

**CNER CODE, IAȘI**  
**CLASA A X-A**  
**DESCRIEREA SOLUȚIILOR**

COMISIA ȘTIINȚIFICĂ

**PROBLEMA A: ENIGMA OTILIEI**

*Propusă de: Andrei Boacă, Colegiul Național "Emil Racoviță", Iași*

**Subtask-ul 1.** Pentru acest subtask este suficientă generarea tuturor drumurilor folosind o metodă de tip backtracking.

**Subtask-urile 2 și 4.** În continuare vom folosi metoda Programării dinamice.

Fie  $dp[i][j]$  numărul de moduri în care calul poate ajunge în celula  $(i, j)$ .

Observăm că în celula  $(i, j)$  se poate ajunge din celulele  $(i - 1, j), (i - 2, j), \dots, (i - K, j), (i, j - 1), (i, j - 2), \dots, (i, j - K)$ .

Deci,  $dp[i][j] = dp[i - 1][j] + dp[i - 2][j] + \dots + dp[i - K][j] + dp[i][j - 1] + dp[i][j - 2] + \dots + dp[i][j - K]$ .

Este de remarcat faptul că o celulă în care se află o groapă va avea  $dp[i][j] = 0$ .

Complexitate teoretică:  $O(N * M * K)$

**Subtask-urile 3 și 5.** Pentru a rezolva problema atunci când  $K$  este mai mare, vom folosi sumele parțiale.

Fie  $lin[i][j] = dp[i][1] + dp[i][2] + \dots + dp[i][j]$  (sumele parțiale pe fiecare linie), respectiv  $col[i][j] = dp[1][j] + dp[2][j] + \dots + dp[i][j]$ .

Acum,  $dp[i][j]$  se poate scrie ca  $lin[i][j - 1] - lin[i][j - k - 1] + col[i - 1][j] - col[i - k - 1][j]$ .

Matricile  $lin$  și  $col$  pot fi actualizate în timp ce calculăm dinamica.

Trebuie avut grijă la cazurile în care  $j - k - 1$  sau  $i - k - 1$  sunt mai mici decât 0. În acel caz ele nu se mai scad.

Complexitate teoretică:  $O(N * M)$

**PROBLEMA B: HARAP ALB**

*Propusă de: Răileanu Alin-Gabriel, Colegiul Național "Emil Racoviță", Iași*

**Subtask-urile 1 și 5.**  $1 \leq N \leq 100$

Pentru acest subtask se pot utiliza 3 for-uri. Primele 2 setează capetele  $L - 1, R + 1$  ale subsecvenței, iar cel de-al treilea va parcurge subsecvența pentru a verifica dacă toate valorile corespunzătoare indicilor dintre  $L$  și  $R$  sunt mai mari decât valorile  $A[L - 1], A[R + 1]$ .

Pentru cerința 1 se va afișa maximul diferenței dintre fiecare  $R$  și  $L$  ce delimitează o secvență validă, iar pentru cerința 2 se poate utiliza o variabilă *ans* ce va fi incrementată pentru fiecare pereche de indici ce delimitează o secvență validă.

Complexitate teoretică:  $O(N^3)$ .

**Subtask-urile 2 și 6.**  $101 \leq N \leq 5000$ 

Pentru acest subtask observăm că nu este necesară setarea ambilor indici. Putem seta doar indicele  $L - 1$  din stânga, iar în timp ce parcurgem valorile corespunzătoare indicilor dintre  $L$  și  $N$ , putem reține minimul actual într-o variabilă *minim*, inițial egală cu *infinit*. Pentru fiecare indice  $i$ , dintre  $L$  și  $N$  care respectă condiția  $A[i] < \text{minim}$ , în timp ce se respectă și condiția  $A[L - 1] < \text{minim}$ , putem considera secvența delimitată de indicii  $L$  și  $i$  ca fiind validă, iar apoi variabila *minim* va lua valoarea  $\min(\text{minim}, A[i])$ .

Se poate observa că dacă valoarea variabilei *minim* ajunge să fie mai mică sau egală decât valoarea lui  $A[L - 1]$  într-un punct, nu mai este necesară continuarea parcurgerii elementelor până la  $N$ . Cu toate acestea, optimizarea prezentată funcționează bine doar pentru unele variante de input, așadar complexitatea teoretică va rămâne aceeași.

Răspunsurile pentru cele 2 cerințe se calculează analog explicației ce la subtask-urile 1 și 5.

Complexitate teoretică:  $O(N^2)$ .

**Subtask-urile 3, 4, 7, 8.**  $5001 \leq N \leq 100000$  și  $100001 \leq N \leq 1000000$ 

Departajarea acestor subtask-uri este făcută prin mici detalii de implementare.

Ideea de rezolvare pentru această problemă se bazează pe câteva observații:

- (1) Pentru fiecare subsecvență ce respectă proprietatea dată în enunț va exista cel puțin un indice  $i$  a cărui valoare asociată este minimă pentru acea subsecvență, iar indicii  $L - 1$  și  $R + 1$  ce delimitează subsecvența au proprietatea că sunt:
  - $L - 1$  = cel mai mare indice strict mai mic decât  $i$  cu proprietatea  $A[L - 1] < A[i]^*$ ;
  - $R + 1$  = cel mai mic indice strict mai mare decât  $i$  cu proprietatea  $A[R + 1] < A[i]^{**}$ ;
- (2) Din  $*$  și  $^{**} \Rightarrow$  există maxim  $N - 2$  secvențe care respectă proprietatea din enunț ( $N$ -dimensiunea șirului).

În continuare, pentru simplitate, vom utiliza un vector  $v$  de tip struct cu 2 câmpuri:  $st$  și  $dr$ , ce va avea proprietatea:

- $v[i].st$  = cel mai mare indice strict mai mic decât  $i$  cu proprietatea  $A[v[i].st] < A[i]$ ;
- $v[i].dr$  = cel mai mic indice strict mai mare decât  $i$  cu proprietatea  $A[v[i].dr] < A[i]$ ;

Pentru a descoperi într-o complexitate bună care sunt  $v[i].st$  și  $v[i].dr$  pentru fiecare  $i$  de la 2 la  $N - 1$  putem utiliza o stivă.

În primă fază vom parcurge vectorul  $a$  de la stânga la dreapta punând în stivă inițial elementul cu indicele 1, iar apoi de la dreapta la stânga punând în stivă inițial elementul cu indicele  $N$ .

Când suntem la un indice  $i$ , acesta va elimina din stivă elemente cât timp stiva nu este goală, iar valoarea corespunzătoare indicelui reținut în vârful stivei este strict mai mare decât  $A[i]$ .

În momentul în care indicele  $i$  scoate din stivă un alt indice  $j$ , indicele  $i$  devine pentru  $j$ :

- $v[j].dr$ , dacă parcurgerea este efectuată de la stânga la dreapta;
- $v[i].st$ , dacă parcurgerea este efectuată de la dreapta la stânga;

După ce indicele  $i$  nu mai poate elimina elemente din stivă, îl introducem pe acesta în vârful stivei și continuăm procesul cu următorul indice.

Pentru cerința 1 este suficient să afișăm maximul diferenței  $v[i].dr - v[i].st$ .

Pentru cerința 2, acum că am rezolvat această parte din soluție, este necesar să parcurgem perechile de indici și să numărăm câte subevenimente diferite sunt delimitate de acestea.

Putem rezolva partea aceasta sortând vectorul de tip struct. După ce l-am sortat va trebui să-l parcurgem și să incrementăm variabila *ans* doar când valorile asociate unei perechi de indici vecini  $(i, i + 1)$  sunt diferite.  $(v[i].st \neq v[i + 1].st \vee v[i].dr \neq v[i + 1].dr)$  În final, răspunsul pentru cerința 2 se va găsi în variabila *ans*.

Numărarea perechilor diferite de indici din vectorul *v* se poate rezolva și în complexitate liniară folosind liste, dar această metodă depășește materia specifică clasei a 10-a.

Complexitate teoretică:  $O(N)$  sau  $O(N * \log(N))$  în funcție de implementare.

### PROBLEMA C: MOARA CU NOROC

*Propusă de: Răileanu Alin-Gabriel, Colegiul Național "Emil Racoviță", Iași*

#### Subtask-ul 1. $N, M \leq 4, C \leq 4$

Pentru acest subtask se poate utiliza o metodă de tip backtracking.

Pentru fiecare cal generăm toate drumurile din celula  $(1, 1)$ , în celula  $(n, m)$  și verificăm dacă există măcar un drum valid care să respecte condițiile din enunț.

#### Subtask-ul 2. $N, M \leq 700, C \leq 10$

Ținând cont de restricțiile semnificativ mai mari pentru dimensiunile matricei, observăm că un algoritm de tip backtracking nu mai este potrivit.

Acest subtask urmărește modificarea algoritmului lui Lee, pentru a optimiza costurile.

Pentru fiecare cal, vom verifica dacă pornind cu acesta din celula  $(1, 1)$ , putem ajunge în celula  $(n, m)$  utilizând o coadă și o matrice separată, numită *fm*, cu următoarea semnificație:

- $fm[i][j]$  = forța maximă pe care calul o poate avea când ajunge în celula  $(i, j)$ , plecând din celula  $(1, 1)$ ;
- $fm[i][j] = -1$ , dacă celula  $(i, j)$  nu poate fi accesată de cal, plecând din celula  $(1, 1)$  și respectând toate restricțiile precizate de enunț.

Pentru fiecare cal care are o forță suficient de mare cât să reziste unui drum din celula  $(1, 1)$ , în celula  $(n, m)$ , variabila *ans* va fi incrementată, iar la final, în aceasta se va găsi răspunsul cerut de problemă.

Complexitate:  $O(C * n * m * \max(n, m))$ , dar se comportă mult mai bine în practică.

#### Subtask-ul 3. $N, M \leq 700, C \leq 1000000$

Singura diferență dintre acest subtask și subtask-ul precedent este restricția pentru numărul de cai.

Pentru a putea rezolva problema acum, vom utiliza algoritmul lui Lee modificat pentru optimizarea costurilor, doar că observăm că nu mai este potrivită iterarea prin vectorul de forțe ale cailor.

Soluția se bazează pe sortarea acelui vector și utilizarea unei căutări binare pe rezultat.

Vom căuta binar pe vectorul sortat poziția minimă pentru care forța calului de pe acea poziție este suficient de mare pentru a putea parcurge un drum de la  $(1, 1)$  la  $(n, m)$  și vom reține această poziție în variabila  $poz$ .

Răspunsul problemei va fi  $C - poz + 1$ .

Complexitate:  $O(\log(C) * n * m * \max(n, m))$  în cazul cel mai rău, dar în practică se comportă mult mai bine.