RoAlgo Contest 2 - Editorial

Horia Andrei Boeriu, Ștefan-Cosmin Dăscălescu, Darius Hanganu,
Rareș Hanganu, Andrei-Cristian Ivan, Radu Vasile, Ștefan Vîlcescu

15 Iulie 2023

1 Mulțumiri

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

- Horia Andrei Boeriu, Darius Hanganu, Rareș Hanganu, Andrei-Cristian Ivan, Radu Vasile, Ștefan Vîlcescu, autorii problemelor și laureați la concursurile de informatică și membri activi ai comunității RoAlgo.
- Alex Vasiluță, fondatorul și dezvoltatorul principal al Kilonova
- Andrei Chertes, David-Ioan Curcă, Luca Valentin Mureșan și Vlad Tutunaru testerii concursului, care au dat numeroase sugestii și sfaturi utile pentru buna desfășurare a rundei, precum și ajutor la crearea unor teste.
- Ștefan-Cosmin Dăscălescu, coordonatorul rundei și autorul a două dintre probleme.
- Comunității RoAlgo, pentru participarea la acest concurs.

2 Problema Somnoros

Autor: Andrei - Cristian Ivan

2.1 Subtask 1 ($N, Q \leq 10^3$)

Pentru această grupă de teste, se poate realiza câte o parcurgere din fiecare nod X, iar în urma parcurgerii putem obține răspunsurile la queryuri. Complexitatea soluției este $O(Q * (N + M))$, care datorită valorilor mai mari pe care le poate lua Q în testele mai mari ($Q \leq 10^5$), aceasta soluție nu poate obține mai mult de 35 de puncte.

Sursa de la această soluție poate fi accesată aici.

2.2 Subtask 2 - Soluția oficială

Ideea de bază a acestei soluții este utilizarea bitsetului. Vom reține pentru fiecare nod câte un bitset, în care vom reține pentru fiecare nod al grafului dacă poate accesa nodul curent. Inițial, fiecare nod va avea in bitsetul său valoarea 0 peste tot, cu exceptia poziției nodului curent, care va fi 1. Înainte de a precalcula răspunsurile pentru fiecare nod, va trebui realizată sortarea topologică a grafului. După realizarea sortării, noi vom parcurge nodurile în ordinea obținută și vom calcula pentru fiecare nod răspunsul, luând în calcul bitseturile nodurilor *"predecesoare"* nodului curent, acestea fiind deja calculate datorită sortării topologice.

După ce am făcut acest proces, putem începe să răspundem la queryuri: pentru fiecare pereche de noduri X Y din queryuri, ne vom uita în bitsetul nodului Y, iar răspunsul va fi valoarea de pe poziția X din bitset. Această soluție are complexitate pătratică, dar este puternic redusă de constanta bitsetului, așadar complexitatea finală este $O(\frac{N^2}{64})$ pentru precalcularea răspunsurilor, cu $O(1)$ pe query.

Sursa de la această soluție poate fi accesată aici.

3 Problema Strehaia

Autor: Rareș Hanganu

Această problemă este o aplicație a problemei rucsacului, dar implementarea clasică nu este îndeajuns de bună pentru a obține punctajul maxim.

Pentru a obține punctajul maxim, va trebui să observăm că numărul de valori distincte din rucsac este cel mult 100, ceea ce ne permite să compresăm rucsacul astfel încât numărul de obiecte de care ținem cont să fie mult redus, ceea ce va transforma factorul de x în unul de $\log x$.

Pentru mai multe detalii puteti urmari acest videoclip: Link.

Solutie

4 Problema Gadfadar4

Autor: Horia Andrei Boeriu

Vom face un vector tridimensional de frecventa pentru fiecare grupa a fiecarui numar in care punem 1 pentru fiecare cifra care apare si 0 pentru fiecare cifra care nu apare.

La fiecare cifra din fiecare query vom parcurge cele n numere si cu vectorul tridimensional vom afla puterea numerelor. La sfarsit comparam puterile si

afisam pozitia primului numar care are puterea maxima. Complexitate $O(Q * X * N)$;

Cod sursa: Solutie

5 Problema Munte

Autor: Ștefan Dăscălescu, Ștefan Vîlcescu

Pentru rezolvarea primelor câteva subtaskuri, se pot folosi diverse abordări bazate pe metode de tipul brute-force, precum generarea tuturor permutărilor sau aflarea cu ajutorul unei dinamici pe stări exponențiale a numărului de permutări ce respectă condiția din enunț.

Pentru obținerea punctajului maxim, se poate observa faptul că aceste secvente sunt asemănătoare cu Euler zigzags, recurența folosită pentru obținerea punctajului maxim fiind similară cu cea de la numerele Catalan.

Astfel, dacă definim ans_i ca fiind răspunsul pentru permutările de ordinul i,

$$2 * ans_i = \sum_{k=0}^{n} C_n^k \cdot ans_k \cdot ans_{n-k}$$

Această recurență se poate calcula în $O(n^2)$ folosind triunghiul lui Pascal, precum și calcularea răspunsurilor în ordine crescătoare.

Cod sursă: Solutie

6 Problema Vrajitoare

Autor: Darius Hanganu, Ștefan Dăscălescu

Pentru rezolvarea primelor două subtaskuri, se pot folosi metode de tipul brute force.

Pentru următoarele două subtaskuri, deoarece numărul de cifre de 1 în baza 2 este foarte mic, putem verifica fiecare număr posibil ușor folosindu-ne de șirul inițial precum și de pozițiile cifrelor de 1 atât din numărul încercat, cât și din șirul inițial.

Cel de-al cincilea subtask poate fi rezolvat și folosind o metodă de tipul digit dp unde aflăm câte numere de i cifre au j de 1 în baza dată, având grijă să nu considerăm numere mai mari decât cel dat.

Pentru obținerea punctajului maxim vom fixa la fiecare pas cifra care face diferența între numărul dat și numerele care vor fi considerate, iar pentru a calcula numărul de variante pe care îl avem la dispoziție, dacă mai avem de

fixat x cifre dintre care y dintre ele trebuie să fie 1, numărul de variante este $C_x^y \cdot (B-1)^{x-y}$.

Solutie

7 Problema Vp

Autor: Ștefan Vîlcescu, Luca Valentin Mureșan

Pentru rezolvarea acestei probleme, vom urma urmatorii pasi:

7.1 Pasul 1

Mergem prin vector si reconstituim toate valorile care se pot reconstitui. De exemplu, pentru $N = 2$, $M = 8$ si sirurile

-1 4

2 -1

Putem reconstituii valoarea -1 de pe a doua pozitie din al doilea sir, deoarece $P_i = (V_i * P_{i-1}) (\text{mod } M)$, aceasta fiind 0. si valoarea -1 de pe prima pozitie din primul sir, deoarece $P_1 = V_1$, aceasta fiind 2. Aceasta parcurgere ar trebui sa ia $O(N)$.

7.2 Pasul 2

Vom merge prin sir pana cand gasim o pereche de nr V_i si P_i sa fie diferite de -1. Cand am gasit perechea aceasta, reconstituim P_{i-1}, si vom merge inapoi in sir, pentru a reconstitui valori pe care nu le-am putut reconstitui. De exemplu, pentru $N = 4$, $M = 11$ si sirurile

-1 -1 -1 3

-1 2 -1 3

Vom da de perechea (3, 3), si atunci putem reconstitui valoarea -1 de pe pozitia a treia, aceasta fiind 1. Stiind ca acum $P_3 = 1$, putem reconstitui valoarea V_3, aceasta fiind 6. Exista doua tipuri de implementari la acest pas:

Forta bruta, cu complexitatea de $O(N^2)$.

Two pointers, cu complexitatea de $O(N)$.

7.3 Pasul 3

Daca exista o pereche de nr (V_i, P_i), atunci V_i il pun 1, si $P_i = P_{i-1}$. De exemplu, pentru $N = 3$, $M = 5$ si sirurile

$4\ -1\ -1$

$4\ -1\ 4$

Pentru perechea de nr (V_2 si P_2), V_2 il pot face 1 si $P_2 = P_1 = 4$. Aceasta parcurgere ar trebui sa aiba complexitatea de $O(N)$.

7.4 Pasul 4

Daca exista -1 pe V_i sau P_i, il reconstituiesc, deoarece stiu ca cel putin una dintre valorile V_i sau P_i sunt reconstituite, iar P_{i-1} este reconstituit deoarece tocmai l-am reconstituit la unul dintre pasii anteriori, sau la acest pas, cu o iterare inainte. De exemplu, pentru $N = 3$, $M = 7$ si sirurile

$1\ -1\ 3\ 3$

$-1\ 1\ -1\ -1$

Putem reconstitui valoarea -1 de pe prima pozitie din al doilea sir, aceasta fiind 1, apoi putem reconstitui valoarea -1 de pe a doua pozitie din primul sir, aceasta fiind 3, etc. Aceasta parcurgere ar trebui sa aiba complexitatea de $O(N)$.

Daca adunam complexitatile de la Pasul 1, Pasul 2, Pasul 3, Pasul 4, cea mai proasta este $O(N)$, daca la pasul 2 implementam two pointers, altfel ar fi $O(N^2)$, daca implementam forta bruta.

Pentru reconstituirea numerelor, se pot folosi doua metode:

Daca trebuie sa reconstituim valoarea P_i si stim valorile V_i si P_{i-1}, atunci se poate aplica formula din cerinta, complexitatea fiind desigur $O(1)$.

Daca trebuie sa reconstituim valorea V_i si stim valorile P_i si P_{i-1} sau P_{i-1} si stim valorile V_i si P_i, se poate face in doua moduri:

Forta bruta, unde luam un indice $j = 0$, si parcurgem cu el pana cand $j * P_{i-1} = P_i$ sau $V_i * j = P_i$. In cel mai rau caz, aceasta complexitate ar trebui sa ia $O(M)$.

Precalcularea valorilor, unde pentru fiecare pereche de indici (i, j), $pc_{i,j} = K$, cu proprietatea ca $K * i = j$. Precalcularea va lua $O(M^2)$, dar pentru a reconstitui valorile ne va lua O(1) complexitate.

Deci, complexitatile care ne pot iesi sunt $O(N^2 * M)$, $O(N^2 + M^2)$, $O(N * M)$ si $O(N + M^2)$. Cea mai buna complexitate este, evident, $O(N + M^2)$.

Totusi, complexitatea de $O(N^2 * M)$ nu exista, deoarece fiecare reconstituire ia $O(M)$, si cum sunt $2 * N$ elemente, inseamna ca complexitatea de $O(N^2 * M)$ este gresita, iar adevarata complexitate este $O(N^2 + N * M)$, dar tot nu este o complexitate mai buna decat $O(N + M^2)$, care este si complexitatea ceruta.

Cod sursa: Solutie.