A számítástudomány alapjai 2022. I. félév

4. gyakorlat. Összeállította: Fleiner Tamás (`fleiner@cs.bme.hu`)

Tudnivalók

Ford-algoritmus Input: $G = (V, E)$ (ir) gráf, $\ell : E \to \mathbb{R}$ konzervatív hosszfv, $r \in V$ gyökérpont. Output: $dist_\ell(r, v)$ minden $v \in V$-re. Működés: Legyen $E = \{e_1, e_2, \ldots, e_m\}$. Kezdetben legyen $f(r) = 0$ és $v \neq r$ esetén $f(v) = \infty$. Az i-dik fázis $i = 1, 2, \ldots, n-1$ esetén abból áll, hogy elvégezzük az $e_1, e_2, \ldots, e_m$ élek menti javításokat. A végén az output $dist_\ell(r, v) = f(v)$ minden v-re.

Állítás: (1) A Ford-algoritmus i-dik fázisa után $dist_\ell(r, v) = f(v)$ minden olyan v-re, ahova van legfeljebb i élű legrövidebb út v-ből. (2) A Ford-algoritmus lépésszáma legfeljebb $konst \cdot n^3$.
(3) Ahogy Dijkstra esetén, itt is legrövidebb utak fáját alkotják a végső $f(v)$ értékeket beállító élek.

Floyd-algoritmus Input: $G = (V, E)$ (ir) gráf, $\ell : E \to \mathbb{R}$ konz. Output: $dist_\ell(u, v)$ $\forall u, v \in V$. Működés: Legyen $V = \{v_1, v_2, \ldots, v_n\}$ és $d^{(k)}(i, j)$ a legrövidebb olyan v_iv_j út hossza, aminek belső pontjai csak $v_1, v_2, \ldots v_k$ lehetnek. Kezdetben $d^{(0)}(i, j) = \ell(v_i, v_j)$, ha $v_iv_j \in E$, különben $d^{(0)}(v_i, v_j) = \infty$. A k-dik fázisban

$$d^{(k)}(i, j) = \min\{d^{(k-1)}(i, j), d^{(k-1)}(i, k) + d^{(k-1)}(k, j)\} \tag{1}$$

alapján a $d^{(k)}$ függvényt határozzuk meg. Az n-dik fázis után $dist_\ell(v_i, v_j) = d^{(n)}(i, j)$ az output.

Állítás: Az (1) fennáll, tehát a Floyd-algoritmus helyes. Lépésszáma pedig legfeljebb $konst \cdot n^3$.

Def: *Mélységi bejárás* (avagy *DFS*) alatt olyan gráfbejárást értünk, amikor mindig a lehető legkésőbb elért csúcsból kerül elérésre a soron következőnek elért csúcs. Az elérési illetve befejezési sorrendből adódik minden v csúcshoz egy $m(v)$ *mélységi* ill. $b(v)$ *befejezési szám.*

Megfigyelés: (1) A mélységi bejárás lépésszáma lineáris, azaz van olyan c konstans, hogy tetszőleges n csúcsú, m élű gráf mélységi bejárásához legfeljebb $c(n + m)$ lépés szükséges.
(2) Ha uv faél vagy előreél, akkor $m(u) < m(v)$ és $b(u) > b(v)$, ha uv visszaél, akkor $m(u) > m(v)$ és $b(u) < b(v)$, ha pedig uv keresztél, akkor $m(u) > m(v)$ és $b(u) > b(v)$.
(3) Következmény: irányítatlan gráf DFS bejárása után nincs keresztél.
(4) Ha G-ben van visszaél, akkor G tartalmaz irányított kört.
(5) Ha G-ben nincs irányított kör, akkor nincs visszaél, így tetszőleges $uv \in E$ esetén $b(u) > b(v)$.

Def: A $G = (V, E)$ irányított gráf *aciklikus* avagy *DAG (directed acyclic graph)*, ha G-ben nincs irányított kör. A $v_1, v_2, \ldots, v_n$ *topologikus sorrend*, ha $V = \{v_1, v_2, \ldots, v_n\}$ és $v_iv_j \in E \Rightarrow i < j$, azaz ha G minden éle „jobbra" mutat.

Köv.: (1) Ha G DAG, akkor a DFS utáni befejezési sorend megfordítása topologikus sorrend.
(2) Tetszőleges $G = (V, E)$ irányított gráfra az alábbi három tulajdonság ekvivalens:
(a) G DAG, (b) G csúcsainak van topologikus sorrendje, (c) G DFS bejárása után nincs visszaél.

A PERT probléma: Input: a $G = (V, E)$ DAG és egy $c : E \to \mathbb{R}_+$ hosszfüggvény.
Output: Minden $v \in V$ csúcsra egy v-be vezető leghosszabb irányított út és annak a hossza.
(Az órai mese szerint a csúcsok „projekttevékenységek", a $c(uv)$ „élhossz" pedig azt mutatja, hogy u megkezdése után legalább mennyi időnek kell eltelnie ahhoz, hogy v elkezdődhessen. Ezért ha a v tevékenység legkorábbi kezdési idejét $k(v)$ jelöli, akkor $k(v) \geq k(u) + c(uv)$ teljesül minden $uv \in E$ élre. Következésképp $k(v)$ minden v csúcs esetén megegyezik a v-ben végződő leghosszabb út hosszával. (A projektmenedzsment szakirodalom máshogyan tekint a feladatra: a csúcsok a projekt mérföldkövei, az élek az egyes projekttevékenységek, $c(e)$ pedig az e tevékenység végrehajtási ideje. Ebben a terminológiában $k(v)$ az a legkorábbi időpont, amikor a v mérföldkő elérhető.))

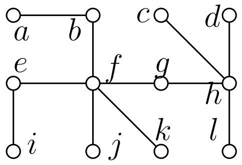
A PERT módszer: Meghatározzuk G egy $v_1, v_2, \ldots, v_n$ topologikus sorrendjét (pl DFS-sel). A $k(v_i) = \max\left(\{k(v_j) + c(v_jv_i) : v_jv_i \in E\} \cup \{0\}\right)$ formulával sora kiszámítjuk a $k(v_1), k(v_2), \ldots$ kezdési időket ill. megjelöljük mindazon v_iv_j éleket, amelyek mentén a maximum eléretik.

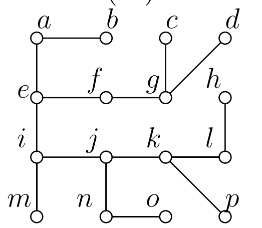
Def: A PERT *kritikus útja* a G egy leghosszabb útja, *kritikus tevékenység* pedig olyan csúcs, ami kritikus úton van.

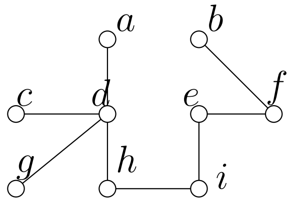
Megfigyelés: (1) Kritikus út forrásból nyelőbe vezet, és (2) több kritikus út is lehetséges.
(3) A kritikus utak minden élét megjelöltük a PERT módszer során.
(4) Egy tevékenység pontosan akkor kritikus, ha annak megkezdésében a legkisebb mértékű csúszás is a teljes projekt befejezésének késését okozza.

Gyakorlatok

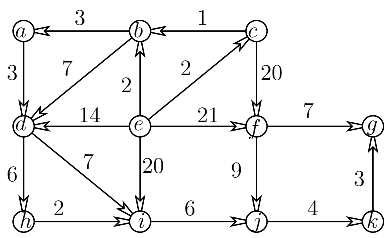
1. Bizonyítsuk be, hogy a Ford-algoritmus minden gráf és minden konzervatív élhosszfüggvény esetén futtatható úgy, hogy a második fázisban a d felső becslés már ne változzon. Bizonyítsuk be azt is, hogy ha minden legrövidebb rv-útnak legalább k éle van, akkor G éleinek van olyan sorrendje, hogy ezzel a sorrenddel a Ford-algoritmusnak legalább k fázisra van szüksége a végső d felső becslés megtalálásához. (*)
2. A bal oldali ábrán látható a G gráf egy mélységi fája. Honnan indulhatott a bejárás, ha tudjuk, hogy b és c ill. a és e szomszédosak G-ben? (✓) (ZH ’14)

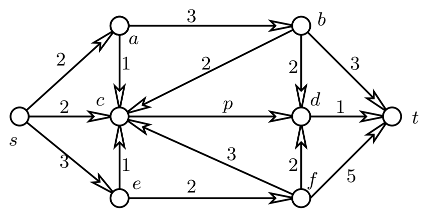


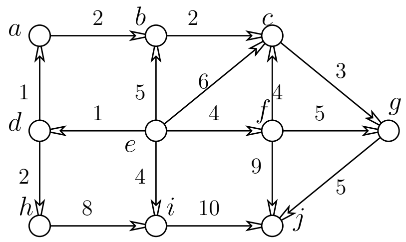




3. A középső ábrán látható a G irányítatlan gráfnak egy i gyökérű DFS fája (azaz egy i-ből indított mélységi bejárása után kapott feszítőfa). Tudjuk, hogy $d_G(e) = 7$. Határozzuk meg a G gráf e-ből induló éleit.
4. Tegyük fel, hogy a jobb oldali ábrán látható F fa a G gráfnak egyszerre h-gyökerű BFS fája és d-gyökerű DFS fája. Legfeljebb hány éle lehet G-nek?
5. Rajzoljunk egy irányított gráfot, végezzük el a mélységi bejárását. Ha a mélységi fa minden élét meg kell hagyni, akkor legalább hány élét kell törölni G-nek, hogy DAG-ot kapjunk? Mik a törlendő élek? Mi a helyzet akkor, ha nem a mélységi fából indulunk ki?
6. Igaz-e, hogy minden aciklikus, irányított G gráf csúcsainak pontosan egy topologikus sorrendje van? (pZH ’14)
7. Igaz-e, hogy ha egy n csúcsú, aciklikus, irányított G gráfban van egy $n-1$ élű irányított út, akkor G csúcsainak pontosan egy topologikus sorrendje van? (ppZH ’14)
8. Legyen G DAG, és tegyük fel, hogy az u és v csúcsok között egyik irányban sincs irányított út G-ben. Mutassuk meg, hogy G-nek van olyat topologikus sorrendje, amelyben u megelőzi v-t, és olyan is, amelyben v előzi meg u-t. (!)
9. Határozzuk meg az első két ábrán látható PERT problémában a legrövidebb végrehajtási időt és a kritikus tevékenységeket. (✓)







10. A jobb oldali ábrán látható G gráf egyes éleire írt számok azt jelentik, hogy hány kincset tudunk összegyűjteni az adott élen. Határozzuk meg, mennyi az összesen összegyűjthető kincsek száma, ha a gráf tetszőleges pontjából indulhatunk, de csak irányított élek mentén haladhatunk. (!)
11. Adjunk példát olyan PERT feladatra, ahol minden tevékenység kritikus, még sincs minden tevékenység egy kritikus úton. (✓!)
12. Adjunk olyan eljárást, amely tetszőleges PERT probléma esetén minden tevékenységhez meghatározza azt a legkésőbbi időpontot, amikor az adott tevékenységet elkezdve a teljes PERT feladat legrövidebb idő alatti végrehajtása még éppen nem kerül veszélybe.(!)
13. Adott a PERT problémát leíró G DAG és a G egy $e = uv$ éle. Tudjuk, hogy x összeg kifizetésével a $c(e)$ érték x-szel csökken. (A többi c érték adott, azokra nincs ráhatásunk.) Adjunk olyan eljárást, amelynek segítségével meghatározható az a legkisebb x érték, aminek kifizetésével a PERT feladat a lehető legrövidebb idő alatt végrehajthatóvá válik.(*)
14. Adott $G = (V, E)$ DAG, $\ell : E \to \mathbb{R}_+$ nemnegatív élhosszok és $u, v \in V$ esetén hogyan lehet a Ford ill. Floyd algoritmusoknál gyorsabb dinamikus programozáson alapuló módszerrel meghatározni egy leghosszabb uv-utat G-ben?(*)
15. Hogyan lehet a Floyd-algoritmust kiterjeszteni úgy, hogy G bármely két csúcsa között legrövidebb utat is könnyen találjunk az output segítségével?