IIOT 2022-23 Runda 2 - Soluții

Comisia IIOT

Noiembrie 2022

1 Problema Construct Arrays

Autor: Mihnea-Vicentiu Buca, Zsolt Német

Descrierea soluției

Dificultate: Medie-Grea

Observăm că, nu ne interesează *numerele propiu-zise* din *vector*, ci doar restul *sumei* **modulo M**, astfel ne vine ideea să aflăm câte numere din intervalul $[L, R]$, au restul împărțirii la M egal cu x ($0 \leq x < M$).

După calcul acestor valori, pentru 35 de puncte, putem rezolva problema prin metoda *programării dinamice*.

Definim $dp_{i,j}$, ca find suma primilor i termeni, cu restul **modulo M** egal cu j. Răspunul final va fi $dp_{N,k}$.

Complexitate $O(M^2 * N)$.

Pentru 52 de puncte, pornim de la soluția de 35 de puncte. Observăm că pentru a calcula o noua stare i depindem doar de starea $i-1$. Din moment ce avem doar M resturi și ($M \leq 100$); ne vine ideea următoare, anume să construim rezultatul cu ajutorul înmultirilor de matrici. Notam cu R,

matricea *"raspuns"* $R = \begin{pmatrix} p[0] & p[1] & ... & p[M-2] & p[M-1] \\ p[M-1] & p[0] & ... & p[M-3] & p[M-2] \\ & & ..... & & \\ p[1] & p[2] & ... & p[M-1] & p[0] \end{pmatrix}$,

unde $p[x]$ reprezintă câte numere din intervalul $[L, R]$, au restul împărțirii la M egal cu x ($0 \leq x < M$). Răspunsul final va fi $R^n_{0,k}$

Complexitate $O(M^3 * log_2 N)$.

Pentru a putea rezolva problema complet, ne folosim de urmatoarea **observație**: *diagonalele sunt constante pentru fiecare putere a matricii*

Demonstratie: Fie $R^{x_{i,j}}$ numărul de moduri de a face suma *"intrărilor"* să treacă de la **modulo** i la **modulo** j prin adăugarea a x elemente. Deoarece numerele din intervalul $[L, R]$ (pentru fiecare operație **modulo**), sunt independente de N, numărul de moduri de a trecere de la **modulo** i la **modulo** j este același cu trecerea de la **modulo** $i+1$ la **modulo** $j+1$; deci diagonalele sunt constante în fiecare putere R^x.

Acum, când exponențiem R, la calculul unui produs de matrici $B * C$ unde $B = R^x$ și $C = R^y$, putem selecta prima coloană v a matricei C și calcula vectorul Bv, din care rezultă $B * C$.

Complexitate $O(M^2 * log_2 N)$

2 Problema Juggling Shows

Autor: Edoardo Morassutto

Dificultate: Usoara

Această problemă poate fi redusă cu ușurință la problema spectacolelor The Scheduling Problem.

Astfel, vom ordona crescător intervalele în funcție de capătul din dreapta și atunci când le parcurgem, vom avea grijă să scoatem din numărul total de intervale, cele care au capătul din stânga egal cu un capăt din dreapta prezent anterior.

3 Problema DinoLand Fossils

Autor: Péter Varga

Descrierea soluției

Dificultate: Grea

Vom nota cei doi arbori cu A si B.

Pentru fiecare muchie cu cost (x, y, w), o vom pastra in nodul x din arborele A, iar in mod similar vom proceda si pentru query-uri, pastrand query-ul (u, v) in nodul u din arborele A.

Pentru a rezolva problema vom rula un DFS pe arborele A, iar cand intram

intr-un nod, activam muchiile cu cost situate in acel nod, urmand ca la final sa le dezactivam.

Atunci cand activam muchia (x, y, w), vom aduna w la toate valorile din toate nodurile din subarborele lui y din arborele B, folosind o structura de date care accepta actualizari pe intervale (arbori de intervale sau arbori indexati binari), urmand ca la dezactivare sa scadem w din acele noduri, intr-o maniera similara.

Astfel, cand ajungem la nodul u din arborele A, raspunsul pentru fiecare query (u, v) va fi valoarea lui v din arborele B.

Motivul pentru care aceasta abordare functioneaza este acela ca suma greutatilor muchiilor (x, y, w), unde x este un stramos al lui u (doar aceste muchii sunt activate), iar y este un stramos al lui v, deoarece v trebuie sa se gaseasca pe subarborele lui y daca valoarea acestuia a crescut.

4 Problema Casino

Autor: Vlad Mihai Bogdan

Descrierea soluției

Dificultate: Usoara - Medie

Prima observatie este ca daca un string A este similar cu un string B, dar B este similar cu C, atunci si A este similar cu C.

Prin urmare, trebuie sa determinam pentru fiecare string din ce componenta face parte. Asadar, vom transforma fiecare string in rotatia sa minima lexicografica. Problema se reduce acum la cate string-uri egale avem in lista, care se poate rezolva cu un trie sau cu hash-uri.

Pentru o solutie de 100 de puncte, este necesara calcularea rotatiei minime lexicografice in $O(N)$, complexitatea finala fiind $O(N * M)$.

De asemenea, o alta solutie de 100 de puncte presupune calcularea a M hash-uri (cate unul pentru fiecare rotatiei) a unui string, iar mai apoi hash-uriea vectorului de hash-uri format.

La toate solutiile cu hash-uri, trebuie, insa, avut grija la coliziunile pe hash-uri. Este de recomandat sa folosim doua hash-uri, cu baze si module diferite (de preferat numere prime).

5 Problema Scoazze

Autor: Carlo Collodel

Descrierea soluției

Dificultate: Medie

Pentru a rezolva problema se va folosi o soluție greedy. Astfel, în fiecare zi în care primim gunoi, dacă suntem forțați să golim tomberonul curent, o vom face, altfel adăugând la dimensiunea tomberonului cantitatea de gunoi care s-a adunat la pasul curent.

La final, vom avea grijă să golim toate tomberoanele, pentru a respecta condiția problemei.

Astfel, noi am avut grijă să golim fiecare tomberon cât de târziu s-a putut, pentru a minimiza răspunsul obținut.

6 Problema CFG

Autor: Bogdan Ioan Popa

Descrierea soluției

Dificultate: Grea

Pentru a rezolva problema trebuiesc făcute următoarele observații:

- Fie string-ul $S = s_1 s_2 ... s_n$, unde s_i este o literă. Atunci $F(S) = F(s_1)F(s_2)...F(s_n)$. Deoarece pentru orice string $x, |x| < |F(x)|$ rezultă că $|F(x)| >= 2 * |x|$
- Fie $X(B) = \{B, F(B), F(F(B)), F(F(F(B))), ...\}$. Orice string din L va fi o concatenare a string-urilor din $X(B)$.

Pentru a verifica dacă un string a se găsește în L vom calcula $dp[i] = true/false$ dacă prefixul $a[1...i]$ se află în L sau nu. Știm că $dp[i] = true$ dacă există string x din $X(B)$ de lungime l astfel încât x se potrivește peste a la poziția i și $dp[i - l] = true$.

7 Problema High Speed Railroad

Autori: Alessandro Bortolin

Descrierea soluției

Dificultate: Medie

Mai întâi vom rula Dijkstra din nodurile 0 și $N - 1$ pentru a afla distantele fata de toate celelalte noduri.

Apoi, pentru fiecare muchie care nu aparține drumului minim, verificăm folosind distanțele precalculate inițial în ce măsură putem folosi acea muchie pentru a obține un nou drum minim de la 0 la $N-1$, noua distanță de la 0 la $N-1$ dacă trecem prin muchia (i, j) este $dist[i] + dist2[j] + costnou(i, j)$, unde $dist[i]$ e distanta de la 0 la i, $dist2[j]$ e distanta de la $N-1$ la j iar $costnou(i, j)$ e noul cost al muchiei (i, j).