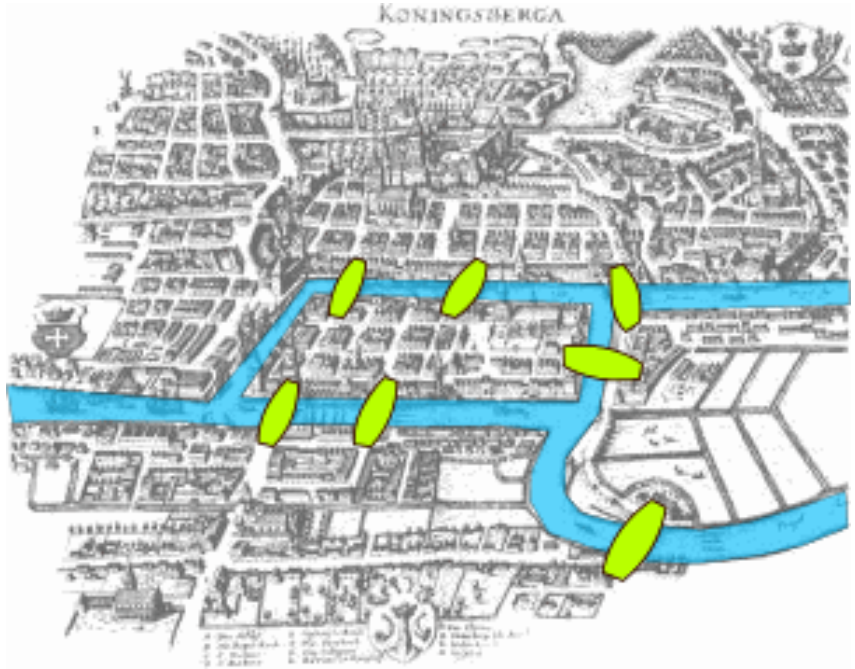
Programiranje v višji predstavi: Grafi in njihovo preiskovanje

Vid Kocijan

25. junij 2017

Kaj so grafi in zakaj jih sploh potrebujemo

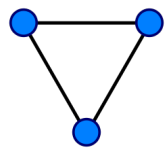
- Množica vozlišč in povezav med njimi.
- Vsako vozlišče ali povezava lahko nosi podatke (npr. teža povezave).
- Z njimi lahko opišemo cestna omrežja, socialna omrežja, interakcije proteinov, internet, stanja igre, itd.

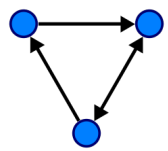


Slika : Problem Königsberških mostov, prvi grafovski problem, objavljen v 1736. Vir: Wikipedia

Nekaj osnovne terminologije

- Graf je sestavljen iz vozlišč (angl. vertices/nodes) in povezav (angl. edges)
- Povezave so lahko usmerjene ali neusmerjene. Grafu z usmerjenimi povezavami pravimo usmerjen graf.
- Povezave so lahko neobtežene ali obtežene. Grafu z obteženimi povezavami pravimo utežen graf.
- Običajno med vsakim parom vozlišč obstaja največ ena povezava. če jih je več, govorimo o multigrafu.
- Ko nekdo reče "graf" ima tipično v mislih neusmerjen, neobtežen, preprost, končen graf.





Še malo terminologije

- Graf brez ciklov je drevo. Vsakemu vozlišču lahko določimo "starša" in "otroke". Vozlišče brez staršev je koren, vozlišče brez otrok je list.
- Drevo, kjer ima vsako vozlišče natanko 2 ali 0 otrok je binarno drevo.
- Graf, kjer so vsi pari vozlišč povezani, je poln graf.
- Pot je zaporedje povezav, kjer se nobena od njih ne ponovi.
- Sprehod je zaporedje povezav, kjer se kakšna od njih lahko ponovi.
- Izrazov je še ogromno, a so na srečo relativno intuitivni.

Kako predstaviti grafe v računalniku?

- n vozlišč oštevilčimo s števili od 0 do $n - 1$.
- Seznam povezav - v tabeli hranimo pare povezanih vozlišč
- Matrika sosednosti - v tabeli velikosti $n \times n$ označimo, ali sta vozlišči povezani ali ne.
- Seznam sosednosti - za vsako vozlišče si hranimo seznam sosedov
- Ker animacije in slike povedo več od besedila: grafični prikaz

Pogosti problemi na grafih

- Najkrajša pot med dvemi vozlišči (GPS)
- Vpeta drevesa (Najcenejše napeljevanje elektrike)
- Povezane komponente v usmerjenih grafih (Ali lahko pridemo iz poljubne točke v poljubno drugo točko?)
- Iskanje prereznih točk, mostov, ciklov (Iskanje kritičnih točk omrežja, ki ne smejo odpovedati)
- Barvanje grafov (Kako pobarvati zemljevid, da nobeni sosednji državi nista enake barve)
- Problem potujočega trgovca (Verjetno najbolj znani NP problem)
- Računanje pretokov (Koliko vode steče po vodovodnem omrežju)

Pogosti problemi na grafih

- Najkrajša pot med dvemi vozlišči (GPS)
- Vpeta drevesa (Najcenejše napeljevanje elektrike)
- Povezane komponente v usmerjenih grafih (Ali lahko pridemo iz poljubne točke v poljubno drugo točko?)
- Iskanje prereznih točk, mostov, ciklov (Iskanje kritičnih točk omrežja, ki ne smejo odpovedati)
- Barvanje grafov (Kako pobarvati zemljevid, da nobeni sosednji državi nista enake barve)
- Problem potujočega trgovca (Verjetno najbolj znani NP problem)
- Računanje pretokov (Koliko vode steče po vodovodnem omrežju)
- Problemov je veliko, posvetili se bomo iskanju najkrajših poti.

Preiskovanje neuteženih grafov

- Določimo začetno vozlišče s in končno vozlišče t.
- Najti želimo pot med njima z minimalnim številom povezav.
- Obstajata dve strategiji preiskovanja: preiskovanje v širino in preiskovanje v globino.

Preiskovanje v širino (angl. Breadth first search)

- Ideja: Najprej bomo poiskali vse sosede vozlišča s (razdalja 1), nato njihove sosede, ki jih še nismo obiskali (razdalja 2), itd.
- Na koncu dobimo ne le razdaljo od s do t, pač pa razdaljo od s do vseh ostalih vozlišč.
- Vozlišča, ki jih še moramo pregledati, hranimo v vrsti.
- V vsakem koraku vzamemo prvo vozlišče v vrsti in vse njegove neobiskane sosede dodamo v vrsto.
- Ponavljamo, dokler ne zmanjka vozlišč.
- Skupna časovna zahtevnost je $O(E + V)$.
- Ker animacije in slike povedo več od besedila: grafični prikaz

Preiskovanje v širino (angl. Breadth first search)

- Ideja: Najprej bomo poiskali vse sosede vozlišča s (razdalja 1), nato njihove sosede, ki jih še nismo obiskali (razdalja 2), itd.
- Na koncu dobimo ne le razdaljo od s do t, pač pa razdaljo od s do vseh ostalih vozlišč.
- Vozlišča, ki jih še moramo pregledati, hranimo v vrsti.
- V vsakem koraku vzamemo prvo vozlišče v vrsti in vse njegove neobiskane sosede dodamo v vrsto.
- Ponavljamo, dokler ne zmanjka vozlišč.
- Skupna časovna zahtevnost je $O(E + V)$.
- Ker animacije in slike povedo več od besedila: grafični prikaz
- Primer implementacije je na voljo na spletni učilnici

Preiskovanje v globino (angl. Depth first search)

- Ideja: Premikamo se kar naprej po grafu dokler gre, ko nimamo kam, se vračamo.
- To nam ne zagotavlja najkrajše poti (razen na drevesih). Zakaj je ta pristop potem uporaben?

Preiskovanje v globino (angl. Depth first search)

- Ideja: Premikamo se kar naprej po grafu dokler gre, ko nimamo kam, se vračamo.
- To nam ne zagotavlja najkrajše poti (razen na drevesih). Zakaj je ta pristop potem uporaben?
- Je osnova drugih algoritmov in dinamičnega programiranja na drevesih in usmerjenih grafih.
- Algoritem je zelo podoben preiskovanju v širino, le da vrsto nadomestimo s skladom.
- Časovna zahtevnost je $O(E + V)$.
- grafični prikaz

Preiskovanje v globino (angl. Depth first search)

- Ideja: Premikamo se kar naprej po grafu dokler gre, ko nimamo kam, se vračamo.
- To nam ne zagotavlja najkrajše poti (razen na drevesih). Zakaj je ta pristop potem uporaben?
- Je osnova drugih algoritmov in dinamičnega programiranja na drevesih in usmerjenih grafih.
- Algoritem je zelo podoben preiskovanju v širino, le da vrsto nadomestimo s skladom.
- Časovna zahtevnost je $O(E + V)$.
- grafični prikaz
- Preiskovanje v globino pa ima tudi lepo rekurzivno implementacijo.

Iterativno poglabljanje

- V primerih, kjer velikost grafa eksponentno narašča (stanja igre) nobeden od prejšnjih algoritmov ni primeren.
- Delamo več iteracij iskanja v globino. Prvič do globine 1, drugič do globine 2, itd. dokler ne najdemo iskanega vozlišča.
- Časovna zahtevnost je $O(E \cdot V)$, vendar pri eksponentno naraščujočih grafih to nima hudega vpliva.
- Implementacija je prepuščena bralcem v premislek.

Dijkstrov algoritem

- Kaj pa ko imajo povezave zraven še (pozitivne) uteži?
- Ideja: Enaka strategija, kot pri preiskovanju v širino, le da gledamo vsoto uteži na poti.
- V vsaki iteraciji izmed neobiskanih vozlišč vzamemo tistega, do katerega vodi najkrajša pot.
- Isti pristop kot pri preiskovanju v širino, le da namesto vrste uporabimo kopico, ki neobiskana vozlišča ureja po razdalji.
- Časovna zahtevnost je $O((E + V) \log(V))$.
- grafični prikaz

Dijkstrov algoritem

- Kaj pa ko imajo povezave zraven še (pozitivne) uteži?
- Ideja: Enaka strategija, kot pri preiskovanju v širino, le da gledamo vsoto uteži na poti.
- V vsaki iteraciji izmed neobiskanih vozlišč vzamemo tistega, do katerega vodi najkrajša pot.
- Isti pristop kot pri preiskovanju v širino, le da namesto vrste uporabimo kopico, ki neobiskana vozlišča ureja po razdalji.
- Časovna zahtevnost je $O((E + V) \log(V))$.
- grafični prikaz
- Kopice v standardnih knjižnicah tipično sortirajo tako, da je na vrhu največji element. Pravilno bi bilo, da spišemo lastno primerjalno funkcijo, vendar je lažje, če samo vstavljamo negativne vrednosti.

Naloge za vajo

- Link do codeforces skupine, kjer so vaje in bo tudi tekma link