Tema 1

Drumuri minime în graf
Costul minim între oricare 2 noduri

Bobocu Alexandra-Florentina, 321CA

Facultatea de Automatică și Calculatoare
Universitatea POLITEHNICA București
alexandra.bobocu@stud.acs.upb.ro
Noiembrie 2021

1 Descrierea problemei rezolvate

În teoria grafurilor, problema drumului minim în graf se referă la a găsi drumul între două noduri ale grafului, astfel încât suma totală a ponderilor muchiilor să fie minimă. Într-un graf orientat G = (V, E) (V = mulțimea de noduri, E = mulțimea de muchii) cu n noduri, fiecărui arc îi este asociat un număr întreg numit cost. Dacă toate muchiile ar avea același cost, această problemă ar putea fi rezolvată cu ușurință folosind parcurgerea în lățime a grafului (BFS: Breadth First Search), însă având grafuri cu muchii de costuri diferite, sunt necesari algoritmi mai avansați, dintre care voi prezenta 3 în cele ce urmează.
Semnificaţia acestui cost poate fi foarte variată, în funcţie de domeniul pe care îl descrie graful. De exemplu, dacă graful reprezintă harta unui oraş în care arcele sunt străzile, iar nodurile sunt intersecţiile dintre străzi, atunci putem vorbi despre costul deplasării unui automobil între două intersecţii, de-a lungul unei străzi. Acesta s-ar putea măsura în cantitatea de benzină consumată, calculată prin prisma lungimii străzii în metrii sau în kilometrii.

Google Maps, Moovit sau Waze sunt doar câteva din aplicațiile practice pe care le folosim în viața de zi cu zi pentru a găsi o rută posibilă, de preferat minimă între două locații pe hartă. Aceste aplicații arată utilizatorului timpul minim în care ajunge la destinație, respectiv traseul optim pe care să îl parcurgă.

2 Specificarea soluțiilor alese

2.1 Floyd-Warshall

Algoritmul Floyd-Warshall compară toate drumurile posibile din graf dintre fiecare 2 noduri. Acesta poate fi aplicat pe grafurile orientate sau neorientate ce au muchii de cost pozitiv sau negativ, însă nu funcționează pentru grafurile cu cicluri în care costul total este negativ. Motivul constă în faptul că nu există o cale cea mai scurtă între nicio pereche de noduri care fac parte dintr-un ciclu negativ, deoarece lungimea căii între 2 noduri poate fi arbitrar de mică (negativă).

Pentru o ieșire semnificativă numeric, algoritmul Floyd-Warshall presupune că nu există cicluri negative. Dacă totuși există, algoritmul poate fi folosit pentru a le detecta.

Acest algoritm este o alegere bună pentru calcularea căilor între toate perechile de noduri în grafurile dense, în care majoritatea sau toate perechile de noduri sunt conectate prin muchii. În general, preferăm reprezentarea grafurilor prin liste de adiacență deoarece au o complexitate mai bună în cazul parcurgerilor. Totuși, în cazul acestui algoritm, alegerea reprezentării prin matrice de adiacență simplifică mult rezolvarea unei probleme, deoarece putem obține ușor distanța dintre două noduri pe baza matricei în care reținem, adițional, și costurile muchiilor.

2.2 Dijkstra

Algoritmul există în multe variante. Algoritmul original al lui Dijkstra a găsit cea mai scurtă cale între două noduri date, dar o variantă mai comună fixează un singur nod drept sursă și găsește cele mai scurte căi de la sursă la toate celelalte noduri din graf. Dijkstra poate fi folosit doar în grafuri care au toate muchiile nenegative, deoarece algoritmul se bazează pe un fapt simplu: dacă costurile muchiilor sunt nenegative, adăugând o muchie grafului nu putem găsi o cale mai scurtă decât cea pe care am ales-o deja. Alegerea de la început a unei muchii ca fiind calea cea mai scurtă de la un nod la alt nod (optim local) ne conduce către rezultatul corect (optim global).

Odată ce algoritmul a găsit cea mai scurtă cale între nodul sursă și un alt nod, acel nod este marcat ca "vizitat" și adăugat la calea minimă. În momentul în care un nod a fost marcat ca "vizitat", calea curentă către acel nod este marcată ca fiind cea mai scurtă cale pentru a ajunge la acel nod. Procedeul continuă până când toate nodurile din graf au fost adăugate la cale. În acest fel, am format un arbore de drumuri minime (SPT: Shortest Path Tree), care conectează nodul sursă la toate celelalte noduri, urmând cea mai scurtă cale posibilă pentru a ajunge la fiecare nod.

2.3 Johnson

Algoritmul Johnson se aplică pe grafuri orientate și găsește cele mai scurte căi între toate perechile de noduri într-un astfel de graf. Permite ca unele dintre costurile muchiilor să fie negative, dar nu pot exista cicluri negative.
Ideea algoritmului Johnson este de a repondera muchiile și de a le face pozitive, utilizând în acest sens algoritmul Bellman-Ford, după care folosește algoritmul Dijkstra pe graful transformat.

Algoritmul Johnson are 3 pași:

1. Se adaugă un nou nod în graf, care se conectează la toate celelealte noduri din graf prin muchii cu costul 0.

2. Se recalculează toate costurile muchiilor pe noul graf, aplicând algoritmul Bellman-Ford luând nodul nou adăugat drept sursă. În urma acestui pas, costurile muchiilor devin nenegative.
3. Nodul adăugat în plus la primul pas este eliminat, urmând să aplicăm algoritmul Dijkstra pentru fiecare nod.

Spre deosebire de algoritmul Floyd-Warshall care este cel mai eficient pentru grafuri dense, algoritmul Johnson este cel mai eficient pentru grafuri rare, sparse (cu câteva muchii), deoarece complexitatea sa temporală depinde de numărul de muchii din graf, în timp ce complexitatea algoritmului Floyd-Warshall nu.

3 Criteriile de evaluare pentru soluția propusă

Pentru a testa corectitudinea, eficiența și performanța algoritmilor aleși, o să pun la dispoziție teste cât mai numeroase și cât mai variate, cu grafuri rare sau complete, cu costuri/cicluri pozitive sau negative. Voi trata și cazurile particulare, cum ar fi grafurile cu costuri negative pentru algoritmul Dijkstra, grafurile cu cicluri negative pentru algoritmii Floyd-Warshall, respectiv Johnson. În cazul în care un algoritm va da de un astfel de caz particular, se va afișa un mesaj sugestiv.
În fișierele de intrare, pe prima linie vor fi numărul de noduri și de muchii, după care următoarele linii vor fi reprezentate de modelul: nod sursă - nod destinație - cost muchie. În fiecare fișier de ieșire corespunzător unui anumit fișier de intrare, îmi propun să afișez matricile de adiacență rezultate în urma aplicării celor 3 algoritmi descriși pe graful considerat ca input.
O să încep cu câteva teste cu date puține de intrare, astfel încât să fie asigurat faptul că algoritmii pot rezolva corect un graf simplu. Pe parcurs, dimensiunea inputului va crește, deoarece voi genera grafuri cu un număr semnificativ mai mare de noduri și muchii, pentru a evidenția o deosebire mai vizibilă în ceea ce privește eficiența fiecărui algoritm.

Referințe

1. https://www.adamconrad.dev/blog/shortest-paths/. Ultima accesare: 26 Oct 2021
2. https://www.hackerearth.com/practice/algorithms/graphs/shortest-path-algorithms/tutorial/. Ultima accesare: 26 Oct 2021
3. https://www.programiz.com/dsa/floyd-warshall-algorithm. Ultima accesare: 29 Oct 2021
4. https://www.geeksforgeeks.org/johnsons-algorithm/. Ultima accesare: 30 Oct 2021
5. Thomas H. Cormen, Charles E. Leiserson, Roland L. Rivest, Cliord Stein: Introduction to Algorithms, 3rd edition, The MIT Press, Massachusetts Institute of Techology (2009)