Editorial FiFo | RoAlgo Educational Contest 1

13 Iulie 2024

Cuprins

1 Mulțumiri

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

- Ardelean Raul, Pascale Cosmin-Gabriel, Ștefan-Cosmin Dăscălescu, 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;
- Adelin Hojda, Matei Ionescu, Susan, Traian Mihai Danciu, testerii concursului, care au dat numeroase sugestii și sfaturi utile pentru buna desfășurare a rundei;
- Ardelean Raul, coordonatorul rundei;
- Comunității RoAlgo, pentru participarea la acest concurs.

2 SumDivK

Autor: Ardelean Raul

2.1 Soluții 30 puncte

2.1.1 Brute Force

Cum $N \leq 1\,000$, ne permite sa abordăm un algoritm de tip brute force, astfel complexitatea va fi $\mathcal{O}\left(N^3\right)$.

2.1.2 $\mathcal{O}\left(R^2\right)$

Vom folosi o abordare bazată pe contorizarea modului în care se distribuie resturile modulo R a valorilor din șir. Vom crea un vector de frecvență care contorizează de câte ori apare fiecare rest r de la 0 la $R-1$. Apoi vom parcurge fiecare pereche (i, j) unde $i < j$ și vom calcula restul necesar pentru un număr k astfel încât suma să fie un multiplu de R. Apoi va trebui să luăm câteva cazuri:

- $v[i] = v[j] = v[k]$
- $v[i] = v[j], v[i] \neq v[k], v[j] \neq v[k]$
- $v[i] = v[k], v[i] \neq v[j], v[j] \neq v[k]$

- $v[i] \neq v[j] \neq v[k]$

2.2 Soluția oficială

Vom avea 2 cazuri de indeplinit:

- Primul caz ar fi să creăm o pereche de 2 valori (i, j), având restul sumei la R o valoare q. Aici pentru o poziție j, vom adăuga într-un vector $M2$ câte numere avem în intervalul $[1, j)$ astfel încât, dacă adunăm cu valoarea de pe poziția j va da restul sumei la R valoarea q.
- Al doilea caz va fi asemănător cu primul caz, doar că aici vom calcula pentru tripletul (i, j, k). Pentru o poziție k, vom afla câte perechi de forma (i, j) avem în intervalul $[1, k)$ astfel încât dacă adunăm cele 3 numere să rezulte un multiplu de-ai lui R, iar aici ne vom folosi de valorile obținute din vectorul $M2$ din primul caz.

Complexitate: $\mathcal{O}(N \cdot R)$

2.2.1 O mică optimizare

Putem observa că algoritmul de complexitate $\mathcal{O}(N \cdot R)$ poate să fie mai încet față de cel de complexitate $\mathcal{O}\left(R^2\right)$, doar atunci când $R < N$. Astfel putem folosi codul de complexitate $\mathcal{O}\left(R^2\right)$ când $R < N$, iar codul de complexitate $\mathcal{O}(N \cdot R)$ când $R > N$.

2.2.2 Cod sursă

Soluție de 100, amortizare
Soluție de 100, $\mathcal{O}(N \cdot R)$

3 binarray

Autor: Ștefan-Cosmin Dăscălescu

3.1 Observație cheie

Cea mai importantă observație pentru a rezolva această problemă este aceea că în loc să calculăm câte subsecvențe au OR-ul egal cu 1, vom calcula câte au OR-ul egal cu 0 și le vom scădea din numărul total de secvențe. Pentru ca o secvență să aibă OR-ul egal cu 0, trebuie ca toate valorile să fie egale cu 0. Aflarea secvențelor pline de 0 va fi ceea ce vom încerca să optimizăm de-a lungul problemei.

3.2 Soluții parțiale

Pentru subtask-urile mici, putem parcurge secvența din query și să calculăm lungimile secvențelor cu zerouri, folosind o sumă Gauss în proces. Fie că facem asta pentru fiecare subsecvență sau facem asta doar pentru secvențele din query, aceste soluții obțin între 7 și 20 de puncte.

3.3 Soluții folosind structuri de date

Pentru următoarele două subtaskuri, se pot ține structuri de date care să păstreze date legate de lungimile secvențelor care au zerouri, precum și a numărului de subsecvențe cu OR-ul 0. Pentru a păstra aceste lucruri, se poate folosi un arbore de intervale sau pentru subtaskul 3, o prelucrare a query-urilor folosind algoritmul lui Mo. Aceste soluții obțin 56 sau 70 de puncte, în funcție de implementare.

3.4 Soluție optimă

Pentru a răspunde la query-uri în timp constant, va trebui să ne folosim de o serie de sume parțiale. Mai întâi, putem observa că dacă ținem pentru fiecare poziție inițială numărul de zerouri consecutive până la acea poziție, putem afla foarte ușor răspunsul pentru query scăzând din numărul secvențelor până la R, numărul secvențelor până la $L - 1$, ajustând apoi la început contribuția pozițiilor inițiale.
Pentru a ajusta contribuțiile pozițiilor inițiale, trebuie să aflăm lungimea primei secvențe de zerouri, dacă secvența din query începe cu un zero. Pentru a face asta, putem precalcula în mod similar, numărul de zerouri consecutive până la acea poziție de la dreapta la stânga, iar acum vom putea să aflăm cât trebuie să scădem din răspuns. Dacă de exemplu am avut până la poziția L un număr de a zerouri și avem b zerouri la dreapta lui L, inclusiv el, lungimea finală este $a + b - 1$, deci din fiecare din cele b poziții din șir, va trebui să scădem $a - 1$ din răspuns. Pentru mai multe detalii, verificați soluția oficială.

3.4.1 Cod sursă

Soluție de 100

4 Zorg

Autor: Ardelean Raul

4.1 Soluția subtask 1

Observăm din restricții că tărâmul este un tablou bidimensional. Armamentul din fiecare zonă (indice) din tărâm este o valoare fixată G și nu există danger zones și safe zones. În aceste condiți, va trebui să aflăm cu un algoritm de tip Lee/Fill cel mai scurt drum ca și timp de la Costel la Suzi. Complexitate: $\mathcal{O}(N \cdot M)$

4.2 Soluția subtask 2

Este asemănător cu subtask-ul 1, doar că avem un tablou tridimensional. Complexitate: $\mathcal{O}(N \cdot M \cdot K)$

4.3 Soluția oficială

Vom utiliza aceași idee ca la subtask-ul 2, dar se adaugă danger zones, safe zones și diferite limite de greutate în fiecare zonă al tărâmului. Pentru a eficientiza algoritmul de căutare al celui mai bun traseu, raportat la timp, vom precalcula pentru fiecare safe zone timpul minim pentru a ajunge la un

danger zone. În Lee/Fill, când ajungem la o poziție danger vom utiliza datele precalculate pentru fiecare safe zone, astfel ne vom „teleporta” într-un safe zone, doar dacă durata minimă spre acel safe zone este $\leq T$. În plus va trebui să aflăm cea mai mare cantitate de armament pe care o putem lua, ca să ajungem la Suzi și a-l repune pe tron pe Zorg. Vom folosi căutarea binară pentru a determina valoarea maximală de armament.
Complexitate: $\mathcal{O}\left(log(G) \cdot (N \cdot M \cdot K + Q \cdot S)\right)$

4.3.1 Cod sursă

Soluție de 100
Soluție de 20 și 40

5 artclass

Autor: Pascale Cosmin-Gabriel

Observația principală: formele pot fi reprezentate într-o matrice care simulează foaia pe care se „desenează”, memoria permițând acest lucru.
Astfel, problema constă în două părți:
I. delimitarea și „umplerea” formelor citite
II. determinarea valorii ariei cerute

5.1 Subtask 1 - 1 punct

Din moment ce $q = 0$, aria cerută este reprezentată de aria întregii suprafețe date, adică $n * m$.
Complexitatea temporală: $O(1)$

5.2 Subtask 2 - 20 de puncte

În primul rând, trebuie să „marcăm” în matrice formele citite. Vom realiza acest lucru cu ajutorul unui tablou tridimensional de dimensiune $n \cdot m \cdot 4$, care ne ajută să știm, pentru o celulă curentă, dacă are linii (ce fac parte din forme) pe lateralul acesteia. Având această informație, putem folosi un algoritm de tip fill pentru a ”umple” formele în matrice. Putem face asta, de

exemplu, folosind o matrice cu valori inițial nule în care celulele „umplute” vor lua valoarea 1.
Observăm că valorile lui n și m sunt relativ mici, acestea fiind maxim 50. Astfel, parcurgem fiecare submatrice din matricea inițială și verificăm dacă matricea curentă conține valori de 1. Dacă da, o ignorăm și mergem mai departe. Dacă nu, actualizăm valoarea ariei cerute de problemă cu aria submatricei curente (doar dacă aria celei curente este mai mare).
Notă: Pentru a îmbunătăți complexitatea temporală putem face sume parțiale pe matrice pentru a afla dacă o submatrice conține valori de 1.
Complexitatea temporală: $O(N \cdot (N+1) \cdot M \cdot (M+1))$

5.3 Subtask 3 - 48 de puncte

Pentru a obține punctele de la acest subtask, păstrăm metoda prin care „marcăm” în matrice formele, dar schimbăm modul în care determinăm aria cerută. Pe fiecare linie vom pretinde că avem m blocuri, fiecare din aceste blocuri având dimensiunea egală cu numărul de celule consecutive care conțin valoarea 0 pe coloana respectivă, numărând de la linia curentă în sus. După ce stabilim dimensiunea fiecărui bloc, încercăm să îl „extindem” în stânga și în dreapta, obținând astfel aria maximă ce cuprinde complet blocul respectiv. După ce aflăm această arie, actualizăm valoarea ariei cerute de problemă cu aceasta (doar dacă este mai mare).
Complexitatea temporală: $O(N \cdot M^2)$

5.4 Subtask 4 - 31 de puncte

Pentru a obține punctele de la acest subtask, schimbăm atât metoda prin care „marcăm” în matrice formele, dar și modul în care determinăm aria cerută,

întrucât limitele mici de timp și memorie nu ne permit abordările de la subtask-urile anterioare.
În ceea ce privește marcarea formelor, putem scăpa de tabloul tridimensional, înlocuindu-l cu o matrice de tip bool, în care o celulă are valoarea 1 dacă în dreapta ei are un „perete” sau valoarea 0 în caz contrar. Astfel, o valoare de 1 indică faptul că începând cu celula aflată pe aceeași linie, coloana următoare, începem, respectiv încetăm, marcarea cu valori de 1. Orice alt aspect este la fel ca în cazul subtask-urilor anterioare.
În continuare, vom înlocui modul în care determinăm aria cerută. Observăm că în locul abordării de la subtask-ul anterior, putem folosi un algoritm de tip skyline pentru a reduce timpul de execuție.
Complexitatea temporală: $O(N \cdot M)$

5.4.1 Cod sursă

Soluție pentru 1 punct
Soluție pentru 21 de puncte
Soluție pentru 69 de puncte
Soluție pentru 100 de puncte

6 Peridea

Autor: Ardelean Raul

6.1 Soluția subtask 1

Din restricții observăm că subtask-ul se rezumă la aflarea celui mai scurt drum de la artefactul inițital la cel final, raportat la timpul de descifrare. Vom folosi un algoritm de tip shortest path pentru a afla cel mai scurt drum de la src la dst. Putem aborda și un DFS / BFS.
Complexitate: $\mathcal{O}(N + M)$

6.2 Soluția subtask 2

Vom folosi aceași idee ca la subtask-ul 1, doar că avem maxim 3 artefacte demonice prin care trebuie să trecem. Putem afla toate cele 6 cazuri prin if-uri, cazurile fiind:
Pentru $K = 3$
$src \to 1 \to 2 \to 3 \to dst$
$src \to 1 \to 3 \to 2 \to dst$
$src \to 2 \to 1 \to 3 \to dst$
$src \to 2 \to 3 \to 1 \to dst$
$src \to 3 \to 1 \to 2 \to dst$

$src \to 3 \to 2 \to 1 \to dst$
Pentru $K = 2$:
$src \to 1 \to 2 \to dst$
$src \to 2 \to 1 \to dst$
Pentru $K = 1$:
$src \to 1 \to dst$
Complexitate: $\mathcal{O}((K+1)! \cdot (N+M))$

6.3 Soluția subtask 3

Ne vom folosi de ideea de la subtask-ul 2, doar că avem în plus cele T drumuri pe unde putem trece cu ajutorul lui Huyang. Ne vom folosi de programarea dinamică pentru a afla cel mai scurt drum de la un artfecat la alt artefact. Dinamica va fi de tipul: dp[i][j][k] - timpul minim de descifrare de la al i-lea artefact demonic, la artefcatul j cu k apeluri la Huyang.
Complexitate: $\mathcal{O}\left(N \cdot (N+M+T) \cdot log(M+T)\right)$

6.4 Soluția oficială

Vom folosi aceași idee ca la subtask-ul 3, doar că ne vom folosi de tehnica programare dinamică pe stări exponențiale pentru a afla cel mai scurt timp de descifrare a blestemului.
Complexitate: $\mathcal{O}\left(K! \cdot K^2 \cdot LIM^2 + N \cdot (N+M+T) \cdot log(M+T)\right)$

6.4.1 Cod sursă

Soluție de 100 puncte
Soluție de 25 puncte

Soluție de 10 puncte

7 Struct arbore{}

Autor: Ardelean Raul

7.1 Soluția subtask 1

Observăm din restricții că numărul de query-uri este ≤ 5 000, astfel în cel mai rău caz, când arborele nostru este un lanț, putem avea o complexitate liniară pentru fiecare query, astfel un algoritm de tip brute force va intra cu aceste limite.
Complexitate: $\mathcal{O}(N \cdot Q)$

7.2 Soluția subtask 2

Tipul de query-uri este doar 1, adică lucrăm pe un arbore offline. Cum noi putem avea un arbore cu 100 000 noduri, intuiți că și aici putem avea worst case scenario un arbore de tip lanț, astfel un brute force nu va intra cu siguranță pe un astfel de test. Ne vom folosi de tehnica string hashing, astfel reducem aflarea de egalitate lexicografică a două stringuri în $\mathcal{O}(1)$. Însă mai avem o problemă, cum aflăm eficient strămoșul celor 2 noduri. Ne vom folosi de tehnica binary lifting pentru a afla al k-lea strămoș al unui nod, astfel vom obține o complexitate $\mathcal{O}(logN)$. Pentru a căuta eficient al k-lea

strămoș, ne vom folosi de căutarea binară. Combinând cele 3 idei, rezultă o complexitate: $\mathcal{O}\left(Q \cdot log^2 N\right)$

7.3 Soluția oficială

Ne vom folosi de ideea de la subtask-ul 2, doar că va trebui să facem update-uri în timp eficient. Vom utiliza tehnica Heavy Path Decomposition / Heavy Light Decomposition pentru a afla în timp logaritmic valoarea string hash-ului, inclusiv update-ul unui nod va fi logaritmic.
Complexitate: $\mathcal{O}\left(Q \cdot log^3 N\right)$

7.3.1 Cod sursă

Soluție de 100
Soluție de 40
Soluție de 10