus 2. Pentru fiecare vârf v calculăm

$$d(v) = 1 + \big(\text{numărul de apariții ale lui } v \text{ în cod}\big).$$

Pentru $i = 1, \ldots, n-2$:

1. Fie x cel mai mic vârf nește rs cu $d(x) = 1$.
2. Adăugăm muchia $x\,a_i$.
3. Ștergem x și micșorăm $d(a_i)$ cu 1.

La final rămân exact două vârfuri neșterse, cu $d = 1$, pe care le unim printr-o muchie.

Decodificarea refă pas cu pas codificarea: la pasul i, vârfurile cu $d = 1$ sunt exact frunzele arborelui curent, deoarece un vârf care mai apare în restul codului mai are vecini de șters.

Teorema lui Cayley. Codificarea și decodificarea sunt inverse una alteia, deci stabilesc o bijecție între arborii cu vârfurile $1, \ldots, n$ și șirurile de lungime $n-2$ cu valori în $\{1, \ldots, n\}$. Prin urmare, numărul arborilor cu n vârfuri etichetate este

$$n^{\,n-2}.$$

Folosind proprietatea gradelor, numărul arborilor în care vârfurile au gradele date $d_1, \ldots, d_n \geq 1$, cu $d_1 + \cdots + d_n = 2n-2$, este numărul de șiruri în care v apare de $d_v - 1$ ori:

$$\frac{(n-2)!}{(d_1-1)!\,(d_2-1)!\cdots(d_n-1)!}.$$

Arbori parțiali de cost minim (algoritmii lui Kruskal și Prim)

Fie $G = (V, E)$ un graf conex, cu ponderi $w(e)$ pe muchii. Un *arbore parțial* al lui G este un subgraf care este arbore și conține toate vârfurile lui G. El are deci exact $n - 1$ muchii. *Costul* unui arbore parțial este suma ponderilor muchiilor sale. Se cere un *arbore parțial de cost minim* (APM). Aplicația tipică: legarea a n localități printr-o rețea (de cablu, de conducte, de drumuri) de cost total minim, în care din orice localitate se poate ajunge în oricare alta.

Proprietatea tăieturii. Fie S o mulțime de vârfuri, cu $S \neq \emptyset$ și $S \neq V$, și fie $e = xy$ o muchie de pondere minimă dintre cele cu un capăt în S și celălalt în afara lui S. Atunci există un APM care conține muchia e. Dacă toate ponderile sunt distincte, e aparține oricărui APM.

Justificare. Fie T un APM care nu conține e. În T există un drum unic de la x la y. Acest drum pleacă din S și ajunge în afara lui S, deci conține o muchie f cu un capăt în S și celălalt în afara lui S. Eliminând f din T, arborele se rupe în două componente, cu x și y în componente diferite. Adăugând e, ele se unesc din nou și obținem un arbore parțial T'. Cum $w(e) \leq w(f)$, costul lui T' nu depășește costul lui T, deci T' este tot un APM și conține e.

Algoritmul lui Kruskal

Pasul 1. *Sortare.* Ordonăm muchiile crescător după pondere.

Pasul 2. *Selecție.* Parcurgem muchiile în această ordine. O muchie uv se adaugă la arbore dacă u și v se află în componente diferite ale arborelui construit până atunci, adică dacă nu formează un ciclu cu muchiile deja alese. Altfel, muchia se ignoră. La adăugarea lui uv, componentele lui u și v se reunesc.

Pasul 3. *Oprire.* Ne oprim după ce am ales $n - 1$ muchii.

Inițial fiecare vârf formează singur o componentă. Când se adaugă muchia uv, ea este o muchie de pondere minimă care leagă componenta lui u de restul vârfurilor. Într-adevăr, o muchie mai ieftină care pleacă din această componentă a fost examinată mai devreme, când capetele ei erau tot în componente diferite, deci ar fi fost deja adăugată. Conform proprietății tăieturii, alegerea este corectă.

Algoritmul lui Prim

Se pornește dintr-un vârf s. Pentru fiecare vârf v din afara arborelui se reține eticheta $c(v)$, ponderea minimă a unei muchii de la v la arborele curent, și vârful $p(v)$ la care duce această muchie.

Pasul 1. *Inițializare.* Arborele conține doar vârful s, iar $c(s) = 0$. Pentru fiecare vecin v al lui s punem $c(v) = w(sv)$ și $p(v) = s$. Pentru vârfurile $v \neq s$ care nu sunt legate printr-o muchie de s punem $c(v) = \infty$ și $p(v) = -$.

Pasul 2. *Alegere.* Alegem vârful u din afara arborelui cu eticheta $c(u)$ minimă și îl adăugăm la arbore, împreună cu muchia $p(u)\,u$.

Pasul 3. *Actualizare.* Pentru fiecare vecin z al lui u din afara arborelui: dacă $w(uz) < c(z)$, punem $c(z) = w(uz)$ și $p(z) = u$.

Pasul 4. *Oprire.* Repetăm pașii 2 și 3 până când toate vârfurile sunt în arbore.

La fiecare pas se adaugă muchia de pondere minimă dintre arborele curent și restul vârfurilor, deci alegerea este corectă, tot conform proprietății tăieturii.

Observații.

- Dacă toate ponderile sunt distincte, APM este unic. Atunci Kruskal și Prim dau același arbore, chiar dacă aleg muchiile în altă ordine.
- Algoritmul lui Prim are aceeași structură ca algoritmul lui Dijkstra. Diferența este doar

la actualizare: Prim compară ponderea muchiei, $w(uz)$, iar Dijkstra compară lungimea întregului drum, $d(u) + w(uz)$. De aceea APM și arborele drumurilor minime sunt, în general, diferiți.

Parcurgerea în lățime (BFS)

Parcurgerea în lățime (*breadth-first search*) vizitează vârfurile unui graf pornind dintr-un vârf s, „în valuri": întâi s, apoi vecinii lui s, apoi vecinii acestora încă nevizitați etc. Se folosește o *coadă* Q (primul vârf intrat este primul extras) și se reține, pentru fiecare vârf v, nivelul $\ell(v)$ și predecesorul $p(v)$.

Algoritmul BFS

Pasul 1. *Inițializare.* Marcăm s ca vizitat, punem $\ell(s) = 0$, $p(s) = -$ și $Q = (s)$. Toate celelalte vârfuri sunt nevizitate.

Pasul 2. *Extragere.* Cât timp Q nu este vidă, extragem vârful u aflat la începutul cozii.

Pasul 3. *Explorare.* Pentru fiecare vecin w al lui u încă nevizitat: îl marcăm ca vizitat, punem $\ell(w) = \ell(u) + 1$, $p(w) = u$ și îl adăugăm la sfârșitul cozii. Revenim la Pasul 2.

Proprietăți.

- Vârfurile sunt vizitate în ordinea crescătoare a nivelurilor, iar $\ell(v)$ este *numărul minim de muchii* al unui drum de la s la v. Un astfel de drum se reconstituie mergând înapoi pe predecesori: $v,\ p(v),\ p(p(v)), \ldots, s$.
- Muchiile $p(v)\,v$ formează un arbore, numit *arborele BFS*, care conține drumuri de lungime minimă de la s la toate vârfurile vizitate.
- Sunt vizitate exact vârfurile din componenta conexă a lui s.
- Orice muchie unește două vârfuri de pe același nivel sau de pe niveluri consecutive. Un graf conex este *bipartit* dacă și numai dacă nicio muchie nu unește două vârfuri de pe același nivel. În acest caz, nivelurile pare și cele impare dau bipartiția.

Două variante utile.

- *Mai multe surse.* Dacă parcurgerea pornește simultan din vârfurile $s_1, \ldots, s_k$, le punem pe toate în coadă de la început, cu nivelul 0. Nivelul $\ell(v)$ obținut este distanța de la v la cea mai apropiată sursă. Este ca și cum am adăuga un vârf nou S, legat de toate sursele, și am aplica BFS din S (Problema 12).
- *Numărul drumurilor minime.* Fie $N(v)$ numărul drumurilor de lungime minimă de la s la v. Avem $N(s) = 1$, iar pentru $v \neq s$, $N(v)$ este suma valorilor $N(u)$ pentru vecinii u ai lui v cu $\ell(u) = \ell(v) - 1$. Valorile se calculează în timpul parcurgerii, nivel cu nivel (Problema 13).

Parcurgerea în adâncime (DFS)

Parcurgerea în adâncime (*depth-first search*) înaintează cât de mult se poate de-a lungul unui drum și revine (face un pas înapoi) doar când vârful curent nu mai are vecini nevizitați. Se folosește un contor de timp t și se reține, pentru fiecare vârf v, momentul descoperirii $t_{\text{in}}(v)$, momentul terminării $t_{\text{out}}(v)$ și predecesorul $p(v)$.

Algoritmul DFS

Folosim un contor global t (un „ceas"), care numără evenimentele parcurgerii: fiecare descoperire și fiecare terminare a unui vârf. Notația $t \leftarrow t+1$ înseamnă că valoarea lui t crește cu 1 (atribuire), adică ceasul avansează cu o unitate.
Procedura recursivă DFS(u):

1. Marcăm u ca vizitat, apoi $t \leftarrow t+1$ și $t_{\text{in}}(u) = t$.
2. Pentru fiecare vecin w al lui u, în ordine: dacă w este nevizitat, punem $p(w) = u$ și apelăm DFS(w).
3. La final, $t \leftarrow t+1$ și $t_{\text{out}}(u) = t$ (vârful u este *terminat*).

Parcurgerea pornește cu $t = 0$ și apelul DFS(s). Dacă rămân vârfuri nevizitate, se relansează DFS din cel mai mic dintre ele.

Recursivitatea poate fi înlocuită cu o *stivă* (ultimul vârf intrat este primul extras).

Proprietăți (graf neorientat).

- Muchiile $p(w)\,w$ formează *arborele DFS* (câte un arbore pentru fiecare lansare). Fiecare lansare vizitează exact o componentă conexă, deci numărul de lansări este numărul componentelor conexe.
- Orice muchie care nu este în arbore unește un vârf cu un *strămoș* al său în arbore. O astfel de muchie se numește *muchie de întoarcere*. Graful are cicluri dacă și numai dacă există muchii de întoarcere: muchia de întoarcere uv, cu v strămoș al lui u, formează un ciclu împreună cu drumul din arbore de la v la u.
- *Proprietatea parantezelor:* pentru oricare două vârfuri, intervalele $[t_{\text{in}}, t_{\text{out}}]$ sunt fie disjuncte, fie incluse unul în celălalt (al doilea caz apare exact când un vârf este descendent al celuilalt).

Legături critice: punți și puncte de articulație

Fie G un graf conex. O muchie e este o *punte* dacă graful $G - e$, obținut prin eliminarea ei, nu mai este conex. Un vârf v este un *punct de articulație* dacă graful $G - v$, obținut prin eliminarea lui v și a muchiilor sale, nu mai este conex. Într-o rețea, punțile sunt legăturile critice, iar punctele de articulație sunt nodurile critice: defectarea uneia singure împarte rețeaua în bucăți care nu mai comunică.

Caracterizare. O muchie este punte dacă și numai dacă nu aparține niciunui ciclu. Într-adevăr, dacă uv se află pe un ciclu, după eliminarea ei u și v rămân legate prin restul ciclului, deci graful rămâne conex. Reciproc, dacă după eliminarea lui uv există încă un drum de la u la v, acest drum împreună cu uv formează un ciclu.

Valorile *low*. Aplicăm DFS și notăm $t_{\text{in}}(v)$ momentele descoperirii. Pentru fiecare vârf v definim $low(v)$ ca fiind cel mai mic moment t_{in} al unui vârf la care se poate ajunge din subarborele DFS al lui v folosind cel mult o muchie de întoarcere. Ele se calculează în momentul terminării lui v:

$$low(v) = \min\Big(t_{\text{in}}(v),\ \ t_{\text{in}}(w) \text{ pentru muchiile de întoarcere } vw,\ \ low(c) \text{ pentru fiii } c \text{ ai lui } v\Big).$$

Valoarea $low(v)$ arată cât de „sus" în arborele DFS se poate întoarce subarborele lui v fără a folosi muchia către părintele lui v.

Determinarea punților și a punctelor de articulație

Pasul 1. Aplicăm DFS și reținem $t_{\text{in}}(v)$ și $p(v)$ pentru fiecare vârf.

Pasul 2. La terminarea fiecărui vârf v, calculăm $low(v)$ cu formula de mai sus.

Pasul 3. *Punți.* O muchie de arbore $p(v)\,v$ este punte dacă și numai dacă $low(v) > t_{\text{in}}(p(v))$. Muchiile de întoarcere nu sunt niciodată punți.

Pasul 4. *Puncte de articulație.* Rădăcina arborelui DFS este punct de articulație dacă și numai dacă are cel puțin doi fii. Un vârf u diferit de rădăcină este punct de articulație dacă și numai dacă are un fiu v cu $low(v) \geq t_{\text{in}}(u)$.

De ce funcționează. Dacă $low(v) > t_{\text{in}}(p(v))$, subarborele lui v nu are nicio muchie spre $p(v)$ sau spre strămoșii acestuia, în afară de muchia $p(v)\,v$. Fără această muchie, subarborele lui v rămâne izolat de restul grafului (în DFS nu există muchii între ramuri diferite). Analog, dacă $low(v) \geq t_{\text{in}}(u)$, subarborele lui v nu poate urca deasupra lui u decât trecând prin u, deci eliminarea lui u îl izolează. Rădăcina nu are strămoși, deci contează doar dacă are cel puțin două ramuri.

O aplicație: străzi cu sens unic. A *orienta* un graf înseamnă a da fiecărei muchii un sens. *Teorema lui Robbins*: un graf conex se poate orienta astfel încât din orice vârf să se poată ajunge în oricare altul dacă și numai dacă nu are punți. O orientare potrivită se obține chiar din DFS: muchiile de arbore se orientează de la părinte la fiu, iar muchiile de întoarcere de la descendent la strămoș (Problema 20).

Complexitatea este cea a DFS, $O(n + m)$.

Drumuri minime: algoritmul lui Dijkstra

Problema. Fie un graf (orientat sau neorientat) cu ponderi *nenegative* $w(uv) \geq 0$ și un vârf sursă s. Se cer distanțele $\text{dist}(s, v)$ pentru toate vârfurile v și câte un drum minim pentru fiecare.

Algoritmul păstrează pentru fiecare vârf o *etichetă* $d(v)$, lungimea celui mai scurt drum de la s la v găsit până la momentul curent, și un predecesor $p(v)$. Mulțimea S conține vârfurile a căror etichetă este deja definitivă.

Algoritmul lui Dijkstra

Pasul 1. *Inițializare.* $d(s) = 0$, $d(v) = \infty$ pentru $v \neq s$, $p(v) = -$ pentru toți v, $S = \emptyset$.

Pasul 2. *Alegere.* Alegem vârful $u \notin S$ cu eticheta $d(u)$ minimă (la egalitate, pe cel mai mic) și îl adăugăm în S: eticheta lui devine definitivă.

Pasul 3. *Relaxare.* Pentru fiecare vecin $z \notin S$ al lui u: dacă $d(u) + w(uz) < d(z)$, punem $d(z) = d(u) + w(uz)$ și $p(z) = u$ (am găsit un drum mai scurt către z, prin u).

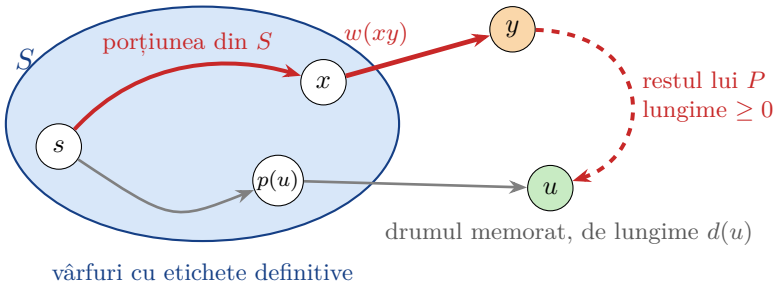
Pasul 4. *Oprire.* Repetăm pașii 2 și 3 până când $S = V$ sau până când toate etichetele din afara lui S sunt egale cu ∞. În al doilea caz, vârfurile rămase nu sunt accesibile din s.

La final, $d(v) = \text{dist}(s, v)$, iar drumul minim se reconstituie din predecesori, mergând înapoi: $v,\ p(v),\ p(p(v)), \ldots, s$. Muchiile $p(v)\,v$ formează *arborele drumurilor minime*.

De ce funcționează. Arătăm că, în momentul în care u este ales la Pasul 2, $d(u) = \text{dist}(s, u)$. Eticheta $d(u)$ este lungimea unui drum real, deci $d(u) \geq \text{dist}(s, u)$. Fie acum un drum oarecare P de la s la u. El pleacă din S și ajunge în afara lui S, deci are o primă muchie xy cu $x \in S$ și $y \notin S$. Presupunem, prin inducție, că etichetele vârfurilor din S sunt deja corecte, deci $d(x) = \text{dist}(s, x)$. Când x a fost adăugat în S, s-a făcut relaxarea muchiei xy, deci $d(y) \leq d(x) + w(xy)$, care este cel mult lungimea porțiunii lui P de la s la y. Restul lui P, de la y la u, are lungime ≥ 0, deoarece ponderile sunt nenegative. Așadar

$$\text{lungimea lui } P \ \geq\ d(y) \ \geq\ d(u),$$

ultima inegalitate rezultând din alegerea lui u cu eticheta minimă. Deci niciun drum nu este mai scurt decât $d(u)$.



În desen, drumul oarecare P este roșu. El părăsește mulțimea S prima dată pe muchia xy. Lungimea lui până în y este cel puțin $d(x) + w(xy) \geq d(y)$, iar porțiunea punctată de la y la u nu poate micșora lungimea. Drumul gri, care ajunge în u prin predecesorul $p(u) \in S$, este cel memorat de algoritm. Lungimea lui este $d(u) \leq d(y)$, deci el nu este mai lung decât P.

Observații.

- Implementarea directă (căutarea minimului printre etichete) are complexitatea $O(n^2)$. Cu o coadă cu priorități se obține $O((n+m)\log n)$.
- Dacă toate ponderile sunt egale cu 1, algoritmul lui Dijkstra coincide cu BFS.
- Pentru ponderi negative algoritmul poate da rezultate greșite, deoarece argumentul de mai sus folosește esențial faptul că porțiunea de la y la u are lungime nenegativă (vezi Problema 23). Pentru astfel de grafuri se folosește algoritmul Bellman–Ford, prezentat mai jos.

Drumuri minime cu ponderi negative: algoritmul Bellman–Ford

Problema. Fie un graf orientat cu ponderi $w(uv)$ care pot fi și negative (de exemplu, costuri din care se scad subvenții sau bonusuri) și un vârf sursă s. Un *circuit negativ* este un circuit cu suma ponderilor strict negativă. Dacă un astfel de circuit este accesibil din s, parcurgându-l de oricâte ori obținem drumuri oricât de „scurte”, deci distanțele nu mai sunt definite. Dacă nu există circuite negative, un drum minim poate fi ales fără repetarea vârfurilor, deci are cel mult $n-1$ arce.

Notăm cu $d_k(v)$ lungimea minimă a unui drum de la s la v cu *cel mult k arce*. Un astfel de drum fie are cel mult $k-1$ arce, fie se termină cu un arc uv precedat de un drum cu cel mult $k-1$ arce până la u. Rezultă recurența

$$d_0(s) = 0, \quad d_0(v) = \infty \ (v \neq s), \qquad d_k(v) = \min\Big(d_{k-1}(v),\ \min_{(u,v)\in E}\big(d_{k-1}(u) + w(uv)\big)\Big).$$

Algoritmul Bellman–Ford

Pasul 1. *Inițializare.* $d_0(s) = 0$, $d_0(v) = \infty$ pentru $v \neq s$.

Pasul 2. *Runde de relaxare.* Pentru $k = 1, 2, \ldots, n-1$ calculăm etichetele d_k din d_{k-1}, relaxând *toate* arcele. Când o etichetă scade, reținem și predecesorul.

Pasul 3. *Testul circuitelor negative.* Mai facem o rundă. Dacă vreo etichetă mai scade, graful are un circuit negativ accesibil din s. Altfel, $d_{n-1}(v) = \mathrm{dist}(s, v)$ pentru toți v.

Observații.

- Dacă o rundă nu modifică nicio etichetă, nici rundele următoare nu le mai modifică, deci algoritmul se poate opri mai devreme.
- *De ce detectează circuitele negative.* Dacă etichetele s-au stabilizat, pentru orice arc avem $d(v) \leq d(u) + w(uv)$. Adunând aceste inegalități pe un circuit C, etichetele se reduc și

obținem $0 \leq w(C)$. Deci, dacă există un circuit negativ accesibil din s, etichetele nu se pot stabiliza.

- Complexitatea este $O(nm)$, mai mare decât la Dijkstra. În schimb, algoritmul funcționează și cu ponderi negative.
- Un graf neorientat se poate trata înlocuind fiecare muchie cu două arce opuse. O muchie negativă uv dă însă circuitul negativ $u \to v \to u$, deci pentru grafuri neorientate cu muchii negative drumurile minime nu sunt definite în sensul de mai sus.

Drumul critic al unui proiect (metoda CPM)

Un proiect este format din activități. Fiecare activitate X are o durată $t(X)$, iar unele activități nu pot începe decât după terminarea altora. Construim un graf orientat cu activitățile ca vârfuri și cu un arc $X \to Y$ dacă X trebuie terminată înainte de începerea lui Y. Acest graf trebuie să fie aciclic (altfel proiectul este imposibil), deci admite o sortare topologică (vezi DFS). Se cer durata minimă a proiectului și activitățile care nu pot fi întârziate fără a întârzia întregul proiect.

Metoda drumului critic (CPM)

Pasul 1. *Sortare topologică* a activităților.

Pasul 2. *Calculul înainte*, în ordine topologică. Momentul cel mai devreme de începere este $ES(X) = \max EF(Y)$, după toți predecesorii Y ai lui X (sau 0, dacă X nu are predecesori). Momentul cel mai devreme de terminare este $EF(X) = ES(X) + t(X)$. Durata minimă a proiectului este $T = \max_X EF(X)$.

Pasul 3. *Calculul înapoi*, în ordine topologică inversă. Momentul cel mai târziu de terminare este $LF(X) = \min LS(Z)$, după toți succesorii Z ai lui X (sau T, dacă X nu are succesori). Momentul cel mai târziu de începere este $LS(X) = LF(X) - t(X)$.

Pasul 4. *Rezerve.* Rezerva activității X este $R(X) = LS(X) - ES(X)$. Activitățile cu $R(X) = 0$ se numesc *critice*.

Interpretare.

- $ES(X)$ este lungimea celui mai lung drum (suma duratelor) care se termină chiar înainte de X: activitatea X trebuie să aștepte terminarea tuturor lanțurilor de activități care o preced. Prin urmare T este lungimea *celui mai lung drum* din graf. Un astfel de drum este format numai din activități critice și se numește *drum critic*.
- O activitate X poate fi întârziată cu cel mult $R(X)$ fără a modifica durata T. Dacă o activitate critică întârzie cu δ, întregul proiect întârzie cu δ.
- Proiectul se poate scurta numai prin scurtarea activităților critice, și numai până când un alt drum devine critic.
- În grafuri oarecare, determinarea celui mai lung drum este o problemă dificilă. În grafurile aciclice ea se rezolvă ușor, în $O(n + m)$, prin parcurgerea vârfurilor în ordine topologică. Aceeași metodă dă și drumurile minime într-un graf aciclic, chiar cu ponderi negative.

Probleme rezolvate

Problemele marcate cu ★ au chenarul portocaliu. Ele conțin o idee mai deosebită sau o aplicație mai surprinzătoare a algoritmilor și pot fi tratate ca probleme de aprofundare.

Problema 1. Codificarea unei rețele de conducte

Enunț. O rețea de conducte leagă 8 puncte de distribuție, numerotate $1, \ldots, 8$, prin conductele

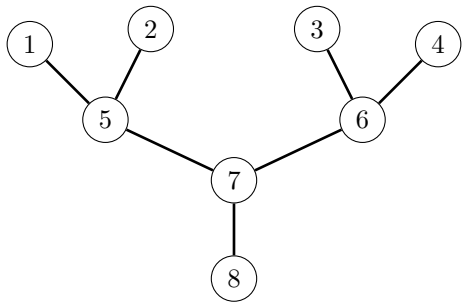
$$1\text{–}5, \quad 2\text{–}5, \quad 3\text{–}6, \quad 4\text{–}6, \quad 5\text{–}7, \quad 6\text{–}7, \quad 7\text{–}8.$$

a) Arătați că rețeaua este un arbore.

b) Determinați codul Prüfer al rețelei.

c) Verificați proprietatea gradelor pe codul obținut.

Rezolvare. a) Graful are $n = 8$ vârfuri și $7 = n - 1$ muchii. Este conex: vârfurile 1 și 2 sunt legate de 5, vârfurile 3 și 4 de 6, iar 5, 6 și 8 sunt legate de 7. Un graf conex cu n vârfuri și $n - 1$ muchii nu are cicluri (altfel, eliminând o muchie de pe un ciclu, am obține un graf conex cu $n - 2$ muchii, imposibil), deci este un arbore.



b) Aplicăm codificarea. La fiecare pas listăm frunzele arborelui curent, ștergem frunza cea mai mică și scriem vecinul ei.

Pas	Frunzele arborelui curent	Frunza ștearsă	Se scrie (vecinul)
1	$1, 2, 3, 4, 8$	1	5
2	$2, 3, 4, 8$	2	5
3	$3, 4, 5, 8$	3	6
4	$4, 5, 8$	4	6
5	$5, 6, 8$	5	7
6	$6, 8$	6	7

După pasul 2, vârful 5 a pierdut vecinii 1 și 2 și a devenit frunză. Analog, 6 a devenit frunză după pasul 4. După cei $n - 2 = 6$ pași rămân vârfurile 7 și 8, unite prin muchia 7–8. Codul Prüfer este

$$(5,\ 5,\ 6,\ 6,\ 7,\ 7).$$

c) Gradele sunt $d(5) = d(6) = d(7) = 3$, iar vârfurile $1, 2, 3, 4, 8$ sunt frunze. Conform proprietății gradelor, 5, 6 și 7 trebuie să apară de câte $3 - 1 = 2$ ori, iar frunzele deloc, ceea ce se verifică. Lungimea codului este $\sum_v (d(v) - 1) = 2(n - 1) - n = n - 2 = 6$.

Problema 2. Reconstruirea unei rețele de senzori

Enunț. O rețea de senzori în formă de arbore a fost transmisă comprimat, prin codul Prüfer

$$(4,\ 4,\ 2,\ 6,\ 6).$$

a) Câți senzori are rețeaua și care este gradul fiecăruia?

b) Reconstruiți rețeaua.

c) Verificați rezultatul, recalculând codul.

Rezolvare. a) Codul are lungime 5, deci rețeaua are $n = 5 + 2 = 7$ senzori. Gradul fiecărui senzor este $1 +$ numărul aparițiilor lui în cod:

Senzor	1	2	3	4	5	6	7
Apariții în cod	0	1	0	2	0	2	0
Grad $d(v)$	1	2	1	3	1	3	1

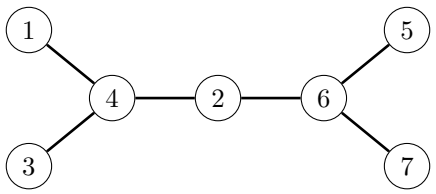
Verificare: suma gradelor este $12 = 2(n-1)$, cum trebuie pentru un arbore cu 7 vârfuri.

b) Aplicăm decodificarea. La fiecare pas, x este cel mai mic senzor nește rs de grad 1, care se leagă de a_i. Apoi ștergem x și micșorăm $d(a_i)$.

i	a_i	Vârfuri cu $d = 1$	Muchie adăugată	Modificări
1	4	$1, 3, 5, 7$	1–4	$d(4) = 2$
2	4	$3, 5, 7$	3–4	$d(4) = 1$
3	2	$4, 5, 7$	4–2	$d(2) = 1$
4	6	$2, 5, 7$	2–6	$d(6) = 2$
5	6	$5, 7$	5–6	$d(6) = 1$

Au rămas nește rse vârfurile 6 și 7, ambele cu $d = 1$. Adăugăm muchia 6–7. Rețeaua are muchiile

$$1\text{–}4,\quad 3\text{–}4,\quad 2\text{–}4,\quad 2\text{–}6,\quad 5\text{–}6,\quad 6\text{–}7.$$



c) Codificăm arborele obținut. Frunzele sunt $1, 3, 5, 7$: ștergem 1 (scriem 4), apoi 3 (scriem 4). Acum 4 este frunză, deci frunzele sunt $4, 5, 7$: ștergem 4 (scriem 2). Vârful 2 devine frunză: ștergem 2 (scriem 6), apoi 5 (scriem 6). Rămân 6 și 7. Codul este $(4, 4, 2, 6, 6)$, cel transmis.

Problema 3. Codul Prüfer al unei rețele ierarhice

Enunț. O rețea ierarhică de distribuție are 15 noduri, numerotate $1, \ldots, 15$. Legăturile sunt date prin listele de vecini:

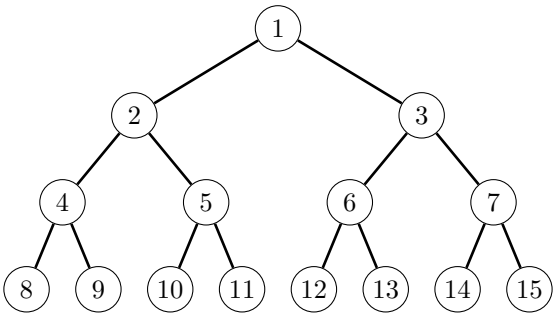
Vârf	Vecini	Vârf	Vecini	Vârf	Vecini	Vârf	Vecini
1	$2, 3$	5	$2, 10, 11$	9	4	13	6
2	$1, 4, 5$	6	$3, 12, 13$	10	5	14	7
3	$1, 6, 7$	7	$3, 14, 15$	11	5	15	7
4	$2, 8, 9$	8	4	12	6		

a) Desenați graful și verificați că este un arbore.

b) Determinați codul Prüfer corespunzător.

c) Verificați proprietatea gradelor pe codul obținut.

Rezolvare. a) Suma lungimilor listelor este $2 + 3 + 3 + 3 + 3 + 3 + 3 + 8 \cdot 1 = 28$, deci graful are $28/2 = 14 = n - 1$ muchii. Observăm o regulă simplă: vecinii vârfului k sunt $\lfloor k/2 \rfloor$ (pentru $k \geq 2$), $2k$ și $2k + 1$ (pentru $k \leq 7$). Așadar fiecare vârf $k \geq 2$ este legat de vârful $\lfloor k/2 \rfloor$, mai mic decât el, iar mergând repetat astfel ajungem în vârful 1. Prin urmare graful este conex. Un graf conex cu n vârfuri și $n - 1$ muchii este un arbore. Desenăm arborele pe niveluri, cu vârful 1 sus:



Este un *arbore binar complet*: vârfurile $1, \ldots, 7$ au câte doi „descendenți”, iar vârfurile $8, \ldots, 15$ sunt frunze.

b) Aplicăm codificarea. La fiecare pas listăm frunzele arborelui curent, ștergem frunza cea mai mică și scriem vecinul ei.

Pas	Frunzele arborelui curent	Frunza ștearsă	Se scrie (vecinul)
1	$8, 9, 10, 11, 12, 13, 14, 15$	8	4
2	$9, 10, 11, 12, 13, 14, 15$	9	4
3	$4, 10, 11, 12, 13, 14, 15$	4	2
4	$10, 11, 12, 13, 14, 15$	10	5
5	$11, 12, 13, 14, 15$	11	5
6	$5, 12, 13, 14, 15$	5	2
7	$2, 12, 13, 14, 15$	2	1
8	$1, 12, 13, 14, 15$	1	3
9	$12, 13, 14, 15$	12	6
10	$13, 14, 15$	13	6
11	$6, 14, 15$	6	3
12	$3, 14, 15$	3	7
13	$14, 15$	14	7

Se observă cum apar frunze noi pe parcurs. După ștergerea lui 8 și 9, vârful 4 devine frunză și este ales la pasul 3, fiind mai mic decât 10. La fel, 5 devine frunză după pasul 5, iar 2 după

pasul 6. După cei $n-2=13$ pași rămân vârfurile 7 și 15, unite prin muchia 7–15. Codul Prüfer este

$$(4,\ 4,\ 2,\ 5,\ 5,\ 2,\ 1,\ 3,\ 6,\ 6,\ 3,\ 7,\ 7).$$

c) Vârfurile $2,\dots,7$ au gradul 3, deci trebuie să apară în cod de câte 2 ori. Vârful 1 are gradul 2, deci apare o singură dată. Frunzele $8,\dots,15$ nu apar. Codul verifică aceste condiții, iar lungimea lui este $6\cdot 2+1=13=n-2$.

Problema 4. Arborele cu codul Prüfer 25121989

Enunț. Un arbore are codul Prüfer

$$(2,\ 5,\ 1,\ 2,\ 1,\ 9,\ 8,\ 9).$$

a) Câte vârfuri are arborele și care este gradul fiecărui vârf?

b) Construiți și desenați arborele.

c) Verificați rezultatul, recalculând codul.

Rezolvare. a) Codul are lungime 8, deci arborele are $n=8+2=10$ vârfuri. Gradul fiecărui vârf este 1 + numărul aparițiilor lui în cod:

Vârf	1	2	3	4	5	6	7	8	9	10
Apariții în cod	2	2	0	0	1	0	0	1	2	0
Grad $d(v)$	3	3	1	1	2	1	1	2	3	1

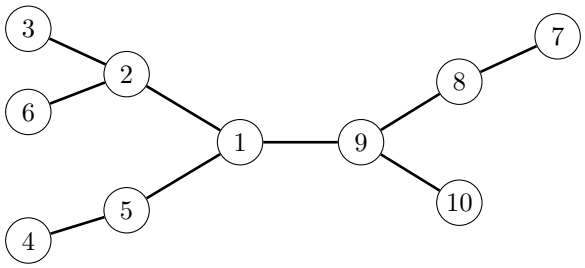
Verificare: suma gradelor este $18=2(n-1)$. Frunzele sunt vârfurile care nu apar în cod: $3,4,6,7,10$.

b) Aplicăm decodificarea. La fiecare pas, x este cel mai mic vârf nesters de grad 1, care se leagă de a_i. Apoi ștergem x și micșorăm $d(a_i)$.

i	a_i	Vârfuri cu $d=1$	Muchie adăugată	Modificări
1	2	$3,4,6,7,10$	3–2	$d(2)=2$
2	5	$4,6,7,10$	4–5	$d(5)=1$
3	1	$5,6,7,10$	5–1	$d(1)=2$
4	2	$6,7,10$	6–2	$d(2)=1$
5	1	$2,7,10$	2–1	$d(1)=1$
6	9	$1,7,10$	1–9	$d(9)=2$
7	8	$7,10$	7–8	$d(8)=1$
8	9	$8,10$	8–9	$d(9)=1$

De exemplu, la pasul 3 vârful 5 are deja $d=1$ (după pasul 2) și este mai mic decât 6, deci el se leagă de $a_3=1$. Au rămas nesterse vârfurile 9 și 10, ambele cu $d=1$. Adăugăm muchia 9–10. Arborele are muchiile

$$2\text{–}3,\quad 4\text{–}5,\quad 1\text{–}5,\quad 2\text{–}6,\quad 1\text{–}2,$$
$$1\text{–}9,\quad 7\text{–}8,\quad 8\text{–}9,\quad 9\text{–}10.$$



c) Codificăm arborele obținut. Frunzele sunt $3, 4, 6, 7, 10$: ștergem 3 (scriem 2), apoi 4 (scriem 5). Acum 5 este frunză și este cea mai mică: ștergem 5 (scriem 1), apoi 6 (scriem 2). Vârful 2 devine frunză: ștergem 2 (scriem 1). Vârful 1 devine frunză: ștergem 1 (scriem 9). Ștergem 7 (scriem 8), apoi 8, devenit frunză (scriem 9). Rămân 9 și 10. Codul este $(2, 5, 1, 2, 1, 9, 8, 9)$, cel dat.

Problema 5. Câte rețele arbore există?

Enunț. Cinci orașe, numerotate $1, \ldots, 5$, trebuie legate prin 4 drumuri astfel încât din orice oraș să se poată ajunge în oricare altul (rețeaua este un arbore).

a) Câte astfel de rețele există?

b) În câte dintre ele orașul 1 este capăt de rețea (are un singur drum)?

c) În câte dintre ele din orașul 1 pleacă exact 3 drumuri?

d) Câte rețele au gradele $d(1) = 3$, $d(4) = 2$ și $d(2) = d(3) = d(5) = 1$? Desenați-le.

Rezolvare. Prin codul Prüfer, rețelele corespund bijectiv șirurilor (a_1, a_2, a_3) cu valori în $\{1, \ldots, 5\}$, iar orașul v apare în cod de $d(v) - 1$ ori. Numărăm deci șiruri.

a) Fiecare dintre cele 3 poziții poate lua 5 valori: $5^3 = 125$ rețele (teorema lui Cayley, n^{n-2} pentru $n = 5$).

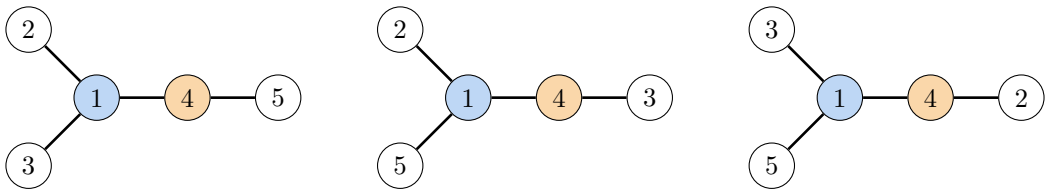
b) Orașul 1 are gradul 1 exact când nu apare în cod. Fiecare poziție ia una dintre celelalte 4 valori: $4^3 = 64$ rețele.

c) $d(1) = 3$ înseamnă că 1 apare de exact două ori. Alegem cele 2 poziții ale lui 1 în $\binom{3}{2} = 3$ moduri, iar poziția rămasă ia una dintre valorile $2, 3, 4, 5$: $3 \cdot 4 = 12$ rețele.

d) Codul conține 1 de două ori, 4 o dată și nicio altă valoare, deci este o permutare a șirului $(1, 1, 4)$. Există $\frac{3!}{2!\,1!} = 3$ astfel de șiruri (formula cu grade date): $(1, 1, 4)$, $(1, 4, 1)$, $(4, 1, 1)$. Prin decodificare obținem rețelele cu muchiile

$$\{12, 13, 14, 45\}, \qquad \{12, 14, 15, 34\}, \qquad \{13, 14, 15, 24\}.$$

În fiecare, orașul 1 este legat de 4 și de două dintre orașele $2, 3, 5$, iar al treilea dintre acestea „atârnă” de 4. Cele trei rețele diferă prin orașul legat de 4.



Problema 6. Rețele în formă de stea și de lanț

Enunț. Șase calculatoare, numerotate $1, \ldots, 6$, sunt legate printr-o rețea arbore.

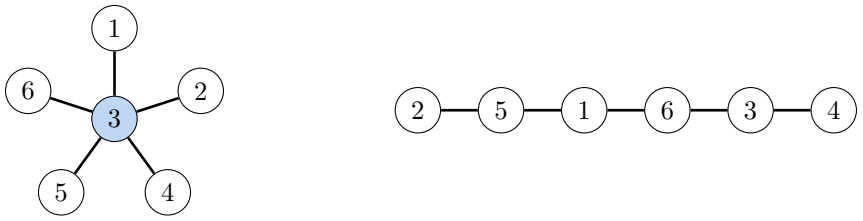
a) Determinați codul Prüfer al rețelei în formă de *stea*, în care calculatorul 3 este legat de toate celelalte.

b) Determinați codul Prüfer al rețelei în formă de *lanț* 2–5–1–6–3–4.

c) Arătați că un arbore este o stea dacă și numai dacă toate elementele codului său sunt egale și că este un lanț dacă și numai dacă elementele codului său sunt distincte.

d) Câte rețele în formă de lanț se pot forma cu cele 6 calculatoare?

Rezolvare.



a) În stea, frunzele sunt $1, 2, 4, 5, 6$, iar fiecare are vecinul 3. La fiecare pas ștergem cea mai mică frunză și scriem 3: ștergem pe rând 1, 2, 4, 5 și rămân 3 și 6. Codul este

$$(3,\ 3,\ 3,\ 3).$$

b) Aplicăm codificarea pe lanț.

Pas	Frunzele arborelui curent	Frunza ștearsă	Se scrie (vecinul)
1	$2, 4$	2	5
2	$4, 5$	4	3
3	$3, 5$	3	6
4	$5, 6$	5	1

Un lanț are mereu exact două frunze, capetele lui. După fiecare ștergere, vecinul frunzei șterse devine noul capăt. Rămân vârfurile 1 și 6, iar codul este

$$(5,\ 3,\ 6,\ 1).$$

c) Folosim proprietatea gradelor: vârful v apare în cod de $d(v) - 1$ ori.

- *Stea.* Centrul unei stele cu n vârfuri are gradul $n - 1$, deci apare de $n - 2$ ori, adică pe toate pozițiile codului. Reciproc, dacă toate elementele codului sunt egale cu c, atunci $d(c) = 1 + (n - 2) = n - 1$. Vârful c este legat de toate celelalte, iar arborele, având $n - 1$ muchii, nu mai are alte muchii. Este deci o stea cu centrul c.
- *Lanț.* Într-un lanț toate vârfurile au gradul cel mult 2, deci fiecare vârf apare în cod cel mult o dată: elementele codului sunt distincte. Reciproc, dacă elementele sunt distincte, fiecare vârf apare cel mult o dată, deci are gradul cel mult 2. Un arbore cu toate gradele cel mult 2 este un lanț. Într-adevăr, pornind de la o frunză și mergând mereu pe muchia nefolosită, nu ne putem ramifica și nici întoarce, deci parcurgem toate vârfurile într-un singur drum.

Codul lanțului de la b), $(5, 3, 6, 1)$, are elementele distincte. Ele sunt exact vârfurile interioare ale lanțului $(5, 1, 6, 3)$, dar nu neapărat în ordinea de pe lanț.

d) După c), rețelele în formă de lanț corespund codurilor de lungime 4 cu elemente distincte din $\{1,\dots,6\}$. Numărul lor este

$$6 \cdot 5 \cdot 4 \cdot 3 = 360.$$

Verificare directă: un lanț este o așezare a celor 6 calculatoare în linie, adică o permutare, iar o permutare și inversa ei dau același lanț. Obținem $6!/2 = 360$.

Problema 7. Cablarea a șapte localități (algoritmul lui Kruskal)

Enunț. Șapte localități $A,\dots,G$ trebuie legate printr-o rețea de fibră optică, astfel încât din orice localitate să se poată comunica cu oricare alta. Traseele posibile și costurile lor (zeci de mii de lei) sunt:

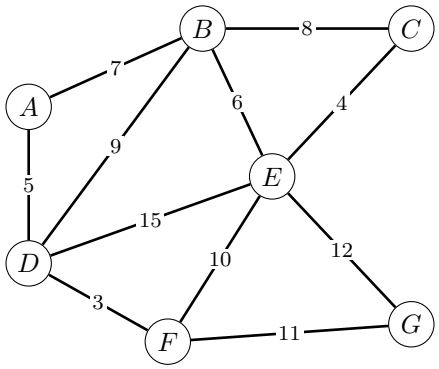
$$AB=7,\quad AD=5,\quad BC=8,\quad BD=9,\quad BE=6,\quad CE=4,$$
$$DE=15,\quad DF=3,\quad EF=10,\quad EG=12,\quad FG=11.$$

a) Aplicați algoritmul lui Kruskal, precizând componentele după fiecare pas.

b) Determinați rețeaua de cost minim și costul ei.

c) Este această rețea unică?

Rezolvare. Localitățile sunt vârfurile unui graf, iar traseele posibile sunt muchiile, ponderate cu costurile. O rețea în care oricare două localități comunică și care are cost minim nu conține cicluri (altfel am putea elimina o muchie de pe ciclu, micșorând costul), deci este un arbore parțial de cost minim.



a) Ordonăm muchiile crescător:

$$DF\,(3),\quad CE\,(4),\quad AD\,(5),\quad BE\,(6),\quad AB\,(7),\quad BC\,(8),$$
$$BD\,(9),\quad EF\,(10),\quad FG\,(11),\quad EG\,(12),\quad DE\,(15).$$

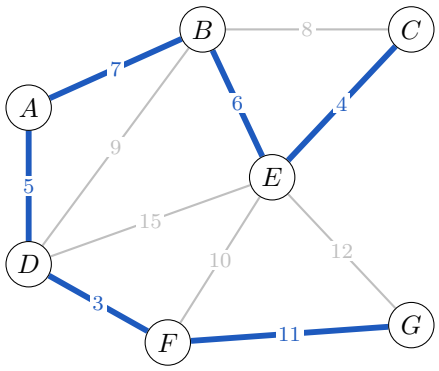
Le examinăm pe rând. Inițial fiecare localitate formează singură o componentă.

Muchie	Cost	Se adaugă?	Componente după pas
DF	3	da	$\{A\},\{B\},\{C\},\{D,F\},\{E\},\{G\}$
CE	4	da	$\{A\},\{B\},\{C,E\},\{D,F\},\{G\}$
AD	5	da	$\{A,D,F\},\{B\},\{C,E\},\{G\}$
BE	6	da	$\{A,D,F\},\{B,C,E\},\{G\}$
AB	7	da	$\{A,B,C,D,E,F\},\{G\}$
BC	8	nu (ciclu $B\,C\,E$)	neschimbate
BD	9	nu (ciclu $B\,A\,D$)	neschimbate
EF	10	nu (ciclu $E\,B\,A\,D\,F$)	neschimbate
FG	11	da	$\{A,B,C,D,E,F,G\}$

O muchie este respinsă când ambele capete sunt deja în aceeași componentă. De exemplu, BC ar închide ciclul $B \to C \to E \to B$, deoarece B și C sunt deja legate prin E. După FG am ales $n-1=6$ muchii, deci ne oprim. Muchiile EG și DE nu mai trebuie examinate.

b) Rețeaua de cost minim are muchiile DF, CE, AD, BE, AB, FG (albastre în desen), cu costul

$$3+4+5+6+7+11=36.$$



c) Toate costurile sunt distincte, deci APM este unic (vezi observațiile din partea teoretică). Orice altă rețea conexă costă mai mult de 36.

Problema 8. Aceeași rețea, cu algoritmul lui Prim

Enunț. Considerăm rețeaua din Problema 7.

a) Aplicați algoritmul lui Prim pornind din localitatea A, precizând etichetele după fiecare pas.

b) Comparați rezultatul și ordinea alegerii muchiilor cu cele obținute prin algoritmul lui Kruskal.

c) Comparați APM cu arborele drumurilor minime din A (algoritmul lui Dijkstra). De ce diferă?

Rezolvare. a) Eticheta $c(v)$ este costul celei mai ieftine muchii de la v la arborele construit până atunci. Valorile îngroșate corespund localităților deja incluse în arbore.

Pas	Ales	Muchia	A	B	C	D	E	F	G
1	A	–	**0**	7	∞	5	∞	∞	∞
2	D	AD	**0**	7	∞	**5**	15	3	∞
3	F	DF	**0**	7	∞	**5**	10	**3**	11
4	B	AB	**0**	**7**	8	**5**	6	**3**	11
5	E	BE	**0**	**7**	4	**5**	**6**	**3**	11
6	C	CE	**0**	**7**	**4**	**5**	**6**	**3**	11
7	G	FG	**0**	**7**	**4**	**5**	**6**	**3**	**11**

Detaliem actualizările.

- Pasul 1: din A obținem $c(B)=7$ și $c(D)=5$.
- Pasul 2: se alege D (5). Obținem $c(E)=15$ și $c(F)=3$. Muchia DB (9) nu îmbunătățește eticheta 7 a lui B.
- Pasul 3: se alege F (3). Pentru E, $10<15$, deci $c(E)=10$. Pentru G obținem 11.
- Pasul 4: se alege B (7). Pentru C obținem 8, iar pentru E, $6<10$, deci $c(E)=6$.

- Pasul 5: se alege E (6). Pentru C, $4 < 8$, deci $c(C) = 4$. Muchia EG (12) nu îmbunătățește eticheta 11 a lui G.
- Pașii 6 și 7: se aleg C (4) și G (11).

Arborele obținut are muchiile AD, DF, AB, BE, CE, FG, cu costul 36.

b) Este exact rețeaua găsită cu algoritmul lui Kruskal, cum era de așteptat, deoarece APM este unic. Ordinea alegerii diferă: Kruskal alege muchiile în ordinea globală a costurilor (DF, CE, AD, BE, AB, FG), construind mai multe „bucăți" care se unesc, în timp ce Prim crește un singur arbore pornind din A (AD, DF, AB, BE, CE, FG).

c) Cei doi arbori răspund la întrebări diferite:

- APM răspunde la întrebarea: *cum legăm toate localitățile cu cost total minim?* Se minimizează suma costurilor tuturor muchiilor alese.
- Arborele drumurilor minime din A răspunde la întrebarea: *care este cel mai scurt drum de la A la fiecare localitate?* Se minimizează, separat, distanța de la A la fiecare vârf.

Pasul 1: aplicăm algoritmul lui Dijkstra din A. Eticheta $d(v)$ este lungimea celui mai scurt drum cunoscut de la A la v. Valorile îngroșate sunt definitive.

Pas	Ales	Muchia	A	B	C	D	E	F	G
1	A	–	**0**	7	∞	5	∞	∞	∞
2	D	AD	**0**	7	∞	**5**	20	8	∞
3	B	AB	**0**	**7**	15	**5**	13	8	∞
4	F	DF	**0**	**7**	15	**5**	13	**8**	19
5	E	BE	**0**	**7**	15	**5**	**13**	**8**	19
6	C	BC	**0**	**7**	**15**	**5**	**13**	**8**	19
7	G	FG	**0**	**7**	**15**	**5**	**13**	**8**	**19**

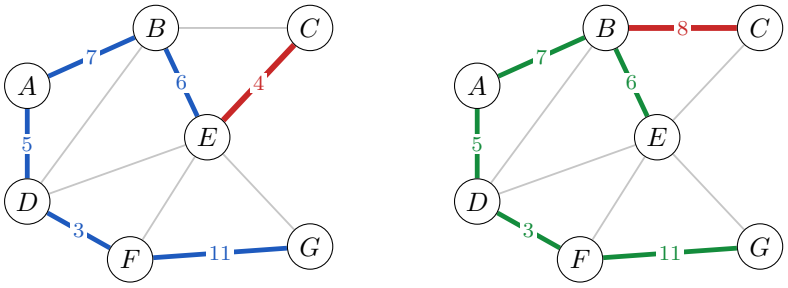
Detaliem relaxările și le comparăm cu actualizările făcute de Prim la punctul a).

- Pasul 2: se alege D ($d = 5$). Pentru E obținem $5 + 15 = 20$, iar pentru F obținem $5 + 3 = 8$. Prim punea aici $c(E) = 15$ și $c(F) = 3$, adică doar costul muchiei, fără distanța deja parcursă de la A la D.
- Pasul 3: se alege B ($d = 7$). Pentru C obținem $7 + 8 = 15$, iar pentru E, $7 + 6 = 13 < 20$, deci $d(E) = 13$.
- Pasul 4: se alege F ($d = 8$). Pentru E, $8 + 10 = 18$ nu este mai mic decât 13. Pentru G obținem $8 + 11 = 19$.
- Pasul 5: se alege E ($d = 13$). Pentru C, $13 + 4 = 17$ *nu* este mai mic decât 15, deci C rămâne legat de B. Pentru G, $13 + 12 = 25$ nu este mai mic decât 19.
- Pașii 6 și 7: se aleg C ($d = 15$, predecesor B) și G ($d = 19$, predecesor F).

Predecesorii dau arborele drumurilor minime, cu muchiile AB, AD, BC, BE, DF, FG și costul total $7 + 5 + 8 + 6 + 3 + 11 = 40$.

Pasul 2: comparăm cei doi arbori.

APM (Prim, Kruskal): cost 36 **Drumuri minime din A: cost 40**



Cei doi arbori au în comun cinci muchii, AB, AD, BE, DF, FG, și diferă printr-o singură muchie (roșie în desen): APM folosește CE, iar arborele drumurilor minime folosește BC.

Pasul 3: comparăm distanțele de la A. În fiecare arbore există un singur drum de la A la fiecare localitate. Lungimile acestor drumuri sunt:

Localitatea	B	C	D	E	F	G
Drumul din APM	7	**17**	5	13	8	19
Drumul din arborele lui Dijkstra	7	**15**	5	13	8	19

Singura diferență este la C:

- În APM, C este legată prin CE, cea mai ieftină muchie a ei (cost 4). Drumul de la A la C este însă ocolit: $A \to B \to E \to C$, de lungime $7 + 6 + 4 = 17$.
- În arborele lui Dijkstra, C este legată prin BC (cost 8), mai scumpă cu 4. În schimb, drumul $A \to B \to C$ are lungimea $7 + 8 = 15$, cel mai scurt drum posibil de la A la C.

Concluzie. Trecerea de la CE la BC crește costul total al rețelei cu $8 - 4 = 4$ (de la 36 la 40), dar scurtează drumul de la A la C cu $17 - 15 = 2$. Prim compară, pentru fiecare vârf nou, doar costul muchiei care îl leagă ($4 < 8$, deci alege CE). Dijkstra compară lungimea întregului drum de la A ($17 > 15$, deci alege BC). APM minimizează costul total al rețelei, iar arborele lui Dijkstra minimizează distanța de la A la fiecare localitate. Sunt două obiective diferite, deci și arborii pot fi diferiți.

Problema 9. Numărul minim de stații într-o rețea de metrou

Enunț. O rețea de metrou are 9 stații $A, \ldots, I$. Tronsoanele directe între stații sunt

$$AB,\ AC,\ AD,\ BE,\ CE,\ CF,\ DF,\ EG,\ EH,\ FH,\ GI,\ HI.$$

a) Aplicați parcurgerea în lățime pornind din stația A, precizând conținutul cozii la fiecare pas.

b) Determinați, pentru fiecare stație, numărul minim de tronsoane de la A și un traseu cu acest număr de tronsoane până la stația I.

c) Desenați arborele BFS.

Rezolvare. Stațiile sunt vârfurile unui graf, iar tronsoanele sunt muchiile. Listele de adiacență, în ordine alfabetică, sunt:

Vârf	A	B	C	D	E	F	G	H	I
Vecini	B, C, D	A, E	A, E, F	A, F	B, C, G, H	C, D, H	E, I	E, F, I	G, H

a) Inițial $\ell(A) = 0$ și $Q = (A)$. La fiecare pas extragem primul vârf din coadă și adăugăm la sfârșitul ei vecinii lui nevizitați.

Pas	Vârf extras	Vecini nevizitați (adăugați)	Niveluri noi	Coada după pas
1	A	B, C, D	$\ell = 1$	B, C, D
2	B	E	$\ell(E) = 2$	C, D, E
3	C	F (E e deja vizitat)	$\ell(F) = 2$	D, E, F
4	D	–	–	E, F
5	E	G, H	$\ell = 3$	F, G, H
6	F	–	–	G, H
7	G	I	$\ell(I) = 4$	H, I
8	H	–	–	I
9	I	–	–	(vidă)

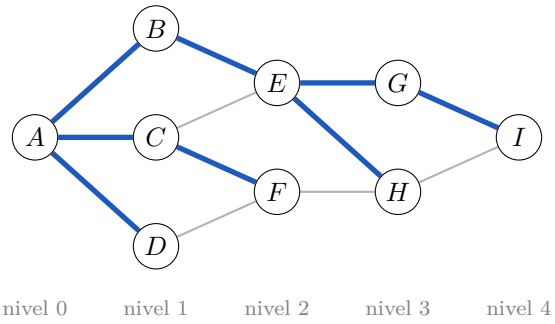
Ordinea de vizitare este $A, B, C, D, E, F, G, H, I$. Predecesorii sunt $p(B) = p(C) = p(D) = A$, $p(E) = B$, $p(F) = C$, $p(G) = p(H) = E$, $p(I) = G$.

b) Numărul minim de tronsoane este nivelul:

Stația	A	B	C	D	E	F	G	H	I
Tronsoane de la A	0	1	1	1	2	2	3	3	4

Pentru I mergem înapoi pe predecesori: $I \leftarrow G \leftarrow E \leftarrow B \leftarrow A$, deci traseul $A \to B \to E \to G \to I$, cu 4 tronsoane. Nu există un traseu mai scurt, deoarece BFS dă exact distanța în număr de muchii. Traseul nu este unic: și $A \to C \to F \to H \to I$ are 4 tronsoane.

c) În desen, muchiile arborelui BFS sunt albastre îngroșate, iar celelalte muchii ale grafului sunt gri. Stațiile sunt așezate pe coloane, după nivel.



Problema 10. Împărțirea în două schimburi, cu BFS

Enunț. Opt angajați, numerotați $1, \ldots, 8$, trebuie împărțiți în două schimburi. Perechile de angajați care nu pot lucra în același schimb sunt

$$12,\ 13,\ 24,\ 35,\ 46,\ 57,\ 68,\ 78.$$

a) Aplicați BFS din vârful 1 și determinați nivelurile.

b) Folosind nivelurile, decideți dacă împărțirea este posibilă și precizați schimburile.

c) Ce se întâmplă dacă nici angajații 4 și 5 nu pot lucra împreună? Găsiți un ciclu impar.

Rezolvare. Construim graful „de conflict”: vârfurile sunt angajații, muchiile sunt perechile de mai sus. O împărțire corectă în două schimburi înseamnă exact că graful este bipartit.

a) BFS din 1: extragem 1 și adăugăm 2 și 3 (nivelul 1). Din 2 adăugăm 4, iar din 3 adăugăm 5 (nivelul 2). Din 4 adăugăm 6, iar din 5 adăugăm 7 (nivelul 3). Din 6 adăugăm 8 (nivelul 4). Din 7 și 8 nu mai apar vârfuri noi. Nivelurile sunt:

Vârf	1	2	3	4	5	6	7	8
Nivel ℓ	0	1	1	2	2	3	3	4

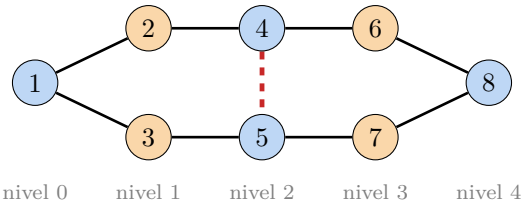
b) Verificăm fiecare muchie: 12, 13 (nivelurile 0 și 1), 24, 35 (1 și 2), 46, 57 (2 și 3), 68, 78 (3 și 4). Nicio muchie nu unește vârfuri de pe același nivel, deci graful este bipartit. Schimburile sunt date de paritatea nivelului:

$$\text{schimbul I: } \{1,4,5,8\} \text{ (niveluri pare)}, \qquad \text{schimbul II: } \{2,3,6,7\} \text{ (niveluri impare)}.$$

c) Nivelurile nu se schimbă, dar noua muchie 45 unește două vârfuri de pe nivelul 2. Drumurile din arborele BFS de la 1 la 4 și la 5 sunt $1,2,4$ și $1,3,5$. Ultimul lor vârf comun este $z=1$, de pe nivelul $j=0$. Obținem ciclul

$$1 \to 2 \to 4 \to 5 \to 3 \to 1,$$

de lungime $2(2-0)+1=5$, impară. Un graf cu un ciclu impar nu este bipartit, deci împărțirea în două schimburi nu mai este posibilă.



Culorile vârfurilor arată schimburile de la b). Muchia nouă 45 (roșie, punctată) leagă două vârfuri de aceeași culoare, de pe același nivel.

★ Problema 11. Problema vaselor, rezolvată cu BFS

Enunț. Avem un vas A de 3 litri și un vas B de 5 litri, ambele goale, fără gradații, și o sursă de apă. O operație constă în: umplerea unui vas, golirea unui vas sau turnarea apei dintr-un vas în celălalt, până când primul se golește sau al doilea se umple.

a) Modelați problema printr-un graf.

b) Aplicați BFS din starea inițială.

c) Care este numărul minim de operații prin care se obțin exact 4 litri într-un vas? Precizați operațiile.

Rezolvare. a) O *stare* este perechea (a,b), unde $a \in \{0,1,2,3\}$ și $b \in \{0,\dots,5\}$ sunt cantitățile din vasele A și B. Stările sunt vârfurile unui graf orientat, cu un arc de la o stare la alta dacă a doua se obține din prima printr-o operație. Starea inițială este $(0,0)$. Din starea (a,b) se poate ajunge în:

- $(3,b)$ sau $(a,5)$, prin umplerea lui A, respectiv B.
- $(0,b)$ sau $(a,0)$, prin golirea lui A, respectiv B.
- $(a-t,\, b+t)$ cu $t=\min(a,5-b)$, prin turnarea din A în B (notată $A \to B$).
- $(a+t,\, b-t)$ cu $t=\min(b,3-a)$, prin turnarea din B în A (notată $B \to A$).

Numărul minim de operații până la o stare este distanța în acest graf, deci o găsim cu BFS. Operațiile sunt examinate în ordinea de mai sus.

b) Tabelul arată, pentru fiecare stare extrasă din coadă, stările noi descoperite, operația folosită și conținutul cozii Q după pas (începutul cozii este la stânga). Inițial coada conține doar starea $(0,0)$. La fiecare pas se extrage prima stare din coadă, iar stările noi se adaugă la sfârșitul ei.

Pas	Stare extrasă	Nivel	Stări noi (operația)	Coada Q după pas
0	–	–	–	$(0,0)$
1	$(0,0)$	0	$(3,0)$ (umple A), $(0,5)$ (umple B)	$(3,0),(0,5)$
2	$(3,0)$	1	$(3,5)$ (umple B), $(0,3)$ ($A \to B$)	$(0,5),(3,5),(0,3)$
3	$(0,5)$	1	$(3,2)$ ($B \to A$)	$(3,5),(0,3),(3,2)$
4	$(3,5)$	2	–	$(0,3),(3,2)$
5	$(0,3)$	2	$(3,3)$ (umple A)	$(3,2),(3,3)$
6	$(3,2)$	2	$(0,2)$ (golește A)	$(3,3),(0,2)$
7	$(3,3)$	3	$(1,5)$ ($A \to B$)	$(0,2),(1,5)$
8	$(0,2)$	3	$(2,0)$ ($B \to A$)	$(1,5),(2,0)$
9	$(1,5)$	4	$(1,0)$ (golește B)	$(2,0),(1,0)$
10	$(2,0)$	4	$(2,5)$ (umple B)	$(1,0),(2,5)$
11	$(1,0)$	5	$(0,1)$ ($A \to B$)	$(2,5),(0,1)$
12	$(2,5)$	5	$(3,4)$ ($B \to A$)	$(0,1),(3,4)$
13	$(0,1)$	6	$(3,1)$ (umple A)	$(3,4),(3,1)$
14	$(3,4)$	6	$(0,4)$ (golește A)	$(3,1),(0,4)$
15	$(3,1)$	7	–	$(0,4)$
16	$(0,4)$	7	–	vidă

Sunt accesibile 16 stări din cele $4 \cdot 6 = 24$. Nu se poate ajunge, de exemplu, în $(1,1)$: după orice operație, cel puțin un vas este gol sau plin.

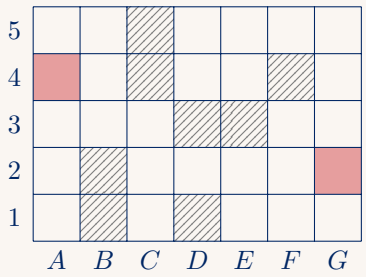
c) Prima stare care conține 4 litri este $(3,4)$, pe nivelul 6. Deci sunt necesare cel puțin 6 operații, iar 6 operații ajung. Mergând înapoi pe predecesori obținem:

Operația	Ce se face	Starea (a,b)
1	umple B	$(0,5)$
2	toarnă din B în A	$(3,2)$
3	golește A	$(0,2)$
4	toarnă din B în A	$(2,0)$
5	umple B	$(2,5)$
6	toarnă din B în A	$(3,4)$

La final vasul B conține 4 litri. Avantajul BFS este că garantează minimalitatea: nicio stare cu 4 litri nu apare pe nivelurile $0,\ldots,5$.

★ Problema 12. Răspândirea unui incendiu (BFS din mai multe surse)

Enunț. Harta unei păduri este împărțită în zone pătrate, ca în figura de mai jos (coloanele $A,\ldots,G$, rândurile $1,\ldots,5$). Zonele hașurate sunt lacuri și stânci, care nu ard. La ora 0 focul izbucnește simultan în zonele $A4$ și $G2$ (roșii în figură). În fiecare oră, focul trece din fiecare zonă care arde în zonele vecine (sus, jos, stânga, dreapta) care pot arde.



a) Modelați răspândirea focului printr-un graf și explicați cum se adaptează BFS pentru două focare.

b) Determinați ora la care ia foc fiecare zonă. Care zonă arde ultima?

c) Dacă ar fi existat doar focarul din $A4$, după câte ore ar fi ars întreaga pădure? Arătați, pentru zonele $D4$ și $C1$, că ora găsită la b) este minimul dintre orele date de fiecare focar separat.

Rezolvare. a) Zonele care pot arde sunt vârfurile unui graf, iar două zone vecine (cu o latură comună) sunt legate printr-o muchie. Ora la care ia foc o zonă v este numărul minim de muchii al unui drum de la un focar la v, adică distanța de la v la *cel mai apropiat* focar.

Aplicăm BFS cu mai multe surse: punem *ambele* focare în coadă de la început, cu nivelul 0. Apoi BFS decurge obișnuit. Justificare: adăugăm un vârf imaginar S, legat de $A4$ și de $G2$, și aplicăm BFS din S. Primul pas al acestui BFS pune în coadă exact cele două focare, iar nivelurile obținute sunt cele căutate, mărite cu 1.

b) Examinăm vecinii în ordinea sus, jos, stânga, dreapta. Primii pași sunt:

Zonă extrasă	Ora	Zone noi	Coada Q după pas
–	–	–	$A4,\ G2$
$A4$	0	$A5,\ A3,\ B4$	$G2,\ A5,\ A3,\ B4$
$G2$	0	$G3,\ G1,\ F2$	$A5,\ A3,\ B4,\ G3,\ G1,\ F2$
$A5$	1	$B5$	$A3,\ B4,\ G3,\ G1,\ F2,\ B5$
$A3$	1	$A2,\ B3$	$B4,\ G3,\ G1,\ F2,\ B5,\ A2,\ B3$
$B4$	1	–	$G3,\ G1,\ F2,\ B5,\ A2,\ B3$

Cele două focare se extind „în paralel”: în coadă alternează zone din vestul și din estul pădurii, dar ordinea pe ore este respectată. Continuând, obținem zonele care iau foc la fiecare oră:

Ora	Zone care iau foc
0	$A4,\ G2$ (focarele)
1	$A3,\ A5,\ B4,\ F2,\ G1,\ G3$
2	$A2,\ B3,\ B5,\ E2,\ F1,\ F3,\ G4$
3	$A1,\ C3,\ D2,\ E1,\ G5$
4	$C2,\ F5$
5	$C1,\ E5$
6	$D5,\ E4$
7	$D4$

5	1	2		6	5	4	3
4	0	1		7	6		2
3	1	2	3			2	1
2	2		4	3	2	1	0
1	3		5		3	2	1
	A	B	C	D	E	F	G

Toate cele 27 de zone care pot arde iau foc. Ultima arde zona $D4$, la ora 7, deci întreaga pădure arde în 7 ore. Focul ajunge în $D4$ din est, pe traseul $G2 \to G3 \to G4 \to G5 \to F5 \to E5 \to E4 \to D4$, ocolind stâncile.

c) Cu un singur focar, în $A4$, aplicăm BFS obișnuit din $A4$. Zona cea mai depărtată este tot $D4$, la distanța 15: focul trebuie să ocolească stâncile prin sud, pe la $C2$ și $E2$, apoi să urce pe coloana G și să revină spre vest. Pădurea ar arde în 15 ore.

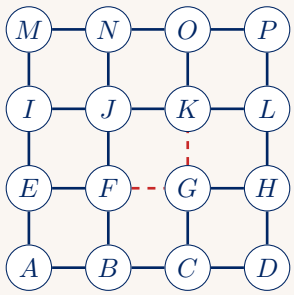
Pentru cele două zone cerute, comparăm orele date de fiecare focar separat:

Zonă	Doar focarul $A4$	Doar focarul $G2$	Ambele focare (punctul b)
$D4$	15	7	$\min(15, 7) = 7$
$C1$	5	5	$\min(5, 5) = 5$

Zona $C1$ este la egală distanță de cele două focare: cele două fronturi ale focului ajung acolo în aceeași oră. În general, distanța de la v la cea mai apropiată sursă este minimul distanțelor de la v la fiecare sursă. BFS cu mai multe surse calculează acest minim pentru toate zonele dintr-o singură parcurgere, în loc de câte o parcurgere pentru fiecare focar.

★ Problema 13. Câte trasee cele mai scurte există?

Enunț. Străzile unui cartier formează un caroiaj cu 16 intersecții, notate ca în figură. Toate tronsoanele de stradă au aceeași lungime. Tronsoanele F–G și G–K sunt închise pentru lucrări (punctate în figură). Un curier pleacă din A spre P.



a) Aplicând BFS din A, determinați distanța de la A la fiecare intersecție și numărul $N(v)$ de trasee de lungime minimă de la A la v.

b) Câte trasee de lungime minimă are curierul de la A la P? Câte ar fi fost fără lucrări?

c) Care intersecție, diferită de A și P, este folosită de cele mai multe trasee minime de la A la P?

Rezolvare. a) Orice traseu minim de la A la v se termină cu un tronson uv, unde u este cu un nivel mai aproape de A: $\ell(u) = \ell(v) - 1$. Porțiunea până la u este un traseu minim până la u. Reciproc, orice traseu minim până la un astfel de u, prelungit cu uv, este un traseu minim până la v. Prin urmare

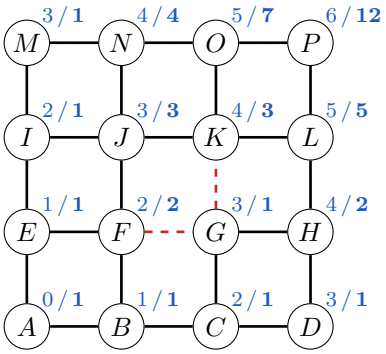
$$N(A) = 1, \qquad N(v) = \sum_{\substack{u \text{ vecin cu } v \\ \ell(u)=\ell(v)-1}} N(u).$$

Calculăm N în timpul BFS: când extragem u și examinăm un vecin v aflat pe nivelul următor (nou descoperit sau descoperit deja de un alt vârf de pe nivelul lui u), adunăm $N(u)$ la $N(v)$. Când v este extras, toți vecinii lui de pe nivelul anterior au fost deja extrași, deci $N(v)$ este complet.

Parcurgem nivel cu nivel:

- Nivelul 0: A, cu $N(A) = 1$.
- Nivelul 1: B și E, fiecare cu un singur vecin pe nivelul 0: $N(B) = N(E) = 1$.
- Nivelul 2: C (din B), F (din B și E), I (din E). Deci $N(C) = 1$, $N(F) = 1 + 1 = 2$, $N(I) = 1$.
- Nivelul 3: D (din C), G (doar din C, deoarece F–G este închis), J (din F și I), M (din I). Deci $N(D) = 1$, $N(G) = 1$, $N(J) = 2 + 1 = 3$, $N(M) = 1$.

- Nivelul 4: H (din D și G), K (doar din J, deoarece G–K este închis), N (din J și M). Deci $N(H) = 2$, $N(K) = 3$, $N(N) = 3 + 1 = 4$.
- Nivelul 5: L (din H și K), O (din K și N). Deci $N(L) = 2 + 3 = 5$, $N(O) = 3 + 4 = 7$.
- Nivelul 6: P (din L și O): $N(P) = 5 + 7 = 12$.



Lângă fiecare intersecție este scris $\ell(v) \,/\, N(v)$. Observăm că, pe fiecare tronson deschis, nivelurile capetelor diferă exact cu 1, deoarece caroiajul este bipartit.

b) Distanța de la A la P este 6, iar numărul traseelor minime este $N(P) = 12$.

Fără lucrări, un traseu minim de la A la P are 3 pași spre dreapta și 3 pași în sus, în orice ordine. Numărul lor este $\binom{6}{3} = 20$. Aceeași valoare se obține cu BFS: pe caroiajul complet, N satisface $N(v) = N(\text{vecinul din stânga}) + N(\text{vecinul de jos})$, adică regula triunghiului lui Pascal. Lucrările elimină deci $20 - 12 = 8$ trasee minime.

c) Un traseu minim de la A la P trece prin v dacă și numai dacă $\ell_A(v) + \ell_P(v) = 6$, unde $\ell_P(v)$ este distanța de la v la P. În acest caz, numărul traseelor minime prin v este

$$N_A(v) \cdot N_P(v),$$

deoarece orice traseu minim de la A la v se poate continua cu orice traseu minim de la v la P. Aplicăm același BFS din P pentru a obține N_P (de exemplu, $N_P(O) = N_P(L) = 1$, $N_P(K) = N_P(O) + N_P(L) = 2$, $N_P(J) = N_P(K) + N_P(N) = 2 + 1 = 3$). Pentru toate intersecțiile avem $\ell_A(v) + \ell_P(v) = 6$, deci toate pot apărea pe un traseu minim:

v	B	C	D	E	F	G	H	I	J	K	L	M	N	O
$N_A(v)$	1	1	1	1	2	1	2	1	3	3	5	1	4	7
$N_P(v)$	5	2	1	7	3	1	1	4	3	2	1	1	1	1
Trasee prin v	5	2	1	7	6	1	2	4	**9**	6	5	1	4	7

Intersecția cea mai folosită este J: 9 dintre cele 12 trasee minime trec prin ea. Verificare: pe fiecare nivel, traseele se împart între intersecțiile nivelului. De exemplu, pe nivelul 3 avem $1 + 1 + 9 + 1 = 12$ (intersecțiile D, G, J, M), iar pe nivelul 5 avem $5 + 7 = 12$ (L și O). Ideea se folosește, de exemplu, pentru a găsi nodurile cele mai solicitate dintr-o rețea.

Problema 14. Explorarea unei peșteri cu DFS

Enunț. O peșteră are 8 săli, numerotate $1, \ldots, 8$, legate prin galeriile

$$12,\ 13,\ 24,\ 25,\ 36,\ 45,\ 57,\ 67,\ 78.$$

Un explorator pornește din sala 1 și aplică parcurgerea în adâncime, alegând mereu galeria spre sala nevizitată cu numărul cel mai mic și întorcându-se când nu mai are săli noi accesibile.

a) Determinați ordinea în care sunt vizitate sălile și momentele t_{in}, t_{out}.

b) Determinați arborele DFS și muchiile de întoarcere.

c) Deduceți că rețeaua de galerii conține cicluri și indicați câte unul pentru fiecare muchie de întoarcere.

Rezolvare. Listele de adiacență sunt:

Sala	1	2	3	4	5	6	7	8
Vecini	2, 3	1, 4, 5	1, 6	2, 5	2, 4, 7	3, 7	5, 6, 8	7

a) Urmărim apelurile recursive. Fiecare chenar descrie un pas: sala curentă (în titlu), predecesorul ei, vecinii ei și ce facem mai departe, precum și sălile încă nevizitate. Vecinii sunt colorați astfel: **verde** pentru sala în care mergem, **portocaliu** pentru o sală nevizitată care rămâne de explorat mai târziu din sala curentă, gri pentru o sală deja vizitată (de obicei predecesorul) și **roșu** pentru un strămoș deja vizitat, care dă o muchie de întoarcere.

$t = 1$: intrăm în sala 1

Predecesor: –
Vecini: **2**, **3**
Acțiune: mergem în **2**
Nevizitate: 3, 4, 5, 6, 7, 8

$t = 2$: intrăm în sala 2

Predecesor: 1
Vecini: 1, **4**, **5**
Acțiune: mergem în **4**
Nevizitate: 3, 5, 6, 7, 8

$t = 3$: intrăm în sala 4

Predecesor: 2
Vecini: 2, **5**
Acțiune: mergem în **5**
Nevizitate: 3, 6, 7, 8

$t = 4$: intrăm în sala 5

Predecesor: 4
Vecini: **2**, 4, **7**
Acțiune: muchia **5–2** este de întoarcere. Mergem în **7**
Nevizitate: 3, 6, 8

$t = 5$: intrăm în sala 7

Predecesor: 5
Vecini: 5, **6**, **8**
Acțiune: mergem în **6**
Nevizitate: 3, 8

$t = 6$: intrăm în sala 6

Predecesor: 7
Vecini: **3**, 7
Acțiune: mergem în **3**
Nevizitate: 3, 8

$t = 7$: intrăm în sala 3

Predecesor: 6
Vecini: **1**, 6
Acțiune: muchia **3–1** este de întoarcere. Nu există vecini noi, deci la $t = 8$ terminăm 3 și revenim în 6
Nevizitate: 8

$t = 9$: terminăm 6, suntem în sala 7

Predecesor: 5
Vecini: 5, 6, **8**
Acțiune: mergem în **8**
Nevizitate: 8

$t = 10$: intrăm în sala 8

Predecesor: 7
Vecini: 7
Acțiune: nu există vecini noi, deci la $t = 11$ terminăm 8 și revenim în 7
Nevizitate: niciuna

$t = 12, \ldots, 16$: întoarcerea

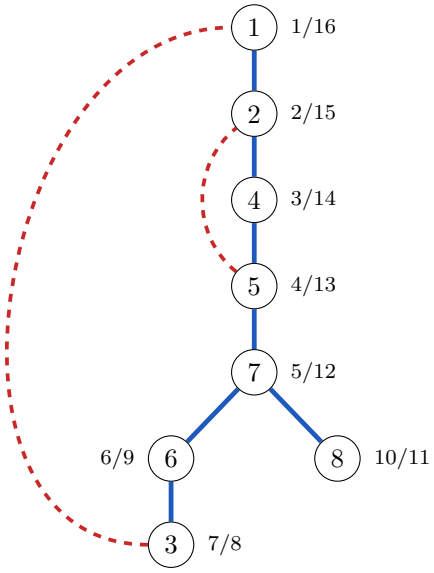
Predecesor: pe rând 5, 4, 2, 1, –
Vecini: toți vizitați
Acțiune: terminăm 7, 5, 4, 2, 1 la momentele 12, 13, 14, 15, 16. Sălile 5 (pentru 2) și 3 (pentru 1), care erau de vizitat, au fost între timp vizitate
Nevizitate: niciuna

În ultimul chenar, sălile 7, 5, 4, 2, 1 se termină pe rând, iar predecesorii lor, 5, 4, 2, 1, sunt sălile în care revenim după fiecare terminare.

Sala	1	2	3	4	5	6	7	8
t_{in}	1	2	7	3	4	6	5	10
t_{out}	16	15	8	14	13	9	12	11
Predecesor	–	1	6	2	4	7	5	7

Ordinea vizitării sălilor este $1, 2, 4, 5, 7, 6, 3, 8$. Se observă proprietatea parantezelor: de exemplu, $[7, 8] \subset [6, 9] \subset [5, 12]$ (3 este descendent al lui 6, care este descendent al lui 7), iar $[6, 9]$ și $[10, 11]$ sunt disjuncte (6 și 8 sunt pe ramuri diferite).

b) Arborele DFS are muchiile 1–2, 2–4, 4–5, 5–7, 7–6, 6–3, 7–8, adică $n-1 = 7$ muchii. Celelalte două galerii, 2–5 și 1–3, sunt muchii de întoarcere: fiecare unește o sală cu un strămoș al ei.



În desen, arborele DFS este albastru, muchiile de întoarcere sunt roșii punctate, iar lângă fiecare sală sunt scrise momentele $t_{\text{in}}/t_{\text{out}}$.

c) Existența muchiilor de întoarcere arată că rețeaua conține cicluri:

- Muchia 5–2, împreună cu drumul din arbore $2 \to 4 \to 5$, dă ciclul $2 \to 4 \to 5 \to 2$, de lungime 3.
- Muchia 3–1, împreună cu drumul din arbore $1 \to 2 \to 4 \to 5 \to 7 \to 6 \to 3$, dă ciclul $1 \to 2 \to 4 \to 5 \to 7 \to 6 \to 3 \to 1$, de lungime 7.

Un explorator care urmează doar galeriile arborelui DFS poate vizita toate sălile. Galeriile 2–5 și 1–3 sunt „redundante” (pot fi închise fără a pierde accesul la vreo sală).

Problema 15. Ordinea cursurilor (sortare topologică)

Enunț. Un program de studii are 8 cursuri. Un arc $X \to Y$ înseamnă că X trebuie promovat înainte de Y:

M_1	Analiză I	PR	Programare
M_2	Analiză II	SD	Structuri de date
AL	Algebră liniară	GR	Teoria grafurilor
EC	Ecuații diferențiale	PO	Optimizare

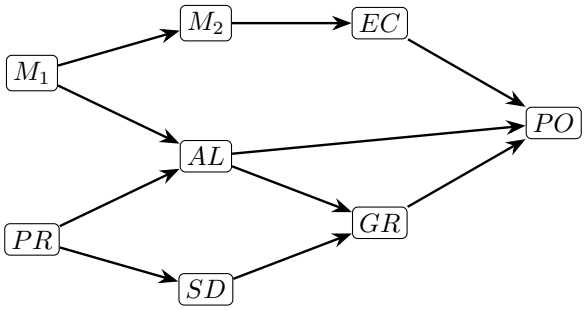
$$M_1 \to M_2,\quad M_1 \to AL,\quad M_2 \to EC,\quad PR \to AL,\quad PR \to SD,$$
$$AL \to GR,\quad AL \to PO,\quad SD \to GR,\quad GR \to PO,\quad EC \to PO.$$

a) Aplicați DFS pe graful orientat, lansând parcurgerea din cursuri în ordine alfabetică $(AL, EC, GR, M_1, M_2, PO, PR, SD)$, și determinați momentele t_{out}.

b) Deduceți o ordine în care pot fi urmate cursurile.

c) Ce se întâmplă dacă se adaugă condiția $PO \to M_1$?

Rezolvare. Graful are ca vârfuri cursurile și ca arce condițiile de precedență. O ordine corectă a cursurilor este o sortare topologică.

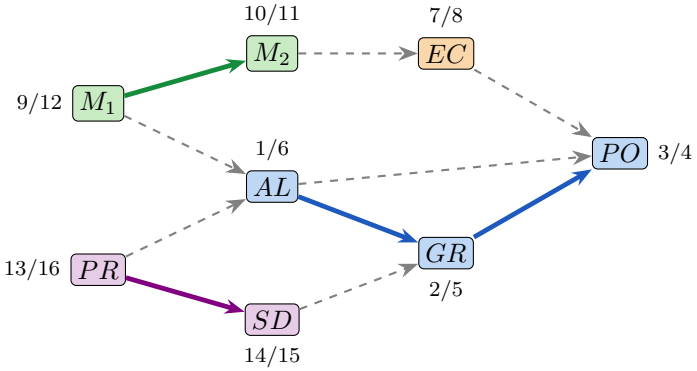


a) Parcurgem arcele în sensul lor, vecinii fiind examinați în ordine alfabetică.

- **Lansarea 1**, din AL, cu $t_{\text{in}}(AL) = 1$. Mergem în GR ($t_{\text{in}} = 2$), apoi în PO ($t_{\text{in}} = 3$). Din PO nu pleacă arce, deci $t_{\text{out}}(PO) = 4$. Revenim și terminăm GR, cu $t_{\text{out}}(GR) = 5$. Arcul $AL \to PO$ duce la un vârf terminat, deci $t_{\text{out}}(AL) = 6$.
- **Lansarea 2**, din EC, cu $t_{\text{in}} = 7$. Vârful PO este terminat, deci $t_{\text{out}}(EC) = 8$.
- Vârful GR este deja vizitat. **Lansarea 3**, din M_1, cu $t_{\text{in}} = 9$. Vârful AL este terminat, deci mergem în M_2 ($t_{\text{in}} = 10$). Vârful EC este terminat, deci $t_{\text{out}}(M_2) = 11$ și $t_{\text{out}}(M_1) = 12$.
- Vârfurile M_2 și PO sunt vizitate. **Lansarea 4**, din PR, cu $t_{\text{in}} = 13$. Vârful AL este terminat, deci mergem în SD ($t_{\text{in}} = 14$). Vârful GR este terminat, deci $t_{\text{out}}(SD) = 15$ și $t_{\text{out}}(PR) = 16$.

Curs	AL	EC	GR	M_1	M_2	PO	PR	SD
t_{in}	1	7	2	9	10	3	13	14
t_{out}	6	8	5	12	11	4	16	15

În desenul următor, vârfurile vizitate în aceeași lansare au aceeași culoare, iar arcele pe care a înaintat parcurgerea (arcele arborilor DFS) sunt îngroșate, în culoarea lansării. Celelalte arce, punctate gri, duc toate spre vârfuri deja terminate. Lângă fiecare vârf sunt scrise momentele $t_{\text{in}}/t_{\text{out}}$.



Avem patru lansări, deci patru arbori DFS: $AL \to GR \to PO$, EC, $M_1 \to M_2$ și $PR \to SD$. Spre deosebire de cazul neorientat, numărul lansărilor nu este numărul componentelor conexe. Graful este conex (ignorând sensul arcelor), dar o lansare atinge doar vârfurile accesibile pe arce din vârful de start și încă nevizitate.

În timpul parcurgerii nu am întâlnit niciun arc către un vârf vizitat, dar neterminat, deci graful nu are circuite.

b) Ordonăm cursurile descrescător după t_{out}:

$$PR\ (16),\ SD\ (15),\ M_1\ (12),\ M_2\ (11),\ EC\ (8),\ AL\ (6),\ GR\ (5),\ PO\ (4).$$

Verificăm toate cele 10 arce: fiecare merge de la un curs aflat mai devreme în listă la unul aflat mai târziu (de exemplu, pentru $PR \to AL$ avem $16 > 6$, pentru $SD \to GR$ avem $15 > 5$, iar pentru $EC \to PO$ avem $8 > 4$). Aceasta este o ordine corectă de parcurgere a cursurilor. Ea nu este unică: de exemplu, M_1 poate fi urmat și înaintea lui PR.

c) Arcul $PO \to M_1$ creează circuitul $M_1 \to AL \to PO \to M_1$: fiecare dintre aceste cursuri ar trebui promovat înaintea celorlalte, ceea ce este imposibil. DFS detectează situația: la lansarea din AL, ajungem în PO pe drumul AL, GR, PO și urmăm arcul nou spre M_1, apoi $M_1 \to AL$ duce la AL, care este vizitat, dar încă neterminat. Acesta este semnalul unui circuit, aici $AL \to GR \to PO \to M_1 \to AL$. Prin urmare nu există nicio sortare topologică.

Problema 16. Rețea fragmentată după o avarie

Enunț. După o avarie, într-o rețea de 10 calculatoare au mai rămas funcționale doar legăturile

1–2, 1–3, 2–3, 4–5, 6–7, 7–8, 6–8, 9–10.

a) Determinați, cu ajutorul parcurgerii în adâncime, componentele conexe ale rețelei.

b) Care este numărul minim de legături noi necesare pentru ca toate calculatoarele să poată comunica? Justificați.

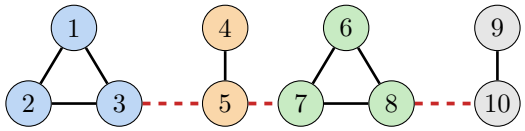
Rezolvare. Listele de adiacență sunt:

Calculator	1	2	3	4	5	6	7	8	9	10
Vecini	2, 3	1, 3	1, 2	5	4	7, 8	6, 8	6, 7	10	9

a) Lansăm DFS din cel mai mic calculator nevizitat, de fiecare dată:

- Din 1 mergem în primul vecin, 2. Din 2, vecinul 1 este vizitat, deci mergem în 3. Vecinii lui 3 sunt deja vizitați, așa că revenim până în 1, care nu mai are vecini noi. Am vizitat $1, 2, 3$, iar prima componentă este $\{1, 2, 3\}$.
- Cel mai mic calculator nevizitat este 4. Din 4 vizităm $4, 5$. A doua componentă este $\{4, 5\}$.
- Din 6 mergem în 7, apoi în 8 (vecinul 6 al lui 7 este vizitat). A treia componentă este $\{6, 7, 8\}$.
- Din 9 vizităm $9, 10$. A patra componentă este $\{9, 10\}$.

Au fost necesare 4 lansări, deci rețeaua are $c = 4$ componente conexe.



b) O legătură nouă unește două calculatoare. Dacă ele sunt în componente diferite, cele două componente devin una. Altfel, numărul componentelor nu se schimbă. Așadar fiecare legătură

nouă micșorează numărul componentelor cu cel mult 1. Pentru a ajunge de la 4 componente la una sunt necesare cel puțin $c - 1 = 3$ legături. Trei legături sunt și suficiente, de exemplu 3–5, 5–7, 8–10 (roșii punctate în desen), care leagă componentele „în lanț”.

Problema 17. Detectarea unui blocaj (deadlock) cu DFS

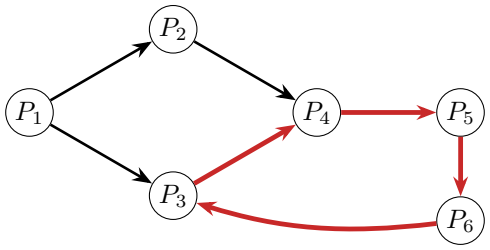
Enunț. Într-un sistem de operare rulează 6 procese $P_1, \ldots, P_6$. Un arc $P_i \to P_j$ înseamnă că P_i așteaptă o resursă ocupată de P_j, deci P_i nu poate continua până când P_j nu se termină. Arcele sunt

$$P_1 \to P_2, \quad P_1 \to P_3, \quad P_2 \to P_4, \quad P_3 \to P_4, \quad P_4 \to P_5, \quad P_5 \to P_6, \quad P_6 \to P_3.$$

a) Aplicați DFS din P_1 și arătați că sistemul este blocat. Precizați procesele implicate în blocaj.

b) Sistemul de operare oprește forțat procesul P_6 și eliberează resursa așteptată de acesta, deci arcul $P_6 \to P_3$ dispare. Arătați că blocajul a fost eliminat și determinați o ordine în care se pot termina procesele.

Rezolvare. Blocajul înseamnă un circuit în graful de așteptare: fiecare proces de pe circuit așteaptă după următorul, deci niciunul nu poate continua. În DFS colorăm vârfurile: *alb* (nevizitat), *gri* (vizitat, în lucru) și *negru* (terminat). Un arc către un vârf gri închide un circuit.



a) Examinăm arcele care pleacă din fiecare proces în ordinea indicilor.

- $t = 1$: intrăm în P_1 (gri). Primul arc duce în P_2.
- $t = 2$: intrăm în P_2, apoi la $t = 3$ în P_4, la $t = 4$ în P_5 și la $t = 5$ în P_6. Toate sunt gri.
- Din P_6 urmăm arcul $P_6 \to P_3$: la $t = 6$ intrăm în P_3 (gri).
- Din P_3 urmăm arcul $P_3 \to P_4$. Vârful P_4 este *gri*, adică vizitat, dar încă neterminat. Am găsit un circuit.

Circuitul este format din vârfurile gri de pe drumul curent de la P_4 la P_3, închis de arcul $P_3 \to P_4$:

$$P_4 \to P_5 \to P_6 \to P_3 \to P_4.$$

Procesele P_3, P_4, P_5, P_6 sunt blocate: fiecare așteaptă după următorul. Procesele P_1 și P_2 nu sunt pe circuit, dar așteaptă (direct sau indirect) după procese blocate, deci nici ele nu pot continua.

b) Fără arcul $P_6 \to P_3$ reluăm DFS din P_1:

- $t = 1$: P_1, $t = 2$: P_2, $t = 3$: P_4, $t = 4$: P_5, $t = 5$: P_6.
- P_6 nu mai așteaptă după nimeni, deci la $t = 6$ se termină. Revenind, la $t = 7$ se termină P_5, la $t = 8$ se termină P_4, iar la $t = 9$ se termină P_2.
- Din P_1 urmăm arcul spre P_3: la $t = 10$ intrăm în P_3. Arcul $P_3 \to P_4$ duce la un vârf negru (terminat), deci nu e circuit. La $t = 11$ se termină P_3, iar la $t = 12$ se termină P_1.

Nu am întâlnit niciun arc către un vârf gri, deci graful nu mai are circuite și blocajul a fost eliminat.

Proces	P_1	P_2	P_3	P_4	P_5	P_6
t_{in}	1	2	10	3	4	5
t_{out}	12	9	11	8	7	6

Un proces se poate termina doar după ce s-au terminat toate procesele după care așteaptă. Pentru orice arc $P_i \to P_j$ avem $t_{\text{out}}(P_i) > t_{\text{out}}(P_j)$, deci ordinea *crescătoare* după t_{out} este o ordine corectă de terminare:

$$P_6,\ P_5,\ P_4,\ P_2,\ P_3,\ P_1.$$

Aceasta este inversa sortării topologice: arcele indică aici „așteaptă după", deci procesul de la capătul arcului se termină primul.

Problema 18. Legături și noduri critice într-o rețea electrică

Enunț. Rețeaua electrică a unui campus are 10 noduri, legate prin liniile

$$1\text{–}2,\quad 1\text{–}3,\quad 2\text{–}3,\quad 3\text{–}4,\quad 4\text{–}5,\quad 4\text{–}6,$$
$$5\text{–}6,\quad 6\text{–}7,\quad 7\text{–}8,\quad 7\text{–}9,\quad 8\text{–}9,\quad 9\text{–}10.$$

a) Aplicați DFS din nodul 1 și calculați valorile t_{in} și *low*.

b) Determinați liniile critice (punțile) și nodurile critice (punctele de articulație).

c) Ce linie nouă ar elimina toate punctele critice ale rețelei?

Rezolvare. Listele de adiacență, cu vecinii în ordine crescătoare, sunt:

Nod	1	2	3	4	5	6	7	8	9	10
Vecini	2, 3	1, 3	1, 2, 4	3, 5, 6	4, 6	4, 5, 7	6, 8, 9	7, 9	7, 8, 10	9

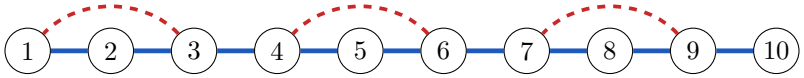
a) Aplicăm DFS din nodul 1, examinând vecinii în ordinea din tabel. Pentru fiecare nod curent, vecinii sunt colorați astfel: gri pentru predecesor (muchia pe care am venit), **roșu** pentru un strămoș deja vizitat, deci o *muchie de întoarcere*, și **verde** pentru vecinul nevizitat în care înaintăm.

t_{in}	Nod curent	Predecesor	Vecini examinați	Acțiune
1	1	–	**2**, 3	înaintăm în 2
2	2	1	1, **3**	înaintăm în 3
3	3	2	**1**, 2, **4**	3–1 **întoarcere**, înaintăm în 4
4	4	3	3, **5**, 6	înaintăm în 5
5	5	4	4, **6**	înaintăm în 6
6	6	5	**4**, 5, **7**	6–4 **întoarcere**, înaintăm în 7
7	7	6	6, **8**, 9	înaintăm în 8
8	8	7	7, **9**	înaintăm în 9
9	9	8	**7**, 8, **10**	9–7 **întoarcere**, înaintăm în 10
10	10	9	9	niciun vecin nou, terminăm 10

Vecinii scriși cu negru (3 pentru nodul 1, 6 pentru nodul 4, 9 pentru nodul 7) nu sunt încă vizitați în momentul primei examinări, dar vor fi vizitați înaintea revenirii în nodul respectiv. După terminarea lui 10 ne întoarcem pe drumul parcurs: 9, 8, 7, …, 1. La revenire, nodul 7

găsește vecinul 9, nodul 4 găsește vecinul 6, iar nodul 1 găsește vecinul 3, toți deja vizitați. Acestea sunt aceleași trei muchii de întoarcere, văzute acum de la strămoș spre descendent, deci nu sunt muchii noi.

Toate nodurile au fost descoperite pe un singur drum, deci $t_{\text{in}}(v) = v$, iar arborele DFS este lanțul 1–2–3–...–10. Celelalte trei muchii ale rețelei, 3–1, 6–4 și 9–7, sunt muchiile de întoarcere.



Arborele DFS este albastru, iar muchiile de întoarcere sunt roșii punctate. Fiecare muchie de întoarcere „sare" înapoi peste o porțiune a lanțului. Aceasta va determina valorile *low*.

Calculăm *low* la terminarea fiecărui nod, de la 10 spre 1:

- $low(10) = 10$ (nu are fii și nici muchii de întoarcere).
- $low(9) = \min(9,\ t_{\text{in}}(7) = 7,\ low(10) = 10) = 7$, iar apoi $low(8) = \min(8, low(9)) = 7$ și $low(7) = \min(7, low(8)) = 7$.
- $low(6) = \min(6,\ t_{\text{in}}(4) = 4,\ low(7) = 7) = 4$, iar apoi $low(5) = 4$ și $low(4) = 4$.
- $low(3) = \min(3,\ t_{\text{in}}(1) = 1,\ low(4) = 4) = 1$, iar apoi $low(2) = 1$ și $low(1) = 1$.

Nod v	1	2	3	4	5	6	7	8	9	10
$t_{\text{in}}(v)$	1	2	3	4	5	6	7	8	9	10
$low(v)$	1	1	1	4	4	4	7	7	7	10

b) *Punți.* Testăm fiecare muchie de arbore $p(v)\,v$, cu condiția $low(v) > t_{\text{in}}(p(v))$:

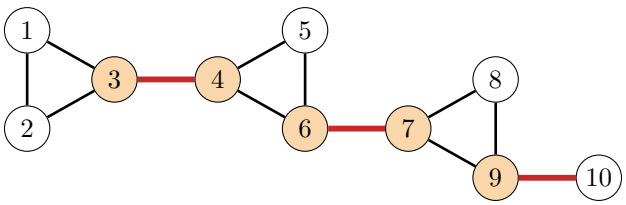
- 3–4: $low(4) = 4 > 3$, deci este punte.
- 6–7: $low(7) = 7 > 6$, deci este punte.
- 9–10: $low(10) = 10 > 9$, deci este punte.
- Pentru celelalte muchii de arbore condiția nu este îndeplinită. De exemplu, pentru 4–5 avem $low(5) = 4$, care nu este mai mare decât $t_{\text{in}}(4) = 4$.

Liniile critice sunt 3–4, 6–7 și 9–10.

Puncte de articulație. Rădăcina 1 are un singur fiu, deci nu este punct de articulație. Un nod $u \neq 1$ este punct de articulație dacă are un fiu v cu $low(v) \geq t_{\text{in}}(u)$:

- nodul 3 (fiul 4: $4 \geq 3$), nodul 4 (fiul 5: $4 \geq 4$), nodul 6 (fiul 7: $7 \geq 6$), nodul 7 (fiul 8: $7 \geq 7$) și nodul 9 (fiul 10: $10 \geq 9$) sunt puncte de articulație.
- Nodurile 2, 5, 8 nu sunt, deoarece pentru fiii lor *low* este mai mic decât propriul t_{in} (de exemplu, $low(3) = 1 < 2$).

Nodurile critice sunt $3, 4, 6, 7, 9$.



Punțile sunt roșii, iar punctele de articulație sunt colorate. Se vede caracterizarea teoretică: punțile sunt exact muchiile care nu se află pe niciun ciclu (rețeaua este formată din trei triunghiuri legate în lanț, plus nodul 10).

c) O linie nouă care, împreună cu rețeaua, formează un ciclu prin toate cele trei punți le transformă în muchii obișnuite. De exemplu, linia 1–10 creează ciclul $1 \to 2 \to 3 \to 4 \to 5 \to 6 \to 7 \to 8 \to 9 \to 10 \to 1$, care conține toate nodurile. După adăugarea ei nu mai există punți. Nu mai există nici puncte de articulație: dacă eliminăm un singur nod de pe acest ciclu, celelalte noduri rămân legate prin restul ciclului.

Problema 19. Galerii și săli critice în peșteră

Enunț. Reluăm peștera din Problema 14, cu galeriile 12, 13, 24, 25, 36, 45, 57, 67, 78.

a) Folosind momentele t_{in} obținute acolo, calculați valorile *low*.

b) Care galerii și care săli sunt critice (a căror prăbușire, respectiv blocare, ar izola o parte a peșterii)?

c) Ce galerie nouă ar elimina toate punctele critice?

Rezolvare. a) Din Problema 14 avem momentele de descoperire

$$t_{\text{in}}: \quad 1 \mapsto 1,\ 2 \mapsto 2,\ 4 \mapsto 3,\ 5 \mapsto 4,\ 7 \mapsto 5,\ 6 \mapsto 6,\ 3 \mapsto 7,\ 8 \mapsto 10,$$

arborele DFS cu muchiile 1–2, 2–4, 4–5, 5–7, 7–6, 6–3, 7–8 și muchiile de întoarcere 5–2 și 3–1. Calculăm *low* în ordinea terminării sălilor (3, 6, 8, 7, 5, 4, 2, 1):

- $low(3) = \min(7,\ t_{\text{in}}(1) = 1) = 1$ (muchia de întoarcere 3–1).
- $low(6) = \min(6,\ low(3)) = 1$.
- $low(8) = 10$ (sala 8 nu are fii și nici muchii de întoarcere).
- $low(7) = \min(5,\ low(6),\ low(8)) = 1$.
- $low(5) = \min(4,\ t_{\text{in}}(2) = 2,\ low(7)) = 1$ (muchia de întoarcere 5–2).
- $low(4) = low(2) = low(1) = 1$.

Sala v	1	2	3	4	5	6	7	8
$t_{\text{in}}(v)$	1	2	7	3	4	6	5	10
$low(v)$	1	1	1	1	1	1	1	10

b) Pentru muchiile de arbore $p(v)\,v$ cu $v \neq 8$ avem $low(v) = 1$, care nu este mai mare decât $t_{\text{in}}(p(v))$, deci nu sunt punți. Pentru galeria 7–8 avem $low(8) = 10 > t_{\text{in}}(7) = 5$, deci aceasta este singura punte. Prăbușirea ei ar izola sala 8.

Rădăcina 1 are un singur fiu (sala 2), deci nu este punct de articulație. Sala 7 are fiul 8 cu $low(8) = 10 \geq t_{\text{in}}(7) = 5$, deci este punct de articulație. Pentru celelalte săli, toți fiii au $low = 1$, mai mic decât propriul t_{in}. Așadar singura sală critică este 7: blocarea ei izolează sala 8.

Rezultatul este în acord cu Problema 14: galeriile 2–5 și 1–3 (muchiile de întoarcere) se află pe cicluri, iar toate celelalte galerii, cu excepția lui 7–8, se află pe ciclul $1 \to 2 \to 4 \to 5 \to 7 \to 6 \to 3 \to 1$.

c) Este suficientă o galerie nouă care să pună sala 8 pe un ciclu, de exemplu 6–8. Atunci 7–8 se află pe ciclul $7 \to 8 \to 6 \to 7$ și nu mai este punte. Nici sala 7 nu mai este critică: după blocarea ei, sala 8 rămâne legată de restul peșterii prin sala 6.

★ Problema 20. Orientarea străzilor unui cartier (teorema lui Robbins)

Enunț. Un cartier are 8 intersecții, legate prin străzile cu dublu sens

$$1\text{–}2,\quad 1\text{–}3,\quad 2\text{–}3,\quad 2\text{–}4,\quad 3\text{–}4,\quad 4\text{–}5,$$
$$5\text{–}6,\quad 5\text{–}7,\quad 6\text{–}7,\quad 6\text{–}8,\quad 7\text{–}8.$$

Primăria vrea să transforme *toate* străzile în străzi cu sens unic, astfel încât din orice intersecție să se poată ajunge în oricare alta.

a) Arătați că, pentru rețeaua actuală, acest lucru este imposibil.

b) Se construiește strada nouă 4–6. Arătați, folosind DFS din intersecția 1, că acum există o orientare potrivită, și determinați-o.

c) Verificați că din fiecare intersecție se poate ajunge în intersecția 1. Este 4 în continuare o intersecție critică (punct de articulație)? Contează acest lucru?

Rezolvare. a) Strada 4–5 este o punte: ea este singura legătură dintre intersecțiile $\{1,2,3,4\}$ și $\{5,6,7,8\}$, deoarece nu se află pe niciun ciclu. Orice sens i-am da, de exemplu $4 \to 5$, din grupul $\{5,6,7,8\}$ nu mai există nicio stradă care să ducă înapoi în $\{1,2,3,4\}$. Pentru sensul $5 \to 4$ situația este simetrică. Deci orientarea cerută este imposibilă. Același argument arată că *orice* punte face imposibilă o astfel de orientare.

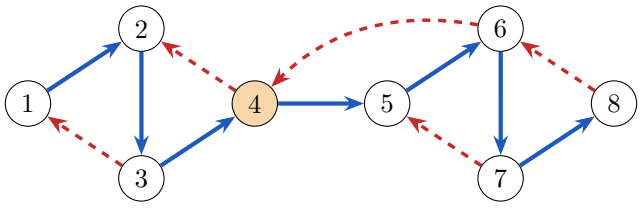
b) Listele de adiacență, cu strada nouă, sunt: 1: 2, 3. 2: 1, 3, 4. 3: 1, 2, 4. 4: 2, 3, 5, 6. 5: 4, 6, 7. 6: 4, 5, 7, 8. 7: 5, 6, 8. 8: 6, 7.

Aplicăm DFS din 1, cu vecinii în ordine crescătoare, și orientăm fiecare stradă în momentul în care o examinăm prima dată:

- *Muchiile de arbore* se orientează de la părinte la fiu, în sensul în care înaintează parcurgerea.
- *Muchiile de întoarcere* se orientează de la descendent la strămoș, în sensul în care sunt descoperite.

t_{in}	Intersecția curentă	Vecini examinați	Arce obținute
1	1	**2**, 3	$1 \to 2$ (arbore)
2	2	1, **3**, 4	$2 \to 3$ (arbore)
3	3	**1**, 2, **4**	$3 \to 1$ (întoarcere), $3 \to 4$ (arbore)
4	4	**2**, 3, **5**, 6	$4 \to 2$ (întoarcere), $4 \to 5$ (arbore)
5	5	4, **6**, 7	$5 \to 6$ (arbore)
6	6	**4**, 5, **7**, 8	$6 \to 4$ (întoarcere), $6 \to 7$ (arbore)
7	7	**5**, 6, **8**	$7 \to 5$ (întoarcere), $7 \to 8$ (arbore)
8	8	**6**, 7	$8 \to 6$ (întoarcere)

Culorile sunt cele din Problema 14: verde pentru vecinul în care înaintăm, gri pentru părinte, roșu pentru un strămoș. La revenire, ceilalți vecini (3 pentru 1, 4 pentru 2, 6 pentru 4, 7 pentru 5, 8 pentru 6) sunt deja vizitați, iar străzile respective au fost deja orientate. Toate cele 12 străzi au primit un sens.



Arcele de arbore sunt albastre, iar arcele de întoarcere sunt roșii punctate. Strada nouă 4–6 a devenit arcul de întoarcere $6 \to 4$. Intersecția 4 este colorată deoarece rămâne punct de articulație (vezi punctul c).

c) Din 1 se ajunge în orice intersecție pe arcele albastre: $1 \to 2 \to 3 \to 4 \to 5 \to 6 \to 7 \to 8$. Drumurile înapoi spre 1 sunt:

Plecare	Drum până la 1
2	$2 \to 3 \to 1$
3	$3 \to 1$
4	$4 \to 2 \to 3 \to 1$
5	$5 \to 6 \to 4 \to 2 \to 3 \to 1$
6	$6 \to 4 \to 2 \to 3 \to 1$
7	$7 \to 5 \to 6 \to 4 \to 2 \to 3 \to 1$
8	$8 \to 6 \to 4 \to 2 \to 3 \to 1$

Prin urmare, din orice intersecție x se poate ajunge în orice intersecție y, mergând întâi din x în 1 și apoi din 1 în y.

De ce funcționează în general. Dacă graful nu are punți, nicio muchie de arbore $p(v)\,v$ nu este punte, deci $low(v) \le t_{\text{in}}(p(v)) < t_{\text{in}}(v)$. Aceasta înseamnă că din subarborele lui v pleacă un arc de întoarcere spre un vârf w cu $t_{\text{in}}(w) < t_{\text{in}}(v)$. Din v coborâm pe arcele de arbore până la capătul acestui arc și urcăm în w. Repetând, momentele t_{in} scad mereu, deci ajungem în rădăcina 1. Împreună cu punctul a), obținem *teorema lui Robbins*: un graf conex se poate orienta astfel încât din orice vârf să se ajungă în oricare altul dacă și numai dacă nu are punți.

Valorile *low* obținute sunt $1, 1, 1, 2, 4, 4, 5, 6$ pentru intersecțiile $1, \dots, 8$. Intersecția 4 are fiul 5 cu $low(5) = 4 \ge t_{\text{in}}(4) = 4$, deci este în continuare punct de articulație: dacă intersecția 4 ar fi blocată, grupul $\{1, 2, 3\}$ ar rămâne izolat de restul cartierului. Pentru orientare acest lucru nu contează. Teorema cere doar ca rețeaua să nu aibă punți, nu și să nu aibă puncte de articulație.

Problema 21. Drumuri minime într-o rețea rutieră (algoritmul lui Dijkstra)

Enunț. Șapte localități $A, \dots, G$ sunt legate prin drumuri, cu lungimile (km):

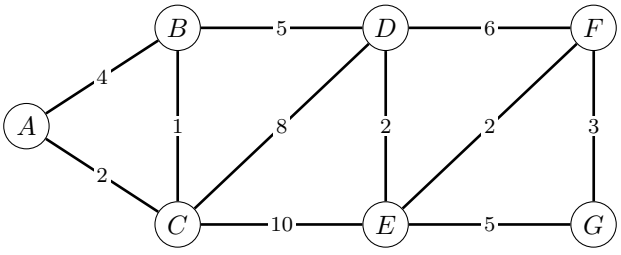
$$AB = 4, \quad AC = 2, \quad BC = 1, \quad BD = 5, \quad CD = 8, \quad CE = 10,$$
$$DE = 2, \quad DF = 6, \quad EF = 2, \quad EG = 5, \quad FG = 3.$$

a) Aplicați algoritmul lui Dijkstra din localitatea A, precizând etichetele după fiecare pas.

b) Determinați distanța de la A la fiecare localitate și un drum minim de la A la G.

c) Desenați arborele drumurilor minime.

Rezolvare. Toate lungimile sunt pozitive, deci algoritmul lui Dijkstra se poate aplica.



a) Inițial $d(A) = 0$ și toate celelalte etichete sunt ∞. La fiecare pas alegem vârful din afara lui

S cu eticheta minimă și relaxăm muchiile lui. În tabel, valorile îngroșate sunt definitive (vârful este în S).

Pas	Ales	A	B	C	D	E	F	G
0	–	0	∞	∞	∞	∞	∞	∞
1	A	**0**	4	2	∞	∞	∞	∞
2	C	**0**	3	**2**	10	12	∞	∞
3	B	**0**	**3**	**2**	8	12	∞	∞
4	D	**0**	**3**	**2**	**8**	10	14	∞
5	E	**0**	**3**	**2**	**8**	**10**	12	15
6	F	**0**	**3**	**2**	**8**	**10**	**12**	15
7	G	**0**	**3**	**2**	**8**	**10**	**12**	**15**

Detaliem relaxările:

- *Pasul 1* (A ales, $d = 0$): $d(B) = 0 + 4 = 4$ și $d(C) = 0 + 2 = 2$, cu $p(B) = p(C) = A$.
- *Pasul 2* (C ales, $d = 2$): pentru B, $2 + 1 = 3 < 4$, deci $d(B) = 3$ și $p(B) = C$. Pentru D obținem $2 + 8 = 10$, iar pentru E obținem $2 + 10 = 12$, cu $p(D) = p(E) = C$.
- *Pasul 3* (B ales, $d = 3$): pentru D, $3 + 5 = 8 < 10$, deci $d(D) = 8$ și $p(D) = B$.
- *Pasul 4* (D ales, $d = 8$): pentru E, $8 + 2 = 10 < 12$, deci $d(E) = 10$ și $p(E) = D$. Pentru F obținem $8 + 6 = 14$, cu $p(F) = D$.
- *Pasul 5* (E ales, $d = 10$): pentru F, $10 + 2 = 12 < 14$, deci $d(F) = 12$ și $p(F) = E$. Pentru G obținem $10 + 5 = 15$, cu $p(G) = E$.
- *Pasul 6* (F ales, $d = 12$): pentru G, $12 + 3 = 15$ nu este strict mai mic decât 15, deci eticheta nu se schimbă.
- *Pasul 7*: se alege G. Toate etichetele sunt definitive.

Se observă că eticheta lui B a scăzut de la 4 la 3: drumul ocolit $A \to C \to B$ este mai scurt decât muchia directă AB.

b) Distanțele de la A sunt

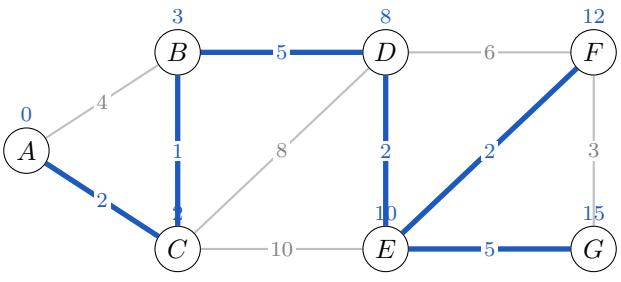
$$\text{dist}(A, \cdot): \quad B = 3,\ C = 2,\ D = 8,\ E = 10,\ F = 12,\ G = 15.$$

Pentru G mergem înapoi pe predecesori: $G \leftarrow E \leftarrow D \leftarrow B \leftarrow C \leftarrow A$, deci un drum minim este

$$A \to C \to B \to D \to E \to G, \qquad 2 + 1 + 5 + 2 + 5 = 15 \text{ km}.$$

La pasul 6 am găsit un drum de aceeași lungime, $A \to C \to B \to D \to E \to F \to G$ ($2 + 1 + 5 + 2 + 2 + 3 = 15$), deci drumul minim nu este unic.

c) Arborele drumurilor minime are muchiile $p(v)\,v$: AC, CB, BD, DE, EF, EG (albastre îngroșate):



Deasupra fiecărei localități este scrisă distanța ei față de A.

Comparație cu arborele parțial de cost minim. Aplicând algoritmul lui Kruskal acestei rețele, se aleg muchiile $BC\,(1)$, $AC\,(2)$, $DE\,(2)$, $EF\,(2)$, $FG\,(3)$, se respinge $AB\,(4)$, care ar închide ciclul $A\,B\,C$, și se alege $BD\,(5)$. APM are costul $1+2+2+2+3+5=15$, mai mic decât costul 17 al arborelui drumurilor minime. Diferența este că APM leagă G prin $FG\,(3)$, cea mai ieftină legătură, pe când arborele lui Dijkstra folosește $EG\,(5)$, care dă drumul cel mai scurt de la A la G (vezi și Problema 8 c)).

Problema 22. Străzi cu sens unic (Dijkstra pe un graf orientat)

Enunț. Într-un oraș, depozitul unei firme de livrări se află în intersecția 1. Străzile cu sens unic dintre intersecțiile $1,\ldots,6$ și timpii de parcurgere (minute) sunt:

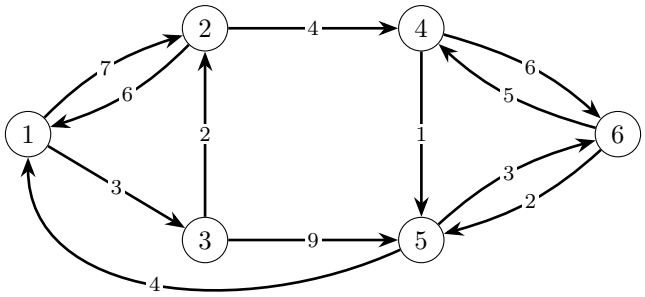
$$1\to 2:\ 7,\quad 1\to 3:\ 3,\quad 2\to 1:\ 6,\quad 2\to 4:\ 4,\quad 3\to 2:\ 2,\quad 3\to 5:\ 9,$$
$$4\to 5:\ 1,\quad 4\to 6:\ 6,\quad 5\to 1:\ 4,\quad 5\to 6:\ 3,\quad 6\to 4:\ 5,\quad 6\to 5:\ 2.$$

a) Aplicați algoritmul lui Dijkstra din intersecția 1 și determinați timpii minimi către toate intersecțiile.

b) Determinați traseul cel mai rapid de la depozit la intersecția 6.

c) Determinați traseul cel mai rapid de întoarcere, de la 6 la depozit. Comparați.

Rezolvare. Graful este orientat: o stradă se parcurge doar în sensul săgeții. La relaxare folosim doar arcele care *pleacă* din vârful ales.



a) Valorile îngroșate sunt definitive.

Pas	Ales	1	2	3	4	5	6
0	–	0	∞	∞	∞	∞	∞
1	1	**0**	7	3	∞	∞	∞
2	3	**0**	5	**3**	∞	12	∞
3	2	**0**	**5**	**3**	9	12	∞
4	4	**0**	**5**	**3**	**9**	10	15
5	5	**0**	**5**	**3**	**9**	**10**	13
6	6	**0**	**5**	**3**	**9**	**10**	**13**

Relaxările sunt următoarele.

- Din 1 (timp 0): $d(2)=7$ și $d(3)=3$.
- Din 3 (timp 3): pentru 2, $3+2=5<7$, deci $d(2)=5$ și $p(2)=3$. Pentru 5 obținem $3+9=12$.
- Din 2 (timp 5): pentru 4 obținem $5+4=9$. Arcul $2\to 1$ duce într-un vârf definitiv.

- Din 4 (timp 9): pentru 5, $9+1=10<12$, deci $d(5)=10$ și $p(5)=4$. Pentru 6 obținem $9+6=15$.
- Din 5 (timp 10): pentru 6, $10+3=13<15$, deci $d(6)=13$ și $p(6)=5$. Arcul $5\to 1$ duce într-un vârf definitiv.
- Se alege 6. Toate etichetele sunt definitive.

Timpii minimi de la depozit sunt $d(2)=5$, $d(3)=3$, $d(4)=9$, $d(5)=10$ și $d(6)=13$ minute.

b) Mergând înapoi pe predecesori, $6\leftarrow 5\leftarrow 4\leftarrow 2\leftarrow 3\leftarrow 1$, obținem traseul

$$1\to 3\to 2\to 4\to 5\to 6, \qquad 3+2+4+1+3=13 \text{ minute}.$$

Traseul ocolește arcul direct $1\to 2$ (7 minute) prin $1\to 3\to 2$ (5 minute) și arcul $4\to 6$ (6 minute) prin $4\to 5\to 6$ (4 minute).

c) Aplicăm algoritmul din 6. Din 6 (timp 0): $d(4)=5$ și $d(5)=2$. Se alege 5 (timp 2): pentru 1 obținem $2+4=6$. Se alege 4 (timp 5): ambele arce, $4\to 5$ și $4\to 6$, duc în vârfuri definitive. Se alege 1 (timp 6): $d(2)=13$ și $d(3)=9$. Se alege 3 (timp 9): pentru 2, $9+2=11<13$, deci $d(2)=11$. În final, timpul minim de întoarcere este

$$\text{dist}(6,1)=6, \qquad \text{pe traseul } 6\to 5\to 1.$$

Într-un graf orientat distanța nu este simetrică: drumul spre intersecția 6 durează 13 minute, iar întoarcerea doar 6 minute, pe alte străzi.

★ Problema 23. Cel mai scurt drum: în ce sens? Ponderi negative

Enunț. Considerăm rețeaua rutieră din Problema 21.

a) Un șofer vrea să ajungă de la A la G trecând prin cât mai puține tronsoane (de exemplu, pentru a plăti cât mai puține taxe de drum). Ce traseu alege? Comparați cu drumul cel mai scurt în kilometri.

b) Pe o altă rută, de la S la A se poate merge direct (cost 2) sau prin B: $S\to B$ costă 3, iar pe porțiunea $B\to A$ șoferul primește un bonus de 2 (cost -2). Ce dă algoritmul lui Dijkstra din S? Este corect?

Rezolvare. a) Acum fiecare tronson „costă” 1, indiferent de lungime, deci aplicăm BFS din A. Nivelurile sunt:

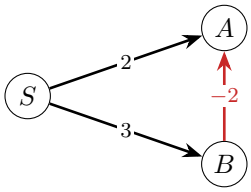
Localitate	A	B	C	D	E	F	G
Nivel ℓ	0	1	1	2	2	3	3
Predecesor	–	A	A	B	C	D	E

Predecesorul lui G este E, al lui E este C, deci traseul cu cele mai puține tronsoane este

$$A\to C\to E\to G: \quad 3 \text{ tronsoane}, \quad 2+10+5=17 \text{ km}.$$

Drumul minim în kilometri, $A\to C\to B\to D\to E\to G$, are 15 km, dar 5 tronsoane. Cele două criterii dau trasee diferite: BFS minimizează numărul de muchii, iar Dijkstra minimizează suma ponderilor. Dacă toate ponderile ar fi egale, cele două criterii ar coincide.

b) Graful orientat are arcele $S\to A$ (2), $S\to B$ (3), $B\to A$ (-2).



Algoritmul lui Dijkstra din S:

- Pasul 1: se alege S ($d = 0$). Se obțin $d(A) = 2$ și $d(B) = 3$.
- Pasul 2: se alege A ($d = 2$, minim), iar eticheta lui devine definitivă: 2.
- Pasul 3: se alege B ($d = 3$). Relaxarea arcului $B \to A$ ar da $3 - 2 = 1 < 2$, dar A este deja în S și eticheta lui nu mai este modificată.

Rezultatul este $d(A) = 2$, dar drumul $S \to B \to A$ are costul $3 + (-2) = 1 < 2$. Algoritmul a greșit. Argumentul de corectitudine folosea faptul că porțiunea de drum de la primul vârf din afara lui S (aici B) până la vârful ales (A) are lungime nenegativă. Aici, această porțiune, arcul $B \to A$, are lungimea -2. Pentru grafuri cu ponderi negative se folosesc alți algoritmi, cum este algoritmul Bellman–Ford (vezi Problema 24 d)).

Problema 24. Transport cu subvenții (algoritmul Bellman–Ford)

Enunț. O firmă transportă marfă între centrele s, a, b, c, d. Costul unui transport direct este dat mai jos. Pe unele rute firma primește o subvenție mai mare decât cheltuielile, deci costul este negativ:

$$s \to a:\ 6, \quad s \to b:\ 7, \quad a \to b:\ 8, \quad a \to c:\ 5, \quad a \to d:\ -4,$$
$$b \to c:\ -3, \quad b \to d:\ 9, \quad c \to a:\ -2, \quad d \to c:\ 7, \quad d \to s:\ 2.$$

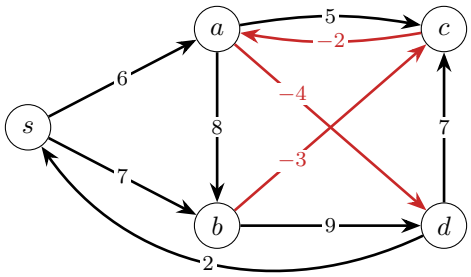
a) Aplicați algoritmul Bellman–Ford din s și determinați costurile minime către toate centrele.

b) Determinați traseul de cost minim de la s la d.

c) Arătați că algoritmul lui Dijkstra ar da un rezultat greșit.

d) Rezolvați cu Bellman–Ford exemplul cu ponderi negative din Problema 23 b).

Rezolvare. Graful are $n = 5$ vârfuri, deci facem $n - 1 = 4$ runde, plus o rundă de verificare.



a) În runda k calculăm etichetele d_k din etichetele rundei precedente, relaxând toate arcele.

Runda k	s	a	b	c	d
0	0	∞	∞	∞	∞
1	0	6	7	∞	∞
2	0	6	7	4	2
3	0	2	7	4	2
4	0	2	7	4	-2

- Runda 1: doar s are etichetă finită, deci $d(a) = 6$ și $d(b) = 7$.

- Runda 2: pentru c, varianta cea mai bună este $d(b)+(-3)=4$ (prin a s-ar obține 11), cu $p(c)=b$. Pentru d, varianta cea mai bună este $d(a)+(-4)=2$ (prin b s-ar obține 16), cu $p(d)=a$.
- Runda 3: pentru a, $d(c)+(-2)=2<6$, deci $d(a)=2$ și $p(a)=c$. Celelalte etichete nu se schimbă (de exemplu, $d(d)+2=4$ nu îmbunătățește eticheta lui s).
- Runda 4: pentru d, $d(a)+(-4)=-2<2$, deci $d(d)=-2$.
- Runda de verificare: nicio etichetă nu mai scade (de exemplu, $d(d)+2=0$ nu este mai mic decât $d(s)=0$, iar $d(d)+7=5$ nu este mai mic decât $d(c)=4$). Deci nu există circuite negative, iar etichetele din runda 4 sunt costurile minime.

Costurile minime de la s sunt $a:2$, $b:7$, $c:4$, $d:-2$.

b) Predecesorii finali sunt $p(a)=c$, $p(b)=s$, $p(c)=b$, $p(d)=a$. Pentru d obținem $d\leftarrow a\leftarrow c\leftarrow b\leftarrow s$, deci traseul

$$s\to b\to c\to a\to d,\qquad 7+(-3)+(-2)+(-4)=-2.$$

Traseul are 4 arce, deci a fost găsit abia în runda 4, ceea ce arată de ce sunt necesare $n-1$ runde.

c) Dijkstra din s ar alege s ($d(a)=6$, $d(b)=7$), apoi a, care ar deveni definitiv cu eticheta 6. Costul real până la a este însă 2, pe traseul $s\to b\to c\to a$, care trece prin arcele negative $b\to c$ și $c\to a$. Eroarea se propagă și la d: Dijkstra ar găsi 2 în loc de -2.

d) Arcele sunt $S\to A$ (2), $S\to B$ (3), $B\to A$ (-2), cu $n=3$. Runda 1 dă $d(A)=2$ și $d(B)=3$. Runda 2 dă $d(A)=\min(2,\ 3-2)=1$, cu $p(A)=B$. Runda de verificare nu mai schimbă nimic. Obținem costul corect 1, pe traseul $S\to B\to A$, acolo unde Dijkstra dădea 2.

★ Problema 25. Arbitraj valutar (circuite negative)

Enunț. O casă de schimb afișează următoarele cursuri: câte unități din moneda de pe coloană se primesc pentru o unitate din moneda de pe linie.

	EUR	USD	RON
EUR	–	1,08	4,97
USD	0,92	–	4,60
RON	0,2005	0,215	–

Un *arbitraj* este un șir de schimburi care pleacă dintr-o monedă și se întoarce în ea cu mai mulți bani decât la început.

a) Modelați problema astfel încât un arbitraj să corespundă unui circuit negativ.

b) Folosind algoritmul Bellman–Ford din EUR, arătați că nu există arbitraj.

c) Cursul USD→RON devine 4,65. Arătați că apare un arbitraj, determinați-l și calculați câștigul pentru 1000 EUR.

Rezolvare. a) Monedele sunt vârfurile unui graf orientat, iar schimbul $X\to Y$ la cursul $r(X,Y)$ este un arc. Un circuit $X_1\to X_2\to\cdots\to X_k\to X_1$ înmulțește suma inițială cu produsul cursurilor de pe circuit, deci este arbitraj dacă acest produs este mai mare decât 1. Dăm fiecărui arc ponderea

$$w(X,Y)=-\ln r(X,Y).$$

Logaritmul transformă produsul în sumă, deci

$$\begin{aligned} r(X_1,X_2)\cdots r(X_k,X_1)>1 &\iff \ln r(X_1,X_2)+\cdots+\ln r(X_k,X_1)>0\\ &\iff w(X_1,X_2)+\cdots+w(X_k,X_1)<0.\end{aligned}$$

Așadar arbitrajele sunt exact circuitele negative. Ponderile, rotunjite la 4 zecimale, sunt:

Arc	w	Arc	w	Arc	w
EUR→USD	$-0{,}0770$	USD→RON	$-1{,}5261$	EUR→RON	$-1{,}6034$
USD→EUR	$0{,}0834$	RON→USD	$1{,}5371$	RON→EUR	$1{,}6069$

b) Avem $n = 3$ vârfuri, deci 2 runde și o rundă de verificare. Pornim din EUR.

- Runda 1: $d(\text{USD}) = -0{,}0770$ și $d(\text{RON}) = -1{,}6034$.
- Runda 2: pentru RON, prin USD obținem $-0{,}0770 - 1{,}5261 = -1{,}6031$, care nu este mai mic decât $-1{,}6034$. Pentru USD, prin RON obținem $-1{,}6034 + 1{,}5371 = -0{,}0663$, nici aceasta mai mică. Pentru EUR, prin USD obținem 0,0064, iar prin RON 0,0035, ambele pozitive. Nicio etichetă nu se schimbă.

Etichetele s-au stabilizat, deci nu există circuit negativ accesibil din EUR. Din EUR se poate ajunge în orice monedă, deci nu există arbitraj. Verificare directă: cele două circuite prin toate monedele au produsele $1{,}08 \cdot 4{,}60 \cdot 0{,}2005 \approx 0{,}9961$ și $4{,}97 \cdot 0{,}215 \cdot 0{,}92 \approx 0{,}9831$, ambele mai mici decât 1.

c) Noul curs dă $w(\text{USD}, \text{RON}) = -\ln 4{,}65 \approx -1{,}5369$.

Runda	EUR	USD	RON
0	0	∞	∞
1	0	$-0{,}0770$	$-1{,}6034$
2	0	$-0{,}0770$	$-1{,}6139$
verificare	$-0{,}0070$	$-0{,}0770$	$-1{,}6139$

În runda 2, eticheta lui RON scade la $-0{,}0770 - 1{,}5369 = -1{,}6139$, cu predecesorul USD. În runda de verificare, eticheta lui EUR scade la $-1{,}6139 + 1{,}6069 = -0{,}0070 < 0$, cu predecesorul RON. O etichetă care scade după $n - 1$ runde semnalează un circuit negativ.

Mergând înapoi pe predecesori, EUR ← RON ← USD ← EUR, obținem circuitul

$$\text{EUR} \to \text{USD} \to \text{RON} \to \text{EUR}, \qquad 1{,}08 \cdot 4{,}65 \cdot 0{,}2005 \approx 1{,}0069.$$

Schimbând 1000 EUR în 1080 USD, apoi în 5022 RON, apoi înapoi în EUR, se obțin aproximativ 1006,9 EUR, deci un câștig de circa 6,9 EUR (0,69%), care se poate repeta. În practică, casele de schimb își ajustează cursurile tocmai pentru a evita astfel de circuite.

Problema 26. Organizarea unei conferințe (drumul critic)

Enunț. Organizarea unei conferințe cuprinde activitățile de mai jos, cu duratele (zile) și condițiile de precedență indicate.

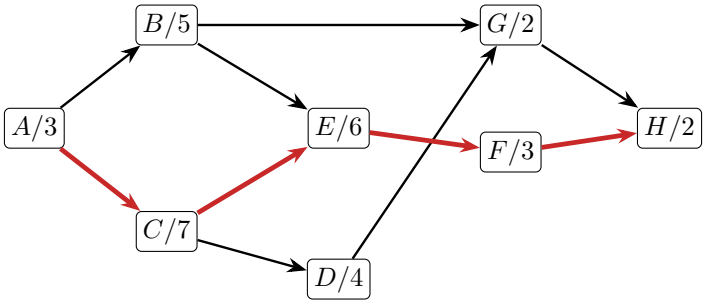
Cod	Activitatea	Precedată de	Durata
A	stabilirea temei și a datei	–	3
B	rezervarea sălii	A	5
C	invitarea vorbitorilor	A	7
D	tipărirea materialelor	C	4
E	înscrierea participanților	B, C	6
F	organizarea mesei	E	3
G	pregătirea sălii	B, D	2
H	desfășurarea conferinței	F, G	2

a) Construiți graful activităților.

b) Determinați durata minimă a pregătirii și a desfășurării conferinței.

c) Determinați activitățile critice și rezervele celorlalte activități.

Rezolvare. a) Fiecare activitate este un vârf, iar fiecare condiție „X precede Y" este un arc $X \to Y$. Lângă fiecare activitate este scrisă durata ei. Ordinea A, B, C, D, E, F, G, H este o sortare topologică (fiecare activitate apare după predecesorii ei), deci graful este aciclic.



b) *Calculul înainte*, în ordinea A, B, C, D, E, F, G, H:

- A: $ES = 0$, $EF = 3$.
- B și C încep după A: $ES = 3$, deci $EF(B) = 8$ și $EF(C) = 10$.
- D începe după C: $ES = 10$, $EF = 14$.
- E începe după B și C: $ES = \max(8, 10) = 10$, $EF = 16$.
- F începe după E: $ES = 16$, $EF = 19$.
- G începe după B și D: $ES = \max(8, 14) = 14$, $EF = 16$.
- H începe după F și G: $ES = \max(19, 16) = 19$, $EF = 21$.

Durata minimă este $T = 21$ de zile.

c) *Calculul înapoi*, în ordinea H, G, F, E, D, C, B, A:

- H: $LF = 21$, $LS = 19$. Apoi G și F, al căror succesor este H: $LF = 19$, deci $LS(G) = 17$ și $LS(F) = 16$.
- E (succesor F): $LF = 16$, $LS = 10$. D (succesor G): $LF = 17$, $LS = 13$.
- C (succesori D, E): $LF = \min(13, 10) = 10$, $LS = 3$. B (succesori E, G): $LF = \min(10, 17) = 10$, $LS = 5$.
- A (succesori B, C): $LF = \min(5, 3) = 3$, $LS = 0$.

Activitate	A	B	C	D	E	F	G	H
Durata	3	5	7	4	6	3	2	2
ES / EF	0 / 3	3 / 8	3 / 10	10 / 14	10 / 16	16 / 19	14 / 16	19 / 21
LS / LF	0 / 3	5 / 10	3 / 10	13 / 17	10 / 16	16 / 19	17 / 19	19 / 21
Rezerva R	**0**	2	**0**	3	**0**	**0**	3	**0**

Activitățile critice sunt A, C, E, F, H. Ele formează drumul critic (roșu în desen)

$$A \to C \to E \to F \to H, \qquad 3 + 7 + 6 + 3 + 2 = 21 \text{ zile},$$

cel mai lung drum din graf. Rezervele sunt 2 zile pentru B și 3 zile pentru D și G. De exemplu, rezervarea sălii poate începe cel mai târziu în ziua 5 fără să întârzie conferința. Invitarea vorbitorilor nu are nicio rezervă: orice întârziere a ei amână conferința.

Problema 27. Lansarea unei aplicații: întârzieri și accelerări

Enunț. Lansarea unei aplicații mobile are următoarele activități (durate în zile):

Cod	Activitatea	Precedată de	Durata
A	analiza cerințelor	–	4
B	proiectarea interfeței	A	5
C	proiectarea bazei de date	A	3
D	programarea interfeței	B	7
E	programarea serverului	C	6
F	testarea	D, E	4
G	redactarea documentației	B	3
H	lansarea	F, G	1

a) Determinați durata minimă a proiectului și drumul critic.

b) Documentația întârzie 5 zile. Se amână lansarea?

c) Programarea serverului întârzie 4 zile. Ce se întâmplă?

d) Echipa poate scurta cu 2 zile fie programarea interfeței, fie programarea serverului. Ce variantă este utilă? Dar dacă programarea interfeței s-ar scurta cu 4 zile?

Rezolvare. a) Aplicăm metoda drumului critic, cu ordinea topologică A, B, C, D, E, F, G, H.

Activitate	A	B	C	D	E	F	G	H
Durata	4	5	3	7	6	4	3	1
ES / EF	0 / 4	4 / 9	4 / 7	9 / 16	7 / 13	16 / 20	9 / 12	20 / 21
LS / LF	0 / 4	4 / 9	7 / 10	9 / 16	10 / 16	16 / 20	17 / 20	20 / 21
Rezerva R	**0**	**0**	3	**0**	3	**0**	8	**0**

De exemplu, testarea începe după D și E, deci $ES(F) = \max(16, 13) = 16$. La calculul înapoi, B are succesorii D și G, deci $LF(B) = \min(9, 17) = 9$. Durata minimă este $T = 21$ de zile, iar drumul critic este

$$A \to B \to D \to F \to H, \qquad 4 + 5 + 7 + 4 + 1 = 21.$$

Celelalte două drumuri de la început la sfârșit au lungimile

$$A \to C \to E \to F \to H : \ 18, \qquad A \to B \to G \to H : \ 13.$$

b) Documentația (G) are rezerva 8 zile. O întârziere de 5 zile se încadrează în rezervă, deci lansarea nu se amână. Într-adevăr, drumul $A \to B \to G \to H$ ar avea $13 + 5 = 18 < 21$ zile.

c) Programarea serverului (E) are rezerva 3 zile. O întârziere de 4 zile depășește rezerva cu o zi. Drumul $A \to C \to E \to F \to H$ ar avea $18 + 4 = 22$ zile, deci proiectul ar dura 22 de zile. Drumul critic s-ar schimba și ar trece acum prin C și E.

d) Programarea serverului nu este critică, deci scurtarea ei nu modifică durata (drumul critic rămâne de 21 de zile). Programarea interfeței este critică. Scurtând-o cu 2 zile, drumurile au lungimile 19, 18 și 13, deci proiectul durează 19 zile. Aceasta este varianta utilă.

Dacă programarea interfeței s-ar scurta cu 4 zile, drumul $A \to B \to D \to F \to H$ ar avea 17 zile, dar drumul $A \to C \to E \to F \to H$ are 18 zile și devine critic. Proiectul ar dura 18 zile, deci câștigul ar fi doar de 3 zile, nu de 4. Aceasta ilustrează observația din partea teoretică: scurtarea are efect doar până când un alt drum devine critic.

Probleme propuse

Problemele marcate cu ★ sunt mai provocatoare intelectual.

1. Codul unei rețele. O rețea arbore cu 9 noduri are legăturile

$$1\text{–}4,\quad 2\text{–}4,\quad 3\text{–}4,\quad 4\text{–}5,\quad 5\text{–}6,\quad 6\text{–}7,\quad 6\text{–}8,\quad 8\text{–}9.$$

Desenați rețeaua și determinați codul ei Prüfer. Verificați proprietatea gradelor.

2. Reconstruirea unei rețele. Desenați arborele al cărui cod Prüfer este $(6, 2, 2, 6, 7, 1)$. Verificați rezultatul, recalculând codul.

3. Numărarea arborilor. Câți arbori cu vârfurile $1, \ldots, 7$ există în care:

a) vârful 1 este frunză?

b) vârful 1 are gradul 3?

c) gradele sunt $d(1) = d(2) = 3$, $d(3) = 2$ și $d(4) = d(5) = d(6) = d(7) = 1$?

d) există exact două frunze?

4. Rețea de irigații. Opt parcele, numerotate $1, \ldots, 8$, trebuie legate prin canale de irigație. Costurile canalelor posibile sunt:

$$12:\ 4,\quad 13:\ 3,\quad 23:\ 2,\quad 24:\ 5,\quad 34:\ 6,\quad 35:\ 4,\quad 45:\ 3,$$
$$46:\ 7,\quad 56:\ 5,\quad 57:\ 6,\quad 67:\ 2,\quad 68:\ 4,\quad 78:\ 3.$$

a) Determinați, cu algoritmul lui Kruskal, o rețea de canale de cost minim.

b) Este această rețea unică? Justificați.

5. Prim și Dijkstra. Pentru rețeaua din Problema propusă 4:

a) Aplicați algoritmul lui Prim pornind din parcela 1 și comparați rezultatul cu cel obținut prin algoritmul lui Kruskal.

b) Determinați arborele drumurilor minime din parcela 1 și comparați-l cu APM.

6. Rețea de comunicații. O rețea are 10 noduri, iar listele de vecini sunt:

Nod	1	2	3	4	5	6	7	8	9	10
Vecini	2, 4, 5	1, 3, 6	2, 7	1, 8	1, 6, 9	2, 5, 10	3, 10	4, 9	5, 8	6, 7

a) Aplicați BFS din nodul 1, precizând conținutul cozii la fiecare pas.

b) Determinați numărul minim de legături de la nodul 1 la fiecare nod și desenați arborele BFS.

c) Determinați un traseu cu număr minim de legături de la nodul 1 la nodul 10.

7. ★ Calul pe o tablă 5×5. Un cal se află în colțul stânga-jos al unei table de șah 5×5. Folosind BFS pe graful săriturilor calului, determinați numărul minim de mutări necesare pentru a ajunge în fiecare pătrat al tablei. Care este numărul minim de mutări până în colțul dreapta-sus?

8. Problema vaselor. Avem un vas de 4 litri și un vas de 9 litri, inițial goale, și o sursă de apă, cu aceleași operații ca în Problema 11. Determinați, cu BFS, numărul minim de operații prin care se obțin exact 6 litri într-un vas și precizați operațiile.

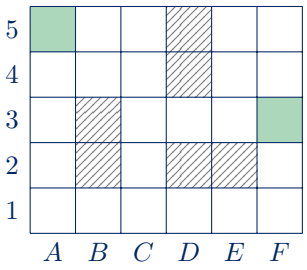
9. Bipartiție cu BFS. Un graf are vârfurile $1, \ldots, 8$ și muchiile

$$12,\ 16,\ 23,\ 27,\ 34,\ 38,\ 45,\ 56,\ 58,\ 78.$$

a) Folosind nivelurile BFS din vârful 1, decideți dacă graful este bipartit.

b) Reluați cerința după adăugarea muchiei 13. Dacă graful nu mai este bipartit, determinați un ciclu impar.

10. Evacuarea unui etaj. Planul unui etaj este împărțit în pătrate, ca în figură (coloanele $A, \ldots, F$, rândurile $1, \ldots, 5$). Pătratele hașurate sunt pereți, iar ieșirile se află în $A5$ și $F3$ (pătratele verzi). Dintr-un pătrat se poate trece în pătratele vecine (sus, jos, stânga, dreapta) care nu sunt pereți.



a) Determinați, cu o singură parcurgere BFS, numărul minim de pași de la fiecare pătrat până la cea mai apropiată ieșire.

b) Care sunt pătratele cele mai defavorizate?

c) Se poate deschide o a treia ieșire, fie în $A1$, fie în $F1$. Care variantă reduce mai mult distanța maximă până la o ieșire?

11. ★ Centrul unei rețele. O rețea arbore are 12 noduri și legăturile

$$1\text{–}2,\ 1\text{–}3,\ 2\text{–}4,\ 2\text{–}5,\ 3\text{–}6,\ 4\text{–}12,\ 5\text{–}7,\ 5\text{–}8,\ 6\text{–}9,\ 8\text{–}11,\ 9\text{–}10.$$

Se știe că, într-un arbore, nodul cel mai depărtat de un nod oarecare este capătul unui drum de lungime maximă.

a) Aplicați BFS din nodul 4 și determinați un nod u cât mai depărtat de 4.

b) Aplicați BFS din u și determinați un nod v cât mai depărtat de u. Deduceți lungimea maximă a unui drum în rețea.

c) Un server trebuie amplasat într-un nod, astfel încât distanța de la server la cel mai depărtat nod să fie cât mai mică. Unde trebuie amplasat? Justificați.

12. Parcurgere în adâncime. Un graf are vârfurile $1, \ldots, 9$ și muchiile

$$12,\ 14,\ 23,\ 25,\ 36,\ 45,\ 56,\ 58,\ 69,\ 78,\ 89.$$

a) Aplicați DFS din vârful 1 și determinați momentele t_{in} și t_{out}.

b) Desenați arborele DFS și precizați muchiile de întoarcere.

c) Pentru fiecare muchie de întoarcere, indicați ciclul pe care îl determină.

13. Construcția unei case. Activitățile unui șantier și condițiile de precedență ($X \to Y$ înseamnă că X trebuie terminată înainte de începerea lui Y) sunt:

F	fundația	T	tencuielile
S	structura	P	pardoseala
A	acoperișul	Z	zugrăveala
I	instalațiile	M	mobilarea

$$F \to S, \quad S \to A, \quad S \to I, \quad A \to T, \quad A \to Z,$$
$$I \to T, \quad T \to P, \quad T \to Z, \quad P \to M, \quad Z \to M.$$

a) Folosind DFS, verificați că nu există condiții circulare.

b) Determinați o ordine în care pot fi executate activitățile.

c) Ce se întâmplă dacă se adaugă condiția $Z \to I$?

14. ★ Ordini de parcurgere. Un graf are vârfurile $1, \ldots, 6$ și muchiile

$$12,\ 13,\ 24,\ 34,\ 35,\ 46,\ 56.$$

a) Pentru fiecare dintre șirurile

$$(1,3,2,5,4,6), \quad (1,2,4,3,5,6), \quad (1,2,3,6,4,5), \quad (1,3,4,2,6,5)$$

decideți dacă poate fi ordinea de vizitare a unei parcurgeri BFS din 1, a unei parcurgeri DFS din 1 sau a niciuneia.

b) Pentru ordinile posibile, precizați în ce ordine trebuie examinați vecinii.

c) Există un șir care să fie atât ordine BFS, cât și ordine DFS din 1? Justificați.

15. Puncte critice. Un graf are vârfurile $1, \ldots, 9$ și muchiile

$$12,\ 13,\ 23,\ 24,\ 45,\ 46,\ 56,\ 57,\ 78,\ 79,\ 89.$$

a) Aplicați DFS din vârful 1 și calculați valorile t_{in} și low.

b) Determinați punțile și punctele de articulație.

c) Adăugați o singură muchie astfel încât graful să nu mai aibă puncte critice.

16. Punți în arbori.

a) Arătați că într-un arbore orice muchie este punte.

b) Arătați că, într-un arbore cu cel puțin 3 vârfuri, punctele de articulație sunt exact vârfurile care nu sunt frunze.

c) Verificați aceste afirmații aplicând criteriile cu valori low (DFS din vârful 1) arborelui din Problema 3.

17. Un cartier cu sens unic. Un cartier are 10 intersecții și străzile

$$1\text{–}2, \quad 1\text{–}3, \quad 2\text{–}3, \quad 3\text{–}4, \quad 4\text{–}5, \quad 5\text{–}6,$$
$$6\text{–}7, \quad 4\text{–}7, \quad 6\text{–}8, \quad 8\text{–}9, \quad 8\text{–}10, \quad 9\text{–}10.$$

a) Determinați punțile rețelei. Se pot transforma toate străzile în străzi cu sens unic, astfel încât din orice intersecție să se ajungă în oricare alta?

b) Arătați că o stradă nouă 2–5 nu rezolvă problema, dar o stradă nouă 2–9 o rezolvă.

c) După construirea străzii 2–9, determinați o orientare potrivită folosind DFS din intersecția 1.

18. Drumuri minime. Opt localități $A, \ldots, H$ sunt legate prin drumuri cu lungimile (km):

$$AB = 3, \quad AC = 6, \quad AD = 9, \quad BC = 2, \quad BE = 7, \quad CD = 2, \quad CF = 8,$$
$$DF = 5, \quad EF = 1, \quad EG = 4, \quad FG = 6, \quad FH = 9, \quad GH = 2.$$

a) Aplicați algoritmul lui Dijkstra din A, precizând etichetele după fiecare pas.

b) Determinați distanțele de la A și un drum minim de la A la H.

c) Desenați arborele drumurilor minime.

19. Rețea cu sens unic. Un graf orientat are vârfurile s, a, b, c, d, t și arcele, cu costurile lor:

$$s \to a:\ 4, \quad s \to b:\ 1, \quad b \to a:\ 2, \quad a \to c:\ 5, \quad b \to d:\ 8,$$
$$c \to d:\ 1, \quad c \to t:\ 6, \quad d \to t:\ 2, \quad t \to s:\ 3, \quad d \to b:\ 2.$$

a) Determinați, cu algoritmul lui Dijkstra, costurile minime de la s la toate vârfurile.
b) Determinați costul minim de la t la s și de la d la a.

20. Modificarea ponderilor. Fie un graf cu ponderi pozitive și două vârfuri u, v.

a) Arătați că, dacă toate ponderile se înmulțesc cu același număr $c > 0$, drumurile minime de la u la v rămân aceleași.
b) Arătați, printr-un exemplu, că dacă la toate ponderile se adună același număr $c > 0$, drumul minim de la u la v se poate schimba.

21. Costuri negative. Un graf orientat are vârfurile $1, \ldots, 5$ și arcele, cu costurile lor:

$$1 \to 2:\ 4, \quad 1 \to 3:\ 5, \quad 2 \to 3:\ -2, \quad 3 \to 2:\ 3,$$
$$2 \to 4:\ 6, \quad 3 \to 5:\ 2, \quad 4 \to 5:\ -3, \quad 5 \to 4:\ 4.$$

a) Aplicați algoritmul Bellman–Ford din vârful 1, precizând etichetele după fiecare rundă.
b) Determinați traseele de cost minim de la 1 la fiecare vârf.
c) Aplicați și algoritmul lui Dijkstra. Dă rezultatul corect?

22. Detectarea unui circuit negativ. Un graf orientat are vârfurile $1, \ldots, 5$ și arcele

$$1 \to 2:\ 2, \quad 2 \to 3:\ -1, \quad 3 \to 4:\ -2, \quad 4 \to 2:\ 1, \quad 1 \to 4:\ 5, \quad 4 \to 5:\ 3.$$

a) Aplicați algoritmul Bellman–Ford din vârful 1 și decideți dacă există un circuit negativ. Dacă da, determinați-l folosind predecesorii.
b) Reluați cerința dacă arcul $4 \to 2$ are costul 4.

23. Durata construcției. Activitățile din Problema propusă 13 au duratele (zile): fundația 10, structura 20, acoperișul 8, instalațiile 12, tencuielile 10, pardoseala 6, zugrăveala 5, mobilarea 3.

a) Determinați durata minimă a construcției și drumul critic.
b) Determinați rezerva fiecărei activități.
c) Cu cât poate întârzia cel mult acoperișul fără a întârzia mutarea în casă?

24. O masă festivă. Pregătirea unei mese festive începe la ora 10:00 și cuprinde activitățile (durate în minute):

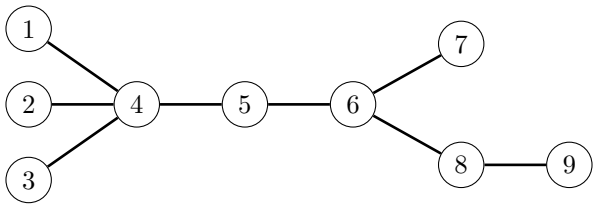
Cod	Activitatea	Precedată de	Durata
A	cumpărăturile	–	60
B	marinarea cărnii	A	120
C	pregătirea legumelor	A	30
D	coacerea cărnii	B	90
E	salata	C	20
F	desertul	A	50
G	aranjarea mesei	E, F	15
H	servirea	D, G	10

a) La ce oră se poate termina cel mai devreme servirea? Care este drumul critic?
b) Cu cât poate întârzia cel mult pregătirea legumelor?
c) Desertul durează cu 40 de minute mai mult decât s-a estimat. Se modifică ora servirii?

Rezolvarea problemelor propuse

Problema propusă 1. Codul unei rețele

Listele de adiacență sunt: 1: 4. 2: 4. 3: 4. 4: 1, 2, 3, 5. 5: 4, 6. 6: 5, 7, 8. 7: 6. 8: 6, 9. 9: 8. Rețeaua are 9 noduri și 8 legături și este conexă, deci este un arbore.



La fiecare pas ștergem frunza cea mai mică și scriem vecinul ei.

Pas	Frunzele arborelui curent	Frunza ștearsă	Se scrie
1	$1, 2, 3, 7, 9$	1	4
2	$2, 3, 7, 9$	2	4
3	$3, 7, 9$	3	4
4	$4, 7, 9$	4	5
5	$5, 7, 9$	5	6
6	$7, 9$	7	6
7	$6, 9$	6	8

După pasul 3, nodul 4 a rămas legat doar de 5, deci a devenit frunză. La fel, 5 devine frunză după pasul 4, iar 6 după pasul 6. După 7 pași rămân nodurile 8 și 9, legate între ele, și ne oprim. Codul Prüfer este

$$(4,\ 4,\ 4,\ 5,\ 6,\ 6,\ 8).$$

Proprietatea gradelor. Gradele sunt $d(4) = 4$, $d(6) = 3$, $d(5) = d(8) = 2$, iar nodurile $1, 2, 3, 7, 9$ au gradul 1. În cod, 4 apare de $3 = d(4) - 1$ ori, 6 apare de $2 = d(6) - 1$ ori, 5 și 8 apar câte o dată, iar frunzele nu apar deloc. Fiecare nod v apare de $d(v) - 1$ ori.

Problema propusă 2. Reconstruirea unei rețele

Codul are lungimea 6, deci arborele are $n = 8$ vârfuri. Calculăm $d(v) = 1+$(numărul de apariții ale lui v în cod):

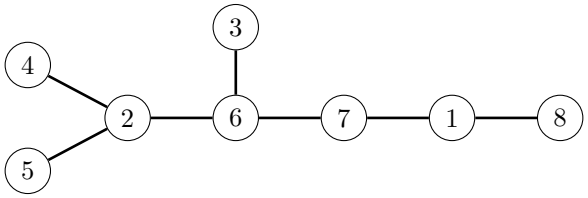
$$d(1) = 2,\ d(2) = 3,\ d(3) = 1,\ d(4) = 1,\ d(5) = 1,\ d(6) = 3,\ d(7) = 2,\ d(8) = 1.$$

La pasul i alegem cel mai mic vârf x cu $d(x) = 1$, adăugăm muchia dintre x și elementul a_i al codului, apoi ștergem x și micșorăm $d(a_i)$ cu 1.

Pas i	a_i	Vârfuri cu $d = 1$	x	Muchia adăugată și modificarea gradelor
1	6	$3, 4, 5, 8$	3	3–6, $d(6) = 2$
2	2	$4, 5, 8$	4	4–2, $d(2) = 2$
3	2	$5, 8$	5	5–2, $d(2) = 1$
4	6	$2, 8$	2	2–6, $d(6) = 1$
5	7	$6, 8$	6	6–7, $d(7) = 1$
6	1	$7, 8$	7	7–1, $d(1) = 1$

La final au rămas vârfurile 1 și 8, ambele cu $d = 1$, pe care le unim prin muchia 1–8. Arborele are muchiile

$$3\text{–}6, \quad 2\text{–}4, \quad 2\text{–}5, \quad 2\text{–}6, \quad 6\text{–}7, \quad 1\text{–}7, \quad 1\text{–}8.$$



Verificare. Codificăm arborele obținut. Frunzele sunt $3, 4, 5, 8$. Ștergem pe rând 3 (scriem 6), 4 (scriem 2), 5 (scriem 2), apoi 2, devenit frunză (scriem 6), apoi 6 (scriem 7) și 7 (scriem 1). Rămân 1 și 8. Regăsim codul $(6, 2, 2, 6, 7, 1)$.

Problema propusă 3. Numărarea arborilor

Folosim corespondența bijectivă dintre arborii cu vârfurile $1, \ldots, 7$ și șirurile de lungime $7-2 = 5$ cu valori în $\{1, \ldots, 7\}$ (codurile Prüfer), precum și proprietatea gradelor: vârful v apare în cod de exact $d(v) - 1$ ori. Numărăm deci coduri.

a) Vârful 1 este frunză dacă și numai dacă $d(1) = 1$, adică 1 nu apare în cod. Fiecare dintre cele 5 poziții poate primi oricare dintre celelalte 6 valori, deci există

$$6^5 = 7776$$

de arbori.

b) Vârful 1 are gradul 3 dacă și numai dacă apare în cod de exact 2 ori. Alegem cele 2 poziții ocupate de 1 în $\binom{5}{2} = 10$ moduri, iar celelalte 3 poziții primesc valori diferite de 1, în $6^3 = 216$ moduri. Rezultă

$$\binom{5}{2} \cdot 6^3 = 10 \cdot 216 = 2160$$

de arbori.

c) Codul conține de 2 ori pe 1, de 2 ori pe 2, o dată pe 3 și deloc vârfurile $4, 5, 6, 7$ (frunzele). Numărul acestor șiruri este numărul de aranjări ale „cuvântului” $1, 1, 2, 2, 3$:

$$\frac{5!}{2!\,2!\,1!} = \frac{120}{4} = 30.$$

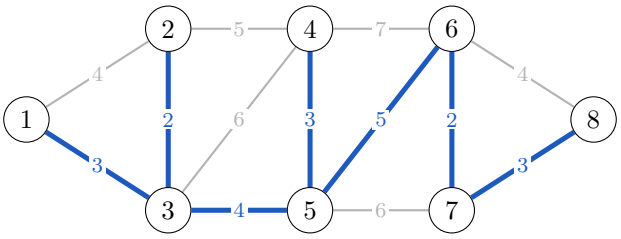
Verificare: suma gradelor este $3 + 3 + 2 + 1 + 1 + 1 + 1 = 12 = 2(7 - 1)$, cum trebuie pentru un arbore.

d) Frunzele sunt exact vârfurile care nu apar în cod. Exact două frunze înseamnă că exact 5 vârfuri apar în cod, iar cum codul are 5 poziții, ele apar câte o dată, deci codul are toate elementele distincte. Numărul acestor coduri este

$$7 \cdot 6 \cdot 5 \cdot 4 \cdot 3 = 2520.$$

Observație: un arbore cu exact două frunze este un lanț (celelalte 5 vârfuri au gradul 2). Numărul lanțurilor cu 7 vârfuri etichetate este $7!/2 = 2520$, deoarece fiecare lanț se poate citi în două sensuri. Cele două numărări coincid.

Problema propusă 4. Rețea de irigații



a) Ordonăm canalele crescător după cost (la egalitate, în ordinea lexicografică):

23 (2), 67 (2), 13 (3), 45 (3), 78 (3), 12 (4), 35 (4), 68 (4), 24 (5), 56 (5), 34 (6), 57 (6), 46 (7).

Canal	Cost	Se adaugă?	Componente după pas
23	2	da	$\{2,3\}$, restul izolate
67	2	da	$\{2,3\}, \{6,7\}$
13	3	da	$\{1,2,3\}, \{6,7\}$
45	3	da	$\{1,2,3\}, \{4,5\}, \{6,7\}$
78	3	da	$\{1,2,3\}, \{4,5\}, \{6,7,8\}$
12	4	nu (ciclu 1 2 3)	neschimbate
35	4	da	$\{1,2,3,4,5\}, \{6,7,8\}$
68	4	nu (ciclu 6 7 8)	neschimbate
24	5	nu (ciclu 2 3 5 4)	neschimbate
56	5	da	$\{1,\ldots,8\}$

Am ales $n-1=7$ canale și ne oprim. Rețeaua de cost minim (albastră în desen) are canalele

$$23,\ 67,\ 13,\ 45,\ 78,\ 35,\ 56, \qquad \text{cost } 2+2+3+3+3+4+5=22.$$

b) Costurile nu sunt distincte, deci nu putem aplica direct observația din teorie. Folosim următorul criteriu. Fiecare canal f care nu este în arbore formează cu arborele un ciclu. Dacă f este *strict* mai scump decât toate celelalte canale ale acestui ciclu, pentru fiecare f, atunci APM este unic. Verificăm:

Canal în afara arborelui	Cost	Ciclul format și costurile canalelor din arbore
12	4	1 2 3: 23 (2), 13 (3)
68	4	6 7 8: 67 (2), 78 (3)
24	5	2 3 5 4: 23 (2), 35 (4), 45 (3)
34	6	3 5 4: 35 (4), 45 (3)
57	6	5 6 7: 56 (5), 67 (2)
46	7	4 5 6: 45 (3), 56 (5)

În fiecare rând costul canalului din afara arborelui este mai mare decât toate costurile din ultima coloană. Justificarea criteriului: fie T arborele găsit și T' un alt arbore parțial. Atunci T' conține un canal f care nu este în T. Eliminând f din T', acesta se rupe în două componente. Ciclul format de f cu T trece de la o componentă la cealaltă și prin f, deci conține și un alt canal g din T care leagă cele două componente. Cum g este mai ieftin decât f, arborele obținut din T' înlocuind f cu g are cost mai mic. Deci T' nu este de cost minim. Prin urmare rețeaua de cost minim este unică.

Problema propusă 5. Prim și Dijkstra

a) Algoritmul lui Prim din parcela 1. Eticheta $c(v)$ este costul celui mai ieftin canal de la v la arborele curent. Valorile îngroșate corespund parcelelor deja incluse.

Pas	Ales	Canal	1	2	3	4	5	6	7	8
1	1	–	**0**	4	3	∞	∞	∞	∞	∞
2	3	13	**0**	2	**3**	6	4	∞	∞	∞
3	2	23	**0**	**2**	**3**	5	4	∞	∞	∞
4	5	35	**0**	**2**	**3**	3	**4**	5	6	∞
5	4	45	**0**	**2**	**3**	**3**	**4**	5	6	∞
6	6	56	**0**	**2**	**3**	**3**	**4**	**5**	2	4
7	7	67	**0**	**2**	**3**	**3**	**4**	**5**	**2**	3
8	8	78	**0**	**2**	**3**	**3**	**4**	**5**	**2**	**3**

Actualizările importante:

- Pasul 2: se alege 3. Pentru 2, canalul 23 costă $2 < 4$, deci $c(2) = 2$. Obținem și $c(4) = 6$, $c(5) = 4$.
- Pasul 3: se alege 2. Pentru 4, canalul 24 costă $5 < 6$, deci $c(4) = 5$.
- Pasul 4: se alege 5. Pentru 4, canalul 45 costă $3 < 5$. Obținem și $c(6) = 5$, $c(7) = 6$.
- Pasul 5: se alege 4. Canalul 46 (7) nu îmbunătățește $c(6) = 5$.
- Pasul 6: se alege 6. Pentru 7, $2 < 6$, deci $c(7) = 2$. Pentru 8 obținem 4.
- Pasul 7: se alege 7. Pentru 8, $3 < 4$, deci $c(8) = 3$. Pasul 8: se alege 8.

Prim obține canalele 13, 23, 35, 45, 56, 67, 78, cu costul 22, adică aceeași rețea ca algoritmul lui Kruskal (era de așteptat, deoarece APM este unic). Diferă doar ordinea: Kruskal a ales $23, 67, 13, 45, 78, 35, 56$, construind întâi mai multe „bucăți”, în timp ce Prim crește un singur arbore pornind din 1.

b) Algoritmul lui Dijkstra din parcela 1. Eticheta $d(v)$ este lungimea celui mai scurt drum cunoscut de la 1 la v.

Pas	Ales	Canal	1	2	3	4	5	6	7	8
1	1	–	**0**	4	3	∞	∞	∞	∞	∞
2	3	13	**0**	4	**3**	9	7	∞	∞	∞
3	2	12	**0**	**4**	**3**	9	7	∞	∞	∞
4	5	35	**0**	**4**	**3**	9	**7**	12	13	∞
5	4	34	**0**	**4**	**3**	**9**	**7**	12	13	∞
6	6	56	**0**	**4**	**3**	**9**	**7**	**12**	13	16
7	7	57	**0**	**4**	**3**	**9**	**7**	**12**	**13**	16
8	8	68	**0**	**4**	**3**	**9**	**7**	**12**	**13**	**16**

Câteva comparații:

- Pasul 2: din 3, pentru 2 obținem $3 + 2 = 5$, care *nu* este mai mic decât 4. Prim schimba aici eticheta lui 2, deoarece compara doar costul canalului ($2 < 4$).
- Pasul 3: din 2, pentru 4 obținem $4 + 5 = 9$, nu mai puțin decât 9.
- Pasul 5: din 4, pentru 5 obținem $9 + 3 = 12 > 7$, iar pentru 6, $9 + 7 = 16 > 12$.

- Pasul 7: din 7, pentru 8 obținem $13 + 3 = 16$, egal cu eticheta existentă, deci 8 rămâne legat de 6. Există două drumuri minime de la 1 la 8, de lungime 16.

Arborele drumurilor minime are canalele 12, 13, 34, 35, 56, 57, 68, cu costul total $4 + 3 + 6 + 4 + 5 + 6 + 4 = 32 > 22$. Comparăm distanțele de la parcela 1:

Parcela	2	3	4	5	6	7	8
Drumul din APM	5	3	10	7	12	14	17
Drumul din arborele lui Dijkstra	4	3	9	7	12	13	16

De exemplu, în APM apa ajunge în parcela 2 pe traseul $1 \to 3 \to 2$, de lungime 5, deși canalul direct 12 are lungimea 4. Arborele lui Dijkstra este mai scump cu 10, dar oferă de la parcela 1 cele mai scurte trasee spre toate parcelele. APM minimizează costul total al rețelei, iar arborele lui Dijkstra minimizează fiecare distanță de la 1.

Problema propusă 6. Rețea de comunicații

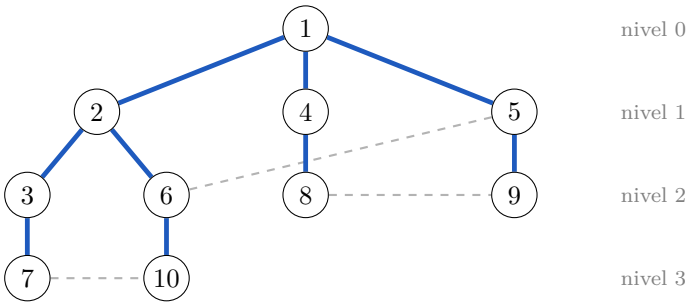
a) Pornim cu $Q = (1)$ și $\ell(1) = 0$. La fiecare pas extragem primul nod din coadă și adăugăm la sfârșitul ei vecinii lui încă nevizitați, în ordinea din liste.

Nod extras	Nivel	Noduri noi (predecesor)	Coada Q după pas
1	0	$2, 4, 5$ (1)	$2, 4, 5$
2	1	$3, 6$ (2)	$4, 5, 3, 6$
4	1	8 (4)	$5, 3, 6, 8$
5	1	9 (5)	$3, 6, 8, 9$
3	2	7 (3)	$6, 8, 9, 7$
6	2	10 (6)	$8, 9, 7, 10$
8	2	–	$9, 7, 10$
9	2	–	$7, 10$
7	3	–	10
10	3	–	vidă

De exemplu, când extragem 5, vecinii lui sunt 1, 6, 9, dar 1 și 6 sunt deja vizitați, deci doar 9 este nou.

b) Numărul minim de legături de la nodul 1 este nivelul BFS:

Nod	1	2	3	4	5	6	7	8	9	10
ℓ	0	1	2	1	1	2	3	2	2	3
Predecesor	–	1	2	1	1	2	3	4	5	6



Arborele BFS este albastru. Legăturile punctate, 5–6, 8–9 și 7–10, nu fac parte din arbore. Fiecare dintre ele unește noduri de pe niveluri alăturate sau de pe același nivel.

c) Mergem înapoi pe predecesori: $10 \leftarrow 6 \leftarrow 2 \leftarrow 1$. Un traseu cu număr minim de legături este $1 \to 2 \to 6 \to 10$, cu 3 legături. Nu există un traseu mai scurt, deoarece $\ell(10) = 3$. Traseul nu este unic: $1 \to 5 \to 6 \to 10$ are tot 3 legături.

Problema propusă 7. Calul pe o tablă 5×5 ★

Graful săriturilor are ca vârfuri cele 25 de pătrate. Două pătrate sunt adiacente dacă un cal poate sări dintr-unul în celălalt, adică diferă cu 1 pe o direcție și cu 2 pe cealaltă. Numărul minim de mutări până la un pătrat este nivelul lui în BFS pornit din $a1$.

Aplicăm BFS. Pe nivelul 0 se află $a1$. Din $a1$ calul poate sări doar în $b3$ și $c2$, care formează nivelul 1. Vecinii lor nevizitați formează nivelul 2 și așa mai departe:

Nivel	Pătrate
0	$a1$
1	$b3,\ c2$
2	$a3,\ a5,\ b4,\ c1,\ c5,\ d2,\ d4,\ e1,\ e3$
3	$a2,\ a4,\ b1,\ b5,\ c4,\ d1,\ d3,\ d5,\ e2,\ e4$
4	$b2,\ c3,\ e5$

Toate cele $1 + 2 + 9 + 10 + 3 = 25$ de pătrate sunt atinse. Scriem nivelurile pe tablă:

	a	b	c	d	e
5	2	3	2	3	4
4	3	2	3	2	3
3	2	1	4	3	2
2	3	4	1	2	3
1	0	3	2	3	2

Colțul dreapta-sus, $e5$, se află pe nivelul 4, deci sunt necesare minim 4 mutări. Un traseu optim, obținut mergând înapoi pe predecesori, este

$$a1 \to c2 \to e3 \to c4 \to e5.$$

Observație: deși se află „aproape” de $a1$, pătratele $b2$ și $c3$ cer tot câte 4 mutări. Distanța în graful săriturilor nu are legătură directă cu distanța pe tablă.

Problema propusă 8. Problema vaselor

Notăm cu A vasul de 4 litri și cu B vasul de 9 litri. O stare este perechea (a, b), cu $0 \le a \le 4$ și $0 \le b \le 9$. Ca în Problema 11, din (a, b) putem ajunge în $(4, b)$, $(a, 9)$ (umplere), $(0, b)$, $(a, 0)$ (golire), $(a - t, b + t)$ cu $t = \min(a, 9 - b)$ (turnare $A \to B$) și $(a + t, b - t)$ cu $t = \min(b, 4 - a)$ (turnare $B \to A$). Examinăm operațiile în această ordine și aplicăm BFS din $(0, 0)$ până când apare o stare cu 6 litri.

Pas	Stare extrasă	Nivel	Stări noi (operația)	Coada Q după pas
1	$(0,0)$	0	$(4,0)$ (umple A), $(0,9)$ (umple B)	$(4,0),(0,9)$
2	$(4,0)$	1	$(4,9)$ (umple B), $(0,4)$ ($A \to B$)	$(0,9),(4,9),(0,4)$
3	$(0,9)$	1	$(4,5)$ ($B \to A$)	$(4,9),(0,4),(4,5)$
4	$(4,9)$	2	–	$(0,4),(4,5)$
5	$(0,4)$	2	$(4,4)$ (umple A)	$(4,5),(4,4)$
6	$(4,5)$	2	$(0,5)$ (golește A)	$(4,4),(0,5)$
7	$(4,4)$	3	$(0,8)$ ($A \to B$)	$(0,5),(0,8)$
8	$(0,5)$	3	$(4,1)$ ($B \to A$)	$(0,8),(4,1)$
9	$(0,8)$	4	$(4,8)$ (umple A)	$(4,1),(4,8)$
10	$(4,1)$	4	$(0,1)$ (golește A)	$(4,8),(0,1)$
11	$(4,8)$	5	$(3,9)$ ($A \to B$)	$(0,1),(3,9)$
12	$(0,1)$	5	$(1,0)$ ($B \to A$)	$(3,9),(1,0)$
13	$(3,9)$	6	$(3,0)$ (golește B)	$(1,0),(3,0)$
14	$(1,0)$	6	$(1,9)$ (umple B)	$(3,0),(1,9)$
15	$(3,0)$	7	$(0,3)$ ($A \to B$)	$(1,9),(0,3)$
16	$(1,9)$	7	$(4,6)$ ($B \to A$)	$(0,3),(4,6)$

Starea $(4,6)$ apare pe nivelul 8 și este prima stare cu 6 litri. Toate stările de pe nivelurile $0,\dots,7$ apar deja în tabel și niciuna nu conține 6 litri, deci sunt necesare minim 8 operații. Mergând înapoi pe predecesori obținem:

Operația	Ce se face	Starea (a,b)
1	umple B	$(0,9)$
2	toarnă din B în A	$(4,5)$
3	golește A	$(0,5)$
4	toarnă din B în A	$(4,1)$
5	golește A	$(0,1)$
6	toarnă din B în A	$(1,0)$
7	umple B	$(1,9)$
8	toarnă din B în A	$(4,6)$

La final vasul B conține 6 litri. Ideea traseului: scoțând de două ori câte 4 litri din vasul plin B rămâne 1 litru, pe care îl mutăm în A. Apoi umplem B și completăm A cu încă 3 litri, deci în B rămân $9-3=6$ litri.

Problema propusă 9. Bipartiție cu BFS

Listele de adiacență sunt: 1: 2, 6. 2: 1, 3, 7. 3: 2, 4, 8. 4: 3, 5. 5: 4, 6, 8. 6: 1, 5. 7: 2, 8. 8: 3, 5, 7.

a) BFS din 1, cu vecinii examinați în ordine crescătoare:

- nivelul 0: 1. Din 1 descoperim 2 și 6.
- nivelul 1: 2, 6. Din 2 descoperim 3 și 7, iar din 6 descoperim 5.
- nivelul 2: 3, 7, 5. Din 3 descoperim 4 și 8. Din 7 și 5 nu apar vârfuri noi.
- nivelul 3: 4, 8.

Verificăm muchiile: 12 (nivelurile 0 și 1), 16 (0, 1), 23 (1, 2), 27 (1, 2), 34 (2, 3), 38 (2, 3), 45 (3, 2), 56 (2, 1), 58 (2, 3), 78 (2, 3). Nicio muchie nu unește două vârfuri de pe același nivel, deci graful este bipartit. Bipartiția este dată de nivelurile pare și impare:

$$X=\{1,3,5,7\}\ \text{(nivelurile 0 și 2)},\qquad Y=\{2,4,6,8\}\ \text{(nivelurile 1 și 3)}.$$

Fiecare muchie are un capăt impar și unul par, deci, în acest caz, chiar paritatea etichetelor dă bipartiția.

b) Cu muchia 13, vecinii lui 1 sunt 2, 3, 6. BFS din 1 dă:

- nivelul 0: 1. Din 1 descoperim 2, 3, 6.
- nivelul 1: 2, 3, 6. Din 2 descoperim 7. Vecinul 3 al lui 2 este deja pe nivelul 1. Din 3 descoperim 4 și 8, iar din 6 descoperim 5.
- nivelul 2: 7, 4, 8, 5.

Muchia 23 unește două vârfuri de pe nivelul 1, deci graful nu mai este bipartit. (La fel muchiile 45, 58 și 78, care unesc vârfuri de pe nivelul 2.) Ciclul impar se obține din muchia 23 și drumurile din arborele BFS de la 1 la 2 și la 3:

$$1 \to 2 \to 3 \to 1, \qquad \text{de lungime 3.}$$

Explicație directă: un graf bipartit nu poate conține un ciclu de lungime impară, deoarece pe un ciclu vârfurile alternează între X și Y.

Problema propusă 10. Evacuarea unui etaj

a) Pătratele care nu sunt pereți sunt vârfurile unui graf, iar pătratele vecine sunt legate prin muchii. Distanța de la un pătrat la cea mai apropiată ieșire se obține printr-un BFS cu mai multe surse, ca în Problema 12: punem ambele ieșiri, $A5$ și $F3$, în coadă de la început, cu nivelul 0. Valul 1 conține vecinii celor două ieșiri, valul 2 vecinii nevizitați ai acestora și așa mai departe:

Pași	Pătrate
0	$A5$, $F3$ (ieșirile)
1	$A4$, $B5$, $E3$, $F2$, $F4$
2	$A3$, $B4$, $C5$, $D3$, $E4$, $F1$, $F5$
3	$A2$, $C3$, $C4$, $E1$, $E5$
4	$A1$, $C2$, $D1$
5	$B1$, $C1$

5	0	1	2		3	2
4	1	2	3		2	1
3	2		3	2	1	0
2	3		4			1
1	4	5	5	4	3	2
	A	B	C	D	E	F

b) Cele mai defavorizate sunt pătratele $B1$ și $C1$ (portocalii în figură), aflate la 5 pași de cea mai apropiată ieșire. De exemplu, din $C1$ un drum optim este $C1 \to C2 \to C3 \to D3 \to E3 \to F3$. Peretele $D2$–$E2$ obligă la acest ocol.

c) Reluăm BFS cu trei surse.

- Cu ieșirea suplimentară $A1$, valurile sunt: 0: $A1, A5, F3$. 1: $A2, A4, B1, B5, E3, F2, F4$. 2: $A3, B4, C1, C5, D3, E4, F1, F5$. 3: $C2, C3, C4, D1, E1, E5$. Distanța maximă scade de la 5 la 3.
- Cu ieșirea suplimentară $F1$, distanța maximă devine 4, atinsă în $A1$, $B1$ și $C2$. Ieșirea $F1$ este aproape de ieșirea existentă $F3$, deci ajută puțin.

Varianta mai bună este ieșirea din $A1$. Observație: ieșirea trebuie pusă lângă zona cea mai depărtată de ieșirile existente, nu oriunde pe margine.

Problema propusă 11. Centrul unei rețele ★

Listele de vecini sunt: 1: 2, 3. 2: 1, 4, 5. 3: 1, 6. 4: 2, 12. 5: 2, 7, 8. 6: 3, 9. 7: 5. 8: 5, 11. 9: 6, 10. 10: 9. 11: 8. 12: 4.

a) BFS din 4 dă nivelurile:

Nivel	Noduri
0	4
1	2, 12
2	1, 5
3	3, 7, 8
4	6, 11
5	9
6	10

Nodul cel mai depărtat de 4 este $u = 10$, la distanța 6, pe drumul $4 \to 2 \to 1 \to 3 \to 6 \to 9 \to 10$.

b) BFS din 10 dă nivelurile: 0: 10. 1: 9. 2: 6. 3: 3. 4: 1. 5: 2. 6: 4, 5. 7: 12, 7, 8. 8: 11. Nodul cel mai depărtat de 10 este $v = 11$, la distanța 8. Conform proprietății din enunț, $u = 10$ este capătul unui drum de lungime maximă, deci cel mai lung drum din rețea are lungimea 8:

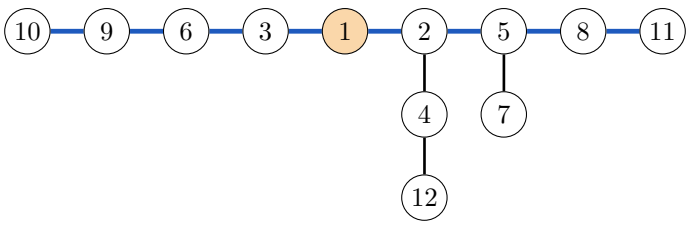
$$10 \to 9 \to 6 \to 3 \to 1 \to 2 \to 5 \to 8 \to 11.$$

Lungimea maximă a unui drum dintr-un graf se numește *diametrul* grafului.

Justificarea proprietății din enunț. Fie a un nod oarecare, u un nod cât mai depărtat de a și P un drum de lungime maximă D, cu capetele x și y. Notăm cu c nodul lui P cel mai apropiat de a. Într-un arbore drumul dintre două noduri este unic, deci drumul de la a la orice nod al lui P trece prin c. Putem presupune $d(c, y) \geq d(c, x)$, deci $d(c, y) \geq D/2$. Arătăm că din u pleacă un drum de lungime cel puțin D, deci exact D. Cum u este cel mai depărtat de a, avem $d(a, u) \geq d(a, y) = d(a, c) + d(c, y)$. Distingem după modul în care drumul de la a la u întâlnește drumul de la a la c și pe P.

- Drumul de la a la u părăsește drumul de la a la c într-un nod $m \neq c$. Atunci $d(a, u) = d(a, m) + d(m, u)$ și $d(a, y) = d(a, m) + d(m, c) + d(c, y)$, deci $d(m, u) \geq d(m, c) + d(c, y)$. Drumul $u \to m \to c \to x$ are lungimea $d(u, m) + d(m, c) + d(c, x) \geq d(c, y) + d(c, x) = D$.
- Drumul de la a la u trece prin c, merge pe P până la un nod r și apoi părăsește P. Atunci $d(c, u) = d(c, r) + d(r, u) \geq d(c, y)$. Dacă r se află între c și y, drumul $u \to r \to c \to x$ are lungimea $d(c, u) + d(c, x) \geq D$. Dacă r se află între c și x, drumul $u \to r \to c \to y$ are lungimea $d(c, u) + d(c, y) \geq 2\, d(c, y) \geq d(c, x) + d(c, y) = D$.

În toate cazurile u este capătul unui drum de lungime D, ceea ce trebuia arătat.



c) Pentru un nod x, notăm cu $e(x)$ distanța de la x la cel mai depărtat nod. Deoarece $d(10, 11) = 8$, pentru orice nod x avem $d(x, 10) + d(x, 11) \geq 8$, deci unul dintre cei doi termeni este cel puțin

4 și $e(x) \geq 4$. Nodul 1, aflat la mijlocul drumului de mai sus, are $e(1) = 4$: BFS din 1 dă nivelurile 0: 1. 1: 2, 3. 2: 4, 5, 6. 3: 12, 7, 8, 9. 4: 11, 10. Serverul trebuie amplasat în nodul 1, cel mai depărtat nod fiind la 4 legături. Un nod cu această proprietate se numește *centrul* arborelui. Celelalte noduri au $e(x) \geq 5$, de exemplu $e(2) = 5$ (până la 10) și $e(3) = 5$ (până la 11).

Problema propusă 12. Parcurgere în adâncime

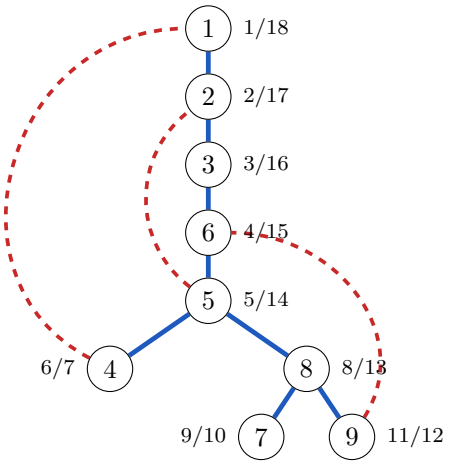
Listele de adiacență sunt: 1: 2, 4. 2: 1, 3, 5. 3: 2, 6. 4: 1, 5. 5: 2, 4, 6, 8. 6: 3, 5, 9. 7: 8. 8: 5, 7, 9. 9: 6, 8.

a) Aplicăm DFS din 1, cu ceasul t crescut la fiecare intrare și la fiecare terminare.

- $t = 1$: intrăm în 1, apoi în 2 ($t = 2$), în 3 ($t = 3$), în 6 ($t = 4$) și în 5 ($t = 5$). La fiecare pas am mers în primul vecin nevizitat.
- În 5, vecinul 2 este vizitat și este strămoș: *muchia de întoarcere* 5–2. Mergem în 4 ($t = 6$).
- În 4, vecinul 1 este strămoș: muchia de întoarcere 4–1. Vecinul 5 este părintele. Terminăm 4 la $t = 7$.
- Revenim în 5. Vecinul 6 este părintele, deci mergem în 8 ($t = 8$), apoi în 7 ($t = 9$). Vârful 7 nu are alți vecini, deci îl terminăm la $t = 10$.
- Revenim în 8 și mergem în 9 ($t = 11$). Vecinul 6 al lui 9 este strămoș: muchia de întoarcere 9–6. Terminăm 9 la $t = 12$.
- Urmează terminările: 8 ($t = 13$), 5 ($t = 14$), 6 ($t = 15$), 3 ($t = 16$), 2 ($t = 17$), 1 ($t = 18$).

Vârf	1	2	3	4	5	6	7	8	9
t_{in}	1	2	3	6	5	4	9	8	11
t_{out}	18	17	16	7	14	15	10	13	12
Predecesor	–	1	2	5	6	3	8	5	8

b) Arborele DFS are muchiile 1–2, 2–3, 3–6, 6–5, 5–4, 5–8, 8–7, 8–9 (8 muchii, $n - 1$). Muchiile de întoarcere sunt 5–2, 4–1 și 9–6.



c) Fiecare muchie de întoarcere închide un ciclu împreună cu drumul din arbore dintre capetele ei:

- 5–2: ciclul $2 \to 3 \to 6 \to 5 \to 2$, de lungime 4.
- 4–1: ciclul $1 \to 2 \to 3 \to 6 \to 5 \to 4 \to 1$, de lungime 6.
- 9–6: ciclul $6 \to 5 \to 8 \to 9 \to 6$, de lungime 4.

Problema propusă 13. Construcția unei case

Vârfurile grafului orientat sunt activitățile, iar arcele sunt condițiile de precedență. Listele de succesori sunt: F: S. S: A, I. A: T, Z. I: T. T: P, Z. P: M. Z: M. M: niciunul.

a) Lansăm DFS din F, urmând arcele în sensul lor, cu succesorii examinați în ordinea de mai sus. Un vârf este *gri* cât timp este vizitat, dar neterminat, și *negru* după terminare.

- Intrăm în F ($t = 1$), S ($t = 2$), A ($t = 3$), T ($t = 4$), P ($t = 5$) și M ($t = 6$). Vârful M nu are succesori: îl terminăm la $t = 7$, apoi terminăm P la $t = 8$.
- Revenim în T și mergem în Z ($t = 9$). Arcul $Z \to M$ duce la un vârf negru. Terminăm Z ($t = 10$), T ($t = 11$). Arcul $A \to Z$ duce la un vârf negru, deci terminăm A ($t = 12$).
- Revenim în S și mergem în I ($t = 13$). Arcul $I \to T$ duce la un vârf negru. Terminăm I ($t = 14$), S ($t = 15$) și F ($t = 16$).

Activitate	F	S	A	I	T	P	Z	M
t_{in}	1	2	3	13	4	5	9	6
t_{out}	16	15	12	14	11	8	10	7

Nu am întâlnit niciun arc către un vârf gri, deci nu există condiții circulare.

b) Ordonăm activitățile descrescător după t_{out}:

$$F\ (16),\ S\ (15),\ I\ (14),\ A\ (12),\ T\ (11),\ Z\ (10),\ P\ (8),\ M\ (7).$$

Verificăm că fiecare arc merge spre o activitate aflată mai târziu în listă: de exemplu $I \to T$ ($14 > 11$), $A \to Z$ ($12 > 10$), $T \to P$ ($11 > 8$), $Z \to M$ ($10 > 7$). Aceasta este o ordine posibilă: fundația, structura, instalațiile, acoperișul, tencuielile, zugrăveala, pardoseala, mobilarea.

c) Parcurgem din nou graful cu arcul nou $Z \to I$. Până la $t = 8$ totul decurge la fel. Din T intrăm în Z ($t = 9$), iar acum Z are succesorii M și I. Vârful M este negru, deci intrăm în I ($t = 10$). Din I urmăm arcul $I \to T$, iar T este gri (vizitat, dar neterminat). Am găsit circuitul

$$T \to Z \to I \to T.$$

Condițiile cer ca tencuielile să fie gata înaintea zugrăvelii, zugrăveala înaintea instalațiilor, iar instalațiile înaintea tencuielilor, ceea ce este imposibil. Nu mai există nicio ordine de execuție a lucrărilor.

Problema propusă 14. Ordini de parcurgere ★

Listele de vecini sunt: 1: 2, 3. 2: 1, 4. 3: 1, 4, 5. 4: 2, 3, 6. 5: 3, 6. 6: 4, 5. Nivelurile BFS din 1 sunt: nivelul 0: 1, nivelul 1: 2, 3, nivelul 2: 4, 5, nivelul 3: 6. Ele nu depind de ordinea vecinilor.

Folosim două criterii simple:

- Într-o ordine BFS, vârfurile apar în ordinea crescătoare a nivelurilor.
- Într-o ordine DFS, fiecare vârf nou este un vecin nevizitat al vârfului curent. Parcurgerea revine pe drumul parcurs doar când vârful curent nu mai are vecini nevizitați.

a) și b)

- $(1, 3, 2, 5, 4, 6)$: nivelurile sunt $0, 1, 1, 2, 2, 3$, în ordine crescătoare. Este ordine BFS, dacă examinăm vecinii lui 1 în ordinea $3, 2$ și ai lui 3 în ordinea $5, 4$: din 1 intră în coadă 3 și 2, din 3 intră 5 și 4, din 2 nu intră nimic nou, din 5 intră 6. Nu este ordine DFS: după 3, DFS ar trebui să meargă într-un vecin nevizitat al lui 3 (4 sau 5), nu în 2.

- $(1, 2, 4, 3, 5, 6)$: este ordine DFS, cu vecinii lui 4 examinați în ordinea $3, 6$: $1 \to 2 \to 4 \to 3 \to 5 \to 6$, fiecare vârf fiind vecin cu precedentul. Nu este ordine BFS, deoarece 4 (nivelul 2) apare înaintea lui 3 (nivelul 1).
- $(1, 2, 3, 6, 4, 5)$: nu este ordine BFS, deoarece 6 (nivelul 3) apare înaintea lui 4 (nivelul 2). Nu este nici ordine DFS: după 2, al cărui vecin nevizitat este 4, DFS trebuie să meargă în 4, nu în 3.
- $(1, 3, 4, 2, 6, 5)$: este ordine DFS. Din 1 mergem în 3, din 3 în 4 (vecinii lui 3 în ordinea $4, 5$), din 4 în 2 (vecinii lui 4 în ordinea $2, 6$). Vârful 2 nu mai are vecini nevizitați, revenim în 4 și mergem în 6, iar din 6 în 5. Nu este ordine BFS, deoarece 4 (nivelul 2) apare înaintea lui 2 (nivelul 1).

c) Nu. Într-o ordine BFS din 1, pozițiile a doua și a treia sunt ocupate de vecinii lui 1, adică de 2 și 3, în ordinea dorită. Într-o ordine DFS, al treilea vârf este un vecin nevizitat al celui de-al doilea, deoarece atât 2, cât și 3 au vecini nevizitați după primul pas. Cum 2 și 3 nu sunt vecini, al treilea vârf DFS nu poate fi celălalt dintre ele. Prin urmare nicio ordine nu este simultan BFS și DFS. Aici cele trei ordini BFS posibile sunt $(1, 2, 3, 4, 5, 6)$, $(1, 3, 2, 4, 5, 6)$ și $(1, 3, 2, 5, 4, 6)$.

Problema propusă 15. Puncte critice

Listele de adiacență sunt: 1: 2, 3. 2: 1, 3, 4. 3: 1, 2. 4: 2, 5, 6. 5: 4, 6, 7. 6: 4, 5. 7: 5, 8, 9. 8: 7, 9. 9: 7, 8.

a) Aplicăm DFS din 1. Valoarea $low(v)$ este cea mai mică valoare t_{in} care se poate atinge din subarborele lui v folosind cel mult o muchie de întoarcere. O calculăm la terminarea lui v.

- Intrăm în 1 ($t = 1$), 2 ($t = 2$), 3 ($t = 3$). În 3, vecinul 1 este strămoș (muchia de întoarcere 3–1), deci $low(3) = 1$. Terminăm 3 ($t = 4$).
- Revenim în 2 și intrăm în 4 ($t = 5$), 5 ($t = 6$), 6 ($t = 7$). În 6, vecinul 4 este strămoș (muchia de întoarcere 6–4), deci $low(6) = 5$. Terminăm 6 ($t = 8$).
- Revenim în 5 și intrăm în 7 ($t = 9$), 8 ($t = 10$), 9 ($t = 11$). În 9, vecinul 7 este strămoș (muchia 9–7), deci $low(9) = 9$. Terminăm 9 ($t = 12$).
- La terminări: $low(8) = \min(10, low(9)) = 9$, $low(7) = \min(9, low(8)) = 9$, $low(5) = \min(6, low(6), low(7)) = \min(6, 5, 9) = 5$, $low(4) = \min(5, low(5)) = 5$, $low(2) = \min(2, low(3), low(4)) = 1$, $low(1) = 1$.

Vârf v	1	2	3	4	5	6	7	8	9
$t_{\text{in}}(v)$	1	2	3	5	6	7	9	10	11
Predecesor	–	1	2	2	4	5	5	7	8
$low(v)$	1	1	1	5	5	5	9	9	9

b) *Punți.* O muchie de arbore $p(v)\, v$ este punte dacă $low(v) > t_{\text{in}}(p(v))$:

- 2–4: $low(4) = 5 > 2$, deci este punte.
- 5–7: $low(7) = 9 > 6$, deci este punte.
- Celelalte nu sunt punți. De exemplu, 1–2: $low(2) = 1$, iar 4–5: $low(5) = 5$, care nu este mai mare decât $t_{\text{in}}(4) = 5$.

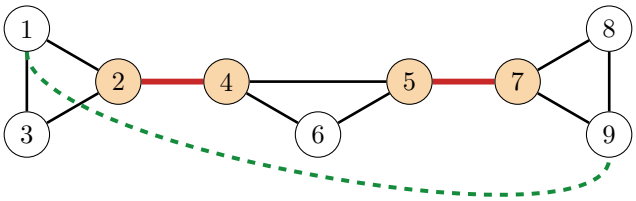
Puncte de articulație. Rădăcina 1 are un singur fiu (2), deci nu este punct de articulație. Un vârf $u \neq 1$ este punct de articulație dacă are un fiu v cu $low(v) \geq t_{\text{in}}(u)$:

- 2 (fiul 4: $5 \geq 2$), 4 (fiul 5: $5 \geq 5$), 5 (fiul 7: $9 \geq 6$) și 7 (fiul 8: $9 \geq 9$) sunt puncte de articulație.

- 3, 6 și 9 nu au fii. Vârful 8 are fiul 9, cu $low(9) = 9 < 10$.

Punțile sunt 2–4 și 5–7, iar punctele de articulație sunt $2, 4, 5, 7$.

c) Graful este format din trei triunghiuri, $\{1, 2, 3\}$, $\{4, 5, 6\}$ și $\{7, 8, 9\}$, legate în lanț prin cele două punți. Adăugăm muchia 1–9 (verde punctat în desen), care unește cele două triunghiuri de la capete.



Muchia nouă formează ciclul $1 \to 2 \to 4 \to 5 \to 7 \to 9 \to 1$, care conține ambele punți și toate punctele de articulație.

- Nu mai există punți: fiecare muchie se află acum pe un ciclu (pe un triunghi sau pe ciclul de mai sus).
- Nu mai există puncte de articulație. Dacă eliminăm, de exemplu, vârful 4, triunghiul $\{1, 2, 3\}$ rămâne legat de restul grafului prin drumul $1 \to 9 \to 7 \to 5$. La fel, eliminând 2, 5 sau 7, cele două părți rămân legate prin cealaltă porțiune a ciclului. Eliminarea unui alt vârf nu deconectează graful nici înainte, nici după adăugare.

Orice muchie între $\{1, 3\}$ și $\{8, 9\}$ are același efect. O muchie care are un capăt într-un punct de articulație nu ajunge. De exemplu, cu muchia 2–9, vârful 2 rămâne punct de articulație, deoarece separă vârfurile 1 și 3 de restul grafului.

Problema propusă 16. Punți în arbori

a) Fie uv o muchie a arborelui T. Presupunem că $T - uv$ este încă conex. Atunci există în $T - uv$ un drum de la u la v, care împreună cu muchia uv formează un ciclu în T. Aceasta contrazice faptul că T este un arbore. Deci $T - uv$ nu este conex și uv este punte.

b) Fie $n \geq 3$.

- Dacă v este o frunză, eliminarea ei lasă un graf conex: orice drum între două alte vârfuri nu trece prin v, deoarece un vârf interior al unui drum are cel puțin doi vecini pe drum. Deci frunzele nu sunt puncte de articulație.
- Dacă v nu este frunză, el are cel puțin doi vecini, a și b. Dacă $T - v$ ar fi conex, un drum de la a la b în $T - v$, împreună cu muchiile va și vb, ar forma un ciclu în T, ceea ce este imposibil. Deci v este punct de articulație.

(Condiția $n \geq 3$ exclude arborele cu două vârfuri, în care ambele vârfuri sunt frunze.)

c) Arborele din Problema 3 este arborele binar complet cu 15 vârfuri, în care vârful k are fiii $2k$ și $2k + 1$. Aplicăm DFS din 1 (cu vecinii în ordine crescătoare) și obținem:

v	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
t_{in}	1	2	16	3	9	17	23	4	6	10	12	18	20	24	26
t_{out}	30	15	29	8	14	22	28	5	7	11	13	19	21	25	27
low	1	2	16	3	9	17	23	4	6	10	12	18	20	24	26

Un arbore nu are muchii de întoarcere, deci $low(v) = t_{\text{in}}(v)$ pentru orice vârf.

- *Punți.* Pentru orice muchie $p(v)\,v$ avem $low(v) = t_{\text{in}}(v) > t_{\text{in}}(p(v))$, deoarece fiul este descoperit după părinte. Toate cele 14 muchii sunt punți, conform punctului a).
- *Puncte de articulație.* Rădăcina 1 are doi fii (2 și 3), deci este punct de articulație. Un vârf $u \in \{2, \ldots, 7\}$ are fiii $2u$ și $2u+1$, iar pentru fiul $v = 2u$ avem $low(v) = t_{\text{in}}(v) > t_{\text{in}}(u)$, deci u este punct de articulație. Frunzele $8, \ldots, 15$ nu au fii, deci nu sunt puncte de articulație.

Punctele de articulație sunt $1, \ldots, 7$, exact vârfurile care nu sunt frunze, conform punctului b).

Problema propusă 17. Un cartier cu sens unic

a) Rețeaua este formată din triunghiul $\{1, 2, 3\}$, patrulaterul 4–5–6–7–4 și triunghiul $\{8, 9, 10\}$, legate prin străzile 3–4 și 6–8. Aceste două străzi nu se află pe niciun ciclu, deci sunt punți. Celelalte străzi se află pe câte un ciclu și nu sunt punți. (Aceleași rezultate se obțin cu valorile *low* dintr-un DFS din 1.) Conform teoremei lui Robbins (Problema 20), orientarea cerută nu există: de exemplu, dacă 3–4 primește sensul $3 \to 4$, din intersecțiile $4, \ldots, 10$ nu se mai poate reveni în $\{1, 2, 3\}$.

b) Strada 2–5 formează ciclul $2 \to 3 \to 4 \to 5 \to 2$, care conține puntea 3–4. Puntea 6–8 rămâne însă singura legătură spre $\{8, 9, 10\}$, deci problema persistă.

Strada 2–9 formează ciclul $2 \to 3 \to 4 \to 5 \to 6 \to 8 \to 9 \to 2$, care conține ambele punți. După construirea ei, fiecare stradă se află pe un ciclu, deci rețeaua nu mai are punți și, conform teoremei lui Robbins, orientarea există.

c) Listele de vecini devin: 1: 2, 3. 2: 1, 3, 9. 3: 1, 2, 4. 4: 3, 5, 7. 5: 4, 6. 6: 5, 7, 8. 7: 4, 6. 8: 6, 9, 10. 9: 2, 8, 10. 10: 8, 9. Aplicăm DFS din 1, orientând muchiile de arbore de la părinte la fiu și muchiile de întoarcere de la descendent la strămoș:

t_{in}	Intersecția	Arce obținute
1	1	$1 \to 2$
2	2	$2 \to 3$
3	3	$3 \to 1$ (întoarcere), $3 \to 4$
4	4	$4 \to 5$
5	5	$5 \to 6$
6	6	$6 \to 7$
7	7	$7 \to 4$ (întoarcere). Terminăm 7 la $t = 8$ și revenim în 6.
9	8	$6 \to 8$ (arbore, din 6), $8 \to 9$
10	9	$9 \to 2$ (întoarcere), $9 \to 10$
11	10	$10 \to 8$ (întoarcere)

Orientarea are arcele de arbore $1 \to 2$, $2 \to 3$, $3 \to 4$, $4 \to 5$, $5 \to 6$, $6 \to 7$, $6 \to 8$, $8 \to 9$, $9 \to 10$ și arcele de întoarcere $3 \to 1$, $7 \to 4$, $9 \to 2$, $10 \to 8$, în total 13 străzi. Din 1 se ajunge în orice intersecție pe arcele de arbore. Drumurile înapoi spre 1 sunt, de exemplu, $7 \to 4 \to 5 \to 6 \to 8 \to 9 \to 2 \to 3 \to 1$ și $10 \to 8 \to 9 \to 2 \to 3 \to 1$. Fiecare intersecție are un astfel de drum, deci din orice intersecție se ajunge în oricare alta. Observați că strada nouă 2–9 este folosită de toate drumurile care pleacă din intersecțiile $4, \ldots, 10$ spre 1.

Problema propusă 18. Drumuri minime

a) Aplicăm algoritmul lui Dijkstra din A. Valorile îngroșate sunt etichete definitive.

Pas	Ales	A	B	C	D	E	F	G	H
1	A	**0**	3	6	9	∞	∞	∞	∞
2	B	**0**	**3**	5	9	10	∞	∞	∞
3	C	**0**	**3**	**5**	7	10	13	∞	∞
4	D	**0**	**3**	**5**	**7**	10	12	∞	∞
5	E	**0**	**3**	**5**	**7**	**10**	11	14	∞
6	F	**0**	**3**	**5**	**7**	**10**	**11**	14	20
7	G	**0**	**3**	**5**	**7**	**10**	**11**	**14**	16
8	H	**0**	**3**	**5**	**7**	**10**	**11**	**14**	**16**

Relaxările, pas cu pas:

- Pasul 1: din A obținem $d(B) = 3$, $d(C) = 6$, $d(D) = 9$.
- Pasul 2: se alege B (3). Pentru C, $3 + 2 = 5 < 6$, deci $d(C) = 5$, $p(C) = B$. Pentru E, $3 + 7 = 10$.
- Pasul 3: se alege C (5). Pentru D, $5 + 2 = 7 < 9$, deci $d(D) = 7$, $p(D) = C$. Pentru F, $5 + 8 = 13$.
- Pasul 4: se alege D (7). Pentru F, $7 + 5 = 12 < 13$, deci $d(F) = 12$, $p(F) = D$.
- Pasul 5: se alege E (10). Pentru F, $10 + 1 = 11 < 12$, deci $d(F) = 11$, $p(F) = E$. Pentru G, $10 + 4 = 14$.
- Pasul 6: se alege F (11). Pentru G, $11 + 6 = 17$ nu este mai mic decât 14. Pentru H, $11 + 9 = 20$.
- Pasul 7: se alege G (14). Pentru H, $14 + 2 = 16 < 20$, deci $d(H) = 16$, $p(H) = G$.
- Pasul 8: se alege H (16).

Observăm că eticheta lui F a scăzut de două ori ($13 \to 12 \to 11$), pe măsură ce s-au găsit drumuri mai bune.

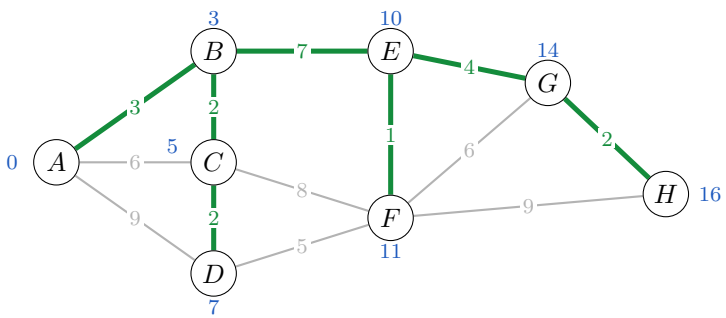
b) Distanțele de la A sunt

$$B = 3, \quad C = 5, \quad D = 7, \quad E = 10, \quad F = 11, \quad G = 14, \quad H = 16.$$

Pentru H mergem înapoi pe predecesori: $H \leftarrow G \leftarrow E \leftarrow B \leftarrow A$. Drumul minim este

$$A \to B \to E \to G \to H, \qquad 3 + 7 + 4 + 2 = 16 \text{ km}.$$

c) Arborele drumurilor minime are muchiile $p(v)\,v$: AB, BC, CD, BE, EF, EG, GH.



În desen, arborele drumurilor minime este verde, iar lângă fiecare localitate este scrisă distanța ei față de A. Fiecare localitate se atinge din A urmând doar muchiile verzi, pe cel mai scurt drum.

Problema propusă 19. Rețea cu sens unic

Listele de succesori, cu costurile, sunt: s: $a(4)$, $b(1)$. a: $c(5)$. b: $a(2)$, $d(8)$. c: $d(1)$, $t(6)$. d: $b(2)$, $t(2)$. t: $s(3)$. Toate costurile sunt pozitive, deci putem aplica algoritmul lui Dijkstra, relaxând doar arcele în sensul lor.

a) Dijkstra din s:

Pas	Ales	s	a	b	c	d	t
1	s	**0**	4	1	∞	∞	∞
2	b	**0**	3	**1**	∞	9	∞
3	a	**0**	**3**	**1**	8	9	∞
4	c	**0**	**3**	**1**	**8**	9	14
5	d	**0**	**3**	**1**	**8**	**9**	11
6	t	**0**	**3**	**1**	**8**	**9**	**11**

- Pasul 2: se alege b (1). Pentru a, $1 + 2 = 3 < 4$, deci $d(a) = 3$, $p(a) = b$. Pentru d, $1 + 8 = 9$.
- Pasul 3: se alege a (3). Pentru c, $3 + 5 = 8$.
- Pasul 4: se alege c (8). Pentru d, $8 + 1 = 9$, egal cu eticheta existentă, deci nu se schimbă nimic. Pentru t, $8 + 6 = 14$.
- Pasul 5: se alege d (9). Arcul $d \to b$ duce la un vârf definitiv. Pentru t, $9 + 2 = 11 < 14$, deci $d(t) = 11$, $p(t) = d$.

Costurile minime de la s sunt $a = 3$, $b = 1$, $c = 8$, $d = 9$, $t = 11$. De exemplu, un traseu optim spre t este $s \to b \to d \to t$, cu costul $1 + 8 + 2 = 11$. Traseul $s \to b \to a \to c \to d \to t$ are tot costul $1 + 2 + 5 + 1 + 2 = 11$, deci drumul minim nu este unic.

b) De la t pleacă un singur arc, $t \to s$, cu costul 3, deci costul minim de la t la s este 3. Observăm că de la s la t costul minim este 11: într-un graf orientat distanța depinde de sens.

Pentru costul de la d la a aplicăm Dijkstra din d. Din d obținem $b = 2$ și $t = 2$. Se alege b (2), care dă $a = 2 + 2 = 4$. Se alege t (2), care dă $s = 2 + 3 = 5$. Se alege a (4), care dă $c = 9$. Etichetele lui s și c nu mai scad. Costul minim de la d la a este 4, pe traseul $d \to b \to a$.

Problema propusă 20. Modificarea ponderilor

Pentru un drum P notăm cu $L(P)$ suma ponderilor muchiilor lui și cu $k(P)$ numărul de muchii.

a) După înmulțirea ponderilor cu $c > 0$, noua lungime a drumului P este $c\,L(P)$. Pentru oricare două drumuri P_1, P_2 de la u la v avem

$$L(P_1) \le L(P_2) \iff c\,L(P_1) \le c\,L(P_2),$$

deoarece $c > 0$. Ordinea drumurilor după lungime nu se schimbă, deci drumurile minime rămân aceleași, iar distanța de la u la v se înmulțește cu c. (Un exemplu practic: trecerea de la kilometri la metri nu schimbă traseul cel mai scurt.)

b) După adunarea lui c la fiecare pondere, noua lungime a lui P este $L(P) + c\,k(P)$. Drumurile cu multe muchii sunt penalizate mai mult decât cele cu puține muchii, deci ordinea se poate schimba. Exemplu: vârfurile u, x, v, cu ponderile $w(ux) = 2$, $w(xv) = 2$, $w(uv) = 5$.

Drum	Lungime inițială	Lungime după adunarea lui $c = 2$
$u \to x \to v$	$2 + 2 = \mathbf{4}$	$4 + 4 = 8$
$u \to v$	5	$5 + 2 = \mathbf{7}$

Inițial drumul minim este $u \to x \to v$, iar după modificare drumul minim devine muchia directă $u \to v$.

Problema propusă 21. Costuri negative

a) Notăm cu $d_k(v)$ costul minim al unui drum de la 1 la v cu cel mult k arce. Avem $n = 5$, deci 4 runde de relaxare, urmate de o rundă de verificare. În fiecare rundă calculăm d_k din d_{k-1}, relaxând toate arcele.

Runda k	1	2	3	4	5
0	0	∞	∞	∞	∞
1	0	4	5	∞	∞
2	0	4	2	10	7
3	0	4	2	10	4
4	0	4	2	8	4

- Runda 1: din 1 obținem $d_1(2) = 4$ și $d_1(3) = 5$.
- Runda 2: $d_2(3) = \min(5,\ d_1(2) - 2) = \min(5, 2) = 2$, cu predecesorul 2. De asemenea $d_2(4) = d_1(2) + 6 = 10$ și $d_2(5) = d_1(3) + 2 = 7$. Pentru 2, $d_1(3) + 3 = 8$ nu este mai mic decât 4.
- Runda 3: $d_3(5) = \min(7,\ d_2(3) + 2) = 4$. Pentru 4, $d_2(5) + 4 = 11$ nu este mai mic decât 10.
- Runda 4: $d_4(4) = \min(10,\ d_3(5) + 4) = 8$, cu predecesorul 5. Pentru 5, $d_3(4) - 3 = 7$ nu este mai mic decât 4.
- Verificare: pentru niciun arc nu mai obținem o etichetă mai mică. De exemplu $d(4) - 3 = 5 \geq d(5) = 4$ și $d(3) + 3 = 5 \geq d(2) = 4$. Nu există circuite negative accesibile din 1, iar etichetele sunt costurile minime.

b) Mergând înapoi pe predecesori obținem traseele de cost minim:

Vârf	Traseu	Cost
2	$1 \to 2$	4
3	$1 \to 2 \to 3$	$4 - 2 = 2$
5	$1 \to 2 \to 3 \to 5$	$4 - 2 + 2 = 4$
4	$1 \to 2 \to 3 \to 5 \to 4$	$4 - 2 + 2 + 4 = 8$

De remarcat că drumul spre 3 trece prin 2, deși există arcul direct $1 \to 3$, iar drumul spre 4 ocolește arcul direct $2 \to 4$, ambele datorită arcului negativ $2 \to 3$.

c) Aplicăm algoritmul lui Dijkstra din 1.

- Pasul 1: din 1, $d(2) = 4$, $d(3) = 5$.
- Pasul 2: se alege 2 (4). Pentru 3, $4 - 2 = 2 < 5$, deci $d(3) = 2$. Pentru 4, $4 + 6 = 10$.
- Pasul 3: se alege 3 (2). Pentru 5, $2 + 2 = 4$.
- Pasul 4: se alege 5 (4). Pentru 4, $4 + 4 = 8 < 10$, deci $d(4) = 8$.

- Pasul 5: se alege 4 (8). Pentru 5, $8 - 3 = 5$, nu mai puțin decât 4.

Dijkstra obține aceleași valori, $0, 4, 2, 8, 4$, deci aici rezultatul este corect. Este însă o întâmplare: arcul negativ $2 \to 3$ a fost relaxat înainte ca 3 să devină definitiv, deoarece 2 a fost ales înaintea lui 3 ($4 < 5$). Garanția de corectitudine nu mai există. De exemplu, dacă arcul $1 \to 3$ ar avea costul 3, Dijkstra ar alege întâi vârful 3, cu eticheta definitivă 3, iar apoi relaxarea $4 - 2 = 2$ ar veni prea târziu. Costul corect ar fi 2, pe traseul $1 \to 2 \to 3$. Pentru grafuri cu costuri negative trebuie folosit Bellman–Ford.

Problema propusă 22. Detectarea unui circuit negativ

a) Avem $n = 5$, deci 4 runde și o rundă de verificare. Calculăm d_k din d_{k-1}, relaxând toate arcele.

Runda k	1	2	3	4	5	Ce se schimbă
0	0	∞	∞	∞	∞	
1	0	2	∞	5	∞	$p(2) = 1$, $p(4) = 1$
2	0	2	1	5	8	3: $2 - 1$, 5: $5 + 3$
3	0	2	1	-1	8	4: $1 - 2 = -1$, $p(4) = 3$
4	0	0	1	-1	2	2: $-1 + 1 = 0$, $p(2) = 4$ și 5: $-1 + 3 = 2$

În runda de verificare, arcul $2 \to 3$ dă $d(2) - 1 = -1 < d(3) = 1$. O etichetă care încă scade după $n - 1 = 4$ runde arată că există un circuit negativ accesibil din 1.

Pentru a-l determina, mergem înapoi pe predecesori pornind din vârful 3, a cărui etichetă a scăzut: $p(3) = 2$, $p(2) = 4$, $p(4) = 3$. Ne-am întors în 3, deci circuitul este

$$2 \to 3 \to 4 \to 2, \qquad -1 - 2 + 1 = -2 < 0.$$

Parcurgând de mai multe ori acest circuit, costul drumurilor spre 2, 3, 4 și 5 scade oricât, deci pentru aceste vârfuri nu există drum de cost minim.

b) Cu $w(4 \to 2) = 4$, circuitul $2 \to 3 \to 4 \to 2$ are costul $-1 - 2 + 4 = 1 > 0$. Rundele 1–3 sunt identice cu cele de la a). În runda 4, pentru 2 obținem $-1 + 4 = 3$, care nu este mai mic decât 2, iar pentru 5 obținem $-1 + 3 = 2$:

Runda k	1	2	3	4	5
3	0	2	1	-1	8
4	0	2	1	-1	2

În runda de verificare nicio etichetă nu mai scade (de exemplu $d(4) + 4 = 3 \geq d(2) = 2$), deci nu există circuit negativ. Costurile minime sunt $d(2) = 2$, $d(3) = 1$, $d(4) = -1$, $d(5) = 2$, pe traseele $1 \to 2$, $1 \to 2 \to 3$, $1 \to 2 \to 3 \to 4$ și $1 \to 2 \to 3 \to 4 \to 5$. Drumul spre 4 prin 2 și 3 (cost -1) este mai bun decât arcul direct $1 \to 4$ (cost 5).

Problema propusă 23. Durata construcției

Ordinea F, S, A, I, T, P, Z, M este o sortare topologică: fiecare activitate apare după predecesorii ei. Predecesorii direcți sunt: S: F. A, I: S. T: A, I. P: T. Z: A, T. M: P, Z.

a) *Calculul înainte*:

- F: $ES = 0$, $EF = 10$. S: $ES = 10$, $EF = 30$.
- A: $ES = 30$, $EF = 38$. I: $ES = 30$, $EF = 42$.

- T începe după A și I: $ES = \max(38, 42) = 42$, $EF = 52$.
- P: $ES = 52$, $EF = 58$. Z începe după A și T: $ES = \max(38, 52) = 52$, $EF = 57$.
- M începe după P și Z: $ES = \max(58, 57) = 58$, $EF = 61$.

Durata minimă a construcției este $T = 61$ de zile.

Calculul înapoi, de la M spre F:

- M: $LF = 61$, $LS = 58$. P și Z (succesor M): $LF = 58$, deci $LS(P) = 52$ și $LS(Z) = 53$.
- T (succesori P, Z): $LF = \min(52, 53) = 52$, $LS = 42$.
- A (succesori T, Z): $LF = \min(42, 53) = 42$, $LS = 34$. I (succesor T): $LF = 42$, $LS = 30$.
- S (succesori A, I): $LF = \min(34, 30) = 30$, $LS = 10$. F: $LF = 10$, $LS = 0$.

Activitate	F	S	A	I	T	P	Z	M
Durata	10	20	8	12	10	6	5	3
ES / EF	0 / 10	10 / 30	30 / 38	30 / 42	42 / 52	52 / 58	52 / 57	58 / 61
LS / LF	0 / 10	10 / 30	34 / 42	30 / 42	42 / 52	52 / 58	53 / 58	58 / 61
Rezerva R	**0**	**0**	4	**0**	**0**	**0**	1	**0**

Drumul critic este

$$F \to S \to I \to T \to P \to M, \qquad 10 + 20 + 12 + 10 + 6 + 3 = 61 \text{ zile.}$$

b) Rezervele sunt: 4 zile pentru acoperiș, 1 zi pentru zugrăveală și 0 pentru celelalte activități, care sunt critice.

c) Acoperișul are rezerva $R(A) = LS(A) - ES(A) = 34 - 30 = 4$, deci poate întârzia cel mult 4 zile. Verificare: cu o întârziere de 4 zile acoperișul se termină în ziua $38 + 4 = 42$, exact când se termină și instalațiile, deci tencuielile încep tot în ziua 42. O întârziere de 5 zile ar amâna tencuielile, deci și mutarea în casă, cu o zi.

Problema propusă 24. O masă festivă

Măsurăm timpul în minute de la ora 10:00. Ordinea A, B, C, D, E, F, G, H este o sortare topologică.

a) *Calculul înainte*:

- A: $ES = 0$, $EF = 60$.
- B, C, F încep după A: $EF(B) = 180$, $EF(C) = 90$, $EF(F) = 110$.
- D (după B): $ES = 180$, $EF = 270$. E (după C): $ES = 90$, $EF = 110$.
- G (după E și F): $ES = \max(110, 110) = 110$, $EF = 125$.
- H (după D și G): $ES = \max(270, 125) = 270$, $EF = 280$.

Servirea se termină cel mai devreme după 280 de minute, adică 4 ore și 40 de minute, deci la ora **14:40**.

Calculul înapoi:

- H: $LF = 280$, $LS = 270$. D și G (succesor H): $LF = 270$, deci $LS(D) = 180$ și $LS(G) = 255$.
- B (succesor D): $LF = 180$, $LS = 60$. E (succesor G): $LF = 255$, $LS = 235$. F (succesor G): $LF = 255$, $LS = 205$.

- C (succesor E): $LF = 235$, $LS = 205$.
- A (succesori B, C, F): $LF = \min(60, 205, 205) = 60$, $LS = 0$.

Activitate	A	B	C	D	E	F	G	H
Durata	60	120	30	90	20	50	15	10
ES / EF	0 / 60	60 / 180	60 / 90	180 / 270	90 / 110	60 / 110	110 / 125	270 / 280
LS / LF	0 / 60	60 / 180	205 / 235	180 / 270	235 / 255	205 / 255	255 / 270	270 / 280
Rezerva R	**0**	**0**	145	**0**	145	145	145	**0**

Drumul critic este

$$A \to B \to D \to H, \qquad 60 + 120 + 90 + 10 = 280 \text{ minute :}$$

cumpărăturile, marinarea, coacerea și servirea. Marinarea și coacerea durează mult, iar celelalte pregătiri se pot face în paralel cu ele.

b) Pregătirea legumelor are rezerva $R(C) = 205 - 60 = 145$ de minute, adică 2 ore și 25 de minute. Ea poate începe cel mai devreme la 11:00 și cel mai târziu la 13:25 fără să întârzie servirea.

c) Desertul ar dura $50 + 40 = 90$ de minute. Întârzierea de 40 de minute este mai mică decât rezerva $R(F) = 145$, deci ora servirii nu se schimbă. Într-adevăr, desertul s-ar termina la minutul $60 + 90 = 150$, aranjarea mesei la $150 + 15 = 165$, mult înainte de minutul 270, când se termină coacerea cărnii. Servirea se încheie tot la ora 14:40.