Editorial RoAlgo Contest 8

23 Decembrie 2023

Cuprins

1 Multumiri

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

- Ștefan-Cosmin Dăscălescu, Matei Ionescu, Lucian Badea, Tudor Morariu și nu în ultimul rând, Radu-Teodor Prosie, 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;
- Traian Mihai Danciu, Andrei Lețu, Andrei Cherteș, Andrei Paul Iorgulescu, Andrei Cosmin Popescu Filimon, Eduard-Lucian Pîrțac, Rareș-Andrei Neculau, Emily Susan, Raul Ardelean, și nu în ultimul rând, Ștefan-Constantin Neagu, testerii concursului, care au dat numeroase sugestii și sfaturi utile pentru buna desfășurare a rundei;
- Ștefan-Cosmin Dăscălescu, coordonatorul rundei, și autorul solutiilor video;
- Comunității RoAlgo, pentru participarea la acest concurs.

2 Mini SuperLiga

Autor: Stefan Dascalescu

2.1 Soluția oficială

Pentru această problemă, trebuie doar să verificăm dacă numărul de meciuri și numărul de puncte corespund cu statisticile descrise în datele de intrare, soluția fiind una simplă din punct de vedere al implementării.

2.1.1 Cod sursă

Soluție de 100

3 SuperLiga

Autor: Stefan Dascalescu

3.1 Soluția oficială

Această problemă este o problemă tipică în care precizia la implementare, precum și atenția la tratarea diferitelor cazuri se dovedește a fi crucială, totuși subtaskurile date în enunț au rolul de a facilita rezolvarea problemei. Astfel, soluția oficială tratează mai întâi cazul când numărul de puncte e necunoscut, fixând numărul de puncte în funcție de celelalte date. Apoi, în funcție de numărul de necunoscute, tratăm diversele cazuri, având grijă să evităm ratarea vreunui caz particular.
Deși soluția este una mai stufoasă, tratarea cazurilor subtask cu subtask, împreună cu subtaskurile pentru punctaje parțiale în care datele de intrare sunt reduse ajută la calcularea rezultatelor într-un mod foarte facil.

3.2 Soluție alternativă

Problema poate fi abordată prin segmentarea în trei scenarii principale, în funcție de setul de informații disponibile:

Când numărul total de puncte este necunoscut

Rezolvarea se bazează pe utilizarea diferențelor pentru a deduce valorile necunoscute, urmată de calculul numărului total de puncte.

Când numărul de puncte este cunoscut, dar lipsește o valoare

Această situație este similară cu cea precedentă, diferența fiind că nu mai este necesar să determinăm numărul total de puncte.

Când numărul de puncte este cunoscut, dar lipsește mai mult de o valoare

Pentru accest caz, vom examina in detailu fiecare situație

Numărul de înfrângeri este cunoscut

Putem formula problema astfel

$$M - I = V + E \Rightarrow V = M - I - E$$

$$P = 3V + E \Rightarrow P = 3M - 3I - 2E$$
$$E = M - I - \frac{P - (M + I)}{2}$$

Determinând E, putem calcula V prin diferență.

Numărul de egaluri este cunoscut

Putem să calculăm V astfel:

$$P = 3V + E \Rightarrow V = \frac{P - E}{3}$$

Cu V cunoscut, I se obține prin diferență.

Numărul de victorii este cunoscut

Putem să aflam E folosind o metodă similară cu cazul anterior:

$$P = 3V + E \Rightarrow E = P - 3V$$

Apoi, I este dedus prin diferență.

Când nu cunoaștem alte valori

Putem stabili V și E în funcție de P:

$$P = 3V + E \Rightarrow V = \left\lfloor \frac{P}{3} \right\rfloor, E = P \bmod 3$$

Și apoi I este calculat prin diferență.
La final, verificăm din nou că toate cazurile trec

3.3 Cod sursă

Soluție de 100
Soluție alternativă

4 Problema Banda Lu' Bahoi

Autor: Lucian Badea

4.1 Subtask 1

Pentru acest subtask se poate precalcula q_k și o sumă pe prefix $sPref_k = q_0 + q_1 + ... + q_k$.

4.2 Soluția oficială

Pentru 100 de puncte, va trebui să folosim faptul că $sPref_k = q_{k+2} - 1$, ce se demonstrează prin inducție. Astfel, va trebui sa aflăm numerele q_{r+2} și q_{l+1} folosind exponențiere de matrice. Mai exact, va trebui să ridicăm matricea $\begin{pmatrix} 0 & 1 \\ 1 & 1 \end{pmatrix}$ la puterile $r + 2$, respectiv $l + 1$.

4.2.1 Cod sursă

Soluție de 100

5 Jocuri cu Kdouri

Autor: Tudor Morariu

5.1 Subtask 1

Se poate implementa caz cu caz si se calculează fiecare probabilitate individual.

5.2 Subtask 2

Se poate itera numărul de pe zar, iar rezultatul poate fi aflat simulând jocul pe stările determinate de numerele Grundy (vezi soluția de 100p pentru strategie)

5.3 Soluție Oficială

Dacă presupunem că zarul are n fețe atunci probabilitatea că zarul să dea numărul i este de $\frac{1}{n}$. În funcție de i jocul se reduce la un bounded nim. Câștigătorul poate fi aflat în $O(1)$ pași folosind numerele Grundy. Strategia optimă pentru jucătorul care mută este să-l ducă pe adversar într-o poziție pierzătoare. Spre exemplu dacă grămada are 5 cadouri și Alice poate să ia încat Bob să fie fortat să lase între 1 și 3 cadouri, orice alegere fiind pierzătoare

pentru el pentru ca la următoarea mutare Alice va putea să ia toate cadourile rămase.

Pentru a calcula răspunsul, dacă N = Numărul de cazuri posibile (egal cu câte fețe are zarul) și k = Numărul de cazuri favorabile atunci răspunsul este:

$$\frac{k}{N} \mod M = ((k \mod M) * (N^{M-2} \mod M)) \mod M$$

Spre exemplu, dacă $P_1 = 3$, $P_2 = 3$ și $N = 3$ avem 3 cazuri:

Zarul dă numărul 1 (P_2 este eliminat):

Dacă P_1 se divide cu 2 atunci Alice câștigă, iar dacă nu, Bob câștigă.

In acst caz P_1 este 3 deci Bob câștigă.

Zarul dă numărul 2 (P_1 este eliminat):

Dacă P_2 se divide cu 3 atunci Alice câștigă, iar dacă nu, Bob câștigă.

In acest caz P_2 este 3 deci Alice câștigă.

Zarul dă numărul 3 (P_2 este eliminat):

Dacă P_1 se divide cu 4 atunci Alice câștigă, iar dacă nu, Bob câstigă.

In acest caz P_1 este 3 deci Bob câștigă.

Datorită faptului că $M = 10^9 + 7$ se folosește exponențiere rapidă.

5.3.1 Cod sursă

Soluție de 100

6 Problema Beculete de brad

Autor: Tudor Morariu

6.1 Soluția oficială

Să presupunem că avem un lanț de relații de tipul

$$S_{a_1} \neq S_{a_2}; S_{a_2} \neq S_{a_3} ... S_{a_{n-1}} \neq S_{a_n}$$

atunci din moment ce avem doar 2 culori, răspunsul este 2. Dacă aflăm fiecare lanț de tipul acesta, fie acest număr k, răspunsul este 2^k, motivația fiind similară cu cea de la algoritmul pentru determinarea proprietății unui graf de a fi bipartit.
Folosind această abordare apare un caz special, cand $S_{a_n} \neq S_{a_1}$. Dacă se întamplă asta se observă că sunt 2 răspunsuri doar dacă n este par. De exemplu dacă $n = 4$ atunci avem: $ARAR; RARA$.
Aceasta se poate calcula eficient folosind un DFS cu marcare în complexitate de $O(N + M)$.

6.1.1 Cod sursă

Soluție de 100

7 Problema Lost&Found

Autor: Prosie Radu-Teodor

Soluție subtask 1

Dacă notam cu F_i de câte ori apare elementul i în vectorul inițial, atunci observăm că $Z_1 = F_1 + F_3, Z_2 = F_2, Z_3 = F_1 + F_2 + F_3$. Acest lucru ne permite să calculăm variabilele F_1, F_2, F_3,și mai trebuie doar să verificăm dacă există un șir inițial cu aceste valori. O sursă demonstrativă se poate găsi aici.

Soluție completă

Ne vom propune să calculăm F_i pe rând pentru fiecare i de la 1 la V. Spunem că $j \subseteq i$ dacă j este submască a lui i. Fie $sos_i = \sum_{j \subseteq i} F_j$ (câte elemente $j \neq i$ respectă $i\ OR\ j = i$). Să presupunem că am calculat deja $F_1, F_2, ...F_{i-1}$. Atunci avem $F_i = sos_i - \sum_{j \subset i} F_j$ (atenție! semnul este $\subset$, nu $\subseteq$). Acum problema se reduce la a calcula $sos_i, \forall i, 1 <= i <= V$. Se observă ca $sos_i = N - Z_{i \oplus V}$, unde $\oplus$ este operația XOR pe biți.
Dacă notam $K = \log_2 V$, atunci această soluție poate fi implementată in $O(4^K)$ sau în $O(3^K)$, în funcție de modul în care sunt parcurse submaștile, numai cea de-a doua variantă luând 100 de puncte. De asemenea, există și o

soluție bazată pe inversarea ordinii parcurgerii de la algoritmul de sos dp, aceasta având complexitatea $O(K \cdot 2^K)$.
Bonus : demonstrați că șirul inițial ce generează un anumit șir Z este unic.

8 Kiloforces

Autor: Ionescu Matei

8.1 Clarificare a judecății problemei

Observăm mai întai că răspunsul unei întrebari este de forma :

$$(p_l + p_{l+1} + p_{l+2} + \cdots + p_{r-1} + p_r) - ((r-l) \cdot t_{k_1} + (r-l-1) \cdot t_{k_2} + ... + t_{k_{r-l}})$$

Deci pentru a maximiza scorul final, trebuie să minimizăm a doua paranteză, lucru simplu de făcut pentru că putem efectiv lua problemele în ordine crescătoare după t_i. Astfel, k_1 reprezintă poziția problemei cu cel mai mic t_i, k_2 poziția problemei cu al doilea cel mai mic t_i , etc. Acum problema se rezumă la aflarea sumei din cea de-a doua paranteză.

8.2 Soluția 37 puncte

Sortăm intervalul $[l, r]$ și aflăm raspunsul folosind un algoritm de tip brute-force. $\mathcal{O}\left(Q \cdot N \log N\right)$

8.3 Soluția 57

Soluția bună, dar cu constanta proastă. Ne folosim de algoritmul lui MO și de un arbore de intervale pentru a actualiza în $\mathcal{O}(\log N)$. Cum actualizăm? Păi, coeficientul fiecărui t_i este dat de numărul de elemente mai mari ca t_i. Atunci când dam update, adunăm la rezultat ce valori sunt $< t_i + t_i \cdot$ câte numere sunt $> t_i$.

8.4 Solutia 100

Soluția bună, constantă bună, folosirea arborilor indexați binari fiind recomandată în cazul de față. O soluție implementată cu arbori indexați binari poate lua 100 puncte.

8.4.1 Soluție alternativă

Putem împărți valorile din vectorul t pe bucăți de radical, iar când facem update, îl vom efectua doar pe bucketul de care aparține t_i. Apoi putem calcula rezultatul în $\mathcal{O}\left(\sqrt{(valmax)}\right)$. Complexitatea devine $\mathcal{O}\left(Q\sqrt{N} + \frac{Q \cdot \sqrt{N} \log N}{2}\right)$
Demonstrația că soluția aceasta funcționează mai rapid :

$$
\begin{aligned}
&Q\sqrt{N} + \frac{Q\sqrt{N}\log N}{2} \leq Q\sqrt{N}\log N \\
&2Q\sqrt{N} + Q\sqrt{N}\log n \leq 2Q\sqrt{N}\log N \\
&2Q\sqrt{N} \leq Q\sqrt{N}\log N \\
&2 \leq \log N
\end{aligned}
$$

În teorie, pentru $N \geq 4$, vom avea o performanță mai bună.

Bonus: rezolvați problema folosind structuri de date persistente sau online în $\mathcal{O}\left(Q\sqrt{N}\right)$

8.4.2 Cod sursă

Soluție de 100

Soluție alternativă

9 Gya-chan and the gcd operation

Autor: Ionescu Matei

9.1 Soluție 13 puncte

După fiecare modificare apelăm funcția f. Complexitate timp $\mathcal{O}\left(q \cdot n^2\right)$.

9.2 Soluția 69 puncte

Definiția 9.2.1. *$\varphi(n)$ = câte numere mai mici sau egale cu n sunt prime cu acesta.*

Definiția 9.2.2. *$[p]$ = 1, dacă p e o propoziție adevărată, 0 în caz contrar.*

Observație. $\sum_{d|n} \varphi(d) = n$.

Putem deduce deci că

$$\sum_{d|\text{CMMDC}(a,b)} \varphi(d) = \sum_{d=1}^{\min(a,b)} \varphi(d) \cdot [d \mid a] \cdot [d \mid b] = \text{CMMDC}(a,b)$$

Rezultatul problemei devine:

$$ans = \sum_{i=2}^{n} \sum_{d|A_i} \varphi(d) \cdot fr_i(d),$$
$$fr_i(d) = \sum_{j=1}^{i-1} [d \mid A_j]$$

Ca să răspundem la queriuri trebuie să ținem cumva evidența asupra divizorilor din fiecare grupă cu aceeași valoare în B. Deorece numărul de elemente distincte ≤ 300, putem să reținem pt fiecare valoare distinctă din B câte un unordered map. Complexitatea temporală
$\mathcal{O}\left(N + N \cdot \sqrt{N} + Q \cdot \sqrt{N}\right)$.

9.3 Soluția suta de puncte

Dar nu ne trebuie unordered map, **do we?** Punem cu vectori în loc de map. Și aparent intră și fără să precalculezi divizorii. Soluția alternativă e:

$$\mathcal{O}\left(N \cdot logN + (N + Q) \cdot \sqrt[3]{N}\right)$$

9.3.1 Cod sursă

Soluție de 100