OLIMPIADA NAȚIONALĂ DE INFORMATICĂ
PROBA DE BARAJ
DESCRIEREA SOLUȚIILOR

COMISIA ȘTIINȚIFICĂ

”Soldații urcă în tramvai. Ein, zwei, drei... Ein, zwei, drei...” (Nistor Moț)

Problema 1: Drei

Propusă de: prof. Ionel-Vasile Piț-Rada, Colegiul Național ”Traian”, Drobeta Turnu Severin

Observație inițială: fiecare cuvânt de o singură literă corespunde puterii 3^p, unde p este poziția literei în alfabet (începând de la 0): $p_a = 0, p_b = 1, \ldots, p_z = 25$.

Notăm cu literă(p) a p-a literă din alfabet (începând de la 0): literă(0) = a, literă(1) = b, ..., literă(25) = z. În plus, notăm cu $||$ operația de concatenare între două șiruri de caractere (de exemplu: $a||b = ab$).

Termenul al n-lea, DREI(n), se determină unic astfel:

(1) Dacă n este putere a lui 3 (n se poate scrie sub forma 3^p), atunci DREI(n) = literă(p).
(2) Dacă n nu este putere a lui 3, atunci n se află între două puteri consecutive ($3^p < n < 3^{p+1}$) și, relativ la distanța față de puterea 3^{p+1}, avem două cazuri:
   (a) Dacă $2 * n > 3^{p+1}$, atunci îl vom obține pe DREI(n) scăzând din 3^{p+1} diferența $d_{right} = 3^{p+1} - n$, adică vom așeza scrierea DREI(d_{right}) în stânga literei literă($p+1$), deci DREI(n) = DREI(d_{right}) $||$ literă($p+1$)
   (b) Dacă $2 * n < 3^{p+1}$, atunci îl vom obține pe DREI(n) adunând la 3^p diferența $d_{left} = n - 3^p$ și vom așeza scrierea DREI(d_{left}) la dreapta literei literă(p), deci DREI(n) = literă(p) $||$ DREI(d_{left})

Exemplu: Vrem să aflăm DREI(59):

- Deoarece $27 < 59 < 81$ suntem în cazul 2-a). Obținem $d_{right} = 22$ și pentru DREI(59) așezăm scrierea DREI(22) în stânga literei literă(4), astfel:
$$\text{DREI}(59) = \text{DREI}(22) \,||\, \text{literă}(4) = \text{DREI}(22) \,||\, e$$

- Deoarece $9 < 22 < 27$ suntem tot în cazul 2-a). Obținem $d_{right} = 5$ și pentru DREI(22) așezăm scrierea DREI(5) în stânga literei literă(3), astfel:
$$\text{DREI}(22) = \text{DREI}(5) \,||\, \text{literă}(3) = \text{DREI}(5) \,||\, d$$

- Deoarece $3 < 5 < 9$ suntem tot în cazul 2-a). Obținem $d_{right} = 4$ și pentru DREI(5) așezăm scrierea DREI(4) în stânga literei literă(2), astfel:
$$\text{DREI}(5) = \text{DREI}(4) \,||\, \text{literă}(2) = \text{DREI}(4) \,||\, c$$

- Deoarece $3 < 4 < 9$ suntem în cazul 2-b). Obținem $d_{left} = 1$ și pentru DREI(4) așezăm scrierea DREI(1) în dreapta literei literă(1), astfel:
$$\text{DREI}(4) = \text{literă}(1) \,||\, \text{DREI}(1) = b \,||\, \text{DREI}(1)$$

- Deoarece $1 = 3^0$ suntem în cazul 1. Deci: DREI(1) = literă(0) = a. Astfel obținem:

$$\text{DREI}(1) = a$$
$$\text{DREI}(4) = b \,||\, \text{DREI}(1) = ba$$

$$\text{DREI}(5) = \text{DREI}(4) \;||\; c = bac$$
$$\text{DREI}(22) = \text{DREI}(5) \;||\; d = bacd$$
$$\text{DREI}(59) = \text{DREI}(22) \;||\; e = bacde$$

Interogare de tip 1 x: să presupunem că șirul x are literele $x[0], x[1], x[2], \ldots, x[r-1]$.

Observăm că o secvență DREI nu poate conține aceeași literă de mai multe ori, iar litera maximă trebuie sa se afle într-unul dintre capete. Așa că, pornim o verificare simultan de la stânga și de la dreapta, verificând restricțiile de mai sus, iar apoi eliminând litera cea mai mare.

Algorithm 1: Verificarea corectitudinii unui șir DREI

```
p = 0, q = r − 1;
Cât timp p + 1 ≤ q execută:
    Dacă x[p] ≤ x[p + 1] sau x[q − 1] ≥ x[q] atunci
        returnează false;   // Litera maximă nu este la margini
    Dacă x[p] = x[q] atunci
        returnează false;   // Literă duplicată
    Altfel, dacă x[p] > x[q] atunci
        p = p + 1;
    Altfel
        q = q + 1;
returnează true
```

Interogare de tip 2 x y: vom calcula în variabila *rez* rezultatul comparației dintre x și y.

- Dacă $rez = -1$, atunci $x < y$
- Dacă $rez = 0$, atunci $x = y$
- Dacă $rez = 1$, atunci $x > y$;

De asemenea, vom nota cu $max(x)$ cea mai mare literă din reprezentarea DREI x, iar cu $lungime(x)$ lungimea reprezentării DREI x.

Algorithm 2: Compararea a două reprezentări DREI x și y

```
rez = −INF; // Cât timp rez este setat pe −INF, atunci nu știm rezultatul.
Dacă x = y atunci
    rez = 0;
Altfel
    Cât timp rez = −INF execută:
        Dacă max(x) < max(y) atunci
            rez = −1;   // x nu conține cea mai mare literă
        Altfel, dacă max(x) > max(y) atunci
            rez = 1;    // x conține cea mai mare literă
        // Dacă se ajunge în acest punct, înseamnă că x și y au aceeași literă maximă.
        lx = lungime(x);
        ly = lungime(y);
        Dacă x[0] = max(a) și y[ly − 1] = max(y) atunci
            rez = 1;    // x are litera maximă la stânga, iar y nu
        Altfel, dacă x[lx − 1] = max(x) și y[0] = max(y) atunci
            rez = −1;   // x are litera maximă la dreapta, iar y nu
        // Dacă se ajunge în acest punct, înseamnă ca litera maximă se afla pe aceeași
           poziție (prima sau ultima) în x și y. Vom elimina litera din ambele reprezentări
           și comparăm ce rămâne.
        Șterg max(x) din x
        Șterg max(y) din y
```

Alternativ, se poate răspunde la această întrebare folosind algoritmul descris mai jos pentru interogările de tip 4, prin compararea numerelor de ordine asociate fiecărei reprezentări DREI.

Interogare de tip 3 N: Pe parcursul determinării literelor reprezentării DREI vom adăuga unui vector V perechi de forma (literă, direcție), unde direcție arată dacă litera va fi inserată la începutul (-1) sau la sfârșitul$(+1)$ șirului x.

Plecăm de la exemplul descris mai sus – DREI$(59) = bacde$:

$$\begin{gathered}\text{DREI}(59) = \text{DREI}(22) \mid\mid e \\ \text{DREI}(22) = \text{DREI}(5) \mid\mid d \\ \text{DREI}(5) = \text{DREI}(4) \mid\mid c \\ \text{DREI}(4) = b \mid\mid \text{DREI}(1) \\ \text{DREI}(1) = a\end{gathered}$$

.

Adăugăm în vector perechile: $(e,1)$, $(d,1)$, $(c,1)$ și $(b,-1)$.

Algorithm 3: Determinarea reprezentării DREI asociate numărului N

Cât timp N nu este putere a lui 3 execută:
- calculăm p; // Avem $3^p < N < 3^{p+1}$
- Dacă $2 * N > 3^{p+1}$ atunci
  - adăugăm în vectorul V perechea $(p+1,1)$;
  - înlocuim N cu $3^{p+1} - N$;
- Altfel, dacă $2 * N <= 3^{p+1}$ atunci
  - adăugăm în vectorul V perechea $(p,-1)$;
  - înlocuim N cu $N - 3^p$;

Avem $N = 3^p$. Inserăm în x litera literă(p) apoi parcurgem în ordinea inversă în care au fost adăugate perechile în V și dacă avem perechea (literă, direcție) în funcție de direcție adăugăm litera la începutul sau sfârșitul șirului x format până la momentul respectiv.

Alternativ, reconstituirea soluției se poate face folosind o coadă pentru literele din stânga, respectiv o stivă pentru literele din dreapta.

Interogare de tip 4 x: vom folosi și aici un vector V pentru reconstituirea soluției.

Algorithm 4: Determinarea reprezentării DREI asociate numărului N

Se calculează $p = max(x) - a$;
Se inițializează Nr cu 3^p;
Cât timp $lungime(x) > 1$ execută:
- se calculează $p = max(x) - a$; Dacă $x[0] = max(x)$ atunci
  - se adaugă în V perechea $(3^p,1)$;
- Altfel
  - se adaugă în V perechea $(3^p,-1)$;
- se șterge $max(x)$ din x;

Se parcurg în sens invers perechile din V și pentru fiecare pereche se adaugă sau se scade puterea curentă din puterea perechii anterioare, la final se adună/scade ultimul rezultat la/din Nr.

Alternativ, se poate implementa recursiv: (de exemplu, ID_DREI$(abcd) = 27 -$ ID_DREI(abc)).

Problema 2: Soldați

Propusă de: Prof. Marinel Șerban, Colegiul Național "Emil Racoviță", Iaşi
Prof. Adrian Panaete, Colegiul Naţional "August Treboniu Laurian", Botoşani

Prima observație: pentru rezolvarea problemei putem trata separat deplasările celor S soldați pe cele două direcții obținând astfel două probleme.

Problema 1: Alinierea soldaților pe o linie LIN

Considerăm soldații sortați în ordine crescătoare după linie ($L_1 \leq L_2 \leq \cdots \leq L_s$). Formal, dorim să determinăm o linie LIN pentru care valoarea $|LIN - L_1| + |LIN - L_2| + |LIN - L_s|$ să fie minimă.

Problema 2. : Alinierea soldaților pe linia LIN începând cu o coloană COL.

Considerăm soldații sortați în ordine crescătoare după coloană $C_1 \leq C_2 \leq \cdots <= C_S$. Vrem să așezăm soldații pe coloanele $COL, COL+1, \ldots, COL+S-1$. Observăm inițial că așezarea Greedy (primul din stânga se deplasează pe coloana COL, al doilea pe coloana $COL+1$, ..., ultimul pe coloana $COL+S-1$) este optimă pentru o alegere fixată a valorii COL.

În acest context vrem să alegem COL astfel încât valoarea: $|COL - C_1| + |COL + 1 - C_2| + |COL + 2 - C_3| + \cdots + |COL + S - 1 - C_S|$ să fie minimă. Se observă că această problemă este similară cu Problema 1, ca și cum am vrea să aducem pe linia $LIN = COL$ soldați situați la liniile $C_1, C_2 - 1, \ldots, C_S - S + 1$.

Pentru cerința 1 se fixează $COL = 1$.

Pentru cerința 2 trebuie să avem grijă la două situații speciale:

- Dacă aplicând algoritmul de la problema 1 obținem $COL < 1$, trebuie să alegem $COL = 1$. În acest caz algoritmul ne-ar arăta că alinierea optimă s-ar realiza dacă șirul de soldați „ar ieși din cazarmă prin partea stângă".
- Dacă aplicând algoritmul de la problema 1 obținem $COL + S - 1 > m$ trebuie să alegem COL astfel încât $COL + S - 1 = m$, adică să alegem $COL = m - S + 1$. În acest caz, algoritmul ne-ar arăta că alinierea optimă s-ar realiza dacă șirul de soldați „ar ieși din cazarmă prin partea dreaptă".

Astfel, am redus problema 2 la problema 1. Ne ocupăm în continuare de rezolvarea problemei 1. Determinăm LIN astfel încât : $DIST(LIN) = |LIN - L_1| + |LIN - L_2| + \cdots + |LIN - L_S|$ are valoare minimă.

Rezultat ajutător: Fie $a \leq b$. Atunci $|X - a| + |X - b| \geq |(X - a) - (X - b)| = |b - a| = b - a$. Egalitatea are loc dacă $a \leq X \leq b$.

În $DIST(LIN)$ grupăm: primul termen cu ultimul, al doilea cu penultimul și așa mai departe.
$|LIN - L_1| + |LIN - L_S| >= L_S - L_1$ cu egalitate pentru $L_1 \leq LIN \leq L_S$
$|LIN - L_2| + |LIN - L_{S-1}| >= L_{S-1} - L_2$ cu egalitate pentru $L_2 \leq LIN \leq L_S - 1$
...
Și așa mai departe.

Dacă S este par ($S = 2 \cdot k$) atunci $DIST(LIN) \geq L_S - L_1 + L_{S-1} - L_2 + \cdots + L_{k+1} - L_k$ și egalitatea se obține când $L_k \leq LIN \leq L_{k+1}$. Deoarece vrem ca LIN să fie minimă se alege $LIN = L_k$.

Dacă S este impar ($S = 2 \cdot k + 1$) atunci $DIST(LIN) \geq L_S - L_1 + L_{S-1} - L_2 + \cdots + L_k - L_{k+1} + |L - L_{k+1}| \leq L_S - L_1 + L_{S-1} - L2 + \cdots + L_k - L_{k+1}$ cu egalitate dacă sunt îndeplinite simultan condițiile $L_k \leq LIN \leq L_{k+2}$ și $LIN = L_{k+1}$. Se observă că valoarea minimă se obține numai dacă $LIN = L_{k+1}$.

Cele două cazuri ne conduc la rezultatul unic $LIN = L_{(S+1)/2}$, adică LIN trebuie ales mediana șirului.

Problema 3: Tramvaie

Propusă de: Stud. Dumitru Ilie, Facultatea de Matematică-Informatică, Universitatea București

Pentru a determină distanța minimă care poate fi parcursă pe jos între două puncte de coordonate (x_0,y_0) și (x_1,y_1) analizăm două cazuri posibile, minimul dintre distanțele determinate pe cele două cazuri fiind rezultatul final.

Cazul 1. Pentru deplasare nu se folosesc tramvaie. În acest caz răspunsul este $|x_1 - x_0| + |y_1 - y_0|$. Această formulă este cunoscută drept distanța Manhattan.

Cazul 2. Pentru deplasare se folosesc tramvaie. În acest caz trebuie să determinăm distanța de la punctul de coordonate (x_0,y_0) la cel mai apropiat tramvai, respectiv distanța de la punctul de coordonate (x_1,y_1) la cel mai apropiat tramvai, suma acestor două distanțe fiind distanța totală parcursă pe jos între cele două puncte.

Pentru a determina distanța de la un punct la cel mai apropiat tramvai, determinăm distanța minimă de la punctul respectiv la o linie de tramvai paralelă cu Ox, respectiv distanța minimă de la punctul respectiv la o linie de tramvai paralelă cu Oy, rezultatul fiind minimul dintre cele două distanțe.

Pentru aceasta vom sorta liniile de tramvai paralele cu Ox, respectiv liniile de tramvai paralele cu Oy. Determinarea celei mai apropiate linii de tramvai (paralelă cu Ox, respectiv cu Oy) se poate face prin căutare binară. Complexitatea finală este $O((N + Q) \cdot \log_2 N + (M + Q) \cdot \log_2 M)$ datorată sortărilor și căutărilor binare.

O altă variantă posibilă ar fi să sortăm și punctele după abscisă, respectiv după ordonată și să determinăm determinăm cea mai apropiată linie de tramvai paralelă cu Ox, respectiv cu Oy printr-un algoritm similar cu algoritmul de interclasare. Complexitatea acestei soluții este $O(N \cdot \log_2 N + M \cdot \log_2 M + Q \cdot \log_2 Q)$, datorată sortărilor.

Echipa

Problemele pentru această etapă au fost pregătite de:

- Prof. Emanuela Cerchez, Colegiul Național "Emil Racoviță" Iași
- Prof. Adrian Panaete, Colegiul Naţional "August Treboniu Laurian", Botoşani
- Prof. Ionel-Vasile Piț-Rada, Colegiul Naţional "Traian", Drobeta Turnu Severin
- Stud. Dumitru Ilie, Facultatea de Matematică-Informatică, Universitatea București
- Prof. Marinel Șerban, Colegiul Național "Emil Racoviță", Iaşi
- Prof. Flavius Boian, Colegiul Național "Spiru Haret", Târgu-Jiu
- lector dr. Paul Diac, Facultatea de Informatică, Universitatea "Alexandru Ioan Cuza" Iaşi
- Prof. Claudiu-Cristian Gorea-Zamfir, Liceul de Informatică "Grigore Moisil" Iași
- Stud. Ioan-Cristian Pop, Universitatea Politehnica București
- Prof. Nistor Moț, Școala "Dr. Luca" Brăila
- Prof. Dan Pracsiu, Liceul Teoretic "Emil Racoviță", Vaslui
- Prof. Isabela Patricia Coman, Colegiul Naţional de informatică"Tudor Vianu", Bucureşti
- Stud. Bogdan Ioan Popa, Facultatea de Matematică și Informatică, Universitatea București
- Stud. Livia Măgureanu, Facultatea de Limbi şi Literaturi Străine, Universitatea Bucureşti
- Stud. Giulian Buzatu, Facultatea de Matematică-Informatică, Universitatea București
- ing. Theodor-Gabriel Tulbă-Lecu, Jane Street, Londra