Bellman-Ford + Upsolving

NIȚĂ ALEXANDROS

Upsolving

Să rezolvăm problema: https://kilonova.ro/problems/57

Upsolving

Să rezolvăm problema: https://kilonova.ro/problems/57

Hint: Trebuie să facem componente conexe până ne oprim la k.

Upsolving

Să rezolvăm problema: https://kilonova.ro/problems/57

Soluție: Vom aplica Algoritmul lui Kruskal, ținând într-un contor numărul de componente conexe. Inițial acesta va fi n și vom scădea o componentă conexă de fiecare dată când aplicăm o operație de make_union. Ne oprim când contorul ajunge la valoarea k.

Sursă: https://kilonova.ro/submissions/145501

Upsolving

Să rezolvăm problema: https://kilonova.ro/problems/328.

Upsolving

Să rezolvăm problema: https://kilonova.ro/problems/328.

Hint: Vom încerca să trecem prin fiecare muchie care are un nod un oraș nou.

Upsolving

Să rezolvăm problema: https://kilonova.ro/problems/328.

Hint: Cum putem să calculăm din nou distanțele în cazul acesta?

Upsolving

Să rezolvăm problema: https://kilonova.ro/problems/328.

Soluție: Vom folosi Algoritmul lui Dijkstra de două ori pentru a putea să calculăm distanțele după ce considerăm fiecare muchie mai ușor: o dată din nodul 0 și apoi din nodul n – 1. Pentru fiecare nod care nu face parte din drumul cu cost minim de la 0 la n – 1, vom reține diferența minimă.

Sursă: https://kilonova.ro/submissions/145607

Upsolving

Să rezolvăm problema: https://kilonova.ro/problems/1776.

Upsolving

Să rezolvăm problema: https://kilonova.ro/problems/1776.

Hint: Ne interesează numărul minim de mutări, deci vrem ca ciclul generat să fie alcătuit din părți cât mai lungi.

Upsolving

Să rezolvăm problema: https://kilonova.ro/problems/1776.

Soluție: Descompunem graful în lanțuri cât mai lungi, apoi le unim.

Sursă: https://kilonova.ro/submissions/145641

Bellman-Ford

Avem aceeași problemă ca în cazul Algoritmului lui Dijkstra: găsiți distanța de la un nod u la toate celelalte noduri într-un graf orientat cu n noduri și m muchii.

Doar un mic detaliu în plus: costurile muchiilor pot fi negative.

Bellman-Ford

În primul rând, să observăm că există posibilitatea să nu existe o distanță minimă de la un nod la toate celelalte. Acest lucru se întâmplă în cazul unui ciclu în care costul muchiilor este negativ. Numim acest caz ciclu de cost negativ.

Algoritmul pe care îl vom prezenta poate să detecteze aceste cazuri și chiar să deducă ciclurile.

Bellman-Ford

Algoritmul funcționează asemănător cu Algoritmul lui Dijkstra. Încercăm să folosim muchia (a, b) cu cost c pentru a relaxa distanța până la b.

Se poate dovedi matematic că aplicând această metodă de n – 1 ori vom determina distanțele minime.

Pentru determinarea ciclurilor negative parcurgem muchiile și dacă găsim încă o relaxare, înseamnă că am găsit unul.

Bellman-Ford

Descriere algoritm:

1. Parcurgem lista muchiilor.
2. Pentru fiecare muchie verificăm dacă poate să relaxeze drumul.
3. Repetăm pașii 1 și 2 de n – 1 ori.
4. Parcurgem muchiile și dacă găsim o relaxare, ne oprim.
   Am găsit un ciclu negativ.

Complexitate: O(nm)

Bellman-Ford

Să rezolvăm problema: https://www.infoarena.ro/problema/bellmanford

Implementare: https://www.infoarena.ro/job_detail/3182394?action=view-source

Bellman-Ford

Ne putem folosi de faptul că nu toate încercările noastre de relaxare au succes. Creăm o coadă de priorități care conține doar nodurile pentru are am găsit deja răspunsul dar care pot să își relaxeze vecinii. La fiecare relaxare, adăugăm nodul în coadă.

Sursă: https://www.infoarena.ro/job_detail/3182423?action=view-source

Temă

https://open.kattis.com/problems/shortestpath3
https://cses.fi/problemset/task/1197
https://cses.fi/problemset/task/1673
https://www.infoarena.ro/problema/ciclu

Probleme suplimentare

https://infoarena.ro/problema/ciob
https://oj.uz/problem/view/APIO17_merchant

Lectură suplimentară

https://cp-algorithms.com/graph/bellman_ford.html
https://usaco.guide/CP2.pdf#page=109