Algoritmi pe grafuri

Deaconu Ioan

ioan.deaconu@cti.pub.ro

ROSEdu Summer Workshops

Cuprins

- Introducere
- Noțiuni teoretice
- Sesiunea practică
  - 10 probleme (ușor, mediu, avansat)
  - rezolvări

Despre mine

- Sunt student în anul 2 la calculatoare, ACS.
- Îmi place să fac orice legat de acest domeniu, de la roboței la algoritmi, drept dovada stau participările mele la CDL, WebDev, Robochallenge și lista continuă.
- Ultimul proiect la care am lucrat este o aplicație de recunoaștere facială pentru Android, aplicație dezvoltată în cadrul NCIT Summer School .

Utilitatea grafurilor

- Aplicații practice în multe domenii cum ar fi economie, sociologie, inginerie, industria jocurilor video, grafică,etc.
- Drumuri de cost minim, sortări topologice
- Site-urile de socializare, gen Facebook, care folosesc grafurile pentru a sugera prieteni și a putea afișa aceeași postare mai multor utilizatori, doar pe baza relațiilor dintre ei.

ROSEdu Summer Workshops

Moduri de reprezentare

- Grafurile pot fi reținute sub forma de
- matrici de adiacență
    - a[i][j] = 1, dacă există muchie de la i la j
    - a[i][j] = 0, altfel
- liste de adiacență, a[i] = lista nodurilor vecine cu i

- listă de muchii, se reține doar muchiile, sub forma unei perechi de noduri (i,j)

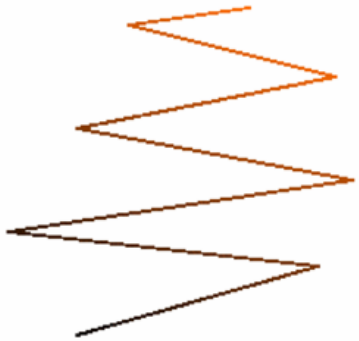
- Notații utile n – nr de noduri , m – nr de muchii

Proprietăți

- Grafuri -orientate / neorientate
- conexitate – drum între oricare 2 noduri
- Ciclu eulerian – vizitează toate muchiile
- Ciclu hamiltonian – vizitează toate nodurile

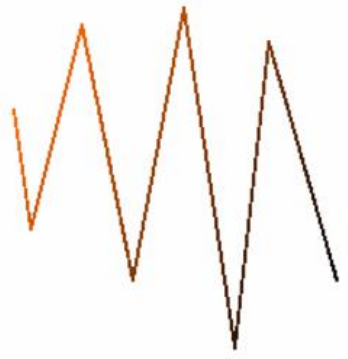
Algoritmi pe grafuri
BFS

- Bfs - cunoscut ca si Lee sau pacurgerea în lățime – se introduce nodul de start într-o coadă. Pentru fiecare nod din coada, se iau toți vecinii nevizitați ai lui și se introduc în coadă . Procedeul se repetă pânâ s-a ajuns la destinație sau nu mai avem noduri de vizitat.
  O(n+m)(lista) O(n^2) (matrice)



DFS

- Pargurerea recursivă a grafului (în adâncime).
- Se ia un nod de start, se consideră vizitat, și pentru fiecare vecin al lui, se reapelează funcția de parcurgere, până ai ajuns la destinație, sau ai parcurs tot graful.
  O(n+m)(lista) O(n^2) (matrice)



Kosaraju
componente tare conexe

- Se parcurge DFS graful și se depun nodurile într-o stivă, care reține ordinea de parcurgere.
- Se pargure DFS graful transpus, pornind cu primul nod din stivă. Toate nodurile scoase din stivă în timpul lui DFST, aparțin unei componente conexe.
- Se repetă procedeul până când nu mai avem noduri în stiva.

Tarjan
componente tare conexe

- Se bazează pe o parcurgere DFS si pe timpii de descoperire și de terminare de explorare a unui nod
- Idx – timpul de descoperire
- Lowlink – nodul cel mai vechi (după timpul de descoperire) cu care are legatură nodul curent

```
tarjan(G, v)
    idx[v] = index
    lowlink[v] = index
    index = index + 1
    push(S, v)
    pentru (v, u) din E
        dacă (idx[u] nu e definit)
            tarjan(G, u)
            lowlink[v] = min(lowlink[v], lowlink[u])
        altfel
            dacă (u e in S)
                lowlink[v] = min(lowlink[v], idx[u])
    dacă (lowlink[v] == idx[v])
        // este v rădăcina unei CTC?
        print "O nouă CTC: "
        repeat
            u = pop(S)
            print u
        until (u == v)
```

Dijkstra
drumuri de cost minim

- Algoritmul este de tip Greedy: optimul local căutat este reprezentat de costul drumului dintre nodul sursă și un nod v. Pentru fiecare nod se reține un cost estimat d[v], inițializat la început cu costul muchiei s → v, sau cu +∞, dacă nu există muchie
- Algoritmul selectează, în mod repetat nodul u care are, costul estimat minim fată de nodul sursă. Se încearcă să se relaxeze restul costurilor d[v]. Dacă d[v] < d[u] + c(u,v) , d[v] ia valoarea d[u] + c(u,v).

Bellman-Ford

- Asemănător cu Dijkstra, doar că realizează relaxările tuturor muchiilor de n ori
- Poate fi folosit în grafuri cu muchii de cost negativ, dar care nu conțin cicluri negative
- Dacă după cele n*m relaxări se mai poate realiza una, graful conține ciclu negativ, iar problema nu are soluție. O(n*m)

Floyd- Warshall

- Compară toate drumurile posibile din graf dintre fiecare 2 noduri folosind relaxări succesive printr-un nod intermediar k
- Poate fi utilizat și în grafuri cu muchii de cost negativ O(n^3)

Edmonds - Karp

- Cât timp există un drum de la sursă la destinație, se ia fluxul minim de pe acel drum, și se scoate din muchiile care compun drumul.
- La final, fluxul maxim va fi reprezentat de suma fluxurilor minime de pe fiecare drum găsit.

1) Metroul

- Avem un număr n de stații de metrou, conectate între ele prin tuneluri. Se numește rută principală în rețea un ciclu. Se garantează ca rețeaua conține un singur ciclu.
- Să se determine distanța fiecărei stații de metrou față de ruta principală.

Rezolvare

- Se face o parcurgere (dfs, sau bfs) pentru a se afla care este ciclul
- Pentru fiecare nod din ciclu, se ia fiecare vecin care nu apare în ciclu, și se incrementeaza distanța față de rută. O(m), unde m e nr de muchii.

2) Arbore5

- Marian deține un arbore cu N noduri, ale cărui ramuri sunt vopsite în alb. Din păcate, vecinul lui, Marius, îi strică arborele, prin K operații
- Pentru o pereche (a,b), vopsește toate muchiile de pe drumul de la a la b în culoare opusă care o aveau (din alb în negru, și invers)
- Ajutați-l pe Marian să numere câte ramuri albe mai are în arbore(arborele poate avea N 100000 de noduri, și are mereu N-1 ramuri)

Rezolvare

- Se memorează graful sub formă de listă de adiacență. Pentru fiecare pereche (a,b), se află drumul (DFS, sau BFS, se realizează în O(N)), apoi se inversează culoarea pentru fiecare muchie din drum.
- La final, se parcurge o singură dată graful, pentru a se afla numărul de ramuri albe rămase O(M*N)

3) Petrecere

- Pentru a sărbători o sesiune fără restanțe, Paul organizează o petrecere la care vor fi invitați doar prietenii lui, și prieteni ai acestora. El are mulți prieteni, dar între ei pot exista relații de ură.
- Determinați grupul maxim de prieteni pe care îi poate invita la petrece, astfel încât să nu existe o relație de ură între doi prieteni, iar toți prietenii prietenului să fie invitați.

ROSEdu Summer Workshops

Rezolvare

- Se determină componenta conexă de dimensiune maximă, care nu conține nici o relație de ură. O(N + M)

4) Graf

- Se știe că într-un graf neorientat conex, între oricare două vârfuri există cel puțin un lanț iar lungimea unui lanț este egală cu numărul muchiilor care-l compun. Definim noțiunea de lanț optim între două vârfuri *X* și *Y* ca fiind un lanț de lungime minimă care are ca extremități vârfurile *X* si *Y*.
- Determinați vârfurile care aparțin tuturor lanțurilor optime dintre *X* și *Y*.

Rezolvare

- Stabilim prin câte o parcurgere în lăţime distanţele tuturor vârfurilor faţă de X şi respectiv Y (capetele lanţului, citite din fişier). Vedem care dintre vârfurile ce aparţin cel puţin unui lanţ de lungime minima între X şi Y au proprietatea ca sunt singurele aflate la o anumită distanţă de X. Acestea sunt vârfurile care aparţin tuturor lanţurilor de lungime minimă dintre X şi Y.
- Algoritmul are complexitatea O(n+m).

5) Distanțe

- Un bun prieten de-al lui Zăhărel, Bronzărel este la spital bolnav de tifos, și are obiceiul să deseneze pe pereții spitalului mai multe grafuri neorientate conexe cu costuri. Mai mult de atât, Bronzărel le-a calculat și distanțele minime de la un nod sursa la celelalte.
- Curios din fire, Zăhărel vrea să verifice dacă distanțele minime au fost calculate corect.

Rezolvare

- Se bazează pe proprietățile algoritmului Dijkstra:
- Ds = 0
- Dx + cost(x,y) ≥ Dy, pentru orice muchie (x,y)
- Pentru fiecare nod y(diferit de S), există un x astfel încât Dx + cost(x,y) = Dy
- O(n+m)

6) Roy- Floyd

- Micul Floyd locuiește într-un oraș mare, în care există *N* intersecții. Fiecare pereche de intersecții este conectată printr-un drum bidirecțional având o lungime pozitivă dată.
- Micul Floyd este un băiat curios și dorește să știe care este distanța minimă pe care cineva ar trebui să o parcurgă de-a lungul drumurilor existente dacă ar vrea să meargă din intersecția *X* în intersecția *Y*, cât și numărul maxim de străzi pe care ar putea să meargă cineva pentru a obține această distanță minimă.

Rezolvare

- Se aplică algoritmul Roy-Floyd standard, la care se adaugă un criteriu suplimentar: în cazul în care se găsește un drum de distanță egală cu distanța minimă curentă, dar având un număr mai mare de străzi, se va alege drumul respectiv. O(n^3)

7) Smen

- Fie un șir de *N* numere naturale (nu neapărat distincte) pe care se pot efectua următoarele operații: la un anumit pas , elementul *i* al șirului poate fi crescut sau scăzut cu o unitate
- Aplicând acestă metodă asupra unor anumite elemente din șir, obțineți (printr-un număr minim de operații) cel puțin *K* elemente distincte, care să aparțină intervalului *[A, B]*.

Rezolvare

- Transformarea problemei într-una de cuplaj de cost minim
- Două mulțimi de noduri : prima mulțime e formată din șirul initial, iar cea de-a doua de valorile întregi din *[A, B]*.
- Se introduc *N**(*B-A*) muchii de capacitate *1*, ce leagă elementul *X* din prima mulțime și elementul *Y* din a doua mulțime având costul */X-Y/*. Prima mulțime se leagă de o sursă fictivă prin muchii de cost *0* și capacitate *1*, iar cea de-a doua mulțime de o destinație fictivă prin muchii de cost *0* și capacitate *1*. Ținându-se cont că în destinație trebuie să se obțină fluxul minim *K*, se aplică acest algoritm având grijă să minimizăm costul. O(N*(N+M))

8) Coach

- Datoria ta de antrenor al unui ciclist, este să îi stabilești traseul de antrenament. Ai la dispoziție un traseu format din intersecții și drumuri între ele. La fiecare intersecție se află o băutură energizantă, de o anumită cantitate, pe care ciclistul o bea ca să poată continua antrenamentul. Dorești ca elevul să parcurgă traseul în timpul T. Ciclistul nu are nevoie decât de intersecția de start și de finish, drumul pe care îl va parcurge fiind mereu cel de timp minim. Ca să obții un drum de timp T, poți să impui o limită minimă și una maximă asupra cantității de energizant pe care îl poate bea într-o intersecție.
- Determină numărul intersecției de start și finish, dar și cantitatea minimă, respectiv maximă de energizant pe care o poate bea într-o intersecție.

Rezolvare

- Se sortează nodurile după valoarea asociată şi se renumerotează (nodul 1 are valoarea cea mai mică). Se fixează un nod de start **s** (si deci un **cmin**) şi se aplică Roy-Floyd, pentru nodurile din graf care au valoarea asociată mai mare sau egală cu **cmin** (noduri de la **s** la **N**). Proprietatea algoritmului Roy-Floyd aplicat de la nodul **s** este următoarea: la pasul **k** (cu **k** de la **s** la **N**), A[**i**][**j**] este drumul cel mai scurt de la **i** la **j** care poate trece prin noduri de la **s** la **k**. O(n^4)

9) Lanterna

- Un agent secret trebuie să ajungă de la baza 1 la baza n, pe timpul nopții. Bazele sunt numerotate de la 1 la n, există drumuri între ele, iar unele baze pot fi prietene, sau inamice. El are o lanternă de capacitate W ($1 \leq W \leq K$), pe care o poate încărcă în bazele prietene. Drumul între 2 baze, durează un timp T, și necesită o lanterna de capacitate minima W.
- Determinați drumul de lungime minimă, dar și capacitatea minimă a lanternei, pentru a ajunge la baza inamică.

Rezolvare

- Căutare binară + belman ford cu coada. O(log K * N*M)

10) Ulei

- Lunasorab dorește să viziteze o galerie de artă care conține N camere și M coridoare,iar pe fiecare coridor se află un tablou picat în ulei. Există mai multe tipuri de tablouri în ulei, iar Lunasorab vrea să le vadă pe toate o singură dată, dar dorește ca întotdeauna următorul tablou vizitat să fie pictat în alt tip de ulei față de anteriorul. O altă cerință este ca el să se întoarcă din locul din care a plecat, iar ultimul tablou văzut să fie pictat în alt tip de ulei față de primul tablou văzut.

Rezolvare

- Se sortează tipuri de ulei după frecvența de apariție și se construiesc noduri cu 2 muchii, una de intrare (cu un tip de tablou), iar cealaltă de ieșire(cu alt tip de tablou).
- Se aleg 2 noduri, se construiește un ciclu din ele, după care se va introduce mereu între primul și al doilea nod din ciclu, un nod, care respectă proprietatea următoare: ce iese dintr-un nod, intră în nodul următor. O(NlogN + M)

- 1) Metrou - http://www.codeforces.com/problemset/problem/131/D
- 2) Arbore5 - http://infoarena.ro/problema/arbore5
- 3) Petrecere - http://codeforces.com/problemset/problem/177/C1
- 4) Graf - http://infoarena.ro/problema/graf
- 5) Distanțe - http://infoarena.ro/problema/distante
- 6) Roy-Floyd - http://infoarena.ro/problema/rf
- 7) Smen - http://infoarena.ro/problema/smen
- 8) Coach - http://infoarena.ro/problema/coach
- 9) Lanterna - http://infoarena.ro/problema/lanterna
- 10) Ulei - http://infoarena.ro/problema/ulei