PA - Drumuri de cost minim

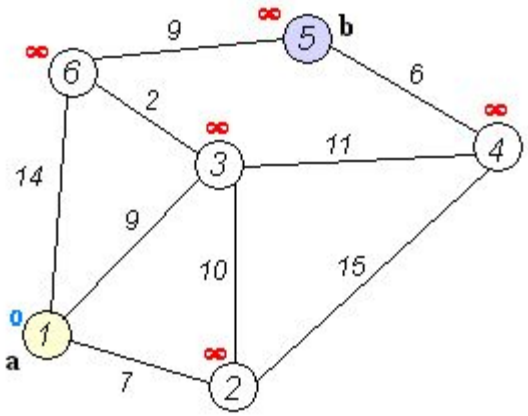
Daniel Chiș - 2022, UPB, ACS, An I, Seria AC

Drumuri de cost minim

Algoritmul Dijkstra

Edsger W. Dijkstra inigner software de origine olandeză.

Algoritmul Dijkstra este folosit pentru a găsi drumul cel mai scurt pentru de la un nod sursă către un alt nod. Algoritmul este asemănător cu cel al lui Prim, astfel generăm arborele de cost minim.



Algoritm

- iniţial, toate nodurile sunt neexplorate şi vom construi arborele, începând de la nodul S
- atribuim un posibil cost (o estimare a distanţei) pentru fiecare nod. (iniţial, S are costul 0, toate celelalte noduri au costul infinit)
- la fiecare pas, alegem cel mai bun candidat dintre nodurile neexplorate, urmând să îl explorăm(să îi evaluăm vecinii), iar acel candidat va rămâne în arbore
- la fiecare explorare, dacă găsim o nouă estimare de cost mai bună decât cea precedentă, folosim, mai departe, noua estimare. Dacă dorim să ţinem evidenţa muchiilor folosite, actualizăm şi nodul părinte al vecinului respectiv.

Diferenţa apare, în algoritmul lui Dijkstra, la funcţia folosită pentru estimarea costurilor, atunci când evaluăm vecinii unui nod. Dacă C este nodul curent(pe care îl explorăm), atunci pentru fiecare nod vecin(V) al lui C, noul cost posibil va fi costul drumului S-V (de la S la V) care trece prin C, mai exact - suma dintre costul drumului S-C şi costul muchiei (C,V)

1. Declarăm două mulţimi:

- mulţimea nodurilor neexplorate(MN), iniţial MN conţine toate nodurile
- mulţimea nodurilor explorate(ME) ce compun arborele, iniţial ME = vidă

2. Atribuim fiecărui nod o estimare iniţială a costului:

- 0 pentru nodul sursă(S)
- infinit pentru toate celelalte

3. Cât timp există noduri în MN

- Alegem, din MN(nodurile neexplorate), nodul cu cel mai mic cost estimat îl numim C(nodul curent)
- pentru fiecare din vecinii lui C care se află în MN
- calculăm noua estimare de cost = cost(drumul S-C) + cost(muchia (C,V))
- comparăm noua estimare cu vechiul cost(drumul S-V): dacă noul cost e mai bun
    - actualizăm cost(drumul S-V) = noul cost
    - actualizăm parinte(V) = C; (pentru păstrarea muchiei folosite) altfel păstrăm vechiul cost
- Marcăm nodul C ca explorat: îl eliminăm din MN şi îl adăugăm în ME.

Algoritmul Bellman-Ford

Principii similare pentru algoritmul Bellman-Ford:

- vom construi arborele, începând de la nodul S
- atribuim un posibil cost (o estimare a distanţei) pentru fiecare nod (iniţial, S are costul 0, toate celelalte noduri au costul infinit)
- la fiecare evaluare, dacă găsim o nouă estimare de cost mai bună decât cea precedentă, folosim, mai departe, noua estimare. Dacă dorim să ţinem evidenţa muchiilor folosite, actualizăm şi nodul părinte
- funcţia de estimare a costului este definită la fel ca la algoritmul lui Dijkstra(costul drumului de la S la nodul respectiv)

Diferenţa apare, în algoritmul Bellman-Ford, la alegerea nodurilor pentru care facem evaluarea:

Algoritmul nu are preferinţe pentru anumite noduri şi nu extrage, la fiecare pas, cel mai bun candidat. În schimb, acest algoritm evaluează toate muchiile la un pas. Folosindu-se de principiul de mai sus, (N-1) astfel de paşi vor fi suficienţi.

1. Atribuim fiecărui nod o estimare inițială a costului:

- 0 pentru nodul sursă(S)
- infinit pentru toate celelalte

2. Executăm de N-1 ori:

    1. Pentru fiecare pereche (u, v) a.i. există muchie de la u la v

        1. calculăm noua estimare de cost = cost(drumul S-u) + cost(muchia (u,v))

        2. comparăm noua estimare cu vechiul cost(drumul S-v):

        dacă noul cost e mai bun

            1. actualizăm cost(drumul S-v) = noul cost

            2. actualizăm parinte(v) = u

Observații

Dijkstra este de preferat în cazul grafurilor în care costurile sunt pozitive.

Bellman-Ford este de preferat în cazul grafurilor în care avem și costuri negative la muchii.

Exerciții

Exerciții

Creați un graf cu minim 10 noduri și 10 muchii. Fiecare muchie va avea un cost asociat. (altul de decât cel de la laboratorul precedent). Trebuie să aveți și muchii cu cost negativ. Aplicati algoritmii pe graful creat acum și cel de la vechea temă.

a) Realizați drumul de cost minim de la un nod ales sursa pana la un altul folosind algoritmul Dijkstra. Afișați ordinea nodurilor, valoarea muchiilor din arbore și costul total. 4.5
b) Realizați drumul de cost minim de la un nod ales sursa pana la un altul folosind algoritmul lui Bellman-Ford. Afișați ordinea nodurilor, valoarea muchiilor din arbore și costul total. 4.5

Tema trebuie să includă și schema grafului.

Trimiteți și o diferență pe care o observați.

Exerciții FIIR

Folosind graful de la laboratorul 7:

a) Realizați drumul de cost minim de la un nod ales sursa pana la un altul folosind algoritmul Dijkstra. Afișați ordinea nodurilor, valoarea muchiilor din arbore și costul total. 4.5
b) Realizați drumul de cost minim de la un nod ales sursa pana la un altul folosind algoritmul lui Bellman-Ford. Afișați ordinea nodurilor, valoarea muchiilor din arbore și costul total. 4.5

Trimiteți și o diferență pe care o observați.

Dijkstra: https://www.geeksforgeeks.org/dijkstras-shortest-path-algorithm-greedy-algo-7/

Bellman Ford: https://www.programiz.com/dsa/bellman-ford-algorithm