

Soluții pentru problema *Medwalk*

Lotul Național de Informatică, Craiova 2025

Subtask 1 ($N \leq 7, Q \leq 1000$)

Operațiile de tipul 1 vor fi procesate trivial (doar modificăm valoarea celulei corespunzătoare din matrice).

Pentru fiecare operație de tipul 2, vom încerca brut toate cele $(dr - st + 1)!$ permutări posibile, iar pentru fiecare permutare posibilă, vom calcula mediana pentru toate cele $dr - st + 1$ drumuri posibile.

Răspunsul pentru o operație de tipul 2 va fi egală cu valoarea minimă a unei mediane obținute anterior.

Complexitate: $O(Q \cdot N! \cdot N^2 \cdot \log(N))$

Subtask 2 ($N, Q \leq 5000$)

Se poate observa că orice drum conține cel puțin o celulă din fiecare coloană a submatricei de query. Pe lângă aceste celule, orice drum conține încă o celulă suplimentară.

Așadar, putem vedea că cea mai mică valoare posibilă pentru mediană ar putea fi atinsă dacă am lua în drum celula minimă de pe fiecare coloană, împreună cu cea mai mică celulă care este maximă pe coloană.

Deoarece putem permuta oricum coloanele submatricei de query, un asemenea drum poate fi construit mereu în felul următor:

- Considerăm coloana i ($st \leq i \leq dr$) ca fiind coloana unde se află cea mai mică celulă care este maximă pe coloană.
- Toate coloanele unde minimul se află pe prima linie vor fi aduse înaintea coloanei i în permutarea finală, iar toate coloanele unde minimul se află pe a doua linie vor fi puse după coloana i în permutarea finală.
- Drumul care are mediana minimă va conține celulele de pe prima linie până la coloana i și celulele de pe a doua linie începând cu coloana i .

Astfel, operațiile de tipul 2 se reduc la a afla mediana șirului următor:

$$[\min(a_{1,st}, a_{1,st}), \min(a_{1,st+1}, a_{1,st+1}), \dots, \min(a_{1,dr}, a_{1,dr}), M]$$

unde $M = \min_{st \leq i \leq dr}(\max(a_{1,i}, a_{2,i}))$.

La fel ca la subtask-ul precedent, operațiile de tipul 1 vor fi procesate trivial (doar modificăm valoarea celulei corespunzătoare din matrice).

Pentru operațiile de tipul 2, este suficient să se calculeze mediana șirului menționat mai sus în $O(N \cdot \log N)$ per query.

Complexitate: $O(QN \cdot \log N)$

Subtask 3 (numerele din matrice ≤ 2 ; pentru toate operațiile de tipul 1, $x \leq 2$)

Bazându-ne pe ideea de la subtask-ul precedent, pentru fiecare operație de tipul 2, va trebui să calculăm următoarele valori:

- a - numărul de coloane $i \in [st, dr]$ pentru care $\min(a_{1,i}, a_{2,i}) = 1$.
- b - numărul de coloane $i \in [st, dr]$ pentru care $\max(a_{1,i}, a_{2,i}) = 1$.

Numărul maxim de 1 posibili dintr-un drum va fi egal cu $cnt_1 = a + [b > 0]$.

Dacă $cnt_1 \cdot 2 \geq dr - st + 2$, atunci răspunsul la query va fi 1. În caz contrar, răspunsul va fi 2.

Cea mai simplă implementare pentru acest subtask ar fi bazată pe două `ordered_set`-uri, unde primul va conține toate coloanele i unde $\min(a_{1,i}, a_{2,i}) = 1$, iar al doilea va conține toate coloanele i unde $\max(a_{1,i}, a_{2,i}) = 1$.

Notă: Printre altele, `ordered_set`-urile pot fi substituite cu arbori de intervale sau arbori indexați binar.

Complexitate: $O((N + Q)\log N)$

Subtask 4 (Nu există operații de tipul 1)

În urma observațiilor de la subtask-ul 2, problema pentru acest subtask se reduce aproximativ la determinarea medianei pentru Q subsecvențe ale unui șir dat.

Această problemă este clasică și se poate rezolva utilizând Arbori de intervale **persistenti** și căutare binară.

Vom reține în fiecare poziție din șir câte un arbore de intervale pe pointeri care să ne ajute să calculăm rapid răspunsul pentru întrebarea: Câte valori mai mici decât V se găsesc pe prefixul $[1, i]$ din șir?

Pentru fiecare query ne putem folosi de interogări de tipul celei precizate mai sus pentru a căuta binar valoarea medianei, utilizând conceptul de sume parțiale.

Pentru a ține cont și de valoarea M în fiecare query, una dintre soluții se bazează pe RMQ pentru determinarea ei, iar apoi în căutarea binară vom crește cu 1 răspunsul pentru verificare atunci când M se găsește în intervalul de valori interogată.

Complexitatea este $O(N \cdot \log(N) + Q \cdot \log(N)^2)$, dar se poate optimiza la $O(N \cdot \log(N) + Q \cdot \log(N))$, căutând binar direct pe structura arborilor de intervale.

Subtask-urile 5, 6 și 7 ($N, Q \leq 60000$, $N, Q \leq 80000$, $N, Q \leq 10^5$)

Soluția pentru aceste subtask-uri este o generalizare a soluției de la subtask-ul 3.

Răspunsul pentru un query va fi egal cu cea mai mică valoare x pentru care $cnt_x \cdot 2 \geq r - l + 2$, unde:

- a este egal numărul de coloane $i \in [st, dr]$ pentru care $\min(a_{1,i}, a_{2,i}) \leq x$.
- b este egal cu numărul de coloane $i \in [st, dr]$ pentru care $\max(a_{1,i}, a_{2,i}) \leq x$.
- $cnt_x = a + [b > 0]$

Valoarea expresiei $[b > 0]$ poate fi calculată ușor folosind un arbore de intervale care admite următoarele operații:

1. Schimbă valoarea unui element din șir.
2. Calculează minimul pe o subsecvență.

Valoarea lui a poate fi aflată (neeficient) în felul următor:

Pentru fiecare valoare $x \in [1, 3 \cdot 10^5]$, vom avea un **ordered_set** care va reține indicii coloanelor i pentru care $\min(a_{1,i}, a_{2,i}) = x$.

Pentru a rezolva (neeficient) un query, putem itera brut prin toate valorile de la 1 la $3 \cdot 10^5$, iar pentru fiecare valoare x vom aduna la a numărul de poziții din **ordered_set**-ul corespunzător care sunt cuprinse între st și dr .

Această abordare poate fi optimizată folosind o structură similară unui arbore indexat binar:

Pentru fiecare valoare $x \in [1, 3 \cdot 10^5]$, vom avea un **ordered_set** care va reține indicii coloanelor i pentru care $\min(a_{1,i}, a_{2,i}) \in (x - lsb(x), x]$, unde $lsb(x) = x \& -x$.

Răspunsul pentru un query se poate afla folosind o căutare binară direct pe structura acestui arbore indexat binar în $O(\log N \cdot \log VMAX)$.

Complexitate finală: $O((N + Q) \cdot \log(N) \cdot \log(VMAX))$.

Notă: În locul unui arbore indexat binar de **ordered_set**-uri, se poate folosi un arbore indexat binar 2d, această soluție având o constantă semnificativ mai bună.

Notă 2: Subtask-urile 5 și 6 sunt gândite pentru a puncta implementări mai puțin eficiente ale acestei idei, cum ar fi:

- $O((N + Q) \cdot \sqrt{VMAX} \cdot \log(N))$ - folosind square root decomposition în loc de arbore indexat binar.
- $O((N + Q) \cdot \log^2(VMAX) \cdot \log(N))$ - căutare binară pe rezultat fără a căuta binar pe structura arborelui indexat binar.
- Diverse implementări mai lente având complexitatea $O((N + Q) \cdot \log(N) \cdot \log(VMAX))$.