Drumuri minime în graf

- Analiza algoritmilor -

Dumitru Elena Bianca
324CD
email: elena.dumitru0312@stud.acs.upb.ro

Noiembrie 2020

Facultatea de Automatica si Calculatoare
Universitatea Politehnica Bucuresti
Noiembrie 2020

Cuprins

1 Introducere

1.1 Importanta gasirii drumului de cost minim

Lumea noastră este interconectată, peste tot în jurul nostru există instanțe ce iau contururi care pot fi ușor încadrate într-un chenar, organizate într-o manieră perceptibilă. Acestea pot lua diverse forme, atât în dimensiunea concretă (mai multe orașe pe o hartă), cât și în cea abstractă (computer networking), iar de aici derivă numeroase concepte matematice și algoritmice. Un exemplu relevant ar fi ideea de *clustering*.

Astfel, din necesitatea înțelegerii, dar și eficientizării datelor, le-am atașat acestora noțiunea de *graf*, unde fiecare entitate are drept corespondent un nod din graf, iar relațiile dintre acestea sunt reprezentate prin atributele muchiilor, respectiv costurile.

De cele mai multe ori, avem nevoie sa inspectăm relațiile dintre două instanțe, respectiv noduri și dorim să facem acest lucru într-o manieră cât mai rapida. Încă de când au fost concretizate aceste noțiuni, mulți reprezentanți ai gândirii algoritmice au încercat să găsească soluții, propunând idei și dând naștere la diverși algoritmi. Această abordare poate fi rezumată ca *Problema găsirii drumului de cost minim* sau *Drumuri minime in graf*.

1.2 Utilizări

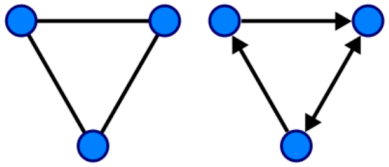
O aplicație practică este reprezentată chiar de sistemul GPS (Global Positioning System) care se folosește de astfel de algoritmi pentru a transmite informații despre cea mai rapidă rută spre un anumit loc. Acest sistem este bazat pe o rețea de sateliți care orbitează în jurul Pământului transmițând semnale receptorilor de pe sol, semnalele conțin un cod de timp și un punct de date geografice, pentru ca mai apoi să fie procesate de algoritmii implementați pe baza grafurilor. De asemenea, acești algoritmi sunt adesea utilizați pentru stabilirea unei agende de zbor în vederea asigurării unor conexiuni optime.

1.3 Graful la nivel teoretic

Ce este un graf?

Un graf este o pereche de mulţimi G = (V, E). Mulțimea V conține nodurile grafului (vertices), iar mulțimea E conține muchiile sale (edges), fiecare muchie stabilind o relație de vecinătate între două noduri. Mulţimea E este inclusă în mulţimea V x V.

Un graf poate fi orientat sau neorientat. Dacă este neorientat înseamnă că muchiile acestuia au asociate un sens.



1.4 Specificarea soluțiilor

Așadar, problema ce va fi expusă în acest raport este următoarea: se dă un graf G cu N noduri și ne propunem să aflăm costul minim de la un nod la toate celelalte.

Algoritmii propuși spre analiză sunt:

- (1): Dijkstra
  În primul rând, algoritmul lui Dijkstra poate fi folosit pentru grafuri care au toate muchiile nenegative.
  Vom avea o coadă de prioritate, care va avea la început un cost estimativ pentru fiecare nod în parte d[v]. Acesta va fi inițializat la început cu costul muchiei respective (sau cu ∞ daca nu există muchie). Costurile vor fi îmbunătățite la fiecare pas, pe baza estimării celorlalte costuri.

- (2): Bellman-Ford
  Deoarece există cazuri când muchiile pot avea costuri negative și nu vom putea aplica Dijkstra (care are o complexitate O((|V| + |E|) * log|V|)), vom folosi algoritmul Bellman-Ford. Spre deosebire de algoritmul Dijkstra, care foloseşte muchiile o singură dată pentru îmbunătăţirea costului, Bellman-Ford foloseşte fiecare muchie de N-1 ori, repetarea asigurând propagarea distanţei minime prin graf. Algoritmul permite detectarea ciclurilor de cost negativ, apărând în cazul în care, după cele N-1 repetiţii, există o muchie care îmbunătăţeşte costul vreunui nod.

- (3): Dijkstra adaptat (pentru costuri foarte mici ale muchiilor)
  Dacă avem costurile numere întregi în intervalul [1, W], putem aduce complexitatea la valorea O((|V| + |E|) * logW)). De această dată, coada de prioritate nu va mai depăși dimensiunea W deoarece elementele acesteia nu vor mai fi nodurile, ci o referință la o structură de date numită bucket. Fiecare bucket va conține un set de noduri care au aceeași distanță estimată, prioritatea cozii fiind această distanță.

1.5 Criterii de evaluare a algoritmilor

Pentru fiecare algoritm, voi avea în vedere:

- teste de dimensiuni mici, care vor fi ușor de urmărit, pentru a putea verifica corectitudinea programelor
- teste in care grafurile vor avea diverse costuri (pozitive sau negative, mari sau mici) pentru a putea vedea comportamentul fiecărui algoritm
- teste de dimensiuni nedeterminate pentru a putea testa situații de tipul best/worst/average complexity

În ceea ce privește generarea testelor, voi lua niște seturi de date de pe internet pe care le voi adapta pentru cazurile mele, cu scopul de a obține o evaluare cât mai variată și corectă.

Bibliografie

[1] https://cs.stackexchange.com/questions/19252/dijkstras-algorithm-for-edge-weights-in-range-0-w.

[2] https://ocw.cs.pub.ro/courses/sd-ca/laboratoare/lab-07.

[3] https://ocw.cs.pub.ro/courses/pa/laboratoare/laborator-09.

[4] https://www.programiz.com/dsa/bellman-ford-algorithm.