DESCRIEREA SOLUȚIILOR, OLIMPIADA NAȚIONALĂ DE INFORMATICĂ CLASA A X-A

COMISIA ȘTIINȚIFICĂ

Problema 1: Arcade

Autor: Stud. Apostol Ilie-Daniel, Facultatea de Matematică și Informatică, Universitatea din București

Pregătită de: Stud. Apostol Ilie-Daniel și Stud. Popescu Ștefan-Alexandru, Facultatea de Matematică și Informatică, Universitatea din București

Soluție $O(N^2)$ pe test.

Observăm că singurele puncte în care putem să ajungem după o mutare sunt (N, N) sau un punct situat imediat lângă o celulă blocată (vecin pe linie sau coloană).

Prin urmare, vom precalcula următorul tablou: $next[i][j][dir]$ = poziția unde voi ajunge dacă sunt în punctul (i, j) și fac o mutare în direcția dir. Acest tablou se poate precalcula în timp $O(N^2)$, simulând o mutare din toți vecinii celulelor blocate și actualizând tabloul $next$.

Pentru cerința 1, pentru fiecare caracter din șirul de mutări, vom face tranzițiile în timp constant, folosindu-ne de tabloul $next$. Pentru aceste restricții, și o soluție în timp $O(N \cdot |S|)$ va obține punctaj maxim pe acest subtask.

Pentru cerința 2, vom folosi algoritmul lui Lee pentru a determina drumul cu număr minim de mutări din celula $(1, 1)$ în celula (N, N). Folosind tabloul $next$, putem determina în timp constant cei patru vecini ai unui punct (i, j), aceștia fiind $next[i][j][up]$, $next[i][j][down]$, $next[i][j][left]$, $next[i][j][right]$.

Soluție completă.

Ne vom baza pe aceeași observație ca la subtask-ul anterior, anume că singurele puncte de interes sunt (N, N) și punctele care sunt vecine cu o celulă blocată. Este util să ne folosim și de faptul că avem exact o celulă blocată pe fiecare linie și exact o celulă blocată pe fiecare coloană. Prin urmare, vom nota cu $line[i]$ indicele coloanei blocate de pe linia i și cu $col[i]$ indicele liniei blocate de pe coloana i.

Pentru a micșora timpul de execuție al soluției precedente, observația cheie este că atunci când suntem într-un punct de interes (i, j), ne putem deplasa doar într-un alt punct de interes (i, y) sau (x, j). Observația pe care ne vom baza este că într-un punct (i, j) suntem restricționați în mișcare doar de elementele $(i, line[i])$ și $(col[j], j)$. Folosindu-ne de aceste observații, putem să obținem complexitate constantă pe fiecare tranziție. Rezolvarea pentru ambele cerințe rămâne la fel, singura diferență fiind modul în care facem tranzițiile.

În funcție de cum reținem identificatorul fiecărei celule de interes din matrice, putem obține complexitatea $O(N \cdot logN)$ (cu un map sau unordered_map din STL) sau $O(N)$ dacă reținem punctele de interes sub forma $(id\_celulă\_blocată, direcție\_de\_deplasare)$.

Problema 2: Comun

Autor: Stud. Bogdan Vlad-Mihai, Facultatea de Matematică și Informatică, Universitatea din București

În cadrul acestei probleme, toate operațiile cu mulțimi se pot face privind mulțimile ca numere în baza 2. Operația de intersecție devine atunci un *și* pe biți, iar operația de reuniune este acum *sau* pe biți.

Subtask 1. Pentru cazul în care $K = N$, trebuie să calculăm numărul de cuvinte care se pot forma pe toate display-urile. Atunci, a i-a literă dintr-un cuvânt care se poate forma pe toate cele N display-uri trebuie să se afle în intersecția mulțimilor de litere corespunzătoare celulei i din fiecare display. Răspunsul pentru această cerință este produsul cardinalelor tuturor celor M intersecții de mulțimi.

Subtask 2. Pentru acest subtask, vom genera toate cuvintele de lungime M cu caractere din mulțimea $\{a, b\}$ și trebuie să verificăm pentru fiecare dintre acestea dacă se poate afișa pe cel puțin K display-uri. Pentru a trece prin toate cuvintele de forma menționată mai sus, este suficient să iterăm prin toate numerele de la 0 la $2^M - 1$, iar pentru fiecare bit de la 0 la $M - 1$, în funcție de valoarea sa, vom adauga caracterul a sau b la șirul corespunzător numărului la care suntem. Complexitatea acestei soluții este $O(2^M \cdot N \cdot M)$.

Subtask 3. Pentru acest subtask, vom genera toate cuvintele distincte de M litere ale alfabetului englez și vom număra pentru fiecare dintre acestea, câte dintre cele N display-uri îl pot forma. Pentru generarea tuturor cuvintelor distincte de lungime M cu litere ale alfabetului englez, este suficient să considerăm toate numerele de cel mult M cifre în baza 26. Complexitatea acestei soluții este $O(26^M \cdot N \cdot M)$

Subtask 4. Pentru o mulțime de display-uri, A, vom nota cu $count_A$ numărul de cuvinte distincte care se pot forma pe toate display-urile din A.

Vom încerca să calculăm pentru un k fixat, numărul de cuvinte distincte care se pot forma pe exact k display-uri. Să considerăm, așadar, o submulțime X de k display-uri. Astfel, numărul de cuvinte care poate să fie format doar de X este dat de formula:

$$\sum_{X \subseteq Y} (-1)^{|Y|-k} \cdot count_Y$$

Răspunsul pentru un k fixat este dat de suma numărului de cuvinte care pot să fie formate de mulțimile X cu $|X| = k$. Pentru fiecare submulțime de k display-uri, trebuie să iterăm prin toate superseturile sale, iar acest lucru trebuie făcut pentru orice k de la K la N. Prin urmare, complexitatea acestei soluții este $O(3^N \cdot N)$.

Subtask 5 și posibil Subtask 6. Pentru acest subtask, observația cheie este că o mulțime Y contribuie la răspuns pentru mai multe submulțimi X ale sale. Așadar, vom inversa formula de la subtask-ul anterior, rezultând că fiecare superset Y apare în rezultat de $\binom{|Y|}{K}$ cu semnul $(-1)^{|Y|-K}$. Așadar, considerând că avem calculate valorile $count_X$ pentru toate submulțimile X, complexitatea pentru calcularea răspunsului este $O(2^N \cdot N)$. Este bine de menționat că acest calcul se poate realiza și în $O(2^N)$, astfel

$$(-1)^{|Y|-K} \cdot \binom{|A|}{K} + (-1)^{|A|-K-1} \cdot \binom{|A|}{K+1} + \ldots + (-1)^{|A|-|A|} \cdot \binom{|A|}{|A|} = (-1)^{|A|-K} \cdot \binom{|A|-1}{|A|-K}$$

Pentru a calcula $count_X$ pentru un X fixat, trebuie să trecem prin toate display-urile din X și să facem intersecția mulțimilor corespunzătoare pentru fiecare celulă j. $count_X$ este produsul cardinalelor intersecțiilor celor M mulțimi. Complexitatea unei astfel de soluții este $O(2^N \cdot N \cdot M)$ și în funcție de implementare, poate obține între 85 și 100 de puncte.

Soluția completă. Pentru acest subtask, trebuie să calculăm mai rapid $count_X$. Observația pe care trebuie să o facem este că putem să calculăm $count_X$ folosindu-ne de $count_{X-\{i\}}$, unde i este un element oarecare din mulțimea X. Pentru a găsi ușor un astfel de i, putem să alegem cel mai semnificativ bit al măștii X.

O altă abordare poate folosi o coadă în care vom reține toate submulțimile pentru care avem deja calculat *count*. Astfel, pentru o submulțime X cu valoarea *count* deja calculată, va trebui să actualizăm (pe baza $count_X$) toate submulțimile de formă $X \cup \{i\}$ (care nu au valoarea *count* deja calculată), unde i este un display din afara mulțimii X.

Complexitatea unor astfel de soluții este $O(2^N \cdot M)$, dar este important de menționat că prima soluție se comportă mult mai bine în practică (deoarece accesează memoria într-un mod mai natural). Este posibil ca o soluție cu coadă care nu este suficient de bine optimizată să nu obțină punctaj maxim.

Problema 3: Matrice

Autor: Stud. Popescu Ștefan-Alexandru, Facultatea de Matematică și Informatică, Universitatea din București

Pregătită de: Stud. Bogdan Vlad-Mihai, Facultatea de Matematică și Informatică, Universitatea din București

Cazul $N = M = 1$ se tratează separat.

Subtask 1. Vom presupune, fără a pierde din generalitate, că $M = 1$. În acest subtask, ni se cere să calculăm numărul de progresii aritmetice cu rația mai mare ca zero, primul element mai mare sau egal ca zero și al N-lea termen mai mic sau egal decât T. Pentru o rație r fixată și un prim element f, trebuie să verificăm dacă $f + r \cdot (N-1) \leq T$. În caz afirmativ, creștem numărul de progresii care se pot forma. Complexitatea unei astfel de soluții este $O(T^2)$.

Subtask 2. Putem modifica soluția de la subtask-ul 1, observând că este suficient să fixăm valoarea primului element. Astfel, pentru un prim element f fixat, vom avea $\lfloor \frac{T-f}{N-1} \rfloor$ progresii posibile care se pot forma. Complexitatea unei astfel de soluții este $O(T)$.

Subtask 3. Notăm rațiile progresiilor pe coloane cu $rc[i]$ și rațiile pe linii cu $rl[i]$. Notăm cu $A[i][j]$ elementul de pe linia i și coloana j (matricea este indexată de la 1). Dacă alegem o pereche oarecare (x, y), cu proprietatea că există atât $A[x][y]$ cât și $A[x+1][y+1]$, putem scrie că $A[x+1][y+1] = A[x][y] + rc[y] + rl[x+1]$ (dacă din $A[x][y]$ facem primul pas în jos) sau $A[x+1][y+1] = A[x][y] + rl[x] + rc[y+1]$ (dacă din $A[x][y]$ facem primul pas spre dreapta). Din cele 2 formule reiese că $rc[y] + rl[x+1] = rc[y+1] + rl[x]$, care e echivalentă cu $rl[x+1] - rl[x] = rc[y+1] - rc[y]$, pentru orice x și y cu $x < N$ și $y < M$. Asta înseamnă că rațiile de pe linii se află la rândul lor într-o progresie aritmetică, cu o rație pe care o vom nota cu *rabs* (la fel și cele de pe coloane).

Prin urmare, o matrice cu proprietățile din cerință este unic determinată de $A[1][1], rc[1], rl[1]$ și *rabs*. Problema se reduce la a număra câte astfel de 4-tupluri există astfel încât matricea generată să aibă toate valorile mai mici sau egale cu T. Având în vedere că toate rațiile sunt pozitive (excepție *rabs*, care poate să fie 0), putem deduce că elementul maxim din matrice este $A[N][M]$.

Formula explicită pentru $A[N][M]$ este $A[N][M] = A[1][1] + (N-1) \cdot rc[1] + (M-1) \cdot rl[N] = A[1][1] + (N-1) \cdot rc[1] + (M-1) \cdot (N-1) \cdot rabs + (M-1) \cdot rl[1]$ (ne ducem în jos și apoi la dreapta).

Ignorăm pentru moment valoarea lui $A[1][1]$, considerând-o a fi 0 (singurul lucru care ne interesează cu adevărat e diferența dintre cel mai mare număr și cel mai mic număr, luăm în calcul ulterior valoarea inițială).

Dacă fixăm *rabs* și considerăm $A[1][1]$ că fiind 0, atunci problema se reduce la a determina toate perechile $(rc[1], rl[1])$ astfel încât $(N-1) \cdot rc[1] + (M-1) \cdot rl[1] \leq A[N][M] - (M-1) \cdot (N-1) \cdot rabs$.

Notăm $A[N][M] - (M-1) \cdot (N-1) \cdot rabs$ cu D. Atunci avem $(N-1) \cdot rc[1] + (M-1) \cdot rl[1] \leq D$, unde D ia valori (cel puțin teoretic) între 1 și T.

Pentru a rezolva acest subtask, este suficient să fixăm fiecare valoare pentru cele 4 necunoascute din 4-tuplu și să verificăm dacă acesta are sens. Complexitatea timp este $O(T^4)$.

Subtask 4.

Pentru acest subtask este suficient să iterăm prin toate valorile *rabs*, $rc[1]$ și $rl[1]$ pentru a calcula numărul de progresii care se pot forma. Complexitatea unei astfel de soluții este $O(T^3)$, dar cu o constantă foarte mică.

Subtask 5. Problema rămâne să calculăm numărul de soluții pentru x și y ale ecuației:

$$(N-1)\cdot x+(M-1)\cdot y=target, 1\le x,y\le T$$

Pentru început, vom încerca să găsim orice soluție a ecuației de mai sus, fără să ținem cont de restricțiile pentru x și y. Acest lucru se poate face folosind varianta extinsă a algoritmului lui Euclid. Să considerăm că am găsit o soluție de forma:

$$(N-1)\cdot x_0\cdot\frac{target}{g}+(M-1)\cdot y_0\cdot\frac{target}{g}=target$$

unde g este $gcd(N-1,M-1)$. Este ușor de observat că atunci când $target$ nu este divizibil cu g, ecuația nu are soluții.

Pornind de la această ecuație, putem să găsim alte soluții, precum:

$$(N-1)\cdot\left(x_0+\frac{M-1}{g}\right)+(M-1)\cdot\left(y_0-\frac{N-1}{g}\right)=target$$

De aici putem deduce că orice soluție a ecuației este de forma următoare:

$$x=x_0+k\cdot\left(\frac{M-1}{g}\right)$$
$$y=y_0-k\cdot\left(\frac{N-1}{g}\right)$$

Folosind relațiile de mai sus, putem găsi soluțiile care au valori minime, respectiv maxime pentru x și y în intervalul $[1,T]$ în timp constant. Vom nota cu $lowX$ valoarea minimă a lui x, iar cu upX valoarea maximă a lui x. Totodată, vom nota cu $lowY$ valoarea lui x corespunzătoare soluției cu y minim și upY valoarea lui x corespunzătoare soluției cu y maxim. Așadar, numărul total de soluții este dat de formula:

$$\frac{min(upX,upY)-max(lowX,lowY)}{M-1}+1$$

Putem, așadar, construi un vector $count$ de sume parțiale, unde $count[x]$ este numărul de soluții ale ecuației de mai sus pentru valori ale $target$ mai mici sau egale ca x. Rămâne să fixăm fiecare $rabs$ de la 0 la T și să verificăm dacă rația respectivă are sens, adică $T-rabs*(N-1)\cdot(M-1)>0$. În caz afirmativ, adăugăm la răspuns $count[T-rabs\cdot(N-1)\cdot(M-1)]$.

Pentru rezolvarea acestui subtask, este important să ținem în memorie un singur vector de numere pe 32 de biți de lungime T. Complexitatea timp este $O(T)$, iar complexitatea spațiu este $O(T)$.

Soluția completă. - Stud. Bogdan Vlad-Mihai

Pentru soluția completă, este nevoie să reducem memoria folosită. Acest lucru se poate face folosind două variabile, $prefixCount$ și $prefixTotal$. La fiecare pas i vom adăuga la $prefixCount$ numărul de soluții ale ecuației de mai sus cu $target=i$, iar mai apoi vom adăuga $prefixCount$ la $prefixTotal$. În acest mod, la orice moment i de timp vom avea în $prefixTotal$ numărul de soluții ale ecuației de mai sus, cu $target\le i$. În acest fel, complexitatea spațiu devine $O(1)$, iar complexitatea timp rămâne $O(T)$.

Soluția alternativă. - Stud. Popescu Ștefan-Alexandru

Soluția pleacă de la ideile de la subtask-ul 3 și se bazează pe programare dinamică. Vom reține $dp[i]$ că fiind numărul de moduri de a alege perechea $(rc[1],rl[1])$ astfel încât $(N-1)\cdot rc[1]+(M-1)\cdot rl[1]=i$ (adică $rc[1]\neq 0$ si $rl[1]\neq 0$).

Relația de recurență pentru dp este următoarea:

- **Cazul I::** $dp[i]=dp[i-(M-1)]+1$, dacă $i\%(N-1)=0$ (putem ajunge în i dintr-o stare în care alegem doar $rl[1]$ și $rc[1]=0$, respectiv putem ajunge în i dintr-o stare în care deja aveam și $rl[1]$ și $rc[1]$ diferite de zero).
- **Cazul II::** $dp[i]=dp[i-(M-1)]$, dacă $i\%(N-1)>0$ (putem ajunge în i doar dintr-o stare în care deja aveam și $rl[1]$ și $rc[1]$ diferite de zero).

Pentru punctaj complet, observația cheie este că ne interesează doar ultimele M elemente din dinamică. Prin urmare, folosind o abordare asemănătoare cu cea de la soluția anterioară (în ceea ce privește calcularea sumelor pe prefixe) și un deque sau un buffer circular pentru a reține doar ultimele M valori din dinamică. Complexitatea timp a unei astfel de soluții este $O(T)$, iar complexitatea spațiu este $O(min(N,M))$.

Echipa

Problemele pentru această etapă au fost pregătite de:

- Prof. Anton Cristiana Elena
- Stud. Apostol Ilie-Daniel
- Stud. Bogdan Vlad-Mihai
- Lect. Diac Paul
- Drd. Ioniță Alexandru
- Prof. Moț Nistor
- Prof. Pintea Rodica
- Stud. Popa Sebastian
- Stud. Popescu Ștefan-Alexandru
- Stud. Răpeanu George