Soluții Olimpiada Naţională de Informatică, Etapa naţională
Clasele 11-12

Comisia științifică

Wednesday 24th April, 2024

Problema 1: DETONATOR

Propusă de: Asist. Drd. Andrei Costin Constantinescu

Subtask 1 (7 puncte)

În acest caz, toate valorile $T_{i,j}$ sunt egale, le notăm valoarea comună cu T. Aici ordinea de dezamorsare a bombelor nu contează, deci răspunsul este $T - \frac{n \cdot (n+1)}{2}$.

Subtasks 2 (13+9 puncte):

Se folosește metoda backtracking pentru a genera toate ordinile posibile de dezamorsare. Pentru fiecare ordine, se calculează în timp liniar X-ul maxim posibil pentru care respectiva ordine respectă, și se ia maximul acestor X-uri.

Subtask 4 (14 puncte):

Pentru rezolvarea acestui subtask se poate folosi o abordare de tip greedy. La fiecare pas dezamorsăm bomba cu cel mai mic timp până la explozie din mulțimea celor care nu sunt așezate pe bombe nedezamorsate încă. Acest greedy este corect în contextul dat deoarece nu are sens să luăm altă valoare decât minimul la fiecare pas. Pe cazul general este posibil însă să fie nevoie să luăm o altă valoare decât minimul dacă undeva deasupra acelei bombe se află o bomba cu valoare mai mică decât minimul nostru (care poate trebuie dezamorsată urgent). Deoarece în acest subtask valoarea unei bombe nu depășește valoarea niciunei bombe care este (direct sau indirect) așezată pe ea, acest caz nu apare și prin urmare strategia greedy funcționează. Având în vedere valoarea mică a lui N, se poate itera prin toate valorile de fiecare dată când se caută minimul sau când se depistează bombele ce pot fi dezamorsate la pasul următor. De asemenea pentru că există doar $2 \cdot |X| + 1 \leq 21$ de valori posibile ale lui X putem încerca să rezolvăm pentru fiecare X și să reținem X-ul maxim pentru care nicio bomba nu a fost dezamorsată după timpul său limită. Complexitate timp este $O(N^4 \cdot |X|)$.

Subtask 5 (13 puncte):

Pentru a rezolva problema pe cazul general vom transforma piramida astfel încât să respecte proprietatea de la subtask-ul 4 ($T_{i,j} \geq \max\{T_{i+1,j}, T_{i+1,j+1}\}$ pentru $1 \leq j \leq i < N$). Dacă o bomba cu valoarea a se află așezată peste o bombă cu valoare mai mare, termenul limită real până la care putem dezamorsa această bombă fără ca bomba de deasupra sa să explodeze ulterior, este de fapt $a - 1$. Parcurgem piramida de sus în jos și determinăm noua valoare a

unei bombe ca fiind minimul dintre valoarea sa actuală și valorile bombelor așezate direct pe aceasta (din care se scade 1). După transformarea valorilor în acest mod, abordarea greedy va funcționa corect. Nici la acest pas nu este necesară optimizarea implementării. Complexitatea timp este $O(N^4 \cdot |X|)$.

Subtask 6 (2 puncte):

Pentru acest subtask în loc să iterăm prin toate valorile posibile ale lui X, căutăm binar valoarea sa în intervalul $[\ell, r]$ unde $\ell = -\frac{N \cdot (N+1)}{2}$ și r este valoarea minimă din piramidă. În rest soluția este analogă cu subtaskul anterior. Complexitatea timp este $O(N^4 \cdot \log |X|)$.

Subtask 7 (9 puncte):

Pentru acest subtask nu este necesară găsirea valorii X prin căutare binară dar trebuie să optimizăm cum determinăm mulțimea de bombe ce pot fi dezamorsate la fiecare pas. Această mulțime va avea mereu maxim N elemente deci astfel reducem căutarea minimului dintre acestea la complexitate $O(N)$. De fiecare dată când dezamorsăm o nouă bombă încercăm să adăugăm în mulțime bombele așezate direct pe aceasta. La acest subtask se poate determina X prin încercarea celor $2 \cdot |X| + 1 \leq 21$ de valori posibile. Complexitatea timp este $O(N^3 \cdot |X|)$.

Subtask 8 (2 puncte):

Pentru acest subtask trebuie să ținem mulțimea de maxim N bombe ce pot fi dezamorsate ca la subtask-ul 7 și în plus să găsim valoarea maximă a lui X prin căutare binară. Complexitatea timp este $O(N^3 \log N)$.

Subtask 9 (19 puncte):

Pentru acest subtask facem la fel ca la subtask-ul 8 dar ținem mulțimea de bombe ce pot fi dezamorsate într-o coadă de priorități (heap). Astfel determinăm minimul la fiecare pas în complexitate $(O(\log N))$. Complexitatea timp devine $O(N^2 \log N \log |X|)$.

Subtask 10 (12 puncte):

Pentru acest subtask optimizăm subtask-ul 9 prin eliminarea căutării binare a lui X. Determinăm X calculând diferența minimă dintre valoarea unei bombe și momentul în care aceasta a fost dezamorsată. Complexitatea timp finală este $O(N^2 \log N)$.

Solutie Alternativa (100 puncte)

În loc să stabilim ordinea de dezamorsare a bombelor ca în soluțiile prezentate anterior, vom pleca în ordine invers temporală, la început având toate bombele dezamorsate, și reamorsând pe rând câte o bombă în fiecare secundă. Partea de găsire a X-ului se poate realiza tot printr-o căutare binară care se poate ulterior elimina analog cu Subtaskul 10. Așadar, descriem restul soluției pentru simplitate presupunând o valoare X fixată, fără a restrânge generalitatea $X = 0$. Într-o ordine de reamorsare (adică o ordine de dezamorsare văzută invers cronologic) se reamorsează o bombă la momentul $T = \frac{N \cdot (N+1)}{2}$ (fiind chiar bomba din vârful piramidei), una la momentul $T - 1$, una la momentul $T - 2$, și așa mai departe, ultima fiind reamorsată la momentul 1 (aceasta fiind o bomba de la baza piramidei). Vom pleca cu $t = T$ și vom decrementa t cu unu la fiecare pas, până când ajunge la 0. La fiecare pas determinăm bombele ce pot fi reamorsate, adică acele bombe peste care sunt așezate doar bombe deja reamorsate, și dintre acestea căutăm dacă există una cu termenul limită cel puțin t. Dacă nu, atunci

dispozitivul nu poate fi dezamorsat, altfel, alegem arbitrar una dintre aceste bombe și o reamorsăm. Corectitudinea soluției vine din faptul că odată ce o bombă devine reamorsabilă, ea poate fi reamorsată oricând odată ce t a coborât la termenul ei limită, iar reamorsarea unei bombe nu micșorează niciodată mulțimea de alegeri pentru pașii ulteriori. Implementarea se poate face eficient cu o coadă de priorități (heap), analog cu soluția precedentă. Observăm că în această soluție nu mai trebuie să schimbăm valorile astfel încât să se respecte proprietatea de la subtask-ul 4 ($T_{i,j} \geq \max\{T_{i+1,j}, T_{i+1,j+1}\}$ pentru $1 \leq j \leq i < N$). Complexitatea timp finală este tot $O(N^2 \log N)$.

Temă de gandire! Există și o solutie cu complexitatea timp minimă posibilă $O(N^2)$, nenecesară pentru 100 de puncte (timpii de rulare sunt în practică comparabili cu soluțiile $O(N^2 \log N)$). Vă invităm să o gasiți!

Problema 2: ZID

Propusă de: Stud. Liviu Silion

Problema se reduce la a calcula răspunsul folosind programare dinamică pentru $M = 1$. Pentru $M > 1$ vom combina doar rezultatele din dinamica pentru $M = 1$.

Prima parte

Vom calcula $dp[i][j] =$ numărul de moduri de a face un turn de înălțimea i cu suma lungimilor pieselor galbene egală cu j.

$O(N^2 \cdot M)$

Facem tranzițiile adăugând efectiv câte o piesă:
$dp[i][j] = \sum_k dp[i-k][j-k]$ pentru o piesă galbenă de lungime k
$dp[i][j] = \sum_k dp[i-k][j]$ pentru o piesă roșie de lungime k

$O(N \cdot M)$

Putem optimiza dinamica anterioară ținând sume parțiale pentru $dp[i-2][j-2] + dp[i-4][j-4] + \ldots$ pentru cazul când adăugam o piesă galbenă, respectiv $dp[i-1][j] + dp[i-3][j] + \ldots$ pentru cazul când adăugam o piesă roșie.

O altă abordare ar fi să ținem și culoarea ultimului turn și să ne gândim că la un pas, putem prelungi ultimul turn cu 2 unități sau să adăugăm unul de culoare nouă de lungimea 1, respectiv 2.

A doua parte

$O(M^2 \cdot K)$

Avem $dp'[i][j] =$ numărul de moduri de a construi un zid (de înălțimea N) cu lățimea i și cu suma lungimilor pieselor galbene j. Actualiazăm dinamica în stil rucsac: $dp'[i][j] = \sum_k dp'[i-1][j-k] \cdot dp[N][k]$.

$O(M^2 \cdot \log K)$

Optimizăm dinamica anterioară folosind o abordare Divide et Impera. Cum mereu facem aceleași tranziții la dinamică, putem folosi un algoritm similar cu ridicare la putere în timp logaritmmic.

Note

Pentru prima parte este probabil să trebuiască ținute doar ultimele 3 linii din dinamică. Prima parte poate de asemenea fi rezolvată în $O(M^3 \cdot \log N)$

Partea a doua poate fi privită ca fiind o convoluție de polinoame și rezolvată în $O(M \cdot \log M \log K)$. De asemenea, se poate realiza elementar mai eficient în $O(M^2)$ printr-o recurență.

Problema 3: ARBORE

Propusă de: Stud. Alexandru Lorintz

Pentru primele 2 subtask-uri, $k_1 = k_2 = 0$, deci operația din enunț nu poate fi folosită. Astfel, având un nod dat, trebuie calculată suma costurilor drumurilor de la respectivul nod la restul nodurilor din arbore. Pentru primul subtask se pot calcula naiv răspunsurile la interogări în complexitate $O(N \cdot Q)$.

Pentru cel de-al doilea subtask este necesară o abordare mai eficientă. Să presupunem că fixăm o rădăcină pentru arbore rezolvăm toate interogările despre această rădăcină. O observație utilă atât în rezolvarea acestui subtask, cât și în rezolvarea întregii probleme este că putem reformula calcularea sumei costurilor tuturor drumurilor în a calcula pentru fiecare muchie contribuția acesteia la rezultat. Mai exact, o muchie contribuie la rezultat cu o valoare egală cu produsul dintre numărul de drumuri care trec prin aceasta și costul său. Numărul de drumuri care trec printr-o muchie fixată este egal cu dimensiunea subarborelui calculată pentru capătul muchiei aflat la adâncime mai mare în arbore(atenție, există o rădăcină fixată). Astfel, pentru o rădăcină fixată, se pot calcula cu o parcurgere în arbore contribuțiile muchiilor și se pot aduna toate la rezultat. Pentru a calcula răspunsurile pentru toate nodurile din interogări, se poate realiza un proces de "mutare a rădăcinii". După ce am calculat răspunsurile pentru prima rădăcină, se face încă o parcurgere și la fiecare pas "se mută" rădăcina în nodul curent din parcurgere. La mutarea rădăcinii se schimbă dimensiunea subarborelui doar pentru vechea rădăcină, deci se schimbă un singur coeficient de muchie, acela al muchiei dintre vechea rădăcină și cea nouă. În concluzie, acest subtask se poate rezolva în complexitate $O(N + Q)$.

Pentru următoarele subtask-uri, problema trebuie rezolvată pentru o singură valoare k. Pentru asta, se poate pleca de la soluția anterioară. Se știe că pentru o muchie, coeficientul său în rezultat este dat de numărul de drumuri care trec prin aceasta, care este egal cu dimensiunea subarborelui calculată pentru capătul muchiei aflat la adâncime mai mare în arbore când rădăcina este fixată. Astfel, se observă că este mereu optim să se decrementeze muchia cu coeficientul cel mai mare, în limita numărului de operații disponibile. Astfel, se vor decrementa muchiile în ordine, de la cea cu coeficientul cel mai mare la cea cu coeficientul cel mai mic.

Pentru a implementa soluția propusă anterior, se poate proceda în diferite moduri, cu complexități diferite. Abordarea cea mai simplă este pentru fiecare interogare să se facă o parcurgere și să se rezolve greedy interogarea curentă cum am menționat anterior. Această abordare are complexitatea $O(N \cdot Q)$. Soluția aceasta poate fi puțin îmbunătățită dacă se face câte o singură parcurgere pentru toate interogările care au același nod și se rezolvă toate acestea deodată folosind doi pointeri sau câte o căutare binară pentru fiecare interogare peste un șir de sume parțiale al costurilor muchiilor sortate după dimensiunea subarborelui. Abordarea din urmă, indiferent de implementare, va avea complexitatea $O(N^2 + Q \cdot \log Q)$.

Pentru o soluție eficientă în cazul $k_1 = k_2$ se poate proceda ca la subtask-ul al doilea, mutându-se rădăcina. Pentru a actualiza coeficienții muchiilor și pentru a se calcula rapid cum se schimbă suma costurilor drumurilor în urma decrementărilor se poate folosi o structură de date arborescentă (arbori indexați binar sau arbori de intervale). Pentru a calcula răspunsul la o interogare se poate face o căutare binară și un query pe structura de date sau să se facă direct o căutare binară pe structura de date. Această soluție are complexitatea $O(Q \cdot \log N)$ sau $O(Q \cdot \log^2 N)$ în funcție de implementare. De menționat este că pentru ultimele soluții menționate se calculează suma decrementărilor muchiilor, deci trebuie să calculăm în paralel și suma costurilor tuturor drumurilor ca la al doilea subtask, din care se scade ceea ce calculăm pentru a se obține rezultatul final.

Pentru a rezolva toată problema se consideră soluția pentru cazul $k_1 = k_2$ și se generalizează. Pentru aceasta o să se folosească un arbore de intervale.

Dacă se notează cu $F(v, k_1, k_2) = f(v, k_1) + f(v, k_1 + 1) + \ldots + f(v, k_2 - 1) + f(v, k_2)$, putem calcula răspunsul pentru o interogare de forma (v, k_1, k_2) ca $F(v, k_2) - F(v, k_1 - 1)$.

Tot timpul la o interogare de tipul $F(v, k)$ vom considera un sufix din structura de date în funcție de cât permite limita k. Se poate imagina o histogramă a acestor valori. Histograma care dă rezultatul este similară, cu mențiunea că prima valoare din sufixul considerat poate fi luată în răspuns incompletă. În continuare, pentru fiecare număr x din sufixul rezultat, o să se adauge într-un nou șir indicele său de x ori (practic, fiecare dimensiune de subarbore apare de atâtea ori cât se pot decrementa muchii care o au ca și coeficient, ținând cont de ordinea optima de decrementare și de limita impusă). Având acest nou șir la dispoziție (să-l notăm cu s) și considerând că este indexat de la 1, rezultatul interogării devine $s[1] \cdot 1 + s[2] \cdot 2 + \ldots + s[m] \cdot m$, unde m este lungimea noului șir, deoare elementul de la poziția i apare în sufixele a i valori mai mici sau egale decât k.

Având în vedere observația anterioară, se poate adăuga în arborele de intervale și un câmp care să rețină rezultatul parțial al unei interogări. Tot ce este rămas este aflarea unei modalități de a combina două elemente în mod eficient. Se vor menține într-un nod variabilele *sumaPonderata*, *sumaMuchii* și *res*, unde *sumaPonderata* este egală cu suma valorilor $s \cdot suma[s]$ pe un interval, unde s este un indice (echivalent cu o dimensiune de subarbore), iar $suma[s]$ este suma costurilor tuturor muchiilor care îl au pe s ca și coeficient, *sumaMuchii* este suma valorilor $suma[s]$ pe un interval și răspunsul *res* pe un interval. Astfel, având două noduri cu variabilele *sumaPonderata*1, *sumaMuchii*1, *res*1, respectiv *sumaPonderata*2, *sumaMuchii*2, *res*2, nodul obținut prin combinarea acestora are variabilele: $sumaMuchii = sumaMuchii1 + sumaMuchii2$, $sumaPonderata = sumaPonderata1 + sumaPonderata2$, $res = res1 + res2 + sumaMuchii1 \cdot sumaPonderata2$. Explicația formulei pentru *res* se poate deduce din interpretarea cu histogramă menționată anterior a rezultatului unei interogări. Complexitatea finală devine $O(Q \cdot \log N)$, dacă pentru o interogare se calculează direct răspunsul printr-un proces similar cu căutarea binară pe structura de date. Dacă se implementează soluția în complexitate $O(Q \cdot \log^2 N)$ cu căutare binară și interogare pe structura de date se va obține punctaj doar pe subtask-urile unde $N, Q \leq 100000$.

Echipa

Problemele pentru această etapă au fost pregătite de:

- Asist. Drd. Andrei Constantinescu
- Prof. Zoltan Szabó
- Prog. Mihai Ciucu
- Stud. Mihaela Cismaru
- Prof. Mihai Bunget
- Stud. Alexandru Lorinz
- Stud. Vlad-Mihai Bogdan
- Stud. Liviu Silion
- Prog. Adrian Budău
- Prog. Daniel Posdărăscu
- Stud. Matei Tinca
- Stud. Bogdan Sitaru
- Stud. Luca Metehău