IIOT 2022-23 Runda Finala - Solutii

Comisia IIOT

Martie 2023

1 Comisia IIOT 2022-23

Problemele din acest an au fost propuse si pregatite de urmatorii membri din 4 tari, Romania, Italia, Ungaria si Siria:

1. Ilie-Daniel Apostol
2. Lucian Bicsi
3. Vlad Mihai Bogdan
4. Mihnea-Vicentiu Buca
5. Vlad-Andrei Butnaru
6. Stefan-Cosmin Dascalescu
7. Adrian Panaete
8. Bogdan-Ioan Popa
9. Ioan Popescu
10. Andrei Alexandru Slatinaru
11. Davide Bartolli
12. Alessandro Bortolin
13. Filippo Casarin
14. Luca Chiodini
15. Carlo Collodel
16. Edoardo Morassutto
17. Bence Deák
18. Péter Gyimesi
19. Zsolt Németh
20. László Nikházy
21. Áron Noszály
22. Péter Varga
23. Rabeeh Jouni

2 Problema Interval Xor 2

Autor: Stefan Dascalescu, Ioan Popescu

Această problemă poate fi abordată în două moduri distincte, în funcție de metoda utilizată.

O primă metodă constă în folosirea unui trie persistent, în care vom insera mai întâi pe rând valorile obținute după calcularea XOR-urilor parțiale din vectorul dat. Această structură de date este necesară deoarece spre deosebire de problemele clasice de trie, avem nevoie să dăm răspunsul și pe un interval anume, deci persistența trie-ului devine mult mai necesară. Operațiile de inserare și query vor fi ambele făcute similar ca la un trie obișnuit, dar ținându-se cont de faptul că avem de-a face cu un interval dat de valori. Complexitatea finală ar fi $O(nlogvalmax)$

O a doua metodă, folosită de concurenți a fost prelucrarea query-urilor în ordine crescătoare a capătului din stânga și rezolvarea lor online folosind un trie obișnuit, complexitatea fiind aceeași.

3 Problema Phone Book

Autor: Bogdan Ioan Popa

Să notăm cu A_i al i-lea string din cele N primite ca și input. Fie S string-ul format prin concatenarea tuturor sirurilor din input. Vom calcula suffix array-ul șirului S. Fiecare sufix va avea o culoare, determinată de indicele i al string-ului din A din care începe. Pentru fiecare sufix din șirul de sufixe este suficient să găsim cel mai apropiat sufix de culoare diferită aflat la stânga, respectiv la dreapta lui, pentru a putea calcula care este cea mai lungă potrivire a acestui sufix cu a altuia din alte cuvinte.

4 Problema Love Journal

Autor: Stefan Dascalescu

Mai întâi, vom avea grijă să procesăm datele din calendar care apar în șirul nostru de caractere. Pentru a face asta, vom codifica acele date folosind numere naturale corespunzătoare cu a câta zi din an este acea dată (de exemplu, 17.04 este a 107-a zi din an, iar 19.12 este cea de-a 353-a zi din an). După ce am prelucrat aceste codificări, vom folosi următoarea dinamică pentru a prelucra răspunsul.

$dp[i]$ = numărul de moduri în care putem împărți șirul de caractere pe date până la a i-a zi din șirul de zile. $dp[i]$ va fi suma tuturor valorilor de forma $dp[j]$, care respectă condiția că data de pe poziția j este data precedentă față

de data de pe poziția i.

Pentru a optimiza implementarea, vom tine și $sum[zi]$, care ne va ține câte variante sunt de a împărți șirul astfel încât ultima dată este zi, iar asta ne va duce la ideea de a folosi vectorul sum pentru a calcula valorile din dinamica de mai sus.

În total, complexitatea algoritmului nostru va fi $O(n)$.

5 Problema Base12

Autor: Bogdan Ioan Popa

Fiecare număr X va fi transformat într-un triplet de forma $(conf, x, y)$ unde $conf$ reprezintă un număr natural în care bit-ul c este setat dacă X conține cifra c, x reprezintă una dintre literele care apare în X, iar y reprezintă cealaltă literă care apare în X. Astfel vom obține un șir de triplete A, unde tripletul A_i este cel generat de al i-lea număr din input.

5.1 Subtask 1

Se generează toate submulțimile posibile de cifre și pentru fiecare submulțime se verifică dacă au fost acoperite toate numerele din input. Verificarea poate fi făcută în $O(1)$, în urma unei precalculări.

5.2 Subtask 2

Vom considera un graf G neorientat în care nodurile sunt litere, inițial gol. Pentru fiecare triplet, vom trage muchie în G de la $A_i.x$ la $A_i.y$. Se observă că pe acest graf va trebui să găsim o submulțime minimă de noduri care să acopere toate muchiile. Această problemă este cunoscută ca și minimum vertex cover. Pe un graf cu N noduri, minimum vertex cover este egal cu N - maximum clique pe graful său complementar. Clica maximă într-un graf cu N noduri poate fi determinată în $O(2^N)$, complexitate suficientă pentru a obține punctaj maxim pe subtask 2, însă poate fi determinată și în $O(2^{N/2})$.

5.3 Subtask 3 și 4

Se combină soluțiile de la primele două subtask-uri. Vom itera prin toate submulțimile de cifre posibile și vom ignora tripletele acoperite de submulțimea curentă de cifre. Pentru tripletele rămase, vom construi graful G și vom determina minimum vertex cover asemănător cu soluția de la subtask 2, însă de această dată vom folosi algoritmul mai rapid pentru a determina

clica maximă. Diferența între subtask 3 și 4 este dată de modalitatea în care se construiește graful G rezultat din tripletele neacoperite.

6 Problema Squirrel

Autor: Adrian Panaete, Lucian Bicsi

Problema cere, în esență, simularea rapidă un proces iterativ pe o matrice de dimensiuni 3×3 care se întâmplă de k ori, unde k este foarte mare ($1 \leq k \leq 10^{18}$). Deși dimensiunile input-ului sunt foarte mici, numărul mare k și valorile mari din cadrul matricei fac simularea să fie una foarte non-trivială.

Cu toate acestea, să observăm mai atent procesul. Intuitiv, după mulți pași, valorile tind să devină foarte apropiate. Continuând, o dată ce valorile sunt foarte apropiate, procesul se va repeta în mod ciclic, cu o perioadă relativ mică.

Mai exact, se poate demonstra că, dacă veverița a redistribuit alune din cadrul a două celule (i_1, j_1) și (i_2, j_2), atunci valorile $M(i_1, j_1)$ și $M(i_2, j_2)$ nu vor fi niciodată la o diferență mai mare decât 7. O intuiție (foarte informală) pentru această propoziție este că, privind primul moment în care veverița a distribuit alune din (i_2, j_2) (presupunând că aceasta distribuise deja în trecut din (i_1, j_1)), atunci $M(i_2, j_2)$, trebuie ca, măcar la un moment dat, $M(i_2, j_2)$ să devină "comparabil de mare" cu $M(i_1, j_1)$. După aceasta, diferența dintre $M(i_1, j_1)$ și $M(i_2, j_2)$ nu poate să depășească 7, deoarece după un pas din proces valoarea maximă scade cu maxim 4 și valoarea celui de-al doilea maxim crește cu maxim 4.

Presupunând că procesul are îndeajuns de multe iterații încât toate celulele să fie atinse, conform propoziției de mai sus, numărul de stări (și, prin urmare, perioada) nu poate depăși 9^7 (deși, în practică, sunt mult mai puține stări valide, perioada fiind de ordinul $\sqrt{9^7}$)

Cu toate acestea, pot dura un număr foarte mare de pași (liniar în mărimea valorilor din matrice) pentru ca valorile din matrice să fie echilibrate și perioada să apară. Cu toate acestea, această soluție ar trebui să obțină 40 de puncte.

Pentru a optimiza soluția, va trebui să înțelegem de ce soluția nu este eficientă. Un exemplu simplu arată în următoarea formă (unde $m \ll M$):

$$\begin{matrix} m & m & m \\ m & M & m \\ m & m & m \end{matrix}$$

Mai exact, pentru acest caz, veverița va face $O(M)$ pași până când valorile vor fi îndeajuns de apropiate și procesul va cicla. Cu toate acestea, putem să rafinăm soluția de mai sus, în modul următor:

Să considerăm S mulțimea celulelor din care veverița a redistribuit alune. Toate valorile din S sunt din mulțimea $\{M-7, M-6, \ldots, M\}$, conform cu propoziția de mai sus. După un număr relativ mici de pași ($O(7^{|S|})$), distribuția diferențelor dintre valorile atinse și valoarea maximă va cicla. Celulele neatinse vor primi un număr de alune, în timp ce valoarea lui M va scădea. Vom folosi această observație pentru a simula mai rapid astfel de transferuri periodice, până când o nouă celulă va trebui considerată pentru veveriță să redistribuie din aceasta (i.e., pentru că valoarea din această celulă a devenit comparabilă cu celelalte, deci nu va mai putea fi ignorată).

Deși numărul de operații din cadrul acestei soluții este de ordinul $O(7^9)$, cu toate acestea în practică perioadele nu vor depăși ordinul miilor, deci soluția este mai mult decât suficientă pentru constrângerile din enunț.

7 Problema String Stack Reduction

Autor: Vlad Mihai Bogdan

Vom nota cu X suma lungimilor cuvintelor din mulțimea S. Se observă că există aproximativ $\sqrt{X}$ lungimi distincte ale cuvintelor. Vom reține câte un hash pentru fiecare cuvânt din mulțimea S și câte un hash pentru fiecare prefix al stivei (de exemplu, numere în baza 37, modulo un număr foarte mare, cu cea mai semnificativă cifra fiind elementul de la baza stivei). Deoarece numărul de lungimi distincte este relativ mic, se poate verifică pentru fiecare lungime dacă există o potrivire cu sufixul stivei. Dacă da, se eleminia cuvnatul și hash-urile pentru prefixele din sufixul eliminat. Cu respectivele hash-uri, atât verificările, cât și operațiile de pop (un număr arbitrat de elemente) și operațiile de push se fac în timp constant.

Complexitatea finala este $O(Q * \sqrt{X})$.