Probleme

BUZATU GIULIAN

Problemă

Să rezolvăm următoarea problemă: https://infoarena.ro/problema/car.

Problemă

Să rezolvăm următoarea problemă: https://infoarena.ro/problema/car.

Hint: Putem extinde algoritmul BFS 0-1 pentru mai multe costuri?

Problemă – Algoritmul lui Dial

Să rezolvăm următoarea problemă: https://infoarena.ro/problema/car.

Soluție: Putem observa că operațiile de tip 3 și 4 se reduc, de fapt, la o combinație de operații de tip 1 și 2. Așadar, trebuie să vedem cum facem să modelăm aceste operații. Putem să ne folosim de principiul de la algoritmul BFS 0-1. Știm că distanțele din coadă vor fi diferite prin cel mult valoarea 2. Deci, putem ține 3 cozi, iar de fiecare dată când nu mai avem elemente într-o coadă, mergem la următoarea, în mod ciclic.

Implementare: https://infoarena.ro/job_detail/3167665?action=view-source

Problemă

Să rezolvăm următoarea problemă: https://infoarena.ro/problema/car.

Putem obține o altă soluție?

Soluție cu BFS 0-1

Să rezolvăm următoarea problemă: https://infoarena.ro/problema/car.

Soluție: Putem să facem și observația că operațiile de tip 1 și 2 pot fi simulate prin simple rotiri ale direcției, în timp ce noi rămânem pe aceeași celulă. Practic, avem 2 variante: continuăm să mergem pe direcția actuală (deci costul este 0) sau schimbăm direcția (fie la stânga, fie la dreapta, cu costul 1 în ambele cazuri). Așadar, problema se reduce la algoritmul BFS 0-1.

Implementare: https://infoarena.ro/job_detail/3167681?action=view-source

Soluții diferite la problema Camionaș

Soluții: 1. https://infoarena.ro/job_detail/3158978?action=view-source (cu BFS 0-1)
2. https://infoarena.ro/job_detail/3167598?action=view-source (cu Dijkstra)
3. https://infoarena.ro/job_detail/3167622?action=view-source (cu Dial)

În cazul de față, nu se observă o diferență considerabilă între timpii de execuție, dar, în practică, dacă putem folosi algoritmul lui Dial, atunci preferăm să facem asta, decât să folosim Dijkstra.

Problemă

Să rezolvăm următoarea problemă: https://cses.fi/problemset/task/1669.

Problemă

Să rezolvăm următoarea problemă: https://cses.fi/problemset/task/1669.

Hint: Putem folosi parcurgerea DFS pentru a determina existența unui ciclu.

Problemă

Să rezolvăm următoarea problemă: https://cses.fi/problemset/task/1669.

Soluție: Facem o parcurgere DFS pentru fiecare componentă conexă până găsim un ciclu. Dacă nu găsim niciun ciclu, afișăm mesajul “IMPOSSIBLE”. Pentru a determina existența unui ciclu este de ajuns ca în cadrul parcurgerii DFS să verificăm dacă un vecin al nodului curent este deja vizitat. Dacă găsim un ciclu, trebuie să-l afișăm. Prin urmare, pentru fiecare nod vom ține minte și care este tatăl său în arborele DFS și după vom reconstrui ciclul folosind acest vector de tați.

Implementare: https://cses.fi/paste/a4091deacf0dfc9faa65e0/

Problemă

Să rezolvăm următoarea problemă: https://cses.fi/problemset/task/1678.

Problemă

Să rezolvăm următoarea problemă: https://cses.fi/problemset/task/1678.

Hint: Putem face similar cu cazul în care graful este neorientat.

Problemă

Să rezolvăm următoarea problemă: https://cses.fi/problemset/task/1678.

Soluție: Facem o parcurgere DFS pentru fiecare componentă conexă până găsim un ciclu. Dacă nu găsim niciun ciclu, afișăm mesajul “IMPOSSIBLE”. Pentru a determina existența unui ciclu vom colora nodurile în 3 culori distincte: alb dacă nu a fost vizitat, gri dacă încă suntem în subarborele său în parcurgerea DFS, negru altfel. Astfel, graful conține un ciclu, dacă de la nodul curent din pacurgere, avem muchie la un nod gri. Dacă găsim un ciclu, trebuie să-l afișăm și pentru asta vom folosi aceeași idee de vector de tați menționată mai sus.

Implementare: https://cses.fi/paste/f1feb52a93ed6e39aa6745/

Problemă

Să rezolvăm următoarea problemă: https://codeforces.com/problemset/problem/510/C.

Problemă

Să rezolvăm următoarea problemă: https://codeforces.com/problemset/problem/510/C.

Hint 1: Pentru a verifica că șirurile sunt în ordine lexicografică este de ajuns să verificăm că șirul i este mai mic lexicografic decât șirul i+1.

Problemă

Să rezolvăm următoarea problemă: https://codeforces.com/problemset/problem/510/C.

Hint 1: Pentru a verifica că șirurile sunt în ordine lexicografică este de ajuns să verificăm că șirul i este mai mic lexicografic decât șirul i+1.
Hint 2: Putem modela un graf folosindu-ne de restricțiile care apar în urma comparării de la punctul 1?

Problemă

Să rezolvăm următoarea problemă: https://codeforces.com/problemset/problem/510/C.

Hint 1: Pentru a verifica că șirurile sunt în ordine lexicografică este de ajuns să verificăm că șirul i este mai mic lexicografic decât șirul i+1.
Hint 2: Putem modela un graf folosindu-ne de restricțiile care apar în urma comparării de la punctul 1?
Hint 3: Dacă cuvântul i și cuvântul i+1 diferă la litera j, atunci ducem o muchie orientată de la s[i][j] la s[i+1][j].

Problemă

Să rezolvăm următoarea problemă: https://codeforces.com/problemset/problem/510/C.

Soluție: Vom compara fiecare șir cu următorul șir din listă și vom găsi caracterul la care diferă. Vom crea un graf în care fiecărei litere din alfabet îi asociem un nod (a -> 0, b -> 1, etc). Dacă cuvântul i și cuvântul i+1 diferă la litera j, atunci ducem o muchie orientată de la s[i][j] la s[i+1][j]. Pentru a putea reordona literele din alfabet astfel încât să obținem o ordine lexicografică a cuvintelor, trebuie ca graful rezultat să admită o sortare topologică. Dacă acest lucru se întâmplă, atunci sortarea topologică este chiar răspunsul la problemă. În cazul în care două cuvinte nu diferă la nicio poziție, iar cuvântul i este mai lung, sau în care graful nu este aciclic, nu avem soluție.
Implementare: https://codeforces.com/contest/510/submission/142788921

Problemă

Să rezolvăm următoarea problemă: https://infoarena.ro/problema/trilant.

Problemă

Să rezolvăm următoarea problemă: https://infoarena.ro/problema/trilant.

Hint 1: Intuitiv, cum ați dori să alegeți nodul X?

Problemă

Să rezolvăm următoarea problemă: https://infoarena.ro/problema/trilant.

Hint 1: Intuitiv, cum ați dori să alegeți nodul X?
Hint 2: Puteți demonstra că acest mod este corect?

Problemă

Să rezolvăm următoarea problemă: https://infoarena.ro/problema/trilant.

Soluție: Intuitiv, am vrea ca nodul X pentru care trilanțul are costul minim să respecte condiția că distanța(A,X) + distanța(B,X) + distanța(C,X) să fie minim. Putem demonstra că prin acest mod, drumurile nu se intersectează? Presupunem că există un nod Y de la care drumurile de la A la X si cel de la B la X se intersectează. Deci, distanța(A,X) + distanța(B,X) + distanța(C,X) = distanța(A,Y) + distanța(B,Y) +2 * distanța(Y,X) + distanța(C,X), dar această sumă este mai mare decât distanța(A,Y) + distanța(B,Y) + distanța(Y,X) + distanța(C,X) = distanța(A,Y) + distanța(B,Y) + distanța(C,Y) => Y este nodul care formează un trilanț de cost minim. Exclus!
Implementare: https://infoarena.ro/job_detail/3172742?action=view-source

Temă + Lectură suplimentară

Temă:

- https://infoarena.ro/problema/patrol2
- https://infoarena.ro/problema/dmin
- https://infoarena.ro/problema/lanterna
- https://infoarena.ro/problema/banuti

Lectură suplimentară:

- https://cp-algorithms.com/graph/finding-cycle.html