Editorial RoAlgo Back To School 2025

6 Septembrie 2025

Copyright © 2025 RoAlgo

Această lucrare este licențiată sub Creative Commons Atribuire-Necomercial-Partajare în Condiții Identice 4.0 Internațional (CC BY-NC-SA 4.0) Aceasta este un sumar al licenței și nu servește ca un substitut al acesteia. Poți să:

- **Distribui:** copiază și redistribuie această operă în orice mediu sau format.
- **Adaptezi:** remixezi, transformi, și construiești pe baza operei.

Licențiatorul nu poate revoca aceste drepturi atât timp cât respectați termenii licenței.

- **Atribuire:** Trebuie să acorzi creditul potrivit, să faci un link spre licență și să indici dacă s-au făcut modificări. Poți face aceste lucruri în orice manieră rezonabilă, dar nu în vreun mod care să sugereze că licențiatorul te sprijină pe tine sau modul tău de folosire a operei.
- **Necomercial:** Nu poți folosi această operă în scopuri comerciale.
- **Partajare în Condiții Identice:** Dacă remixezi, transformi, sau construiești pe baza operei, trebuie să distribui contribuțiile tale sub aceeași licență precum originalul.

Pentru a vedea o copie completă a acestei licențe în original (în limba engleză), vizitează:
https://creativecommons.org/licenses/by-nc-sa/4.0

Cuprins

1 Multumiri

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

- Robert-Sebastian Moldovan, Radu-Teodor Prosie, Flaviu Dimitriu și Iuli Morariu, autorii problemelor și laureați la concursurile de informatică și membri activi ai comunității RoAlgo;
- Susan Margine, coordonatoarea rundei;
- Darius Ramon Bejan, Ștefan Neagu și Raul Ardelean, testerii concursului, care au dat numeroase sugestii și sfaturi utile pentru buna desfășurare a rundei;
- Algolymp, pentru premiile generoase oferite și pentru organizarea webinarelor de după concurs.
- Alex Vasiluță, fondatorul și dezvoltatorul principal al Kilonova;
- Ștefan Alecu, creatorul acestui șablon LaTeXpe care îl folosim;
- Comunității RoAlgo, pentru participarea la acest concurs.

2 Wolves and sheep

Autor: Radu-Teodor Prosie

2.1 Soluția oficială

Notăm cu L numărul de lupi, cu O numărul de oi și cu K capacitatea bărcii. Împărțim problema în 3 cazuri :

Cazul $O > L$
Putem duce animalele unul câte unul astfel : alternăm între oaie și lup până când rămânem fără lupi iar apoi ducem restul oilor.

Cazul $O = L$
Dacă $K > 1$, atunci vom duce cu barca O perechi de oaie - lup. Dacă $K = 1$, atunci putem duce animalele dacă și numai dacă $O \leq 2$.

Cazul $L > O$
Demonstrăm că este necesar și suficient să avem $K \geq O$ și $K \geq L - O$ pentru a putea muta animalele. Necesitate : dacă avem $K < O$, atunci vom fi forțați să despărțim oile pe două maluri la un moment dat și ele vor fi mâncate pe unul dintre maluri. Dacă avem $K < L - O$, atunci o încercare este să mutăm toate oile în prima mutare. Presupunem că putem câștiga. Astfel, la ultima mutare aveam mai puțini de $L - O$ lupi pe malul stâng, dar asta ar însemna că avem

mai mulți de O pe malul drept, deci am fi făcut o mutare pierzătoare înainte, deci cazul în care ducem toate oile primele e imposibil. Dacă în prima mutare decidem să ducem lupi, trebuie să ducem cel puțin $L - O$, pentru a nu lăsa lupii în majoritate pe malul stâng, dar nu putem (din ipoteză). Astfel, ambele condiții sunt necesare, iar suficiența lor este evidentă. O soluție este aici.

3 Up-Right

Autor: Robert-Sebastian Moldovan

3.1 Soluția oficială

Având un query de forma N, M, K, facem următoarele observații:

- Vom avea $N - K$ pași în sus, $M - K$ pași în dreapta, respectiv K pași în diagonala;
- O secvență de mutări poate fi reprezentată unic de o secvență de caractere (U pentru sus, R pentru dreapta și D pentru sus-dreapta).

Așadar, putem reduce problema la a afla câte șiruri de caractere de lungime $N + M - K$ există cu $N - K$ caractere U, $M - K$ caractere R, respectiv K caractere D. Inițial, avem $N + M - K$ locuri libere, din care putem amplasa caracterele U oricum ($\binom{N+M-K}{N-K}$). Rămân M locuri libere, din care plasăm cele $M - K$ caractere R ($\binom{M}{M-K}$). Rămân de amplasat cele K caractere R ($\binom{K}{K} = 1$). Așadar, răspunsul este $\binom{N+M-K}{N-K}\binom{M}{M-K}$.

Punctajul obținut mai departe ține de modul în care sunt calculate combinările. Pentru 100 de puncte, este necesară precalcularea factorialelor, respectiv inverselor modulare ale factorialelor în $O(N_{max} + M_{max})$.

3.1.1 Cod sursă

Soluție de 100

4 LLM

Autori: Flaviu Dimitriu, Robert-Sebastian Moldovan

4.1 Soluția oficială

Construim un graf parțial al arborelui în care adăugăm muchiile între perechile de noduri a și b cu una dintre următoarele proprietăți:

- există muchie între nodurile a și b în arborele dat;
- a sau b este un nod de tip A;
- a și b sunt noduri de tip B și au aceleași valori.

Răspunsul la un query este suma mărimilor componentelor distincte ale nodurilor din query. Componentele se pot construi cu un DFS sau cu un DSU.

4.1.1 Cod sursă

Soluție de 100

5 Make Increasing

Autor: Robert-Sebastian Moldovan

5.1 Soluția 1 ($O(N)$) - 5 puncte

În acest subtask, șirul este sortat crescător. Astfel, bineînțeles că nu avem nevoie să facem modificări niciuneia din secvențe.

5.2 Soluția 2 ($O(N)$) - 15 puncte

În acest subtask, șirul este sortat descrescător. Astfel, parcurgând șirul în dreapta începând din poziția i, maximul pe prefix rămâne mereu A_i. Astfel, pentru un sufix $[i, N-1]$, adăugăm la răspuns $\frac{(N-i)(N-i+1)}{2} A_i - \sum_{j=i}^{N-1} \sum_{k=i}^{j} A_k$. Calculam $S_i = \sum_{j=i}^{N-1} \sum_{k=i}^{j} A_k$, folosind contributia lui A_i, în $O(N)$.

5.3 Solutia 3 ($O(N^2)$) - 30 de puncte

Pornim din fiecare $i < N$ și menținem variabilele *max*, *sum* și *ans*. Parcurgem j de la i la $N-1$ si *max* devine $max(max, A_j)$. La *sum* adaugam costul minim pentru a aduce A_j la *max* ($max - A[j]$). La răspunsul total (*ans*) adaugam *sum*.

5.4 Solutia 4 ($O(NlogMAX)$) - 60 de puncte

In acest subtask, sirul este generat aleator cu valori de la 0 la MAX. Astfel, exista $O(logMAX)$ schimbari de maxim incepand din orice pozitie. Avand precalculate cu o stiva pozitiile $next_i = j$-ul minim pentru care $A_i < A_j$, parcurgem i de la 0 la $N-1$ si j de la i la $N-1$, dar sarim peste toate pozitiile nesemnificative. Calculam contributia pe segmentul $[j, next_j - 1]$ ca fiind $\frac{(next_j - j)(next_j - j + 1)}{2} A_j - \sum_{j=i}^{next[j]-1} \sum_{k=i}^{j} A_k$. Contributia pe segmentul $[next_j, N-1]$ este $(next_j - j)(N - next_j)A_j - (N - next_j) \sum_{k=j}^{next_j - 1} A_k$.

5.5 Solutia 5 ($O(N)$) - 100 de puncte

Folosim o stiva, in mod asemanator solutiei anterioare, doar ca o sa calculam raspunsul in timp ce mentinem stiva. Avand o stiva cu elementele $\{a, b, c, ...\}$, A_a este maxim pe $[a, b-1]$, A_b este maxim pe $[b, c-1]$, etc. Folosim aceeasi formula de mai sus, doar ca mentinem sumele de maxime pe prefix si sumele de prefixe separat. Astfel, cand eliminam un indice din stiva, scadem toata contributia lui. Cand adaugam un indice in stiva, crestem cu contributia lui.

5.5.1 Cod sursă

Soluție de 100

6 John isi renoveaza ferma

Autor: Iuli Morariu

6.1 Subtask 1 - 10 puncte

In acest Subtask, graful este doar o linie si nu avem operatii de tipul 1, deci il putem considera o lista. Operatiile de tipul 2 se manifesta ca XOR pe sufix, de la nodul x, x + 1, ... , n. Puteam chiar efectua operatiile.

6.2 Subtask 2 - 20 puncte

Graful are aceeasi forma ca la subtaskul anterior, doar ca trebuie sa optimizam intrebarile. Putem face asta cu o structura similara cu smenul lui mars.

6.3 Subtask 3 - 30 puncte

Graful nu se modifica in forma, doar in valoare. Pentru fiecare operatie de tipul 2 putem parcurge tot subarborele nodului x.

6.4 Subtask 4 - 50 puncte

Aici, la fel, graful nu se modifica in forma, doar in valoare. O solutie este sa facem turul euler (avem o lista, sa ii zicem v. Parcurgem arborele DF. de fiecare data cand intram intr-un nod si cand iesim, il adaugam in lista. pe exemple unul dintre posibilele tururi ar fi : 1 2 2 3 5 5 3 4 4 1. Ne ajuta pentru ca fiecare subarbore este o secventa continua.) si pentru fiecare operatie updatam cu o structura similara cu smenul lui mars sau un arbore de intervale valorile fiecarui nod. Valoarea unui nod va fi valoarea uneia dintre pozitiile unde apare in tur.

6.5 Subtask 5 - 100 puncte

Initial, parcurgem toate operatiile si construim arborele final (dupa toate operatiile de tipul 1). Apoi, facem turul euler (ca la subtaskul 4). De acolo avem tot ce ne trebuie pentru a rezolva problema. Parcurgem din nou operatiile. Pentru operatiile de tip 2, doar facem XOR pe intervalul care reprezinta subarborele lui x. Pentru operatiile de tip 1, vom reseta toata secventa nodului respectiv (care este adaugat) la 0.
La toate subtaskurile: pentru a determina valoare ceruta in output se va folosi un map.
Soluție de 100

7 Closed room mystery

Autor: Radu-Teodor Prosie

7.1 Definiții

Notăm cu $C(i)$ cea mai mică mulțime închisă care îl conține pe i. Ea poate fi generată incremental în $O(N^2)$, pornind de la elementul i și inserând toate elementele ce devin necesare pe parcurs. Spunem că o mulțime închisă care nu conține strict o altă mulțime închisă este ”bună”.

7.1.1 Subtask 1 - 30 de puncte

Observăm că pentru ca o mulțime să fie bună, atunci ea trebuie sa fie egală cu $C(x)$, pentru toate elementele x din mulțime. Într-adevăr, dacă există x, y în o mulțime bună A pentru care $C(x) \neq C(y)$, una dintre ele, fie ea $C(x)$, trebuie să aiba cardinal mai mare decat $C(y)$, și implicit cardinal mai mare decat A (deoarece nu poate avea cardinalul mai mic), ceea ce ar contrazice faptul ca mulțimea A este închisă (din faptul ca $C(x)$ este mulțimea minimală închisă care îl conține pe x). Astfel, avem următoarea soluție: generăm toate mulțimile $C(i)$ și verificăm pentru fiecare element din ea daca au acelasi $C(i)$. Complexitate temporală : $O(N^3)$

7.1.2 Soluție completă

Avem două soluții, una bazată pe grafuri, alta pe randomizare.

Soluția bazată pe grafuri

Construim un graf pe $N \cdot (N+1)/2$ noduri, fiecare nod reprezentând o pereche (i, j) cu $i \leq j$, iar de la perechea (i, j) tragem următoarele arce de implicație (cu semnificația că dacă avem această pereche, avem și perechile la care se trage arc) : de la (i, j) la fiecare pereche ordonată din mulțimea $\{i, j, f[i][i], f[j][j], f[i][j], f[j][i]\}$, adică (i, i), (i, j), $(i, f[i][i])$ etc... Acum, compresăm acest graf în graful aciclic al componentelor tare conexe. Deoarece o mulțime închisă este mulțimea determinată de un set de noduri cu proprietatea că nu exista arce de la un nod din set către un nod care nu e în set, se observă că mulțimile căutate sunt cele determinate de componentele tare conexe care au gradul de ieșire 0. Pentru a determina mulțimea determinată de o astfel de componentă, este suficient să vedem care perechi de tipul (i, i) aparțin componentei. Complexitatea temporală este $O(N^2)$. De menționat că această soluție nu se încadrează în timp dacă folosiți algoritmul Kosaraju-Sharir pentru determinarea CTC-urilor. Trebuie folosit algoritmul lui Tarjan / altul cel puțin la fel de eficient. O soluție este aici.

Soluția bazată pe randomizare

Presupunem că am avea toate mulțimile $C(i)$ distincte, fie ele V. Atunci am putea aplica următoarea soluție: sortăm crescător după mărime mulțimile din V și le parcurgem pe rând. Pentru ca mulțimea V_i să facă parte din răspuns, este necesar și suficient să nu se intersecteze cu vreo mulțime pe care am desemnat-o ca răspuns înaintea ei deoarece dacă se intersectează în X de exemplu, avem că mulțimea de dinainte, fie ea V_j, este egală cu $C(x)$ (din observația de mai sus) de unde rezultă că $C(x) \subset V_i$ din cauza faptului că

$V_i \neq V_j$ și $|V_j| \leq |V_i|$. Această verificare poate fi facută prin marcarea unui vector caracteristic atunci când adăugăm o mulțime la răspuns și consultarea acestuia atunci când vrem să verificăm dacă mulțimea curentă este parte din răspuns. Astfel, acum trebuie doar să găsim mulțimile $C(i)$ distincte. Vom avea o abordare hibridă : parcurgem fiecare i și încercăm să generăm $C(i)$ cât timp nu depășește mărimea B, ceea ce practic ne lasă doar cu $C(i)$ - urile cu cardinal mai mare decât B, iar la final facem T runde în care alegem un element x la întamplare și generăm $C(x)$ indiferent de cât de mare e. Complexitatea temporală este $O(N \cdot B^2 + T \cdot N^2)$. Noi am ales $B = 300$ și $T = 50$, soluție care are o șansă de 0.001% să pice pe un test, în cel mai rău caz. Soluția o găsiți aici

8 xnsgraph

Autor: Robert-Sebastian Moldovan

8.1 Soluția oficială

Problema cere pentru fiecare query $[l, r]$, sa determinam daca graful partial obtinut prin folosirea muchiilor din $[l, r]$ are un nod cu grad mai mare sau egal cu 4 si un lant de cel putin 6 noduri. In acest editorial vom explora solutiile subtask-urilor si solutia optima.

8.2 Solutia 1 ($O(N^2 * Q)$) - 10 puncte

Construim un graf nou si facem cate o parcurgere din fiecare nod, pentru a afla daca exista un nod in componenta sa care este aflat la distanta >= 5.

8.3 Solutia 2 ($O(N * Q)$) - 30 de puncte

Procedam la fel ca si la solutia 1, dar aflam diametrul fiecarei componente in $O(comp\_size)$, in loc de $O(comp\_size^2)$.

8.4 Solutia 3 ($O((N+Q)logN+Q)$) - 50 de puncte

Prima data vom explora o structura de date care ne lasa sa efectuam urmatoarele operatii

- adaugarea unei muchii intre doua noduri din componente diferite (mai exact, activarea unei muchii deja existente in arbore);
- determinarea unei perechi de noduri aflate la distanta maxima intr-o componenta.

Pentru prima operatie, putem folosi o structura Union-Find. Pentru a doua operatie, sa zicem ca vrem sa activam o muchie intre doua noduri din componentele A, respectiv B. Presupunand ca stim diametrele corespunzatoare componentelor A si B, putem afla diametrul noii componente, incercand 6 perechi candidate ($\{A_x, A_y\}, \{B_x, B_y\}, \{A_x, B_x\}, \{A_x, B_y\}, \{A_y, B_x\}, \{A_y, B_y\}$). Distantele intre acesti candidati pot fi calculate in $O(1)$ folosind LCA cu RMQ pe Euler's Tour.
Deoarece $l_i = 1$ pentru toate query-urile, putem sa sortam query-urile dupa capatul drept si sa adaugam muchii pe masura ce avem nevoie si sa actualizam un vector de frecventa pentru grade si Union-Find-ul mentionat anterior pentru diametre.

8.5 Solutia 4 ($O(?)$) - 69 de puncte

Acest subtask nu corespunde unei solutii anume, el este un simplu failsafe pentru solutii generale mai eficiente decat $O(NQ)$, dar nu suficient de

eficiente pentru 100 de puncte.

8.6 Solutia 5 ($O((N+Q)\sqrt{N}logN)$) - 100 de puncte

In loc de Union-Find-ul mentionat anterior, putem modifica putin structura sa nu mai comprimam lanturile cand le parcurgem. In schimb, este absolut necesar sa unim nodurile dupa marimea componentelor, altfel parcurgem $O(N)$ noduri in cel mai rau caz. Cu aceasta modificare, si tinand o stiva de modificari, putem da foarte usor ”undo” la fiecare ultimă operatie.
Cu aceasta structura, putem folosi algoritmul lui Mo cu rollback (impartim query-urile in $O(\sqrt{N})$ block-uri dupa capatul stang). In cadrul fiecarui block, sortam query-urile crescator dupa capatul drept. Astfel, baleiem peste query-uri si abia la sfarsit scadem pointer-ul stang (de cel mult $O(\sqrt{N})$ ori). Dupa ce obtinem raspunsul query-ului, crestem pointer-ul stang inapoi la loc si dam ”undo” la toate muchiile adaugate in partea stanga.

8.6.1 Cod sursă

Soluție de 100
Nota autorului: folosind o structură DSU cu queue-like undo (implementarea cozii cu o singură stivă), se poate obtine o soluție în $O(Nlog^2(N)+Q)$.