ADATSZERKEZETEK ÉS ALGORITMUSOK

Python nyelven

Gráf adatszerkezet II.

Legrövidebb utak keresése egy forrásból SSSPP (Single source shortest path

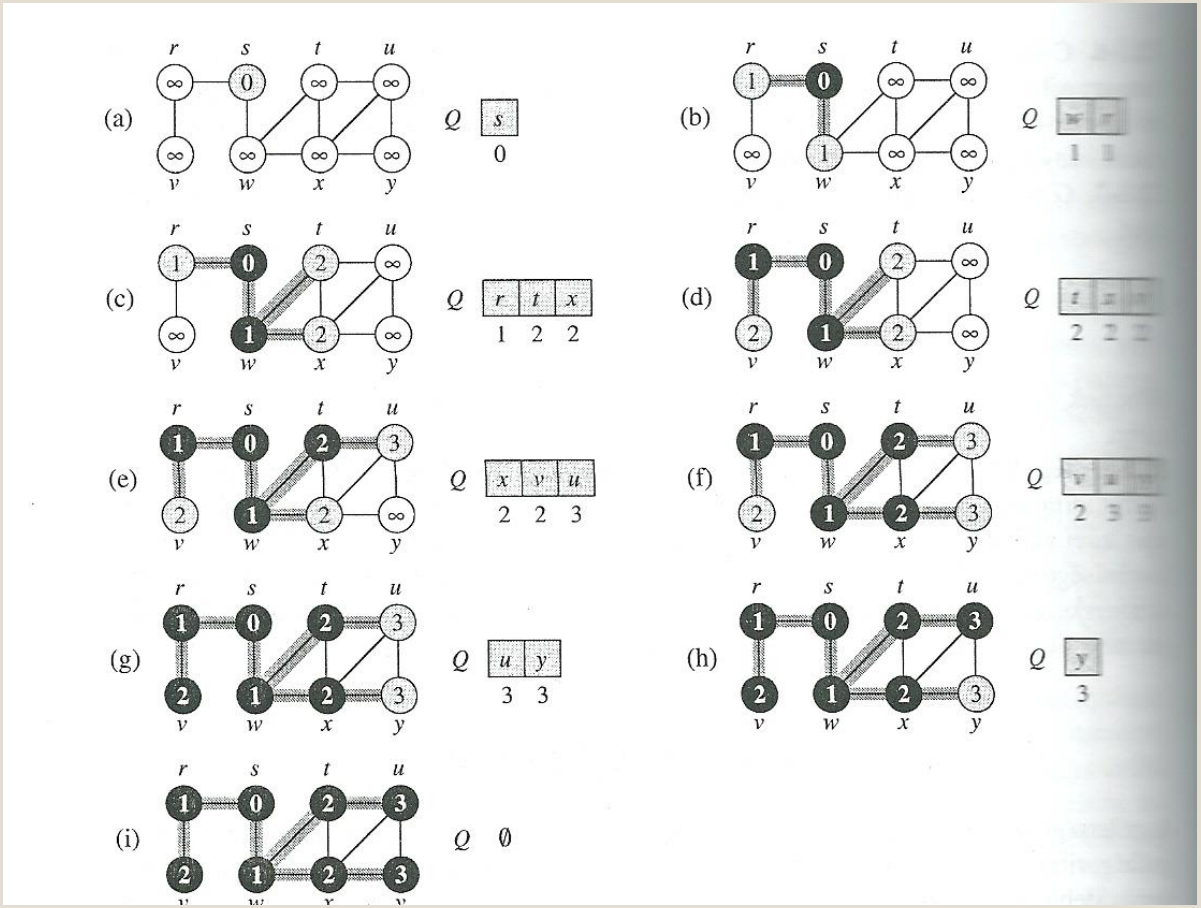
Szélességi bejárás
irányítatlan, súlyozatlan gráfnál

Feszítőfa
Irányítatlan, súlyozott gráfnál

Dijkstra
Irányított pozitív súlyú gráfnál

Bellman-Ford Algoritmus
Irányított pozitív és negatív súlyú gráfnál

Gráf szélességi bejárása



Ötlet:
A csúcsok színezése
Segédadat: sor

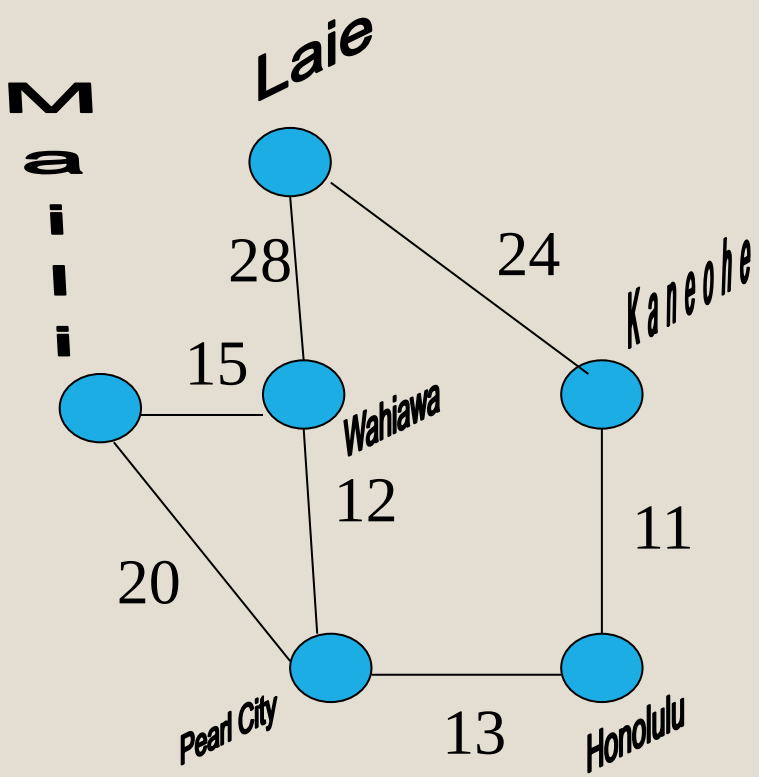
Minimális feszítőfa keresése

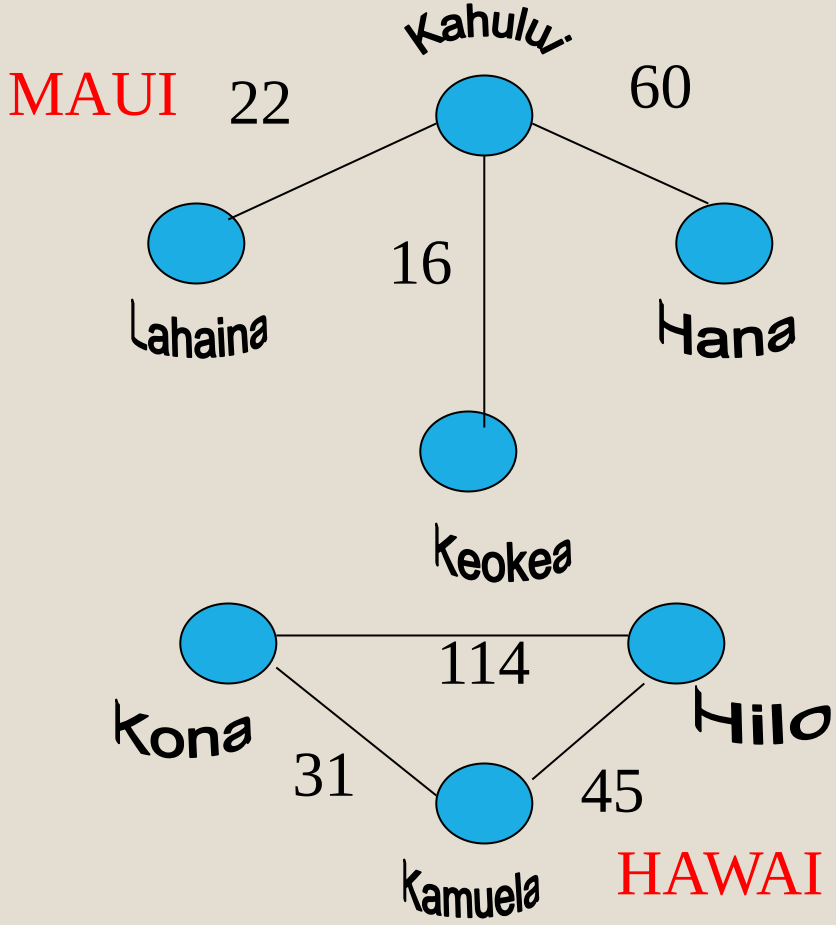
Irányítatlan súlyozott gráfban nemcsak az öf. komponenseket akarjuk megkeresni, hanem a komponens legyen fa (itt: ciklusmentes részgráf, gyökérrel, levéllel, gyerekkel nem foglalkozunk)

Feszítőfa: csúcsai azonosak a komponens csúcsaival, élei a komponens éleinek részhalmazát alkotják, ciklust nem tartalmaz

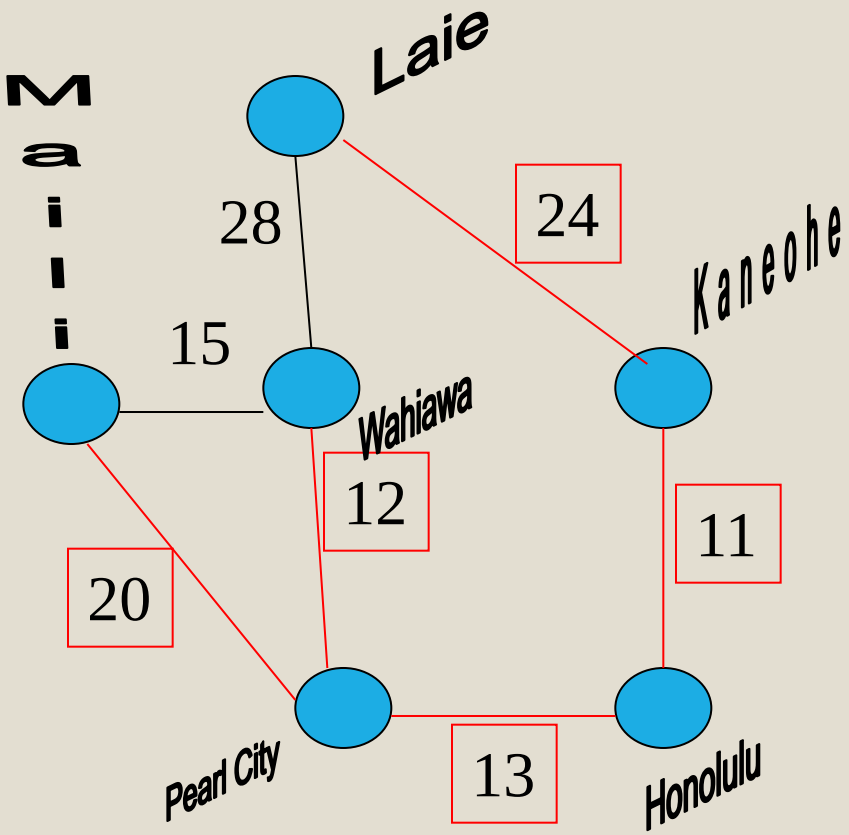
Ráadásul a feszítőfa legyen minimális abban az értelemben, hogy a súlyösszege minimális legyen (a lehetséges feszítőfák közül)

Kövessük nyomon az előbbi algoritmust a Hawai szigetek példáján! Az élek sorrendjét a súlyok nagyságrendje szerint vesszük.

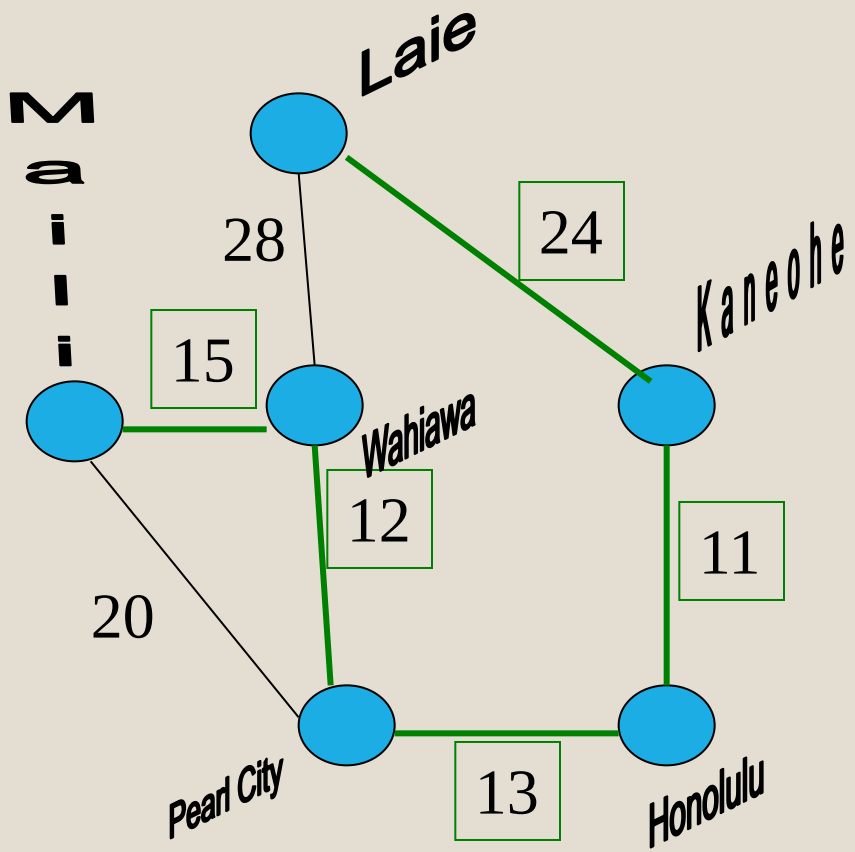




Ugyanazon gráf két különböző feszítőfája

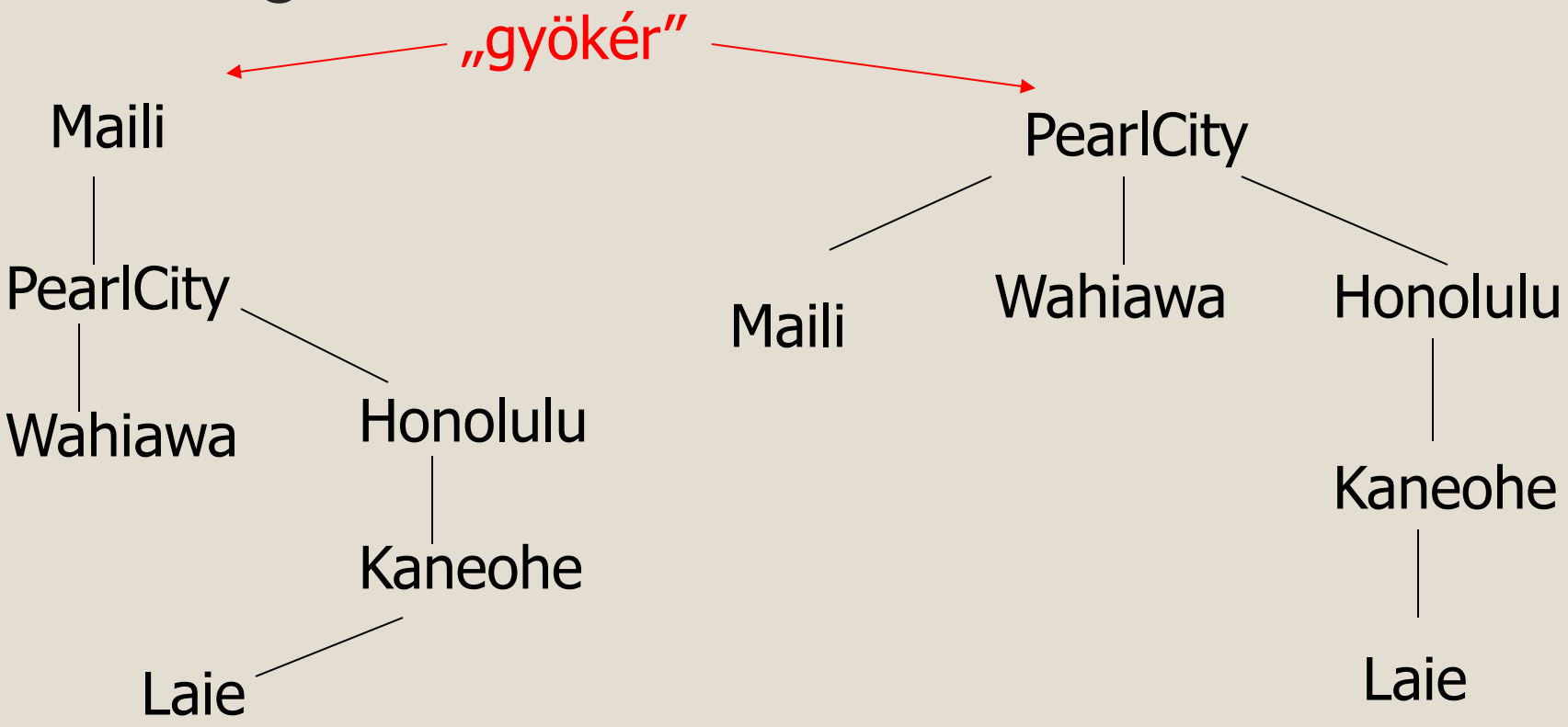


Piros feszítőfa,
összúlya 80



Zöld feszítőfa,
összsúlya 75

A fa gyökértelen, a gyerekek sorrendje tetszőleges



Ha nem kitüntetett a gyökér, a két gráf ugyanaz, mindkettő a piros fa!

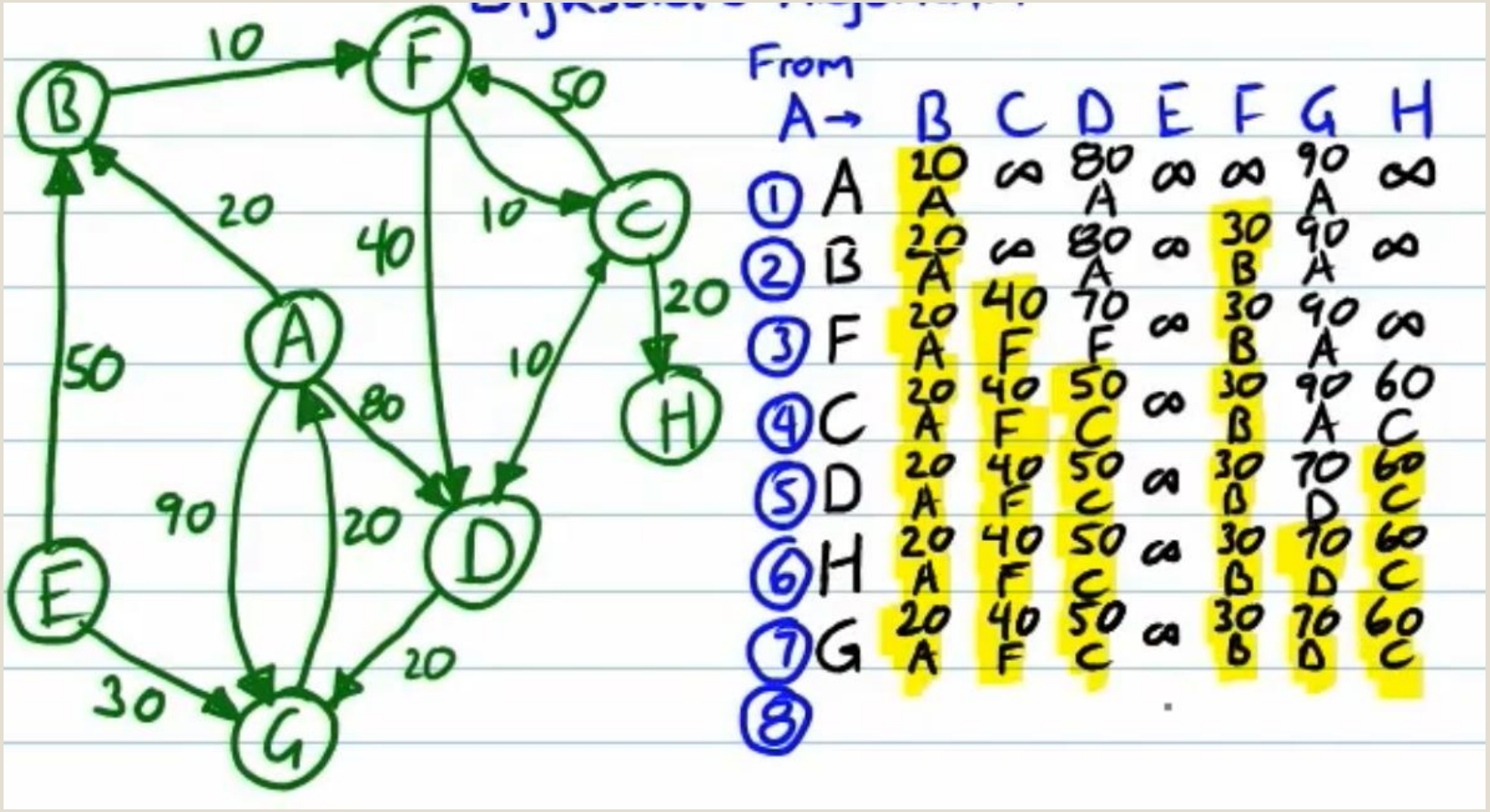
- Az éleket sorbarendezzük a súlyok szerint
- ha egy él két csúcsa különböző komponensbe tartozik, kiválasztjuk a feszítőfába és egyesítjük a két részkomponenst, egyébként nem választjuk ki az élt és nem egyesítünk
- Eredmény OAHU szigetén a zöld feszítőfa

Kruskal algoritmus a minimális feszítőfa megkeresésére

Miért működik a Kruskal algoritmus?

- Legyen G egy összefüggő, irányítatlan súlyozott gráf.
- Rendezzük az éleket súly szerint növekvő sorrendbe. Ha van két azonos súly, az egyikhez adjunk hozzá egy kis ε értéket, hogy a két él különböző súlyú legyen, de a sorrend csak közöttük változzon. Esetleg többször is alkalmazzuk e trükköt úgy, hogy minden él különböző legyen, de $\sum\varepsilon$ még mindig kisebb, mint bármely két „régi" él különbsége.
- Így a generált feszítőfa egyértelmű lesz.
- A Kruskal algoritmus mohó algoritmus (greedy algorithm) abban az értelemben, hogy minden pillanatban az akkor legjobb lépést tesszük meg. Ez nem mindig vezet globális optimumhoz, de a Kruskal algoritmus esetében igen. (Nem bizonyítjuk.)

Dijkstra algoritmusa



From

A→	B	C	D	E	F	G	H
① A	10 A	∞	80 A	∞	∞	90 A	∞
② B	20 A	∞	80 A	∞	30 B	90 A	∞
③ F	20 A	40 F	70 F	∞	30 B	90 A	∞
④ C	20 A	40 F	50 C	∞	30 B	90 A	60 C
⑤ D	20 A	40 F	50 C	∞	30 B	70 D	60 C
⑥ H	20 A	40 F	50 C	∞	30 B	70 D	60 C
⑦ G	20 A	40 F	50 C	∞	30 B	70 D	60 C
⑧							

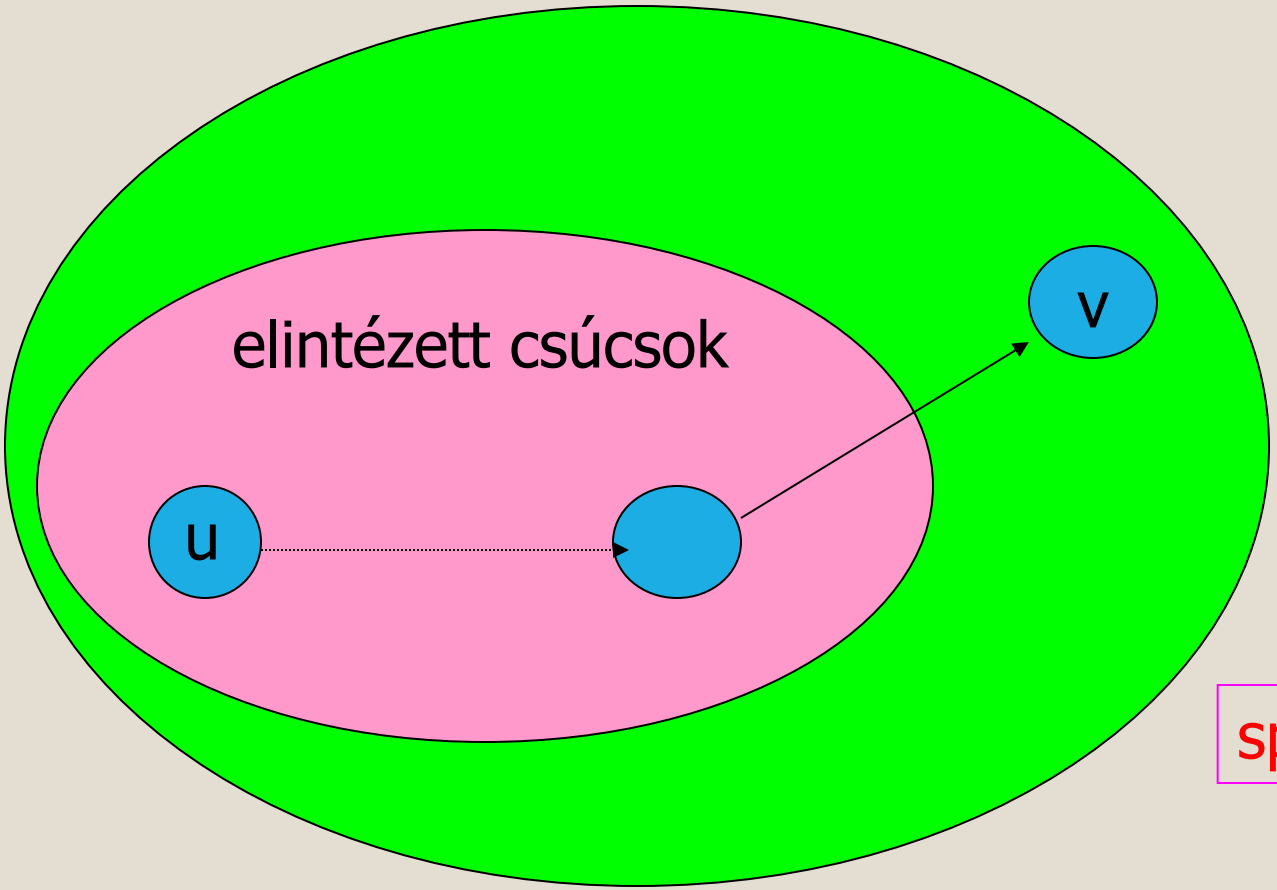
Legrövidebb út keresése 1. (Dijkstra)

- Legyen a G gráf irányított vagy irányítatlan,és az élek legyenek súllyal ellátva (hossz, idő, költség stb.) (pl. Hawai szigetek) Két csúcs között - ha van út a két csúcs között - az *út-menti távolság* az élek súlyainak összege az út mentén haladva.
- Két csúcs közötti (minimális) távolság az összes lehetséges út-menti távolság minimuma.
- Feltesszük, hogy a súlyok nem-negatívak. Ekkor minden útból kihagyhatóak a ciklusok, hiszen ezzel a távolságot csak csökkenthetjük.
- **Feladat**: egy adott **u** csúcsból indulva keressük a minimális távolságot az összes többi csúcshoz.

Legrövidebb út keresése 2. (Dijkstra)

- Kezdetben **u** *elintézett*, a többi csúcs *elintézetlen*. Legyen **S** az eddig elintézett csúcsok halmaza (kezdetben S={u}). Egy **v∉S** csúcsra definiáljuk a **speciális út** fogalmát: ez egy olyan út **u**-ból **v**-be, amely az utolsó él kivételével végig S-beli csúcsok közti éleket használ.
- Bevezetjük minden **v** csúcsra a $\mathbf{d_v}$ távolságot **u**-tól. Ha **v∈S** akkor $\mathbf{d_v}$ a legrövidebb távolság **u**-tól. Ha **v∉S** akkor $\mathbf{d_v}$ a legrövidebb speciális út hossza.
- Jelölje INFTY a "végtelent" (akkora konstans, amelyet semmilyen úthossz nem haladhat meg), $\mathbf{d_v}$ = INFTY ha v (egyelőre) nem érhető el.
- Egy lépés: **v∉S** csúcsra kiszámítjuk $\mathbf{d_v}$ -t. Ezután minden w ∈ **G-S-{v}** csúcsra beállítjuk a távolságot: ha nincs v→ w él, nem csinálunk semmit, ha van, $\mathbf{d_w} = \mathbf{d_v}$ + (v→ w él hossza). Ennél hosszabb utakat a jövőben nem keresünk.

Legrövidebb út keresése 3. (Dijkstra)



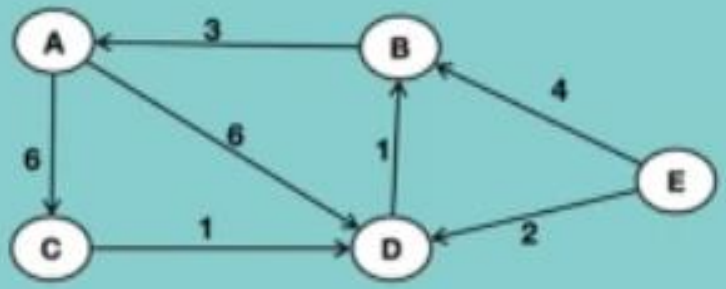
speciális út

Összefüggő komponensek keresése irányítatlan gráfban

- **Eddigi módszer**: (valamilyen súlyozással) alkalmazzuk a Kruskal algoritmust minimális feszítőfa keresésére, ez megadja az összefüggő komponenseket.
- **Új megoldás**: minden élt tekintsünk oda-vissza irányúnak, így irányított gráfot kaptunk. Keressük meg a dfs erdőt, ennek minden fája az eredeti gráf egy-egy öf. komponensét adja.

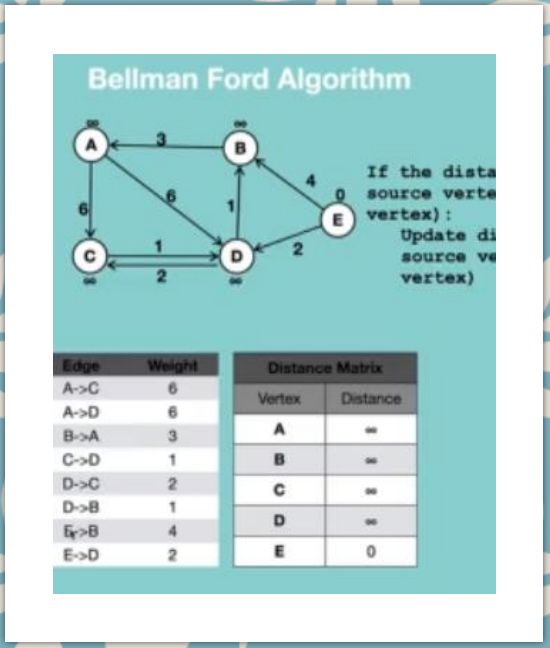
Belmann-Ford algoritmus

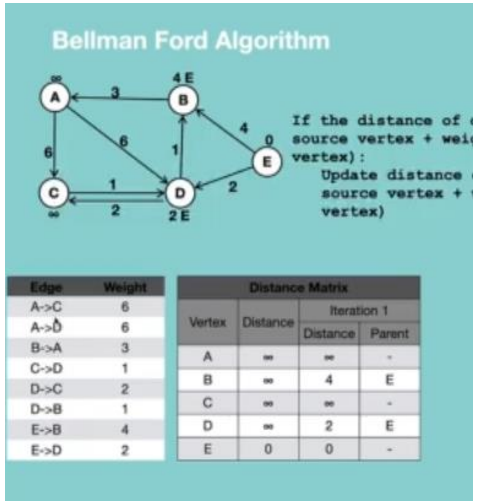
- Annyi ciklus, ahány csomópont van a gráfban
- Kiindulás



Edge	Weight
A->C	6
A->D	6
B->A	3
C->D	1
D->C	2
D->B	1
E->B	4
E->D	2

Distance Matrix	
Vertex	Distance
A	∞
B	∞
C	∞
D	∞
E	0





FOLYTATÁS

Szélességi bejárás szerint haladunk mohó algoritmussal

Ajánlott irodalom

- Cormen, Leiserson, Rivest, Stein: Új algoritmusok, Scolar Informatika, Budapest ,2003.
  ISBN 963 9193 90 9
- Mayeda, W.: Alkalmazott gráfelmélet, Műszaki Könyvkiadó, 1976, ISBN 963-10-1205-0

Köszönöm a figyelmet!