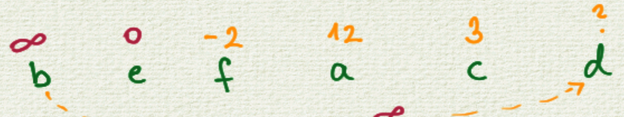
Egy irányított, élsúlyozott G gráfban a csúcsok b, e, f, a, c, d sorrendje topologikus sorrend, ezen topologikus sorrenddel alkalmazzuk a DAG-ban használható tanult eljárást (SzuperCsodás algoritmus) az e csúcsból induló legrövidebb utak meghatározására.
(a) Mennyi lesz a b és az e csúcsokra kiszámolt távolság és miért?
(b) Tegyük fel, hogy az algoritmus már kiszámolta az f, a, c csúcsokra is a távolságokat, ezek a következők: $távolság[f] = -2$, $távolság[a] = 12$, $távolság[c] = 3$. Az algoritmus használatához melyik élek létezését és élsúlyát kell tudnunk ahhoz, hogy ezek alapján kiszámoljuk $távolság[d]$ értékét? Hogyan kell az algoritmus alapján a kapott információból meghatározni $távolság[d]$ értékét?



a) b-be nem lehet éljutni, hiszen top. sorrendben nincs ←
e-be eljutni = sehova se megyek => 0 a legjobb út hossza

b) $távolság[d] = \min_{u \to d} \{ távolság[u] + c(u,d) \}$ → $távolság[d] = \min \{ távolság[c] + c(c,d),\ távolság[a] + c(a,d),\ távolság[f] + c(f,d),\ távolság[e] + c(e,d) \}$

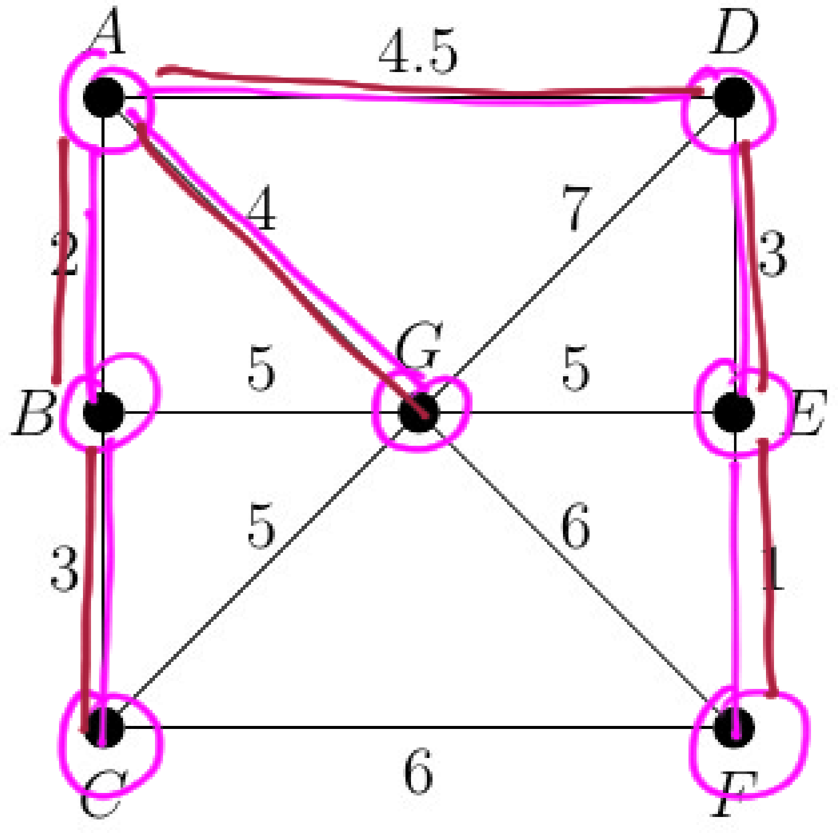
(3, 12, -2, 0)

↓ ezeket kell tudni: $c \to d$, $a \to d$, $f \to d$, $e \to d$

Minimális súlyú feszítőfát keresünk az alábbi gráfban két tanult algoritmus segítségével.

(a) Melyik éleket választja be és milyen sorrendben Prim algoritmusa az A csúcsból indulva? Indoklásképpen 1-2 mondatban írja le, hogy mi alapján választja ki az algoritmus a következő élet.

(b) Adjon meg olyan e és f éleket az a) pontban felsoroltak közül, amikre igaz, hogy Prim algoritmusa előbb választja e-t és valamikor később f-et, de Kruskal algoritmusa előbb választja f-et és csak valamikor később e-t. Indoklásképpen írja le 1-2 mondatban, hogy hogyan megy Kruskal algoritmusa és hogy ebből hogyan következik, hogy az élpár jó.



a) Prim

A már lefedettektől kimenő legkisebb élet veszi be:

AB, BC, AD, DE, EF, AG

b) Kruskal:

- Sorbarakja az éleket:

EF, AB, BC, DE, AG, AD, BG, CG, EG, CF, GF, GD

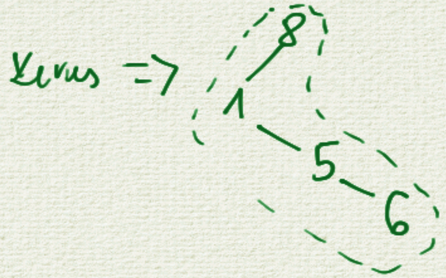
- végigmegy és ? aktuális él + eddig választottban van-e kör

nem → beveszem az aktuálisat

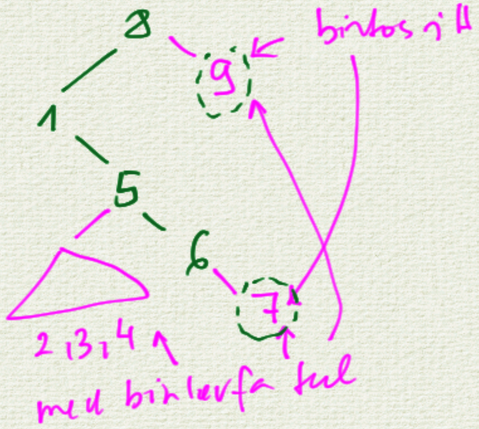
igen → nem veszem be

b) AB és EF él jó válasz

Egy bináris keresőfában az $1, 2, 3, 4, 5, 6, 7, 8, 9$ számokat tároljuk. Tudjuk, hogy a 6-os érték keresése során a $8, 1, 5, 6$ számokat látjuk ebben a sorrendben és azt is tudjuk, hogy a posztorder bejárás a fa csúcsait $3, 2, 4, 7, 6, 5, 1, 9, 8$ sorrendben látogatja meg. Rajzolja fel ezt a bináris keresőfát és indokolja meg, hogy miért csak ez az egy ilyen fa lehetséges.

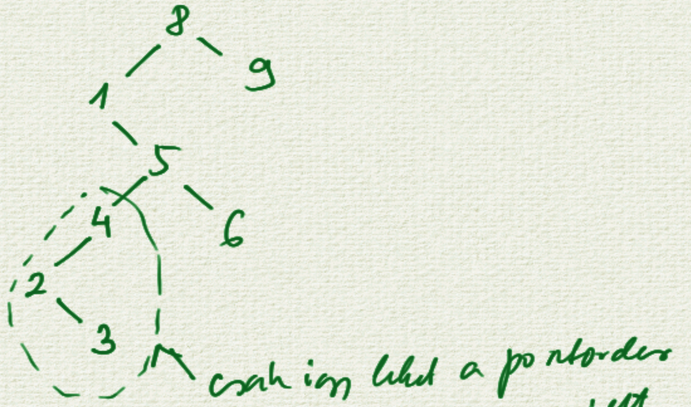


← mely ágaknál kezd, utána valamelyik ágacskába megy a részfa, kvázi >, <



? 2,3,4 helyzete ?

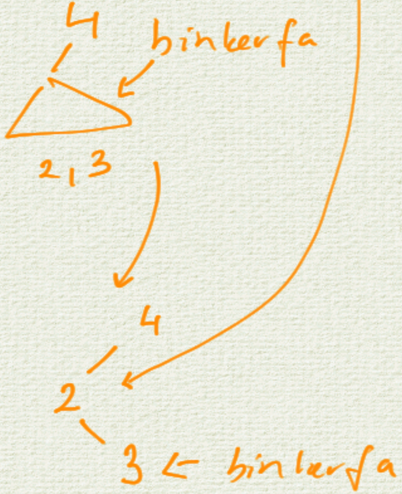
posztorder



posztorder sorrend

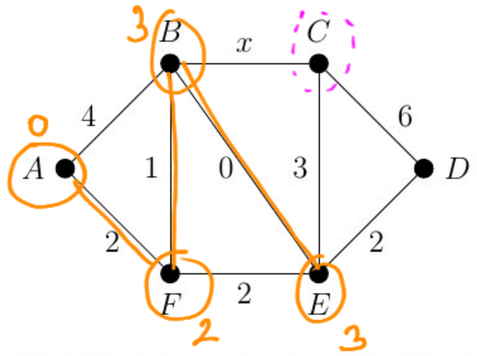
3,2,4 =>

4 ágacskái



ha = lehetséges van, akkor az algo futhat több féleképpen

Az alábbi gráfban az x élsúly nem ismert. Azt tapasztaljuk, hogy bárhogyan futtatjuk Dijkstra algorimusát az A csúcsból induló legrövidebb utak hosszának meghatározására, a C csúcs mindig utolsó előttiként kerül be a KÉSZ halmazba. Adja meg x összes lehetséges értékét és indoklásképpen írja le, hogy a válasza hogyan következik a Dijkstra aloritmus futásából.



A	B	C	D	E	F
*	4	∞	∞	∞	2
x	3	∞	∞	4	y
x	x	3+x	∞	3	*
x	x		5	*	*

Dijkstra: nem KÉSZ-ből közül a legjobbat veszi → ahol eddigi-legjobb minimális

→ min { 3+x, 5 }

Elején: KÉSZ-be $A^{(0)}_1$ utána F(2), B(3), E(3), C(3+x),

E(3) → jutottál

C(3+x) → jutottál → nem a D jön ⇒ 3+x < 5

3+x nagyobb, mint 3 ⇒ x > 0

x < 2

Adott két AVL-fa, mindkettőben ugyanannyi, $n \geq 3$ egész számot tárolunk. Az elemek egy fán belül különböznek, de a két fának lehetnek közös tárolt elemeik.
(a) Adjon $O(\log n)$ lépésszámú eljárást, ami eldönti, hogy igaz-e, hogy az első fa legkisebb eleme benne van-e a második fában.
(b) Adjon $O(\log n)$ lépésszámú eljárást, ami eldönti, hogy igaz-e, hogy az első fa második legkisebb eleme megegyezik-e a második fában tárolt legnagyobb elemmel.

AVL-fa: keresés $O(\log n)$

$\Downarrow$

$\forall$ művelet (keres, min, max, törlés): $O(\log n)$

a) 1-fa min eleme:
- min keresés az 1-fában $O(\log n)$ → balra, amíg lehet
- ezt az elemet keresem 2-fában $O(\log n)$ → van ✓ / nincs ✗

b) 2. legkisebb elem?:
- min keresés 1-fában $O(\log n)$
- törlés ennek $O(1)$ [mert nincs balgyereke] van $O(\log n)$
- új fában min keresés $O(\log n)$

$O(\log n)$

2-fa max eleme: max keresés → jobbra, amíg tudok → $O(\log n)$

. Szomszédossági mátrixával adott egy n csúcsú irányítatlan G gráf.
Adjon $O(n^2)$ lépésszámú algoritmust, ami eldönti, hogy van-e a gráfban pontosan 17 csúcsból álló komponens.

BFS / DFS

• BFS elindul valahonnan => bejárja a kezdőcsúcs komponensét

és közben lehet számolni a csúcsokat

Algo: • BFS újraindítással, amíg ∀ csúcs bejárt lesz

• közben, ∀ újraindításnál számláló = hány csúcsot járok itt be (ennél a kezdésnél / újraindításnál)

ha = 17 → van jó komponens

≠ 17 → tovább, újraindítással, ha lehet (ha elfogyott ∀ csúcs => nincs jó komponens.)

jóság: ciklus amíg $Q \neq 0$: $O(n)$ — lefut n_i-szer

LSZ: 1 indítás során egy n_i csúcsú komponens => $O(n_i \cdot n)$

össz indítás $O(n_1 \cdot n) + O(n_2 \cdot n) + \dots + O(n_k \cdot n)$ =>

$O(\overbrace{(n_1 + \dots + n_k)}^{n} \cdot n) = O(n^2)$

n_i = i. komponens hány csúcs