Požrešni algoritmi v grafih, najčcenejša vpeta drevesa, DSU

Filip Koprivec

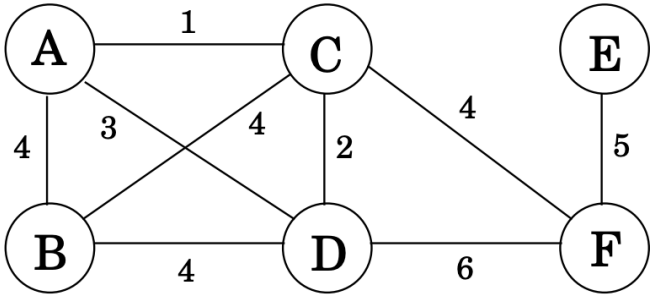
Fakulteta za Matematiko in Fiziko

Marec 2019

Domača naloga

Vprašanja, težave?

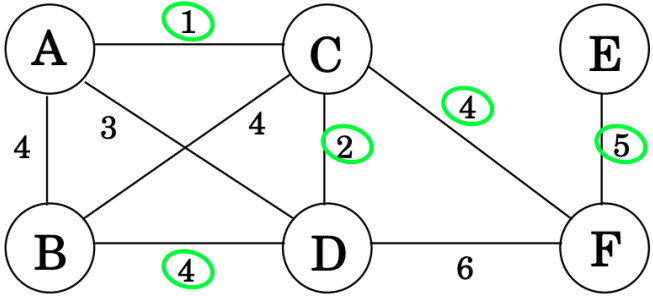
(Najcenejše) vpeto drevo



(Najcenejše) vpeto drevo

V grafu iščemo vpeto drevo, poleg vsega si želimo najcenejše

(Najcenejše) vpeto drevo



Netekmovalna motivacija

- Minimalna vpeta drevesa imajo cel kup možnih praktičnih uporab v računalniških omrežjih
- Postavitev cest
- Električna omrežja
- Aproksimacije za TSP
- Steinerjeva drevesa
- ...

Roller Coaster Railroad - IOI 2016

- Imamo posamezne točke v katere moramo vstopiti z omejeno hitrostjo s_i
- Iz njih izstopimo z hitrostjo t_i
- Dodatno moramo postaviti posamezne proge, kjer na vsakem metru vlakec upočasni za $1\frac{\mathrm{m}}{\mathrm{s}}$
- To seveda želimo narediti čim ceneje

- Kot pri Dijkstri, skoraj nikoli ni rešitev direktno minimalno vpeto drevo (MST)
- Načeloma je treba kar nekaj razmisleka
- MST je pogosto zgolj en korak (ali pa zaključek)
- Uporabna je požrešnost in dodatne strukture

MST

- Dva osnovna algoritma
- Primov algoritem
- Kruskallov algoritem

Opažanja

Opažanja

- Ali je smiselno iskati ne-drevo?

Opažanja

- Ali je smiselno iskati ne-drevo?
- Kaj naredimo če pridemo do cikla?

Opažanja

- Ali je smiselno iskati ne-drevo?
- Kaj naredimo če pridemo do cikla?
- Ali smo lahko požrešni?

Opažanja

- Ali je smiselno iskati ne-drevo?
- Kaj naredimo če pridemo do cikla?
- Ali smo lahko požrešni? DA

Opažanja

- Ali je smiselno iskati ne-drevo?
- Kaj naredimo če pridemo do cikla?
- Ali smo lahko požrešni? DA
- Negativne povezave, negativni cikli?
- Posplošitve?

Kruskall

- V rešitvi bo gotovo vsaj eno vozlišče

Kruskall

- V rešitvi bo gotovo vsaj eno vozlišče
- Če imamo minimalno vpeto na ostalih vozliščih lahko novo drevo povežemo najceneje

Kruskall

- V rešitvi bo gotovo vsaj eno vozlišče
- Če imamo minimalno vpeto na ostalih vozliščih lahko novo drevo povežemo najceneje
- Ključna ideja
- Počasi povečujemo trenutno najdaljše drevo, vedno vzamemo najcenejšo povezavo
- Razen če je ta že obiskana

Kruskall

- Koda

Kruskall

- Koda
- Kako preverimo že obiskana vozlišča?

Kruskall

- Koda
- Kako preverimo že obiskana vozlišča?
- Kako uspešno iščemo najcenejše povezave?
- Časovna zahtevnost
- $\mathrm{E} \log \mathrm{V}$

Kruskall

- Koda
- Kako preverimo že obiskana vozlišča?
- Kako uspešno iščemo najcenejše povezave?
- Časovna zahtevnost
- $\mathrm{E} \log \mathrm{V}$
- V resnici $\mathrm{E} \log \mathrm{E}$

Celotna koda

Primov algoritem

- Drugačna ideja
- Namesto širjenja okoli vozlišča gledamo na povezavo

Primov algoritem

- Drugačna ideja
- Namesto širjenja okoli vozlišča gledamo na povezavo
- Potrebujemo pomožno podatkovno strukturo

Primov algoritem

- Katera povezava bo gotovo v najcenejšem vpetem drevesu

Primov algoritem

- Katera povezava bo gotovo v najcenejšem vpetem drevesu
- Spet ista ideja *združimo* obiskanje povezave

Primov algoritem

- Katera povezava bo gotovo v najcenejšem vpetem drevesu
- Spet ista ideja *združimo* obiskanje povezave
- Ker gradimo gozd, moramo pametno preverjati kdaj smo v isti komponenti!

Združevanje komponent (DSU)

Disjoint set union

- Dve (tri) osnovni operaciji
- `Preveri komponento`
- Združi v isto komponento

UPM 2012, 1.kolo: Glasovanje

- Imamo N strank, vsaka ima svoje mnenje o zakonu (vstavi poljubno stvar, ki je v nedeljo zvečer po televiziji: II Tir, kraja sendviča, ...)
- Stranke se sestajajo in poskušaj odoseči konsenz
- Če se sestanejo tri stranke, bo mnenje spremenila tista, ki je v manjšini

UPM 2012, 1.kolo: Glasovanje

- Imamo N strank, vsaka ima svoje mnenje o zakonu (vstavi poljubno stvar, ki je v nedeljo zvečer po televiziji: II Tir, kraja sendviča, ...)
- Stranke se sestajajo in poskušaj odoseči konsenz
- Če se sestanejo tri stranke, bo mnenje spremenila tista, ki je v manjšini
- Kaj pa če stranke na sestanku sklenejo koalicijo, in ko si 'premisli' ena, to prenese na vse?

Ideje?

Ideje?

- Vsaka stranka je svoje vozliče v grafu

Ideje?

- Vsaka stranka je svoje vozliče v grafu
- Da ju združimo dodamo povezavo
- Preverjanje je malo bolj zahtevno

Ideje?

- Vsaka stranka je svoje vozliče v grafu
- Da ju združimo dodamo povezavo
- Preverjanje je malo bolj zahtevno
- O(1) za združevanje, O(N) za preverjanje

Ideje?

- Vsaka stranka je svoje vozliče v grafu
- Da ju združimo dodamo povezavo
- Preverjanje je malo bolj zahtevno
- O(1) za združevanje, O(N) za preverjanje
- Še kar slabo...

DSU ideja

- Za vsako množico izberimo predstavnika (vsak glasuje tako kot mu reče neposredni vodja)

DSU ideja

- Za vsako množico izberimo predstavnika (vsak glasuje tako kot mu reče neposredni vodja)
- Pri združevanju posodobimo predstavnika (parent)
- Za uskanje se sprehodimo do najvišjega starša (na koncu vsi poslušajo velikega vodjo)
- Kako pametno preurediti združevanje, da je iskanje sedaj hitrejše?

Union by rank

- Kako pametno združiti dve drevesi?
- Manjšega pridružimo večjemo
- Višina se poveča natanko tedaj ko

Union by rank

- Kako pametno združiti dve drevesi?
- Manjšega pridružimo večjemo
- Višina se poveča natanko tedaj ko sta višini obeh enaki

Union by rank

- Kako pametno združiti dve drevesi?
- Manjšega pridružimo večjemo
- Višina se poveča natanko tedaj ko sta višini obeh enaki
- Z malo matematike se da pokazati, da ima drevo reda k vsaj 2^k otrok
- Časovna zahtevnost je $\log n$

Union by rank

- Kako pametno združiti dve drevesi?
- Manjšega pridružimo večjemo
- Višina se poveča natanko tedaj ko sta višini obeh enaki
- Z malo matematike se da pokazati, da ima drevo reda k vsaj 2^k otrok
- Časovna zahtevnost je $\log n$
- Je to vredno?

Izboljšava iskanja

- Težava je v iskanju poti do starša
- Lahko to kako izboljšamo?

Izboljšava iskanja

- Težava je v iskanju poti do starša
- Lahko to kako izboljšamo?
- Med iskanje starša sproti posodabljamo in pot kompresiramo (path compression)
- $O(log^* n)$ ali $O(\alpha(n))$, kar je praktično konstantno (V resnici je optimalno v amortiziranem času)

Izboljšava iskanja

- Težava je v iskanju poti do starša
- Lahko to kako izboljšamo?
- Med iskanje starša sproti posodabljamo in pot kompresiramo (path compression)
- $O(log^* n)$ ali $O(\alpha(n))$, kar je praktično konstantno (V resnici je optimalno v amortiziranem času)
- Ali s tem kaj izgubimo?

Tehnikalije

- V resnici zahtevamo tri operacije
- `make_set`, `find_parent`, `merge`

Tehnikalije

- V resnici zahtevamo tri operacije
- `make_set`, `find_parent`, `merge`
- V resnici uporabljamo kar vektor in indekse (število elementov je praviloma znano vnaprej)

Tehnikalije

- V resnici zahtevamo tri operacije
- `make_set`, `find_parent`, `merge`
- V resnici uporabljamo kar vektor in indekse (število elementov je praviloma znano vnaprej)
-
- Rank ?!?

Stranke

Preprosto uporabimo DSU
DSU je v resnici uporaben za celo vrsto drugih združevanj različnih stvari, je hitra enostavna in lahka za impementacijo

Prim MST

- Podobno kot pri Kruskallu
- Gradimo drevo iz gozda dreves, preverimo če lahko združimo

Prim MST

- Podobno kot pri Kruskallu
- Gradimo drevo iz gozda dreves, preverimo če lahko združimo
- Kje začnemo

Prim

- Koda
- Časovna zahtevnost

Prim

- Koda
- Časovna zahtevnost
- $O(E \log E) + E\alpha(V)$
- Primerjava s Kruskallom?

Prim

- Koda
- Časovna zahtevnost
- $O(E \log E) + E\alpha(V)$
- Primerjava s Kruskallom?
- Slabši za goste grafe, razen če imamo podatke že urejene

Prim

- Koda
- Časovna zahtevnost
- $O(E \log E) + E\alpha(V)$
- Primerjava s Kruskallom?
- Slabši za goste grafe, razen če imamo podatke že urejene
- Negativne povezave?

Naloga

- Predstavljajmo si (neskončen, lahko celo zvezen) graf, kjer so vozlišča možne hitrosti, med seboj pa so povezane začetne in končne hitrosti za vsako 'posebno' točko
- Vsakemu segmentu lahko priredimo utež; kolikokrat moramo pospešiti čez to hitrost in kolikokrat upočasniti. (Pospeševanje je zastonj)
- Dovolj je da gledamo zgolj intervale hitrosti, ki so na voljo

- Predstavljajmo si (neskončen, lahko celo zvezen) graf, kjer so vozlišča možne hitrosti, med seboj pa so povezane začetne in končne hitrosti za vsako 'posebno' točko
- Vsakemu segmentu lahko priredimo utež; kolikokrat moramo pospešiti čez to hitrost in kolikokrat upočasniti. (Pospeševanje je zastonj)
- Dovolj je da gledamo zgolj intervale hitrosti, ki so na voljo
- Cena za interval: $\max(0, w * (x_{i+1} - x_i))$

- Predstavljajmo si (neskončen, lahko celo zvezen) graf, kjer so vozlišča možne hitrosti, med seboj pa so povezane začetne in končne hitrosti za vsako 'posebno' točko
- Vsakemu segmentu lahko priredimo utež; kolikokrat moramo pospešiti čez to hitrost in kolikokrat upočasniti. (Pospeševanje je zastonj)
- Dovolj je da gledamo zgolj intervale hitrosti, ki so na voljo
- Cena za interval: $\max(0, w * (x_{i+1} - x_i))$
- In smo konec

- Predstavljajmo si (neskončen, lahko celo zvezen) graf, kjer so vozlišča možne hitrosti, med seboj pa so povezane začetne in končne hitrosti za vsako 'posebno' točko
- Vsakemu segmentu lahko priredimo utež; kolikokrat moramo pospešiti čez to hitrost in kolikokrat upočasniti. (Pospeševanje je zastonj)
- Dovolj je da gledamo zgolj intervale hitrosti, ki so na voljo
- Cena za interval: $\max(0, w * (x_{i+1} - x_i))$
- In smo konec
- Ne čisto, dobimo koščke grafa, ki niso nujno povezani, da jih povežemo poženemo MST, da dobimo najcenejšo možnost

Najširša (najbolj ozka) pot

Najširša (najbolj ozka) pot

- Taka pot gotovo leži na MST (široka na maksimalnem, ozka na minimalnem)

Dijkstra

- Za samo eno začetno točko lahko razmišljamo tudi drugače
- Posodobimo Dijkstro

Dijkstra

- Za samo eno začetno točko lahko razmišljamo tudi drugače
- Posodobimo Dijkstro
- $\min \mapsto \max$, $(+) \mapsto \min$

UPM 2016, 1.kolo: Kočija

- Najdi najširšo pot med vsemi pari
- Hitreje kot z navadno Dijkstro

UPM 2016, 1.kolo: Kočija

- Najdi najširšo pot med vsemi pari
- Hitreje kot z navadno Dijkstro
- Trik omenjen zadnjič: Imamo le malo vozlišč, namesto vrste s prednostjo indeksiramo po urejenih težah (ker ne moremo dobiti vmesnih rezultatov z min)

Binarna kopica

Domača naloga

1. Dungeons and Candies
   Ključna ideja današnjega predavanja, iščemo MST, le da je treba malo razmisliti, kaj je naš graf, kaj so cene in zakaj je drevo dobra zadeva. Če ne drugega poskusite implementirati oba algoritma in ugotovite kateri je lažji, pri katerem ste ste večkrat zmotili. . . .
2. Swaps in Permutation
   DSU je super struktura, ki nam omogoča preprosto implementacijo nekaterih stvari (le par vrstic kode za razliko od B(D)FS). Razmislite, kako bi si željene stvari predstavili kot graf.
3. The Child and Zoo
   Ko delamo najmanjše vpeto drevo si lahko zraven shranjujemo še cel kup podatkov, če jih znamo smiselno posodabljati. Razmislite, kaj se dogaja pri združevanju dveh komponent in ali znate pametno združiti podatke iz obeh.