Editorial Fall FLASG

21 Decembrie 2024

Cuprins

1 Multumiri *Comisia RoAlgo* **4**

2 Earnings *Dimitriu Flaviu* **5**

2.1 Soluția oficială 5

2.1.1 Cod sursă 5

3 Fortnite *Ardelean Raul* **6**

3.1 Soluția oficială 6

3.1.1 Cod sursă 6

4 Bricodeque *Ardelean Raul* **7**

4.1 Soluția oficială 7

4.1.1 Cod sursă 7

5 Biblioteca *Ardelean Raul* **8**

5.1 Soluția oficială 8

5.2 Soluție parțială 11

5.2.1 Cod sursă 11

6 LAN Party *Ardelean Raul* **12**

6.1 Brut 12

6.2 Soluție parțială 12

6.3 Soluția oficială 14

6.3.1 Cod sursă 15

1 Multumiri

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

- Dimitriu Flaviu, Ardelean Raul, 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;
- Matei Ionescu, Chertes Andrei, Neagu Stefan, Simion, testerii concursului, care au dat numeroase sugestii și sfaturi utile pentru buna desfășurare a rundei;
- Ștefan Dăscălescu, coordonatorul rundei;
- Comunității RoAlgo, pentru participarea la acest concurs.

2 Earnings

Autor: Dimitriu Flaviu

2.1 Soluția oficială

Se folosește un map sau un unordered map pentru a se menține pentru fiecare jucător scorul acestuia. La fiecare final de iterație / kill, se menține jucătorul cu cele mai multe puncte care la final se afișează, precum scorul lui ASG care se afișează indiferent de punctajul său. Acest lucru se face foarte ușor cu line.find() și trebuie să fie ținut în minte faptul că soluția nu va lua maxim dacă nu se ia în calcul faptul că pentru un victory royale (ultimul kill) se duplică punctajul.

2.1.1 Cod sursă

Soluție de 50

3 Fortnite

Autor: Ardelean Raul

3.1 Soluția oficială

Ne vom lua o matrice tridimensională inițializată dinamic. Pentru fiecare insulă în parte, ne vom stoca o matrice și vom marca zona unde se află aceasta insulă.
Pentru a afla în O(1) dacă avem toată insula într-o zonă, vom face sume parțiale pe matrice pentru fiecare insulă în parte. Astfel, vom verifica pentru fiecare insulă dacă se află în totalitate în zona cerută.
Complexitate: $\mathcal{O}(N \cdot M \cdot K)$

3.1.1 Cod sursă

Soluție de 100

4 Bricodeque

Autor: Ardelean Raul

4.1 Soluția oficială

Vom simula cele K case de marcat.
În prima instanță, vom sorta intrările clienților în zona caselor de marcat. Primul venit, primul servit. Dacă sunt 2 sau mai mulți clienți care ajung în același moment, se va sorta crescător în funcție de poziția lor la citire.
Prima dată vom scoate toți clientți din cozile caselor de marcat care au terminat de plătit până la timpul intrări clientului i, inclusiv cei care au terminat de plătit în momentul când intră clientul i în zona de marcat.
Apoi îl vom repartiza la casa la care acesta va ieși cel mai rapid.
Complexitate: $\mathcal{O}(N \cdot K)$

4.1.1 Cod sursă

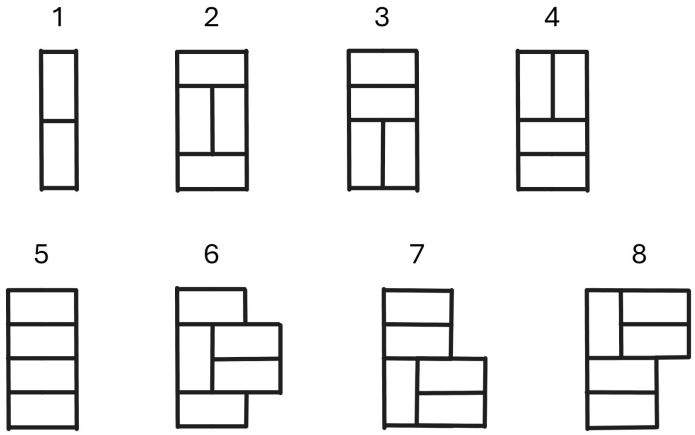
Soluție de 150

5 Biblioteca

Autor: Ardelean Raul

5.1 Soluția oficială

În total există 8 poziționări diferite cu care putem începe, acestea sunt:



Să notăm $F(N)$- numărul de configurații distincte pe care le putem obține cu cele 8 poziționări până la poziția N astfel încât nu există nici-un spațiu gol. Dacă folosim doar primele 5 poziționări, vom avea o formulă liniară de tipul:

$$F(N) = F(N-1) + 4 \cdot F(N-2) + X$$

Să notăm $F_k(N)$ - configurații distincte pe care le putem obține folosind ultima dată a k poziționare.
Formula devine:

$$F(N) = F(N-1) + 4 \cdot F(N-2) + F_{6,7,8}(N)$$

Acuma trebuie să aflăm configurația finală folosind și ultimele 3 poziționări. A 6 - a configurație este o extensie a configurației a 2 - a sau a 6 - a, eliminând cartea din mijloc stânga. Astfel, rezultă formula:

$$F_6(N) = F_{2,6}(N-2)$$

A 7 - a configurație este o extensie a configurației a 1 - a, a 3 - a sau a 7 - a, eliminând cartea de jos stânga. Astfel, rezultă formula:

$$F_7(N) = F_{1,3,7}(N-2)$$

Ultima configurație este o extensie a configurației a 1 - a, a 4 - a sau a 8 - a, eliminând cartea de sus stânga. Astfel, rezultă formula:

$$F_8(N) = F_{1,4,8}(N-2)$$

Formula generală devine:

$$F(N) = F(N-1) + 4 \cdot F(N-2) + F_6(N) + F_7(N) + F_8(N) =$$
$$F(N-1) + 4 \cdot F(N-2) + F_{2,6}(N-2) + F_{1,3,7}(N-2) + F_{1,4,8}(N-2) =$$
$$F(N-1) + 4 \cdot F(N-2) + F(N-2) + F_1(N-2) - F_5(N-2) =$$
$$F(N-1) + 5 \cdot F(N-2) + F(N-3) - F(N-4)$$

Pentru a afla în timp logaritmic valoarea $f(N)$, va trebui să rescriem formula sub forma unei matrice:

$$\begin{pmatrix} 1 & 5 & 1 & -1 \\ 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \end{pmatrix}$$

Dacă o ridicăm la $N-2$ și după înmulțim cu matricea primilor 4 termeni, aceasta fiind

$$\begin{pmatrix} 11 & 5 & 1 & -1 \end{pmatrix}$$

o să obținem pe prima linie a matricei $4 \cdot 4$ valorile: $F(N-1)$, $5 \cdot F(N-2)$, $F(N-3)$, $-F(N-4)$.
Ultima parte a problemei este de a afla în câte modalități putem umple K rafturi astfel încât oricare 2 rafturi să difere ca și aranjare a cărților.
O putem afla ușor aplicând formula: $\binom{F(N)}{K}$.
Atenție la aritmetica modulară.
Complexitate: $\mathcal{O}\left(log(N)\right)$

5.2 Soluție parțială

Se mai poate rezolva utilizân DP on Broken Profile, dar suntem limitați la timp și la memorie.
Complexitate: $\mathcal{O}\left(N \cdot M \cdot 2^N\right)$

5.2.1 Cod sursă

Soluție de 275
Soluție cu DP on Broken Profile

6 LAN Party

Autor: Ardelean Raul
Problema se reduce la forma: ”Aflați tree-ul format din extremitățile nodurilor din secvența [l,r] până la lca-ul lor, iar apoi să calculăm suma nodurilor din acest arbore”.
Cum orice prieten poate aștepta într-un oraș, orice nod din acest arbore poate să fie o variantă validă pentru a organiza LAN party-ul.
Pentru a ne ușura calculele, vom spune că orașul gazdă pentru LAN party va fi LCA-ul nodurilor din secvența [l,r].

6.1 Brut

Vom parcurge lanțurile de la fiecare prieten până la lca-ul lor.
Complexitate: $\mathcal{O}\left(Q \cdot N\right)$

6.2 Soluție parțială

Pentru a afla în $\mathcal{O}$ (1) LCA-ul nodurilor din secvență, vom liniariza arborele folosind tehnica de la ciclul eulerian și ne vom nota timpul în care am intrat într-un nod și ultimul timp cand am ieșit dintr-un nod.
Dacă căutăm în secvența $[tin[node], tout[node]]$ (tin[i] = timpul minim în care am intrat prin parcurgearea euleriană, respectiv tout[i] = timpul cand

am ieșit din nodul i prin parcurgerea euleriană) nodul minim, acesta va fi chiar LCA-ul secvenței.
Acest lucru se poate precalcula în $\mathcal{O}\left(N \cdot log(N)\right)$ folosind Range Minimum Query, astfel vom avea în $\mathcal{O}$ (1) lca-ul nodurilor din [l,r].
Pentru a eficientiza operațiile, vom folosi tehnica Square Root Decomposition. Această ne va aduce o complexitate de $O((F+Q) \cdot sqrt(F) \cdot constupdate)$ pentru toate query-urile.
Acuma trebuie să ne gândim. Cum putem adaugă un prieten sau să îl scoatem într-un timp eficient?
Dacă vrem să adugăm un prieten, vom avea 4 cazuri:

- El este primul prieten, astfel petrecerea se va desfașura în orașul său.
- Există un oraș până la lca-ul celorlalți prieteni prin care trece unul sau mai mulți prieteni
- LCA-ul prietenilor se schimbă. Astfel ceilalți prieteni se vor duce în noul oraș, respectiv prietenul pe care îl vom adăuga va merge direct spre noul oraș
- Orașul în care se află prietenul este parcurs de un alt prieten => nu adaugăm nimic

Daca vrem să ștergem un prieten, vom avea tot 4 cazuri:

- El este ultimul prieten => ștergem nodul curent
- Există un oraș în arborele format din ceilalți prieteni astfel încât unul sau mai mulți prieteni trec prin acel oraș => ștergem nodurile din path-ul format de la nodul curent până la orașul unde se întâlnește cu alți prieteni.

- LCA-ul prietenilor se schimbă. Astfel, ceilalți prieteni nu vor mai merge până la LCA-ul anterior, fiind mai departe decât LCA-ul lor => ștergem path-ul de la node -> LCA anterior, respectiv tatăl LCA-ului nou -> LCA vechi

- Orașul în care se află prietenul este parcurs de un alt prieten => nu ștergem nimic

Update pe path-uri + timp eficient? HLD strikes again, astfel putem updata în $\mathcal{O}\left(log(N)\right)$ orice path din arbore folosind tehnica lazy propagation. Pentru a afla orașul în care ne vom întâlni cu un prieten, vom folosi tehnica Binary Lifting. Așadar, fiecare update poate avea maxim $\mathcal{O}\left((logN)^3\right)$ operații, astfel complexitatea finală este: $\mathcal{O}\left((F+Q)\cdot sqrt(N)\cdot(logN)^3)\right)$
Dacă comparăm complexitățile dintre Brut și această soluție, observăm că în teorie brutul intră mai rapid. Există un caz special când MO intră mai rapid, acesta fiind: arborele este o stea și multe query-uri se repetă, adică numărul de query-uri distincte este mic.

6.3 Soluția oficială

Arbore + subarbore nou format din anumite noduri?
Ne duce cu gândul la Virtual Tree / Auxiliary Tree. Suma prietenilor din query-uri ≤ 1 000 000 ne oferă un hint esențial pentru Virtual Tree, deoarece acesta are complexitate $\mathcal{O}\left(F*log(F)\right)$ pentru fiecare query.
Așadar, complexitatea finală este $\mathcal{O}\left(Q\cdot F\cdot log(F)\right)$, dar este amortizat la $\mathcal{O}\left(S\cdot log(S)\right)$, unde S este suma prietenilor din cele Q query-uri.

6.3.1 Cod sursă

Soluție de 400

Soluție cu HLD

Brut