

Mélységi és széltében bejárás gráfokban

Mélységi keresés

Adott egy gráf.

Algoritmus: kiindulunk egy pontból, majd olyan "mélyre" megyünk előre, amennyire csak tudunk. Utána visszalépetünk addig, amíg el nem érünk egy olyan pontot, amelyben van alternatíva. Ott elindulunk arra is. Ha elértük a gyökeret, és már nincs alternatíva, de még van hátra be nem járt pont, akkor onnan kiindulva megismételjük a fenti műveletet.

Az algoritmus pszeudokódja

```
MélységbenMindentBejár
for minden  $u \in V$  forrás csúcsra
do MélységbenBejár(u)
MélységbenBejár(u)
    bejárt[u]=True, belépés[u]=most,
    most=most+1
    for minden  $v \in \text{utód}(u)$ 
do if not bejárt[v]
then előd[v]=u,
    MélységbenBejár(v)
kilépés(u) = most, most=most+1
```

Megjegyzések

- Az algoritmus előállít egy mélységi erdőt. Ezt azonban nem rétegenként, hanem a legbaloldalibb ágon azonnal leás, majd visszalép az utolsó nyitott csomópontig.
- Időelemzés: 1. A MélységbenBejár eljárás egyszer hívódik meg minden csomópontra ($\Theta(|V|)$) 2. Az utód (U) függvény egyszer hívódik meg minden élre ($\Theta(|E|)$)
- Teljes időszükséglet: ($\Theta(|V|+|E|)$)

Éltípusok a mélységi erdőben

- Fa él: a mélységi erdő élei: $u-v$ a fa éle, ha v -t először az u -ból kiindulva 1 lépésben értük el
- Visszamatató él: $u-v$ visszamatat, ha v megelőzője u -nak a mélységi fában
- Előremutató él: $u-v$ előre mutat, ha v leszármazottja u -nak a mélységi fában
- Kereszt él: az összes többi él
- Színezés/élosztályozás:
 - fehér: még be nem járt él
 - szürke: a mélységi fa gyökeréből vezető út
 - fekete: többi már bejárt csúcs

Széltében keresés

Adott egy súlyozatlan, egyszerű, irányítatlan v. irányított gráf. Adott egy s kezdőcsúcs. Minden csúcsra határozzuk meg a legrövidebb távolságát s -től.

Megoldás: a szélességi keresés.

- Egy adott kezdőpontból kiindulva bejárja a többi csúcsot
- A kezdőcsúcsból vagy az adott rétegből egy lépésben elérhető csúcsok alkotják az első/a következő réteget
- Rátalál minden elérhető csúcsra
- Minden egyes lépésben feljegyezzük az s-től mért távolságot, valamint azt, hogy mely pontból értük el (ős).
- Azoknak a pontoknak nem lesz őse, amelyek nem elérhetőek s-ből, valamint magának s-nek.
- Több ősjelölt esetén elég egyet megjegyezni. Ezt kihasználhatjuk egy legrövidebb útvonal kinyomtatására.
- Ha mindegyik csúcsra ráírjuk az elérési réteg sorszámát, akkor megkapjuk a csúcsok távolságát a kezdőponttól.
- Kiszámítja a legrövidebb (a legkevesebb élből álló) utat
- Létrehoz egy ún. szélességi fát: amelynek a gyökere a kezdőcsúcs, ágai pedig a keresés lépései
- A szélességi fát is rétegenként hozza létre

Elemzés

- Futási idő: függ az adatok ábrázolásától.
- Minden csomópontot legfeljebb egyszer vizsgálunk meg \rightarrow időszükséglete $O(V)$
- Minden ívet legfeljebb csak egyszer vizsgálunk meg (a szomszédok vizsgálatakor) \rightarrow ezek időszükséglete $O(E)$
- Teljes időszükséglet: $O(E+V)$

Szélességi keresés - Legrövidebb utak

- s-ből v-be vezető legrövidebb úthosszat jelöljük $\delta(s,v)$ -vel
- Lemma: ha létezik $u \rightarrow v$ él, akkor $\delta(s,v) \leq \delta(s,u) + 1$
- Biz: Ha u elérhető, akkor legrosszabb esetben az $u \rightarrow v$ plusz lépéssel elérhető v is.

Ha u nem elérhető, akkor $\delta(s,u) = \text{végtelen}$

- Lemma: A Szélességi Keresés algoritmus $\text{táv}[u]$ értékei minden $v \in V$ csúcsra kielégíti a $\text{táv}[v] \geq \delta(s,v)$ egyenlőtlenséget.
- Biz: Teljes indukcióval a réteg-be beállítás szerint
 1. Kiindulás: amikor s-t beállítjuk a „réteg” sorba, akkor $\text{táv}[s]=0$, tehát $\text{táv}[s] \leq 0 \leq \delta(s,s)$ teljesül.
 2. Indukciós lépés: Tfh. Egy u csúcsra $\text{táv}[u] \geq \delta(s,u)$ teljesül, és egy v csúcs az u utódjainak vizsgálata során még nem bejárt. Ekkor (program értékadás, előzőek miatt): $\text{táv}[v] = \text{táv}[u] + 1 \geq \delta(s,u) + 1 \geq \delta(s,v) \Rightarrow \text{táv}[v] \geq \delta(s,v)$
- Lemma: Tfh. „réteg” a (v_1, v_2, \dots, v_r) csúcsokat tartalmazza. Ekkor $\text{táv}[v_r] \leq \text{táv}[v_1] + 1$ és $\text{táv}[v_i] \leq \text{táv}[v_{i+1}]$ bármely $1 \leq i \leq r-1$ értékre
- Biz: Sor műveletek szerinti teljes indukcióval
 1. Kiindulás: a „réteg” sor csak az s csúcsot tartalmazza, az állítás triviálisan teljesül
 2. Indukciós lépés: egy elem kivétele: Ha a sor üres lesz, akkor triviálisan teljesül. Ha nem lesz üres, akkor is teljesül. Egy elem betétele: az új, v_{r+1} csúcs az éppen

vizsgált v_1 utódja. Ezért (algoritmus): $táv[v_{r+1}] = táv[v_1] + 1$ Viszont:
 $táv[v_r] \leq táv[v_1] + 1$ (indukciós előfeltevés miatt), vagyis $táv[v_r] \leq táv[v_{r+1}]$.

- Tétel: $táv[v] = \delta(s, v)$ minden s -ből elérhető $v \in V$ csúcsra.
- Biz: A távolság szerinti indukcióval, csak az elérhető csúcsokra. Jelölje V_k az s -től k távolságra levő csúcsokat. $V_k = \{v \in V : \delta(s, v) = k\}$ Alapeset: $V_0 = \{s\}$, $táv[s] = 0 = \delta(s, s)$, ez a legrövidebb út Indukciós lépés: Tfh., hogy adott a V_k halmaz, vagyis amelyre $\delta(s, u) = k$. Ekkor u utódjainak vizsgálata során a köv. lehetőségek vannak:- v már bejárt. ($v \in V_i$ valamely $i \leq k$ -ra). Ezt ilyenkor nem vesszük hozzá a V_{k+1} halmazhoz.- v még nem járt be. Ilyenkor a $\delta(s, v) \leq \delta(s, u) + 1$ miatt $\delta(s, v)$ legfeljebb 1-gyel nő meg, ha v még nem bejárt, akkor éppen 1-gyel.