Descrierea Soluțiilor
Olimpiada Societății pentru Excelență și Performanță în Informatică, Etapa Națională
Clasa a VI-a

1 Problema Butoi

Propunător: prof. Marinel Șerban – Colegiul Național "Emil Racoviță" Iași

1.1 Pentru cerința1 și cerința2

Notăm cu $t_1, t_2..t_n$ numărul de utilizări pentru fiecare tip de găleată.

Se folosește un **algoritm de tip succesor** prin adunare cu 1 în baza $k+1$ pe n poziții. Algoritmul începe de la combinația nulă:

$$t_1 = 0; t_2 = 0; \ldots; t_{n-1} = 0; t_n = 0 \tag{1}$$

După o adunare cu 1 o să obținem:

$$t_1 = 0; t_2 = 0; \ldots; t_{n-1} = 0; t_n = 1 \tag{2}$$

După k adunări cu 1 o să obținem:

$$t_1 = 0; t_2 = 0; \ldots; t_{n-1} = 0; t_n = k \tag{3}$$

Iar după $k+1$ adunări de 1:

$$t_1 = 0; t_2 = 0; \ldots; t_{n-2} = 0; t_{n-1} = 1; t_n = 0 \tag{4}$$

Cantitatea totală de apă se calculează folosind formula:

$$V = t_1 * c_1 + t_2 * c_2 + ... + t_n * c_n \tag{5}$$

După fiecare dintre cele k^n adunări cu 1 trebuie să verificăm care este volumul total (folosim formula (5)).

Pentru cerința 1 se numără încă o soluție și se trece la combinația următoare.

Pentru cerința 2 se verifică dacă combinația are mai puține operații și se reține minimul apoi se trece la combinația următoare.

Cantitatea totală de apă se calculează folosind formula: $V = t_1 * c_1 + t_2 * c_2 + ... + t_n * c_n$.

1.2 Soluții alternative

- Simularea tuturor combinațiilor posibile folosind $1, 2..9$ structuri repetitive incluse una în alta.
- Soluție de tip backtracking (care nu se încadrează în materia clasei a 6-a)

1.3 Pentru cerința 3

Se calculează suma S a oricăror p capacități succesive. Dacă V se divide la S atunci am detectat o soluție posibilă. Dintre toate soluțiile găsite o reținem și afișăm pe cea care folosește numărul minim de găleți.

Pentru calcularea lui S se pot folosi următoarele metode:

- Se fixează capătul stânga al unui interval de lungime p și se face suma elementelor. (scor obținut 66% puncte pe cerința 3)
- Se calculează suma primelor p elemente, iar apoi se mișcă intervalul câte un singur pas la dreapta. Pentru un intervalul $[st, dr]$ se preia S-ul de la intervalul precedent $[st-1, dr-1]$, se scade elementul eliminat $c[st-1]$ și se adaugă elementul nou introdus $c[dr]$ (scor obținut 100% puncte pe cerința 3)
- Se folosesc sume parțiale pe prefixe. Notăm cu $partial[i] = c[1] + c[2] + .. + c[i]$. Suma elementelor pe intervalul $[st, dr] = partial[dr] - partial[st-1]$ (scor obținut 100% puncte pe cerința 3)

1.4 Observații

Problema testează cunoștințe despre:

- lucrul cu tablouri unidimensionale
- adunare pe vector într-o bază de numerație
- lucrul cu baze de numerație
- determinare minim
- tehnica algoritm de tip succesor
- sume parțiale pe vector

2 Problema Păsări

Propunător: Prof. Daniela Lica, Centrul Județean de Excelență Prahova, Ploiești

2.1 Notații:

- $H[i][j]$ $(1 \leq i \leq N, 1 \leq j \leq N)$ = înălțimea pomului amplasat pe linia i și coloana j. Matricea $H[][]$ este citită din fișierul de intrare;
- $Nord[i][j] = x$ $(1 \leq i \leq N, 1 \leq j \leq N, x < i) \Leftrightarrow x$ este indicele maxim al unei linii, astfel încât $H[x][j] > H[i][j]$; dacă nu există niciun astfel de x, se consideră că $Nord[i][j] = 0$. Mai informal spus, x reprezintă indicele liniei pomului de pe coloana j care va întrerupe supravegherea, pe direcția $Nord$, generată de un sistem situat în pomul de pe linia i și coloana j (adică, primul pom strict mai înalt decât acesta, în sus);
- $Sud[i][j] = x$ $(1 \leq i \leq N, 1 \leq j \leq N, x > i) \Leftrightarrow x$ este indicele minim al unei linii, astfel încât $H[x][j] > H[i][j]$; dacă nu există niciun astfel de x, se consideră că $Sud[i][j] = N+1$. Mai informal spus, x reprezintă indicele liniei pomului de pe coloana j care vă întrerupe supravegherea, pe direcția Sud, generată de un sistem situat în pomul de pe linia i și coloana j (adică, primul pom strict mai înalt decât acesta, în jos);
- $Vest[i][j] = y$ $(1 \leq i \leq N, 1 \leq j \leq N, y < j) \Leftrightarrow y$ este indicele maxim al unei coloane, astfel încât $H[i][y] > H[i][j]$; dacă nu există niciun astfel de y, se consideră că $Vest[i][j] = 0$. Mai informal spus, y reprezintă indicele liniei pomului de pe linia i care va întrerupe supravegherea, pe direcția $Vest$, generată de un sistem situat în pomul de pe linia i și coloana j (adică, primul pom strict mai înalt decât acesta, la stânga);
- $Est[i][j] = y$ $(1 \leq i \leq N, 1 \leq j \leq N, y > j) \Leftrightarrow y$ este indicele minim al unei coloane, astfel încât $H[i][y] > H[i][j]$; dacă nu există niciun astfel de y, se consideră că $Est[i][j] = N + 1$. Mai informal spus, y reprezintă indicele liniei pomului de pe linia i care va întrerupe supravegherea, pe direcția Est, generată de un sistem situat în pomul de pe linia i și coloana j (adică, primul pom strict mai înalt decât acesta, la dreapta);

2.2 Cerința 1 - $C = 1$ (30 de puncte)

- În cadrul acestei cerințe, avem la dispoziție un singur dispozitiv de supraveghere, pe care dorim să îl amplasăm într-un pom situat pe linia L, astfel încât numărul pomilor supravegheați de acesta să fie maxim.

- Putem spune că un dispozitiv amplasat în pomul de pe linia i și coloana j $(1 \leq i \leq N, 1 \leq j \leq N)$ va supraveghea un număr de pomi (inclusiv pomul de pe linia i și coloana j) egal cu $Extindere[i][j]$, după cum arată formula următoare:

  $Extindere[i][j] = (Sud[i][j] - Nord[i][j] - 1) + (Est[i][j] - Vest[i][j] - 1) - 1$

- Prin urmare, pentru a rezolva corect cerința, trebuie să identificăm cel mai mic indice al unei coloane j, pentru care $Extindere[L][j]$ este maxim, după cum urmează:

```
int ans = 0, Max = 0;

for(int j = 1; j <= N; ++j)
    if(Extindere[L][j] > Max)
        Max = Extindere[L][j], ans = j;

afișează ans;
```

- Având în vedere că sistemul de supraveghere trebuie amplasat pe linia L, primită în cadrul datelor de intrare, înseamnă că ne vor interesa doar valorile de pe linia L din matricele $Nord[][]$, $Sud[][]$, $Vest[][]$, $Est[][]$, și implicit $Extindere[][]$.

- Valorile $Nord[L][j]$, $Sud[L][j]$, $Vest[L][j]$, $Est[L][j]$ $(1 \leq j \leq N)$, $j-fixat$ se pot calcula în complexitatea de timp $O(N)$, utilizând o structură repetitivă, iterând un indice până când am întâlnit un pom mai înalt decât cel curent (linia L, coloana j), sau am ieșit din matrice.

- Pentru a calcula $Nord[L][j]$ și $Sud[L][j]$ pentru un $j-fixat$ $(1 \leq j \leq N)$, putem proceda astfel:

```
Nord[L][j] = 0, Sud[L][j] = N + 1;

for(int x = L - 1; x >= 1; --x)
    if(H[x][j] > H[L][j])
    {
        Nord[L][j] = x;

        break;
    }

for(int x = L + 1; x <= N; ++x)
    if(H[x][j] > H[L][j])
    {
```

```
            Sud[L][j] = x;

            break;
        }
```

- În mod similar, se pot calcula valorile $Vest[L][j]$ și $Est[L][j]$ pentru un $j - fixat$ $(1 \leq j \leq N)$.
- O soluție care are complexitatea totală de timp $O(N \cdot N)$ obține punctajul maxim pentru $C = 1$.

2.3 Cerința 2 - $C = 2$ (70 de puncte)

- În cadrul acestei cerințe, se știe faptul că în fiecare pom dintre cei $N \cdot N$ este amplasat câte un dispozitiv de supraveghere. Așadar, pentru fiecare linie i și coloană j $(1 \leq i \leq N, 1 \leq j \leq N)$ avem că: $Extindere[i][j] \geq 1$ (dispozitivul va supraveghea, cu siguranță, cel puțin un pom, și anume pomul de pe linia i și coloana j).
- Se cere să se afle care sunt *coordonatele* (adică linia și coloana) pomului care este supravegheat de cele mai multe dispozitive.
- Dispozitivul de supraveghere amplasat în pomul de pe linia i și coloana j $(1 \leq i \leq N, 1 \leq j \leq N)$ va supraveghea pomii:
  - pe ***verticală*** (direcția $Nord$ - Sud):
    - $(Nord[i][j] + 1, j)$;
    - $(Nord[i][j] + 2, j)$;
    - ...
    - $(i - 1, j)$;
    - $(i + 1, j)$;
    - ...
    - $(Sud[i][j] - 2, j)$;
    - $(Sud[i][j] - 1, j)$.
  - pe ***orizontală*** (direcția $Vest$ - Est):

$$(i, Vest[i][j] + 1); (i, Vest[i][j] + 2); \ldots; (i, j - 1); (i, j + 1); \ldots; (i, Est[i][j] - 2); (i, Est[i][j] - 1) \tag{6}$$

  - în care este amplasat: (i, j).
- Așadar, pentru fiecare pom, vom introduce notația: $cnt[i][j] = V \Leftrightarrow$ pomul situat pe linia i și coloana j va fi supravegheat de V dispozitive

$(1 \le i \le N, 1 \le j \le N)$. Răspunsul va fi dat de indicii liniei x, respectiv coloanei y pentru care valoarea $cnt[x][y]$ este maximă, după cum urmează:

```
int ans_i = 0, ans_j = 0, Max = 0;

for(int x = 1; x <= N; ++x)
    for(int y = 1; y <= N; ++y)
        if(cnt[x][y] > Max)
            Max = cnt[x][y], ans_i = x, ans_j = y;

afișează ans_i ans_j;
```

- O soluție care are complexitatea de timp $O(N^3)$, asemănătoare cu cea de la cerința 1, prin care se utilizează $O(N)$ operații pentru fiecare dintre cele $N * N$ dispozitive, și marchează corespunzător modificările în matricea $cnt[][]$, nu va reuși să obțină punctajul maxim, întrucât va depăși considerabil limita de timp alocată.

- Pentru a obține punctajul maxim, vom alege să calculăm într-un mod mai eficient matricele $Nord[][]$, $Sud[][]$, $Vest[][]$, $Est[][]$. Ne propunem ca fiecare linie din aceste matrice să fie corect calculată folosind doar $O(N)$ pași, reducând astfel, complexitatea totală la $O(N \cdot N)$.

- Pentru a calcula cum va arăta linia i din matricea $Est[][]$, vom folosi un vector auxiliar, procedând astfel:

```
int K = 0;

for(int j = 1; j <= N; ++j)
{
    while(K && H[i][j] > H[i][V[K]])
        Est[i][V[K]] = j, --K;

    V[++K] = j;
}

for(int j = 1; j <= K; ++j)
    Est[i][V[j]] = (N + 1);
```

- Pentru a nu degenera complexitatea totală $O(N*N)$, marcarea pomilor supravegheați de dispozitive se va face utilizând *Șmenul lui Mars*.

3 Problema Puternic

Propunător: Prof. Ana-Maria Arișanu, Colegiul Național "Mircea cel Bătrân", Rm. Vâlcea

3.1 Cerința 1

Descompunem în factori primi fiecare număr **mai mare decât 1** din șir pentru a verifica dacă este puternic.

Dacă prin descompunere obținem un factor prim la puterea 1, numărul respectiv nu e puternic.

3.2 Cerința 2

Memorăm într-un vector numerele din șir care nu sunt puternice.

Concatenăm numerele egal depărtate de capetele noului șir și verificăm dacă numărul obținut prin concatenare este puternic. Numerele din șirul inițial sunt $\leq 10^9$, prin urmare în urma concatenării se obțin numere $\leq 10^{18}$ (long long).

Pentru a verifica eficient dacă un număr x este puternic ținem cont de următoarele observații matematice:

- $4000^5 = 1.024.000.000.000.000.000 \approx 10^{18}$
- numărul x trebuie împărțit, de câte ori este posibil, la toate numere prime mai mici sau egale cu 4000. Dacă în timpul acestei descompuneri obținem un factor prim la putere 1, atunci numărul x nu este puternic
- După împărțiri rămânem cu x fie 1 (și e puternic), fie produs de numere prime **mai mari decât 4000**. Mai exact x poate fi produs de până la 4 numere prime mai mari ca 4000:
  - dacă este număr prim NU poate fi puternic
  - dacă este produs de 4 numere prime, pentru a fi puternic, poate fi de forma p^4 sau $p^2 * q^2$ (adică pătrat perfect, indiferent de formă)
  - daca e produs de 3 numere prime pentru a fi puternic trebuie sa fie de forma p^3 (trebuie sa se divida cu p si cu p^2).
  - daca e produs de 2 numere prime pentru a fi puternic trebuie sa aibă forma p^2

Putem folosi Ciurul lui Eratostene până la 4000, pentru a verifica dacă un număr obținut prin concatenare este puternic.

Pentru a verifica dacă numărul rămas este pătrat perfect sau cub perfect putem folosi o căutare binara ori biblioteca *cmath* și calcula direct rădăcina

pătrată (sqrt) și radical de ordin 3(cbrt) (atenție la erorile de precizie, se poate că uneori $sqrt(16) = 3.99999$).

Echipa. Setul de probleme a fost pregătit de:

- Prof. Daniela Lica, Centrul Județean de Excelență Prahova, Ploiești
- Prof. Cristina Iordaiche, Liceul Teoretic "Grigore Moisil", Timișoara
- Prof. Ana-Maria Arișanu, Colegiul Național "Mircea cel Bătrân", Rm. Vâlcea
- Prof. Roxana Tîmplaru, Liceul Tehnolologic "Ștefan Odobleja"/ ISJ Dolj
- Prof. Cristina Anton, Colegiul Național "Gheorghe Munteanu Murgoci", Brăila
- Prof. Marinel Șerban, Colegiul Național "Emil Racoviță", Iași
- Student Radu Muntean, ETH Zurich
- Student Stelian Chichirim, Universitatea București