Iskanje kritične poti po principu dinamičnega programiranja
(iskanje dolžine najdaljše poti)

POGOJI ZA DELOVANJE
- graf je povezan
- graf je usmerjen
- ne sme biti ciklov

KAJ VPLIVA NA ČASOVNO ZAHTEVNOST (O (n+m))
- število vozlišč n
- število povezav m
- čas za izračun stopnje vozlišč

Če želimo izračunati še kritično pot (poleg dolžine) je potrebno za vsako vozlišče shraniti še predhodnika.

V algoritem za iskanje najkrajše poti ga spremenimo trivialno (vzamemo recipročno težo povezav ali samo obrnemo neenačaj).

Dijsktra (iskanje najkrajših poti)

POGOJI ZA DELOVANJE
- graf je usmerjen
- povezave morajo biti **pozitivne**
- graf mora biti povezan
- graf LAHKO vsebuje cikle
- poti od začetnega vozlišča morajo biti znane
- DECREASE_KEY

KAJ VPLIVA NA ČASOVNO ZAHTEVNOST (O(mlogn))
- število vozlišč n
- število povezav m
- implementacija prioritetne vrste
- operacija za zmanjšanje prioritete
- čas za dodajanje novih vozlišč v drevo povezav

V algoritem za iskanje najkrajše poti ga spremenimo trivialno (vzamemo recipročno težo povezav ali samo obrnemo neenačaj, s tem, da moramo popraviti še prioritetno vrsto, da je na vrhu največji element).

Iskanje v širino
Vse je kot pri Dijsktri, le da so vse povezave enako dolge. Uporabimo lahko kar navadno vrsto. Časovna zahtevnost je O(m).

Primov algoritem (gradi minimalno vpeto drevo)

POGOJI ZA DELOVANJE
- graf je neusmerjen
- DECREASE_KEY
- vsako vozlišče potebuje podatek ali je že v v drevesu, ali je že v prioritetni vrsti in najkrajšo razdaljo do drevesa

- graf mora biti povezan

KAJ VPLIVA NA ČASOVNO ZAHTEVNOST (O(mlogn))

- število vozlišč n
- število povezav m
- implementacija prioritetne vrste
- operacija za zmanjšanje prioritete
- čas za dodajanje novih vozlišč v drevo povezav

Če hočemo iskati maksimalno vpeto drevo lahko vzamemo recipročno vrednosti tež povezav ali pa preuredimo kopico in spremenimo neenačaje.

Kruskalov algoritem (gradi minimalni vpeti gozd)

POGOJI ZA DELOVANJE

- graf je neusmerjen
- vsaka povezava hrani podatek, ki pove če je v enem od vpetih dreves

KAJ VPLIVA NA ČASOVNO ZAHTEVNOST (O(mlogm))

- število povezav (m)
- m operacij INSERT
- m operacij DELETEMIN

Če hočemo iskati maksimalni vpeti gozd lahko vzamemo recipročno vrednosti tež povezav ali pa preuredimo kopico in spremenimo neenačaje.

Ponovno zgoščanje zgoščene tabele

KAJ VPLIVA NA ČASOVNO ZAHTEVNOST (O(n))

- število elementov, ki so že v tabeli (n)
- velikost prvotne tabele m
- izbira zgoščevalne funkcije
- izbira velikosti zgoščene tabele m(dobro je če je praštevilska)
- dolžina seznama, ki ga moramo prehoditi do elementa

ODPRTA ZGOŠČENA TABELA (O(n/m))

- povprečna dolžina vsakega seznama v tabeli je n/m
- iskanje in brisanje je reda O(n/m)
- hranimo kazalce

ZAPRTA ZGOŠČENA TABELA

- uporablja več zgoščevalnih funkcij
- zasede manj prostora kot odprta
- ni seznamov

Optimalna binarna iskalna drevesa (če se slovar ne spreminja)

ČASOVNA ZAHTEVNOST (O(n^2))

- ocenjevanje verjetnostne distribucije podatkov
- število elementov
- gradnja tabele vseh dreves
- iskanje optimalnega drevesa

Kritična pot, pogoji za delovanje:
graf je povezan, graf je usmerjen, ne sme imeti ciklov

dijkstra, pogoji:
graf je usmerjen
vse povezave so pozitivne
graf mora biti povezan
graf lahko vsebuje cikle
poti od začetnega vozlišča morejo biti znane
DECREASE_KEY

primov, pogoji:
graf je neusmerjen
DECREASE_KEY
vsako vozlišče potrebuje podatek ali je že v drevesu, ali je že v prioriteni vrsti in najkrajšo razdaljo do drevesa
graf mora bit povezan

če imajo povezave v grafu lahko dolžino enako 0:
kritična pot ok, dijkstra morejo bit pozitivne, pa baje 0 je pozitivna, primov ok

če je graf nepovezan:
kritične ne delajo, dijkstra ne dela, primov ne dela

če je prvi del grafa popolnoma poravnano drevo višine v in s konstantno stopnjo d, vsa vozlišča tega drevesa (prvega dela grafa) pa so povezana vsako z eno povezavo s skupnim dodatnim vozliščem (torej drugi del grafa vsebuje eno samo vozlišče, ki je povezano z vsemi ostalimi vozlišči):
primov ok, dijkstra ok, kritične poti - lahko se pojavijo cikli v takem grafu, če jih ni ok, če so pa ne dela

Pozdravljeni!

Reševal sem izpitno nalogo, pri kateri je nastalo precej dvomov in bi prosil, če mi lahko pregledate priloženo rešitev in poveste kje se motim.
Zanima me še ali je pri takih nalogah mišljeno, da bi algoritmi vračali nesmiselne rezultate ali da bi program crknil med delovanjem?

3. Ali bi algoritmi:
(a) Za iskanje kriti•cne poti po principu dinami•cnega programiranja
(b) Primov algoritem za iskanje minimalnega vpetega drevesa
(c) Dijkstra za iskanje drevesa najkraj•sih poti
delovali in •ce je odgovor pritrdilen, kak•sna bi bila •casovna zahtevnost algoritma, •ce
je izpolnjen eden od spodnjih pogojev (3 krat 3 je 9 argumentiranih odgovorov):
1 •ce je v grafu en cikel;
2 •ce graf vsebuje 5000 vozli•s•c in 4500 povezav;
3 imajo lahko povezave v grafu dol•zino enako 0.

m – število povezav
n – število vozlišč

a) dinamično programiranje:
1. odvisno od cikla, pri enih deluje, pri enih ne.
2. če začetno in končno vozlišče nista povezani, algoritem ne bo deloval pravilno, rezultat bo brezpomenski, ne bo pa prišlo do napake pri izvajanju programa.
3. deluje – O(n+m) = O(m)

b) Primov algoritem:

1. bi deloval, cas zahtevnost: O((n+m)log n) = O(m log n)- to je obična časovna zahtevnost
2. ne bi deloval, algoritem bi se ustavil delovati (brez napake) takoj, ko bi preiskal vse povezave povezanega podgrafa, ki ni povezan z ostalimi podgrafi. O(m)- v najslabšem primeru zaman preišče vse povezave
3. normalno bi deloval, enaka časovna zahtevnost kot v točki 1.

c) Dijkstra:
1. deluje- O((n+m)*log n)
2. enako kot pri Primovem algoritmu točka 2.
3. normalno deluje

Hvala!

Šikonja je naredu samo ta popravk.

a)1. ne deluje, pri vozliščih v ciklu vhodna stopnja nikoli ne pade na 0; ciklične odvisnosti obenem pomenijo, da je problem napačno zastavljen

Ostali odgovori so pravilni.