Drumuri minime în graf. Costul minim între oricare două noduri

Iordache Mădălina-Gabriela

Universitatea Politehnică din Bucureşti
Facultatea de Automatică şi Calculatoare
madaiorda@yahoo.com
Grupa: 323CA

Abstract. Această temă este dedicată găsirii costului minim in calea unui graf ponderat între nodul sursă și destinație. Au fost aleși trei algoritmi pentru a determina calea minimă: Dijkstra, Bellman-Ford și, nu în ultimul rând, Floyd-Warshall. În cele din urmă, va fi analizată performanța acestor algoritmi.

Keywords: Floyd-Warshall · Bellman-Ford · Dijkstra · graf ponderat · drum minim.

1 Descrierea problemei rezolvate

Principala problemă pe care dorim să o rezolvăm prin intermediul acestui document este găsirea drumului minim în graf și a costului minim între oricare două noduri. Această problemă este specifică grafurilor ponderate, ce au costuri specifice între două vârfuri. Problema drumului minim poate fi privită din perspectiva atât a grafurilor orientate cât și a celor neorientate, scopul temei îl va face analiza pe grafuri orientate, aciclice, ponderate.

Matematic, un graf este o pereche ordonată de mulțimi, $G = (V, E)$, unde V reprezintă o mulțime finită și nevidă de elemente numite vârfuri (sau noduri), iar E este o mulțime de perechi cu elemente din V, numite muchii (dacă sunt neordonate) sau arce (dacă sunt ordonate).

Grafurile au numeroase aplicații[1] în diverse domenii: determinarea celui mai scurt drum dintre două localități (În aplicații precum Waze sau Google Maps această problemă trebuie rezolvată prin algoritmul cel mai optim din punct de vedere temporal. Redactarea celei mai bune soluții este esențială, deoarece, în ziua de azi, traficul suprateran din marile metropole ale lumii este din ce în ce mai aglomerat, fiind destul de importantă ghidarea șoferilor către rute alternative, mai rapide.), rețelele sociale (ex. Facebook, unde nodurile sunt utilizatorii, iar o muchie reprezintă relația de prietenie între doi utilizatori), structura substanțelor chimice, proiectarea circuitelor electrice, etc.

2 Specificarea soluţiilor alese

Floyd-Warshall

Algoritmul Floyd-Warshall întoarce un rezultat de tip matrice, cu dimensiunile VxV (unde V reprezintă numărul de vârfuri ale grafului). De asemenea, acest algoritm este corect din punct de vedere al rezultatului întors și pentru costuri negative pe muchiile grafului.

În primul rând, matricea graficului de intrare este egală cu matricea soluției. Luând în considerare toate vârfurile intermediare, vom actualiza matricea soluției. Aceasta înseamnă că trebuie să alegem toate vârfurile (pe rând) și să calculăm calea cea mai scurtă folosind și nodul ales (cu rol de intermediar). De exemplu, când selectăm nodul *f*, selectăm nodurile $\{0, 1, ..j-1\}$ ca noduri intermediare. Pentru o pereche sursă și destinație (j, l) putem avea următoarele situații: dacă nodul nu este unul intermediar, păstrăm valoarea $(j)(l)$ ca valoare curentă, altfel valoarea $(j)(l)$ este egală cu valoarea $(j)(f)$ + valoarea $(f)(l)$. Considerând V ca număr de noduri, putem spune că complexitatea in timp a acestui algoritm este $O(V^3)$ (pentru construirea matricei de cale).

Bellman-Ford

Algoritmul Bellman-Ford funcţionează pe grafurile cu arce cu cost negativ, cât timp drumurile de cost minim sunt bine definite(fără cicluri cu cost negativ). În principal, algoritmul[2] asociază valoarea 0 cu distanța până la sursă și valoarea infinită tuturor nodurilor rămase. Algoritmul minimizează drumul de la nodul de start la oricare vârf x până la obținerea costului minim. Pentru fiecare arc (j,k) se verifică dacă minimizează distanța de la nodul de start la nodul x, iar această operație se repetă de n ori. Există $|V| - 1$ căi existente și ca rezultat, trebuie să facem același număr de teste înainte pentru a ne asigura că am găsit calea cea mai scurtă. Complexitatea de timp a acestui algoritm este $O(VE)$, unde E este numărul de muchii ale grafului și V numărul de vârfuri.

Dijkstra

Algoritmul lui Dijkstra este specific pentru aflarea celui mai scurt drum de la un nod la celelalte, dar poate fi modificat pentru a afla costul minim între oricare două noduri, ideea algoritmului ramânând aceeași. De asemenea, acest algoritm are o problemă: nu merge pentru costuri ale muchiilor mai mici de 0.

La început, algoritmul consideră toate nodurile nevizitate și creează o mulțime a acestora. După aceea, setează primul nod ca nod curent, îi aribuie valoarea 0 și valoarea infinit celorlalte noduri. Având acest nod, verifică toate nodurile nevizitate și găseste o distanță provizorie[5] față de nodul curent. Compară noul rezultat provizoriu cu cel vechi și preia valoarea celui mai mic.

Când toate nodurile nevizitate sunt verificate, marchează nodul curent ca vizitat și îl scoate din mulțimea nodurilor nevizitate. Dacă distanța cea mai mică dintre nodul curent și destinație este infinită sau dacă nodul destinație a fost vizitat, atunci programul s-a terminat. În mod analog, trece la nodul aflat la cea mai mică distanță provizorie care este nevizitat, calculează din nou distanța provizorie între vecinii lui și așa mai departe. Din punctul de vedere al complexității, Dijkstra este mai bun, deoarece are o complexitate de $O(V^2)$, unde V este numărul de vârfuri ale grafului. În cazul în care se folosește heap-ul pentru Dijkstra, complexitatea devine $O(V * E)$. Acest algoritm nu este corect pentru costuri negative.

3 Criteriile de evaluare pentru soluţia propusă

Cei 3 algoritmi funcţionează bine atunci când toate muchiile au cost pozitiv. Drumul de cost minim nu este bine definit atunci când există cicluri cu cost negativ. Putem ajunge de la un nod la acelaşi nod, folosind acelaşi ciclu de oricâte ori costMinim $= -\infty$; în aceste cazuri, nu putem pune problema găsirii drumurilor de cost minim.

Testele vor fi concepute în aşa fel încât să prezinte într-o manieră cât mai obiectivă acurateţea agloritmilor. Putem începe testarea verificând dacă input-ul se potrivește algoritmului. De exemplu, noi trebuie să avem un graf ponderat ca input pentru a aplica algoritmul Floyd-Warshall și nu putem să avem muchii de cost negativ pentru a aplica Dijkstra.Algoritmii Bellman-Ford şi Floyd-Warshall pot detecta dacă un graf conţine cicluri cu cost negativ. Pornim de la input-uri mici, iar ulterior testăm pe input-uri cât mai complexe pentru a urmări evoluția rezultatului și îl comparăm cu output-ul așteptat. În cazuri speciale putem folosi și grafice pentru a identifica diferențele dintre algoritmi.

Referințe

1. Noțiuni elementare despre grafuri
   `https://infogenius.ro/introducere-teoria-grafurilor/`
   Ultima accesare: 22 Oct 2021.

2. `https://tutoriale-pe.net/algoritmul-bellman-ford-in-c-infoarena/`
   Ultima accesare: 22 Oct 2021.

3. `https://www.programiz.com/dsa/bellman-ford-algorithm`
   Ultima accesare: 22 Oct 2021.

4. `https://www.geeksforgeeks.org/floyd-warshall-algorithm-dp-16/`
   Ultima accesare: 22 Oct 2021.

5. Algoritmi fundamentali. O perspectiva C++ - Razvan Andonie, Ilie Garbacea