Roy-Floyd. Prim. Ciclu eulerian

BUZATU GIULIAN & NIȚĂ ALEXANDROS

Problemă

Dându-se un graf ponderat, aflați distanțele minime pentru fiecare pereche de noduri.

Problemă

Dându-se un graf ponderat, aflați distanțele minime pentru fiecare pereche de noduri.

Cum am putea face?

Problemă

Dându-se un graf ponderat, aflați distanțele minime pentru fiecare pereche de noduri.

Putem implementa algoritmul lui Dijkstra, dacă graful are doar ponderi pozitive, sau algoritmul Bellman-Ford, dacă avem și costuri negative, și după să rulăm pe rând algoritmul pentru fiecare nod. Totuși, acest lucru ne-ar genera o complexitate destul de mare.

Roy-Floyd

Pentru a rezolva problema anterioară, putem folosi algoritmul Roy-Floyd. Acesta are complexitatea $O(n^3)$ și acceptă costuri negative, dar nu și cicluri negative. Acest algoritm este unul de programare dinamică.

Roy-Floyd

Avem o matrice d[i][j] ce reprezintă distanța minimă dintre nodurile i și j. Inițial, matricea va conține doar muchiile directe dintre i și j, 0 pe diagonala principală și ∞ în rest.

Pentru fiecare nod k, considerăm drumurile pe care le putem augmenta folosind nodul k. Practic, la pasul k, noi vom avea deja calculate drumurile de cost minim folosind doar nodurile din mulțimea {1, 2, … , k-1} și vom folosi nodul k pentru a vedea dacă putem îmbunătății distanțele deja obținute. Deci, folosim k pentru a uni lanțul de la i la k și lanțul de la k la j, astfel:

$$d[i][j] = \min(d[i][j], d[i][k] + d[k][j])$$

Roy-Floyd

Problemă: https://infoarena.ro/problema/royfloyd

Implementare: https://www.infoarena.ro/job_detail/3193701?action=view-source

Problemă

Să rezolvăm problema: https://infoarena.ro/problema/coach

Problemă

Să rezolvăm problema: https://infoarena.ro/problema/coach

Hint: Ne-ar ajuta să avem nodurile sortate crescător după valoarea lor calorică.

Problemă

Să rezolvăm problema: https://infoarena.ro/problema/coach

Soluție: Sortăm nodurile după valoarea calorică și le renumerotăm. Acum putem fixa valoarea calorică minimă și să aplicăm algoritmul Roy-Floyd. După fiecare pas k verificăm dacă am atins cumva valoarea dorită T. Dacă da, atunci reținem valorile care ne-au oferit răspunsul și putem să ne oprim din căutare. Deoarece noi am obținut răspunsul folosind noduri din mulțimea {1, 2, … , k}, valoarea calorică maximă este cea a nodului k.

Implementare: https://infoarena.ro/job_detail/3193753?action=view-source

Algoritmul lui Prim

Vom considera aceeași problemă ca în cazul algoritmului lui Kruskal: dându-se un graf ponderat cu n noduri și m muchii, aflați un arbore parțial de cost minim ce conține toate nodurile din graf.

Algoritmul lui Prim

Vom considera aceeași problemă ca în cazul algoritmului lui Kruskal: dându-se un graf ponderat cu n noduri și m muchii, aflați un arbore parțial de cost minim ce conține toate nodurile din graf.

Hint 1: Să presupunem că avem un arbore format din k noduri, $k \leq n$. Cum adăugăm următorul nod în arbore?

Algoritmul lui Prim

Vom considera aceeași problemă ca în cazul algoritmului lui Kruskal: dându-se un graf ponderat cu n noduri și m muchii, aflați un arbore parțial de cost minim ce conține toate nodurile din graf.

Hint 1: Să presupunem că avem un arbore format din k noduri, $k \leq n$. Cum adăugăm următorul nod în arbore?
Hint 2: Intuitiv, vom folosi muchia de cost minim ce leagă un nod care nu este în arbore de arborele deja format pentru a minimiza costul total.

Algoritmul lui Prim

Descriere algoritm:

1. Inițial arborele este vid și putem adăuga orice nod.

2. Adăugăm un nod în arborele format folosind muchia de cost minim ce unește un nod din arbore de alt nod care nu este în arbore.

3. Repetăm pasul 2 până adăugăm toate nodurile în arbore, adică de $n - 1$ ori.

Algoritmul lui Prim

Descriere algoritm:

1. Inițial arborele este vid și putem adăuga orice nod.
2. Adăugăm un nod în arborele format folosind muchia de cost minim ce unește un nod din arbore de alt nod care nu este în arbore.
3. Repetăm pasul 2 până adăugăm toate nodurile în arbore, adică de $n - 1$ ori.

Pasul 2 este descris vag, deoarece există mai multe metode de a-l efectua, ducând la complexități diferite.

Algoritmul lui Prim

Metoda 1

Vom parcurge toate nodurile din arborele creat și, pentru fiecare nod din arbore, toate nodurile din graf care nu fac parte din arbore. Apoi alege minimul muchiilor cu un capăt în arbore și celălalt în afara arborelui. Această implementare are complexitatea $O(nm)$, deoarece pentru fiecare nod din arbore vom verifica toate muchiile adiacente. În cazul unui graf dens, această complexitate tinde spre $O(n^3)$.

Implementare: https://infoarena.ro/job_detail/3194695?action=view-source

Algoritmul lui Prim

Metoda 2

Vom reține într-un vector distanța minimă din arbore la un nod din afară. Când trebuie să decidem prin ce muchie adăugăm noul nod, vom folosi valoarea minimă din vectorul de distanțe minime și după adăugarea lui, vom actualiza acest vector. Această abordare conduce la complexitatea de $O(n^2)$, deoarece căutăm un minim într-un vector de n ori. Această soluție este de preferat pentru grafuri dense.

Implementare: https://infoarena.ro/job_detail/3194704?action=view-source

Algoritmul lui Prim

Metoda 3

Vom folosi un $std :: priority\_queue$ pentru a determina minimul. Apoi parcurgem lista de adiacență a nodului găsit pentru a actualiza distanțele. Conduce la o complexitate de $O(m \log n)$ deoarece pentru fiecare muchie actualizată trebuie să o adaug în $std :: priority\_queue$.

Implementare: https://infoarena.ro/job_detail/3194710?action=view-source

Algoritmul lui Prim

Metoda 3

Vom folosi un $std :: priority\_queue$ pentru a determina minimul. Apoi parcurgem lista de adiacență a nodului găsit pentru a actualiza distanțele. Conduce la o complexitate de $O(m \log n)$ deoarece pentru fiecare muchie actualizată trebuie să o adaug în $std :: priority\_queue$.

Implementare: https://infoarena.ro/job_detail/3194710?action=view-source

Problemă

Să rezolvăm problema: https://infoarena.ro/problema/cablaj

Problemă

Să rezolvăm problema: https://infoarena.ro/problema/cablaj

Soluție: Problem constă în aplicarea algoritmului lui Prim. Graful este complet, deci, implicit, este și dens, deci metoda a 2-a este cea pe care vrem să o folosim.

Implementare ($O(n^2)$): https://infoarena.ro/job_detail/3171297?action=view-source
Implementare ($O(m \log n)$): https://infoarena.ro/job_detail/3194713?action=view-source

Observație: Se poate observa o diferență de timp și de memorie între cele două surse.

Graf eulerian

Un lanț eulerian este un lanț în care apare fix o dată fiecare muchie din graf. Similar definim noțiunea de ciclu eulerian.

Graf eulerian

Un lanț eulerian este un lanț în care apare fix o dată fiecare muchie din graf. Similar definim noțiunea de ciclu eulerian.

Dându-se un multigraf (un graf în care pot exista muchii de la un nod la el însuși și pot exista mai multe muchii între două noduri), aflați dacă acesta este sau nu un graf eulerian (i. e. conține un ciclu eulerian). În caz afirmativ, afișați ciclul eulerian.

Ciclu eulerian

Teoremă: Un multigraf este eulerian dacă:

1. este conex
2. gradul fiecărui nod este par

Ciclu eulerian

Teoremă: Un multigraf este eulerian dacă:

1. este conex
2. gradul fiecărui nod este par

Această teoremă rezolvă prima parte a problemei. Dar cum putem să aflăm ciclul, dacă multigraful nostru este eulerian?

Ciclu eulerian

Teoremă: Un multigraf este eulerian dacă:

1. este conex
2. gradul fiecărui nod este par

Această teoremă rezolvă prima parte a problemei. Dar cum putem să aflăm ciclul, dacă multigraful nostru este eulerian?

Hint: Gândiți-vă la cum arată un ciclu eulerian.

Ciclu eulerian

Algoritmul se bazează pe observația că un ciclu eulerian se poate descompune într-o reuniune de cicluri disjuncte. Astfel, algoritmul va parcurge fiecare ciclu disjunct și va memora într-un vector această parcurgere. Vom face acest lucru recursiv, într-un mod similar cu o parcurgere DFS.

Ciclu eulerian

Descriere algoritm:

1. Pornim dintr-un nod de start u.

2. Mergem într-un vecin nevizitat al nodului de start și ștergem muchia care îi unește.

3. Pornim o altă căutare a unui ciclu eulerian din nodul vizitat.

4. Când terminăm de parcurs ciclul, îl memorăm invers.

Complexitate:$O(n + m)$- algoritmul fiind similar cu un DFS.

Implementare: https://infoarena.ro/job_detail/3194826?action=view-source

Problemă

Să rezolvăm problema: https://codeforces.com/gym/104013/problem/E

Problemă

Să rezolvăm problema: https://codeforces.com/gym/104013/problem/E

Hint 1: Cum am rezolva problema dacă am avea doar operații pe care trebuie să le facem?

Problemă

Să rezolvăm problema: https://codeforces.com/gym/104013/problem/E

Hint 1: Cum am rezolva problema dacă am avea doar operații pe care trebuie să le facem?
Hint 2: Cum putem integra operațiile care nu trebuie făcute în soluția precedentă?

Problemă

Să rezolvăm problema: https://codeforces.com/gym/104013/problem/E

Hint 1: Cum am rezolva problema dacă am avea doar operații pe care trebuie să le facem?
Hint 2: Cum putem integra operațiile care nu trebuie făcute în soluția precedentă?
Hint 3: Există două tipuri de operații pe care nu trebuie să le facem.

Problemă

Să rezolvăm problema: https://codeforces.com/gym/104013/problem/E

Soluție: Putem construi un graf unde punem muchie de la nodul a la nodul b, dacă avem o operație de la a la b cu w=1. Putem găsi un lanț/ciclu eulerian în acest graf. Dacă avem unul, atunci știm că putem face toate operațiile cu w=1. Trebuie să vedem dacă putem integra operațiile cu w=0 în soluția curentă. Putem observa că sunt două tipuri de operații cu w=0, cele care încep din nodul de start c și celelalte. Așadar, dacă putem integra aceste operații în lanțul/ciclul nostru eulerian, avem soluție. Din cauză că datele de intrare sunt mari, trebuie să le normalizăm la început.
Implementare: https://github.com/Giulian617/Hai-la-olimpiada-2023-2024/blob/main/11-12/resources/GYM104013E.cpp

Temă

- https://infoarena.ro/problema/rf
- https://infoarena.ro/problema/rfinv
- https://infoarena.ro/problema/domino
- https://infoarena.ro/problema/cartite

Probleme suplimentare

- https://infoarena.ro/problema/tester
- https://infoarena.ro/problema/ulei
- https://infoarena.ro/problema/mmo

Lectură suplimentară

- https://cp-algorithms.com/graph/all-pair-shortest-path-floyd-warshall.html
- https://arxiv.org/pdf/1904.01210.pdf
- https://cp-algorithms.com/graph/mst_prim.html
- https://cp-algorithms.com/graph/euler_path.html