Editorial Unirea pentru Performanta

6 Iulie 2024

Cuprins

1 Multumiri

Acest concurs nu ar fi putut avea loc fără următoarele persoane:

- Lensu Alexandru și Lețu Andrei autorii problemelor și laureați la concursurile de informatică și membri activi ai comunității RoAlgo;
- Alex Vasiluță, fondatorul și dezvoltatorul principal al Kilonova;
- Ștefan Alecu, creatorul acestui șablon LaTeXpe care îl folosim;
- Cristian Margine, Tănăsescu Andrei Rareș și Ștefan Dăscălescu testerii concursului, care au dat numeroase sugestii și sfaturi utile pentru buna desfășurare a rundei;
- Lensu Alexandru, Lețu Andrei și Ștefan Dăscălescu coordonatorii rundei;
- Comunității RoAlgo și Colegiului Național "Unirea" Focșani pentru participarea la acest concurs.
- Sponsorilor RoAlgo și CodingHeads pentru susținerea financiară și nu numai!

2 Tralala

Autor: Lensu Alexandru

2.1 Soluția oficială

Această problemă este una de optimizare a soluției și a fost creată cu scopul de a antrena concurentul în a găsi o soluție cât mai ingenioasă.
Soluția următoare, fiind cea mai optimă găsită de comisie este adusă de Cristian Margine și folosește vectori de frecvență. Se aleg 7 note cu frecvențele cele mai mari și se procedează astfel:

- Se pune cea mai mică notă rămasă și se scade frecvența
- Se pune cea mai mare notă rămasă și se scade frecvența

La sfârșitul acestui algoritm e posibil să rămână nepoziționată o valoare, deci vom mai trece încă o dată prin șir, iar unde sunt două note vecine diferite de nota rămasă o vom poziționa și vom scădea frecvența.
De exemplu, dacă șirul construit după algoritm este: 1 7 2 7 2 6 3 5 3 4 și au rămas 2 note de 3 nepoziționate le putem așeza pe pozițiile 2 și 3, după prima modificare această poziție fiind defapt 4. Acest lucru este posibil deoarece 1 și 7 sunt diferite de 3, deci 3 poate fi poziționat între ele, analog 7 și 2 sunt diferite de 3.

Noul șir ar arăta astfel: 1 3 7 3 2 7 2 6 3 5 3 4.
Complexitate de timp: O(n)

2.1.1 Cod sursă

Soluție de 100 de puncte

3 Euro 2024

Autor: Lensu Alexandru

3.1 Soluția oficială

Cea mai importantă observație este că metoda optimă a lui Lensu de a se deplasa este una greedy și anume de a găsii cea mai îndepărtată valoare mai mică sau egală decât banii scoși zilnic(rezultatul căutat), în intervalul $[x, x + k]$, unde x este locul unde am staționat ziua precedentă.
Pentru primul subtask este de ajuns să încercăm să scoatem de la bancă toate sumele posibile și să simulăm drumul.
Complexitate de timp: O($Vmax * n * k$) sau O($Vmax * n$) în funcție de implementare. Pentru soluția de 100 de puncte și pentru subtaskul 2 este necesară căutarea binară a rezultatului. În funcție de metoda de verificare a valorii se obțin punctaje diferite.

- O implementare a verificării în O($n * k$) obține 70 de puncte
- O implementare a verificării în O(n) obține 100 de puncte

Complexitate de timp: O($n * k * \log(Vmax)$) sau O($n * \log(Vmax)$) în funcție de implementare.

3.1.1 Cod sursă

Soluție de 40 de puncte - Subtask 1

Soluție de 70 de puncte - Subtask 2

Soluție de 100 de puncte

4 scor pion

Autor: Leţu Andrei

4.1 Soluția oficială

Această problemă se poate rezolva prin metoda programării dinamice. Vom folosi o matrice dp de dimensiunea tablei de șah, pe care o vom inițializa cu valori de 0. Vrem ca $dp[i][j]$ să fie valoarea maximă pe care o poate avea un pion ajuns la poziția (i, j). Vom parcurge această matrice de jos în sus și o vom umple în felul următor: dacă la poziția (i, j) nu se află nicio piesă adversă, atunci $dp[i][j] = dp[i + 1][j]$. Altfel, $dp[i][j]$ va fi maximul dintre $dp[i + 1][j - 1]$ și $dp[i + 1][j + 1]$, la care se adaugă valoarea piesei de pe poziția curentă. Rezultatul va fi valoarea maximă din matricea dp. Complexitatea este $O(N * M)$.

4.1.1 Cod sursă

Soluție de 100 de puncte

5 scorpion

Autor: Leţu Andrei

5.1 Soluția oficială

Subtask #1
Aceast subtask se poate rezolva printr-o parcurgere BFS în care calculăm distanța fiecărui nod față de rădăcină. Pentru a obține rezultatul vom aduna doar distanțele nodurilor care sunt frunze.
Complexitate: O(N).
Subtask #2
Arborele cu scorul cel mai mare este format dintr-un lanț de noduri, în capătul căruia se află toate frunzele. Putem ajunge la această configurație printr-un mod greedy: scorul unei frunze crește când aceasta este mutată la părintele unei frunze aflată mai departe de rădăcină, deci toate frunzele ar trebuie să fie la distanță maximă față de rădăcină. În plus, toate frunzele ar trebui să aibă același părinte: mutând frunzele care sunt fii unui nod, nodul respectiv devine la rândul lui frunză și contribuie la scorul arborelui.
Dacă în arbore există u frunze, acestea se află la distanța $n - u$ față de rădăcină. Scorul arborelui va fi $u * (N - u)$. Aceasta este o funcție de gradul 2, care atinge valoare maximă când $u = \frac{N}{2}$. Dacă N este par, rezultatul este $\frac{N}{2} * \frac{N}{2}$, iar dacă N este impar $\frac{N-1}{2} * \frac{N+1}{2}$.

Complexitate: O(1).

Subtask #3

Am stabilit deja care este configurația care produce costul maxim. Dacă N este par, va trebui să alegem $\frac{N-1}{2}$ noduri care să formeze „coada” arborelui (lanțul de noduri care duce spre rădăcină). Știm că nodul 1 trebuie să fie mereu rădăcina arborelui, deci rezultatul va fi $A_{n-1}^{\frac{n}{2}-1}$.

Dacă N este impar, atunci coada poate avea fie lungimea $\frac{N-1}{2}$, fie $\frac{N+1}{2}$, deci rezultatul este $A_{n-1}^{\frac{n-1}{2}-1} + A_{n-1}^{\frac{n+1}{2}-1}$.

Complexitate: O(N).

5.1.1 Cod sursă

Soluție de 100 de puncte

6 fuziune

Autor: Leţu Andrei

6.1 Soluția oficială

Subtask #1
Aceast subtask se poate rezolva brut: pentru fiecare interogare vom parcurge intervalul dat și vom calcula rezultatul folosind formula oferită.
Complexitate: O($Q * N$).
Subtask #2
Observăm că operația de fuzionare este comutativă și asociativă, adică nu contează în ce ordine fuzionăm luptătorii. Prin urmare, putem precalcula rezultatul pentru secțiuni de lungimea $\sqrt{N}$, pe care să le refolosim în interogări. Pentru fiecare interogare vor fi calculate maxim $3 * \sqrt{N}$ valori.
Complexitate: O($Q * \sqrt{N}$).
Subtask #3
Folosind aceleași observații ca la punctul anterior, putem calcula rezultatul pentru orice interval folosind un arbore de intervale.
Complexitate: O($Q * \log N$).
Subtask #4
Putem observa că $a + b + a * b = (a + 1) * (b + 1) - 1$. Dacă mărim cu 1 toate valorile citite, putem afla rezultatul înmulțind valorile din intervalul $[s, d]$ iar

apoi scăzând 1. Putem folosi vectori de produse de sufixe pentru a afla răspunsul unei interogări în O(1). Vectorul *prod* va reține produsul primelor *i* elemente, iar vectorul *inv* inversul acestora. Rezultatul unei interogări va fi $prod[d_i] * inv[s_i - 1] - 1$.
Complexitate: O($Q + N$).

6.1.1 Cod sursă

Soluție de 15 puncte - Subtask 1
Soluție de 40 de puncte - Subtask 2
Soluție de 70 de puncte - Subtask 3
Soluție de 100 de puncte

7 skill issue

Autor: Lensu Alexandru

7.1 Soluția oficială

Soluția se bazează pe propietățile operației XOR. Suma efectuării operației XOR pe biți între elementele tuturor submulțimilor unei mulțimi este egală cu efectuarea operației pe biți SAU între toate elementele mulțimii înmulțit cu 2^{n-1}, unde n este numărul de elemente din mulțime. Demonstrația o puteți găsi aici.
Datorită acestei propietăți observăm că putem folosi structuri de date pentru operațiile de tip 1 și 2.
Pentru Subtaskul 1 se folosește metoda backtracking și nu este necesară această observație.
Pentru Subtaskul 2 se folosește această observație fără a se implementa o structură de date.
Pentru 80 de puncte se poate folosi câte un arbore de intervale pentru fiecare linie, actualizându-se fiecare celulă la operațiile de tip 1 și la operațiile de tip 2, aflând rezulatul efectuării operației ”sau” pe biți în intervalul $[y1, y2]$ pentru tabla A, respectiv $[y3, y4]$ pentru tabla B , rezultatul final al interogării respective fiind efectuarea operației SAU pe biți între rezultatele liniilor din intervalul $[x1, x2]$ pentru tabla A, respectiv $[x3, x4]$ pentru tabla B,

înmulțite 2^{n-1}, unde n este numărul de elemente din submatricea respectivă.
Fie R_A rezultatul operațiilor de tip 2 ale tablei A.
Fie R_B rezultatul operațiilor de tip 2 ale tablei B.
Se va ține într-un vector $ans[i]$ cașigătorul rundei i de tip 2.
Dacă $R_a > R_B$ atunci $ans[i] = 1$ pentru că va câștiga prima tablă
Dacă $R_a < R_B$ atunci $ans[i] = 2$ pentru că va câștiga a doua tablă
Dacă $R_a = R_B$ atunci $ans[i] = 0$ nu câștigă nicio tablă.
Pentru a determina modul optim în care Ștefan folosește schimbările se va folosi metoda programării dinamice, rezultatul fiind construit cu ajutorul vectorului ans. (dp[i][j] = scorul maxim obținut de Ștefan după i runde de tip 2 folosind j schimbări)
Pentru 100 de puncte este aceeași metodă de rezolvare însă se vor folosi arbori de intervale 2D.

7.1.1 Cod sursă

Soluție de 50 de puncte
Soluție de 80 de puncte
Soluție de 100 de puncte