Exemplu de subiect pentru examenul de licenţă

Informatică

Definiţi următoarele funcţii C++ conform cerinţelor precizate şi scrieţi un program pentru testarea lor.

i) `double val(int a[], int n){...}` returnează *valoarea* v a vectorului `a` de dimensiune `n`, definită de relaţia

$$v = a_0 + a_1 \cdot 3 + a_2 \cdot 3^2 + \cdots + a_{n-1} \cdot 3^{n-1}.$$

ii) `bool esteAdmisibil(int a[], int n){...}` decide dacă vectorul `a` de dimensiune `n` este admisibil. Un vector este *admisibil* dacă are toate componentele egale cu -1, 0 sau 1 iar suma lor este 0.

iii) `bool esteReprezentabil(int v, int a[], int n){...}` decide dacă numărul întreg `v` este *reprezentabil* pe un registru de lungime `n`, adică dacă este egal cu valoarea unui vector admisibil de dimensiune `n`, caz în care funcţia returnează `true` şi încarcă vectorul căutat în tabloul `a`.

Explicaţi algoritmul folosit pentru ultima funcţie. Dimensiunea `n` se consideră mai mică decât `nmax=10`.

Rezolvare. Prezentăm trei variante pentru funcţia `esteReprezentabil()`. Prima, cea mai puţin eficientă, foloseşte *forţa brută*: în registrul `a` de lungime `n` generăm prin incrementare repetată toţi vectorii de componente -1, 0 sau 1, şi pentru fiecare vector format testăm dacă este admisibil şi dacă are valoarea căutată, caz în care oprim generarea.

Varianta a doua constă tot căutarea soluţiei prin verificarea tuturor vectorilor admisibili, dar acum generarea acestora este mai eficientă, prin *backtracking*: ţinem cont că din

$$a_0 + a_1 + \cdots + a_k + a_{k+1} + \cdots a_{n-1} = 0 \text{ şi } a_i \in \{-1, 0, 1\}, i = 0, 1, \ldots, n-1,$$

rezultă că

$$|a_0 + a_1 + \cdots + a_k| \leq |a_{k+1}| + \cdots + |a_{n-1}| \leq n - 1 - k,$$

condiţie care pentru $k > n/2$ restrânge posibilităţile de alegere a candidatului de pe locul k.

În sfârşit, ultima variantă, cea mai eficientă, pleacă de la observaţia că adunând lui v dat de reprezentarea

$$v = a_0 + a_1 \cdot 3 + a_2 \cdot 3^2 + \cdots + a_{n-1} \cdot 3^{n-1}$$

numărul

$$p = 1 + 1 \cdot 3 + 1 \cdot 3^2 + \cdots + 1 \cdot 3^{n-1} = \frac{3^n - 1}{2}$$

obţinem

$$v + p = \alpha_0 + \alpha_1 \cdot 3 + \alpha_2 \cdot 3^2 + \cdots + \alpha_{n-1} \cdot 3^{n-1},$$

cu $\alpha_i = a_i + 1 \in \{0, 1, 2\}$ pentru $i = 0, 1, \ldots, n-1$, adică exact scrierea numărului natural $v + p$ în baza de numeraţie 3.

Aşadar, fiind dat v, testăm dacă $v + p$ este un număr natural care poate fi scris cu n cifre în baza 3, adică dacă $0 \leq v + p \leq 2p$, şi, dacă da, aflăm cifrele acestuia şi le micşorăm cu o unitate. Obţinem astfel în mod unic vectorul a_0, a_1, ..., a_{n-1} pe care, în final, îl testăm dacă este admisibil sau nu.

```
#include<iostream>
using namespace std;

const int nmax = 10;

bool esteAdmisibil(int a[], int n){
    int s = 0;
    for (int i = 0; i < n; i++)
    {
        if (a[i] < -1 || a[i]>1) return false;
        s += a[i];
    }
    if (s != 0) return false;
    return true;
}

double val(int a[], int n){
    double s = 0, t = 1;
    for (int i = 0; i < n; i++){
        s += a[i] * t;
        t *= 3;
    }
    return s;
}

int suma(int a[], int n){
    int s = 0;
    for (int i = 0; i < n; i++) s += a[i];
    return s;
}

bool esteReprezentabil1(int v, int a[], int n){ //forta bruta
    if (n < 1 || n > nmax) return false;
    //initializam registrul a[]
    for (int i = 0; i < n; i++)  a[i] = -1;
    bool avemDepasireRegistru = false;
    while (!avemDepasireRegistru){
        if (esteAdmisibil(a, n) && v == val(a, n)) return true;
        //incrementam registrul a[]
        int k = 0;
        bool avemTransport = true;
        while (k < n && avemTransport){
            a[k]++;
            if (a[k] < 2) avemTransport = false;
            else {
                a[k] = -1;
                k++;
            }
        }
        avemDepasireRegistru = avemTransport;
    }
    return false;
}
```

```
bool esteReprezentabil2(int v, int a[], int n){ //backtracking
    if (n < 1 || n > nmax) return false;
    int k = 0;
    a[k] = -2;  //preinitializare
    while (k >= 0){
        //cautam un candidat valid pentru locul k
        bool amGasit = false;
        while (a[k] < 1 && !amGasit){
            a[k]++;
            if (k <= n / 2 || abs(suma(a, k + 1)) <= n - 1 - k) amGasit = true;
        }
        if (!amGasit){      //ne intoarcem
            k--;
        }
        else if (k == n - 1 && v == val(a, n)) return true;  // avem o solutie
        else if (k < n - 1){    //avansam
            k++;
            a[k] = -2;
        }
    }
    return false;
}

bool esteReprezentabil3(int v, int a[], int n){
    //folosim scrierea numerelor in baza 3
    if (n < 1 || n > nmax) return false;
    int p = 1;
    for (int i = 0; i < n; i++) p *= 3;
    p = (p - 1) / 2;
    v += p;
    if (v < 0 || v > 2 * p) return false;
    for (int i = 0; i < n; i++){
        a[i] = v % 3 - 1;
        v /= 3;
    }
    return esteAdmisibil(a, n);
}

int main(){
    int a[nmax] = {};
    int n = 7;
    int v = 312;
    if (esteReprezentabil3(v, a, n)) {
        cout << "da" << endl;
        for (int i = 0; i < n; i++) cout << a[i] << " ";
        cout << "\nv=" << v << " val=" << val(a, n) << endl;
    }
    else cout << "nu" << endl;
    return 0;
}
```