Algoritmusok és adatszerkeztek II.

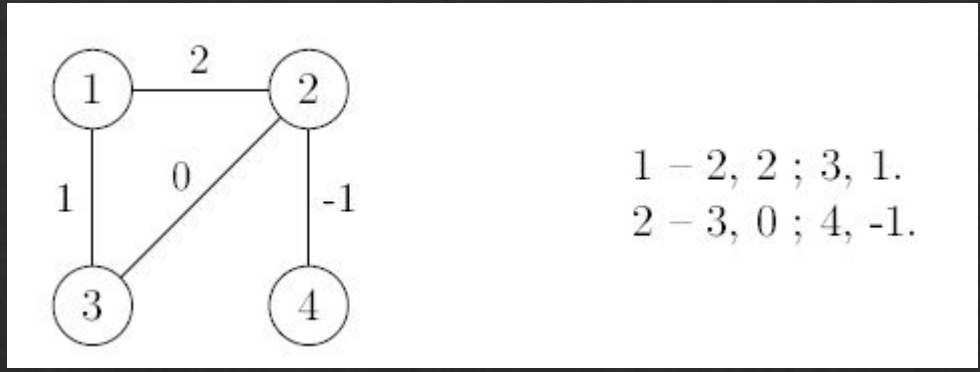
9. gyakorlat

Tartalom:

Minimális összköltségű feszítőfa II.

Élsúlyozott gráfok ábrázolása

- Irányítatlan gráf: G=(V,E), w:E→R

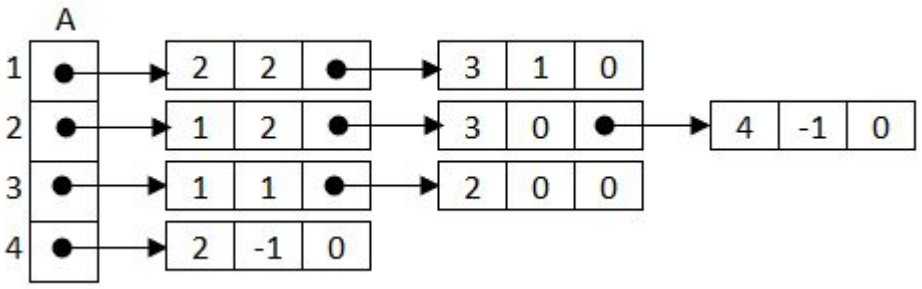


A	1	2	3	4
1	0	2	1	∞
2	2	0	0	-1
3	1	0	0	∞
4	∞	-1	∞	0

Mátrix főátlója mindig csupa nulla!

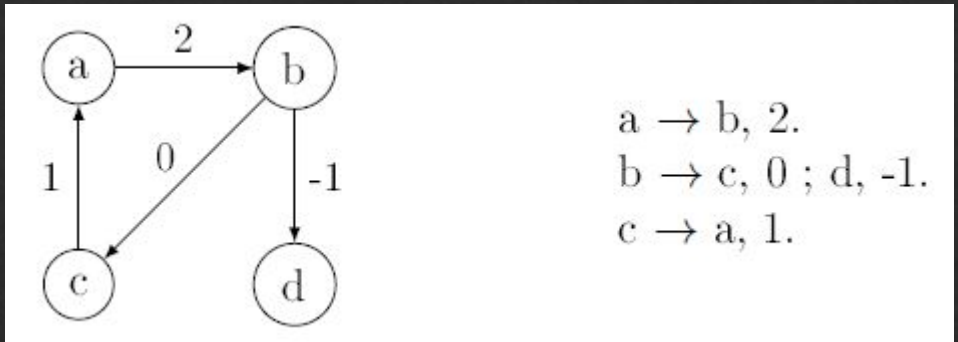
$$A[i,j] = w(v_i, v_j) \iff (v_i, v_j) \in E$$
$$A[i,i] = 0$$
$$A[i,j] = \infty \iff (v_i, v_j) \notin E \wedge i \neq j$$

Edge
$+v : \mathbb{N}$
$+w : \mathbb{R}$
$+next : Edge*$



Élsúlyozott gráfok ábrázolása

- Irányított gráf, G=(V,E), w:E→R

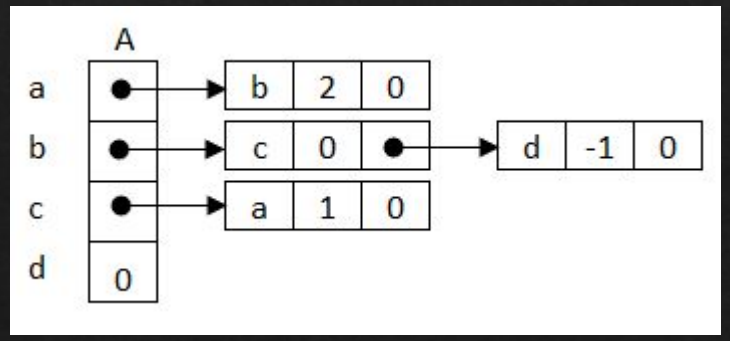


A	a	b	c	d
a	0	2	∞	∞
b	∞	0	0	-1
c	1	∞	0	∞
d	∞	∞	∞	0

Mátrix főátlója mindig csupa nulla!

$$A[i,j] = w(v_i, v_j) \iff (v_i, v_j) \in E$$
$$A[i,i] = 0$$
$$A[i,j] = \infty \iff (v_i, v_j) \notin E \wedge i \neq j$$

Edge
$+v : \mathbb{N}$
$+w : \mathbb{R}$
$+next : Edge*$



Minimális összköltségű feszítőfa meghatározása

- Adott egy élsúlyozott, összefüggő, irányítatlan G gráf. G=(V,E), w:E→R
- Feladat:
  határozzuk meg az (egyik) minimális összköltségű feszítő fáját.
- Villamos hálózat építése, O. Borúvka, 1926
- Nevezetes megoldó algoritmusok:
- J. B. **Kruskal**, 1956
- R. C. **Prim**, 1957
- Érdekesség: piros/kék algoritmus: R. E. Tarjan

Piros/kék algoritmus

- Általános leírása a feladatnak, de végrehajtható algoritmus is
- Két szabály szerint kiszínezi a gráf valamennyi élét.
- Kékre színezi a minimális költségű feszítőfába bekerülő éleket.
- Pirosra színezi azokat az éleket, amelyek már biztosan nem kerülnek be a fába.
- A két szabályt tetszőleges sorrendben és tetszőleges helyen alkalmazhatjuk, akár véletlenített módon.
- Bizonyíthatók a következő állítások:
  - Az eljárás során a színezés mindig megfelelő: azaz létezik olyan minimális feszítőfa, amelynek részfáját alkotják a kék élek.
  - A színezés nem akad el (a gráf minden éle kiszínezhető).
  - Ha kiszíneztük a gráf valamennyi élét, a kék élek egy minimális feszítőfát fognak adni. (Valójában elegendő n-1 kék élt választani.)

A szabályok

- **Kék szabály:**

  Válasszuk a csúcsoknak egy olyan nem üres X részhalmazát, amelyből nem vezet ki kék él. Az -egyik- legkisebb költségű kivezető *színtelen* élt fessük kékre.

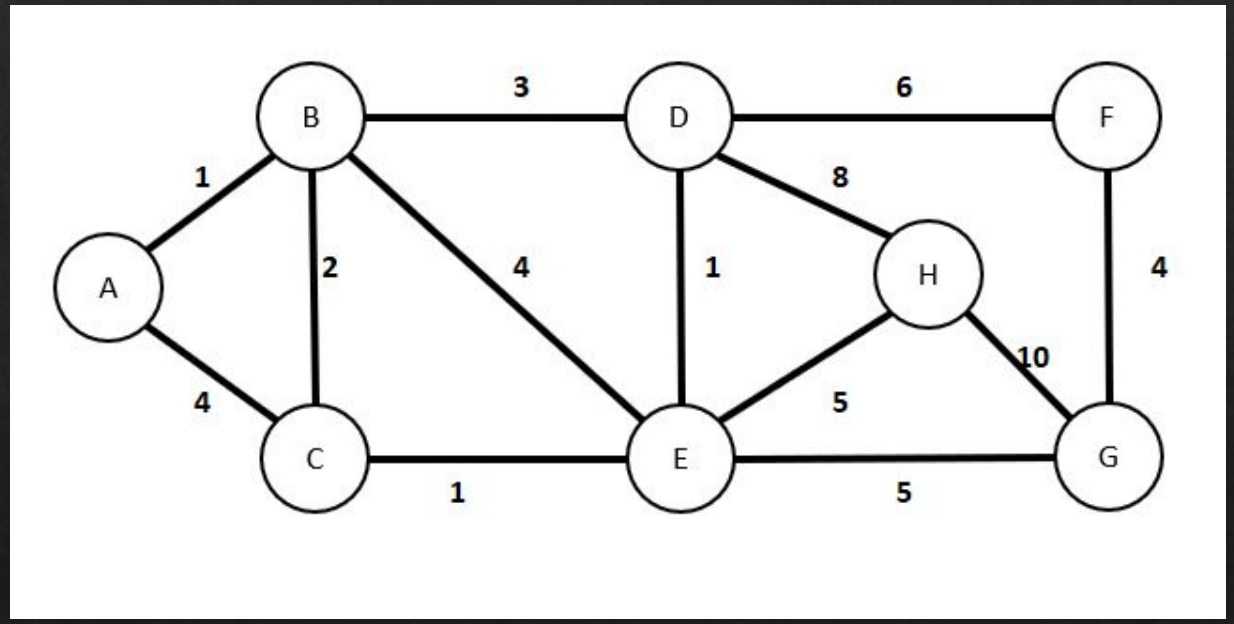
- **Piros szabály:**

  Tekintsünk egy olyan kört a gráfban, amely nem tartalmaz még piros élt. A kör -egyik-legnagyobb költségű *színtelen* élét fessük pirosra.

- A definíciók fontos eleme a „színtelen” jelző. Mindig egy szintelen élt kell beszínezni, így nem akadhat el az algoritmus.

Példa

- Próbáljuk ki a piros/kék algoritmust. A mellékelt gráf valamennyi élét színezzük ki a szabályok szerint.
- Ameddig lehet, felváltva használjuk a szabályokat, kezdjük kék szabállyal.
- Piros_kek.xlsx

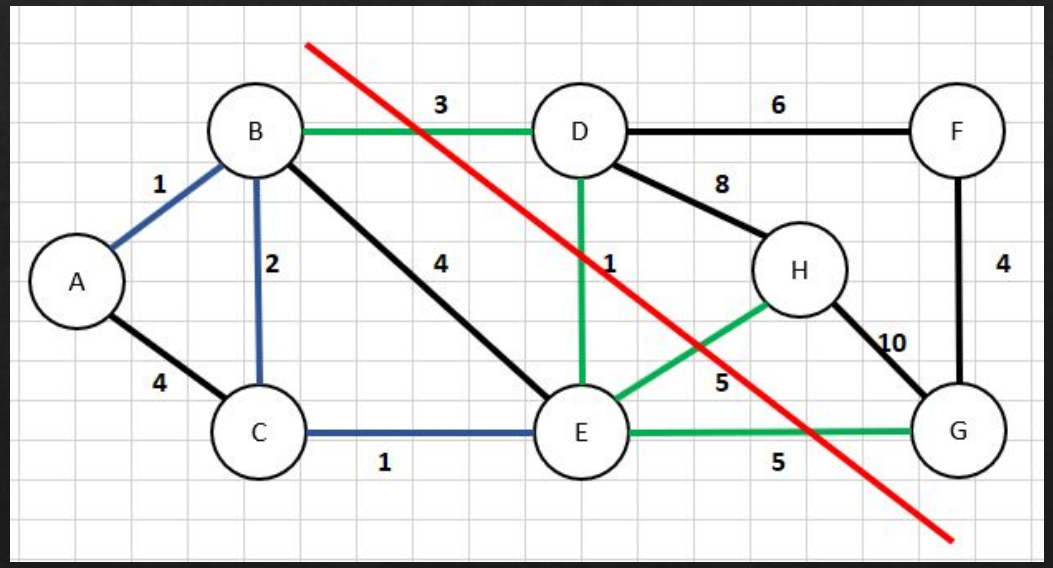


Egy általános algoritmus

2.7. Definíció. *Ha* $G = (V, E)$ *gráf és* $\{\} \subsetneq S \subsetneq V$, *akkor a* G *gráfon* $(S, V \setminus S)$ *egy* vágás.

2.8. Definíció. $G = (V, E)$ *gráfon az* $(u, v) \in E$ *él* keresztezi *az* $(S, V \setminus S)$ vágást, *ha* $(u \in S \wedge v \in V \setminus S) \vee (u \in V \setminus S \wedge v \in S)$.

2.9. Definíció. $G = (V, E)$ *élsúlyozott gráfon az* $(u, v) \in E$ könnyű él *az* $(S, V \setminus S)$ vágásban, *ha* (u, v) *keresztezi a vágást, és* $\forall (p, q)$, *a vágást keresztező élre* $w(u, v) \leq w(p, q)$.

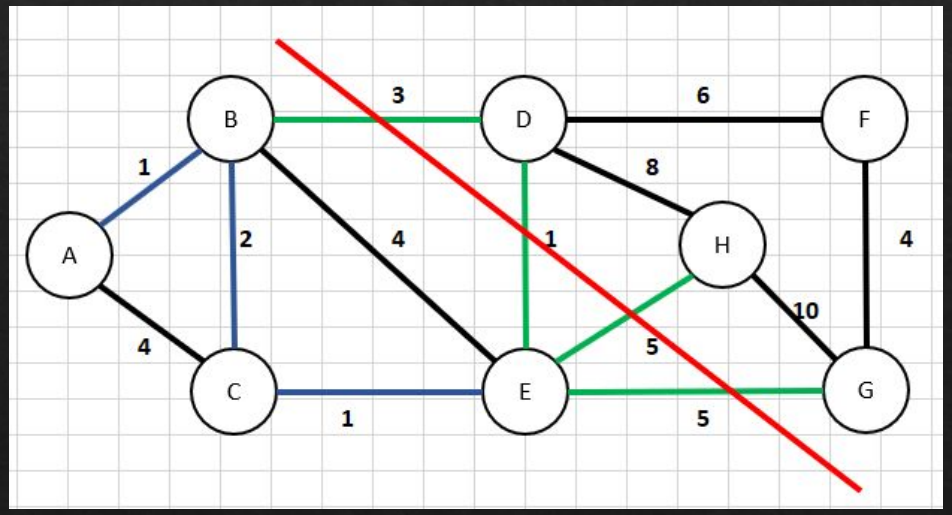


- Vágás szemléltetése: S={A,B,C,E}
- Keresztező élek: zöld színűek
- Könnyű él: (D,E)

Egy általános algoritmus

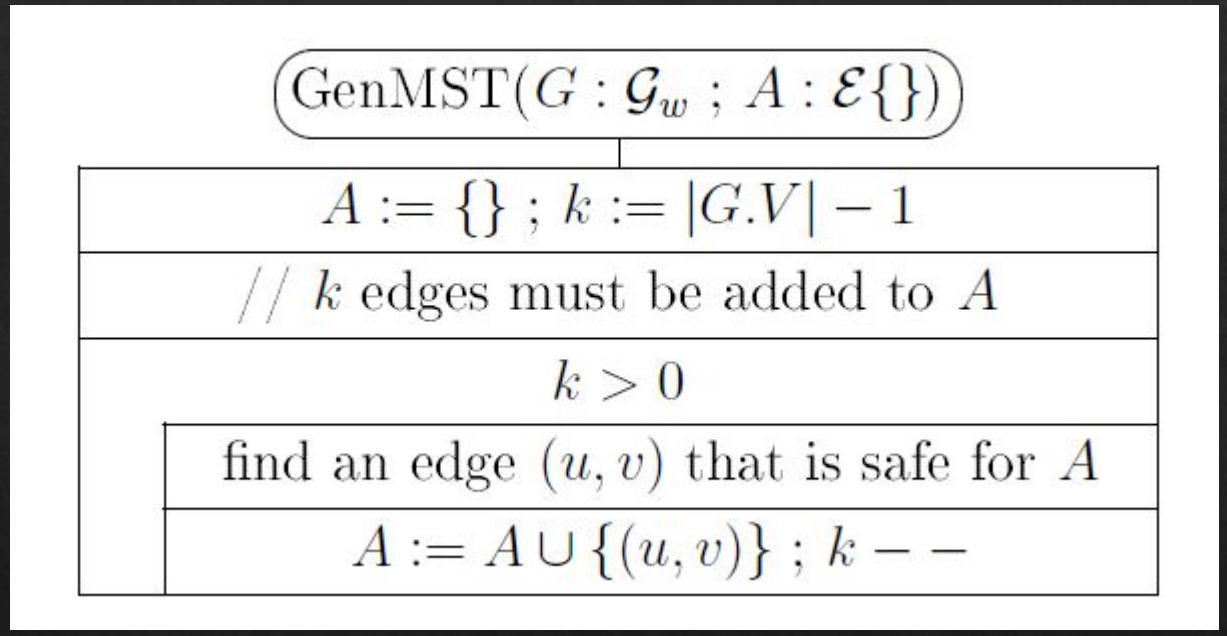
2.10. Definíció. *A $G = (V, E)$ gráfban az $A \subseteq E$ élhalmazt* elkerüli *az $(S, V \setminus S)$* vágás, *ha az A egyetlen éle sem keresztezi a vágást.*

2.11. Tétel. *Ha a $G = (V, E)$ irányítatlan, összefüggő, élsúlyozott gráfon*
(1) A részhalmaza a G valamelyik minimális feszítőfája élhalmazának,
(2) az $(S, V \setminus S)$ vágás elkerüli az A élhalmazt, és
(3) az $(u, v) \in E$ könnyű él *az $(S, V \setminus S)$* vágásban,
$\Longrightarrow$ az (u, v) él biztonságosan hozzávehető az A élhalmazhoz.



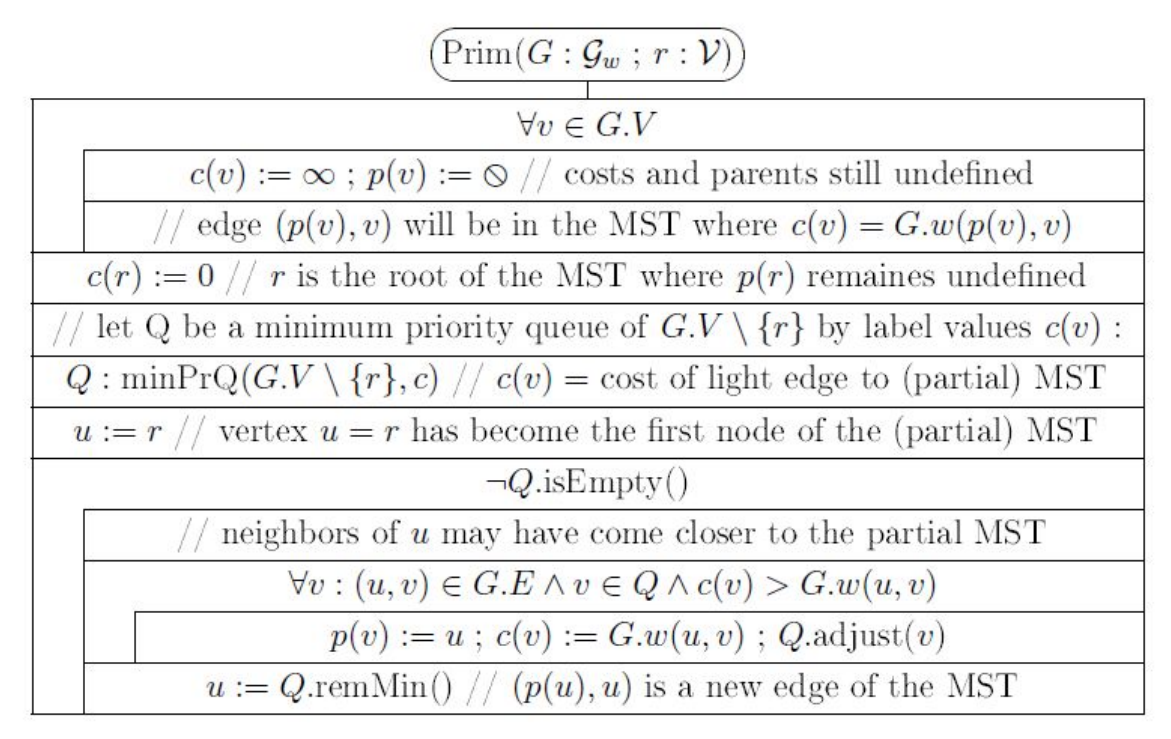
- Az *A* élhalmaz: A={(A,B),(B,C),(C,E)}
- A vágás elkerüli az A élhalmazt,
- A vágás könnyű éle: (D,E), biztonságosan hozzávehető az A halmazhoz.

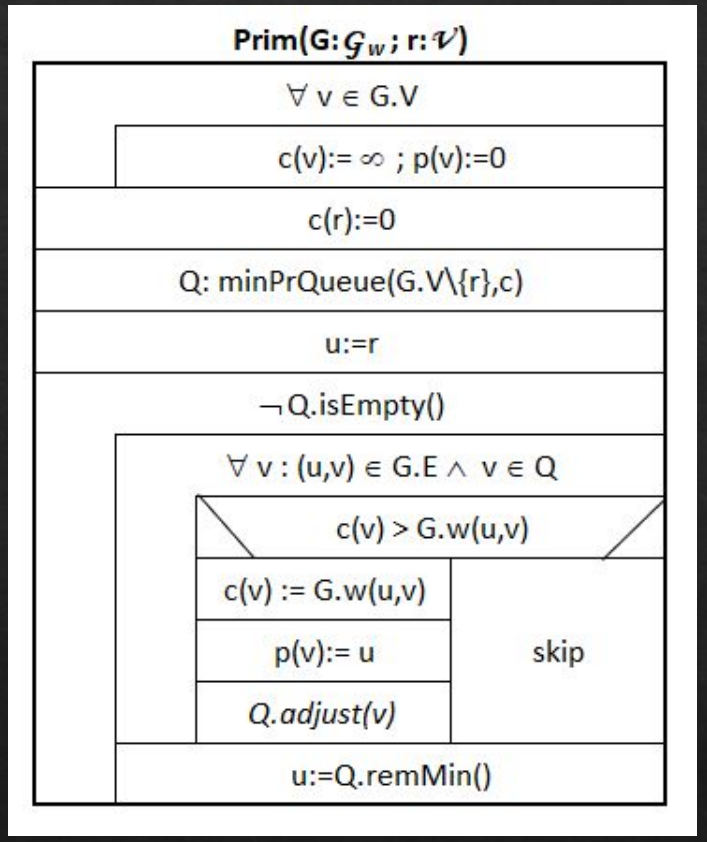
A tételből kapott algoritmus



- Vessük össze a tanult piros-kék algoritmussal:
- Amikor a kék szabályt használjuk, egy vágást hozunk létre a gráfban.
- Ha a választott halmazban vannak már kék élek, akkor azok tartoznak az A halmazba, ha nincsenek, akkor A={}
- Kék él lesz a vágás könnyű éle.
- Ezt biztonságosan hozzávehetjük A-hoz.

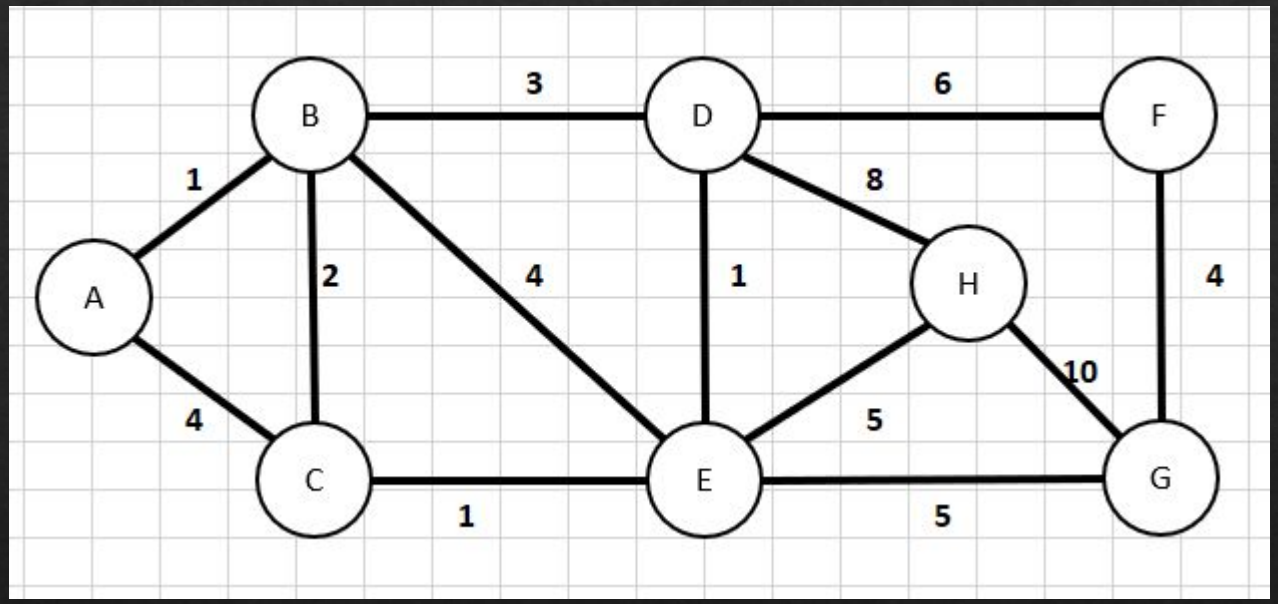
Prim algoritmus





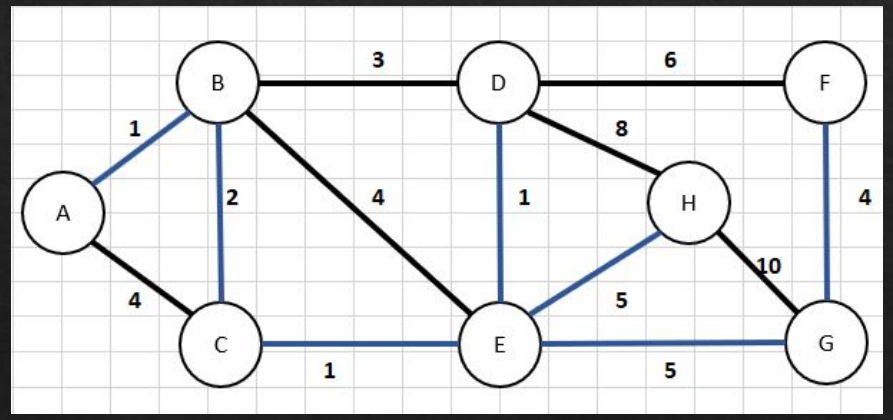
Mutassuk be az algoritmus működését az alábbi gráfon

- Kezdőcsúcs: A
- Lejátszás: Prim.xlsx



A lejátszás végeredménye

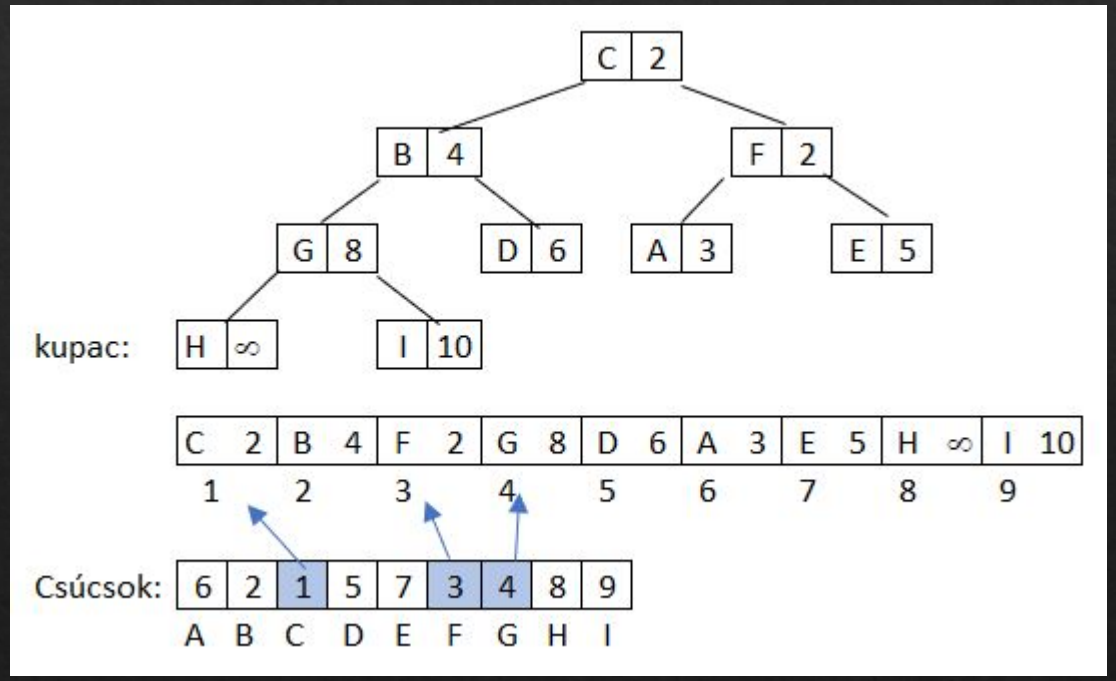
- Kék élek jelzik a kapott feszítőfát
- A csúcsok c() értékeit összeadva kapjuk meg a feszítőfa összköltségét.
- A p() értékek határozzák meg a feszítőfa éleit: (v,p(v))



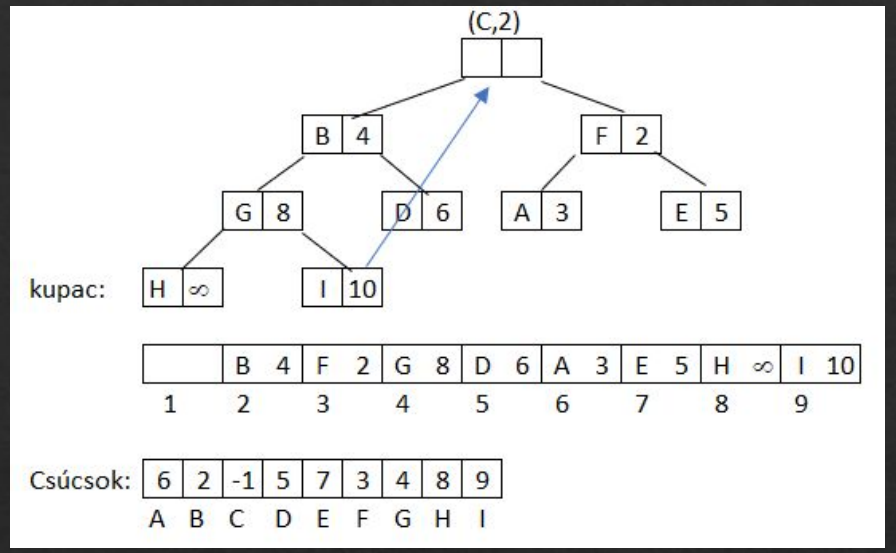
c értékek a Q-ban								Fába kerülő csúcs	p értékek változásai							
A	B	C	D	E	F	G	H		A	B	C	D	E	F	G	H
0	∞	∞	∞	∞	∞	∞	∞		0	0	0	0	0	0	0	0
	1	4	∞	∞	∞	∞	∞	A		A	A					
		2	3	4	∞	∞	∞	B			B	B	B			
			3	1	∞	∞	∞	C					C			
			1		∞	5	5	E				E			E	E
					6	5	5	D						D		
					4		5	G						G		
							5	F								
								H								
0	1	2	1	1	4	5	5		0	A	B	E	C	G	E	E

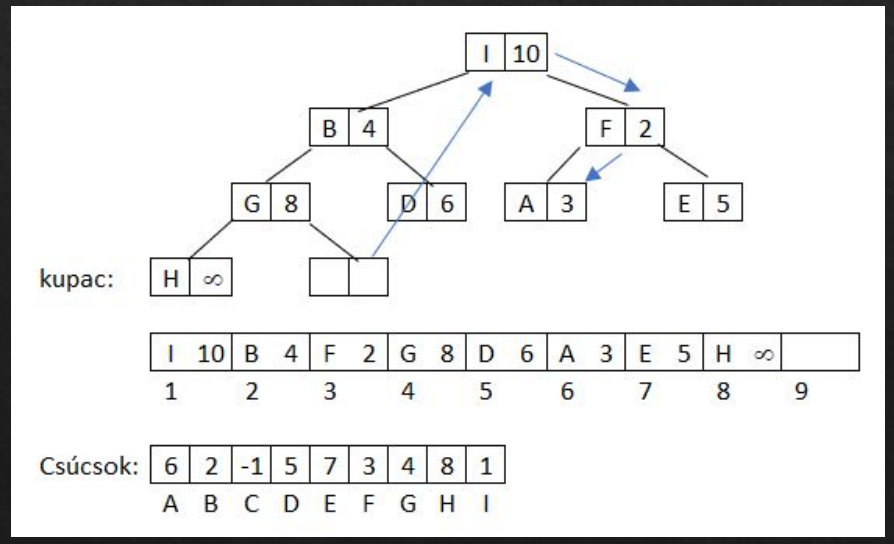
minPrQueue kupacának és a gráfnak a kapcsolata

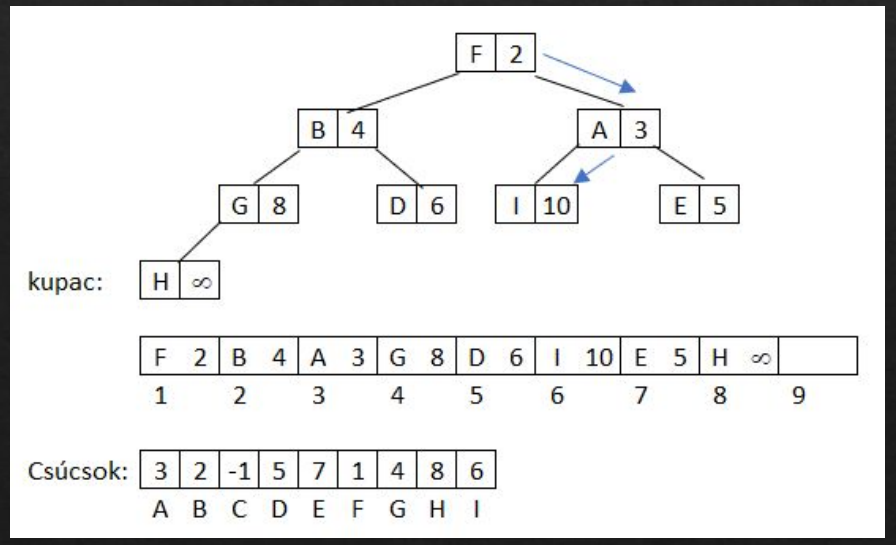
- Kétirányú kapcsolat kell:
- remMin() művelet után meg kell tudni, melyik csúcsot vettük ki: kupac elemei a c() érték mellett a csúcs azonosítóját is tartalmazzák.
- A közelítő ciklusban egy (u,v) él mentén v csúcs vizsgálatakor szükségünk van c(v) értékre. Ez a kupacból olvasható ki! Ezért a csúcsnak „tudnia” kell, hogy hol van a kupacban a helye.



remMin() művelet

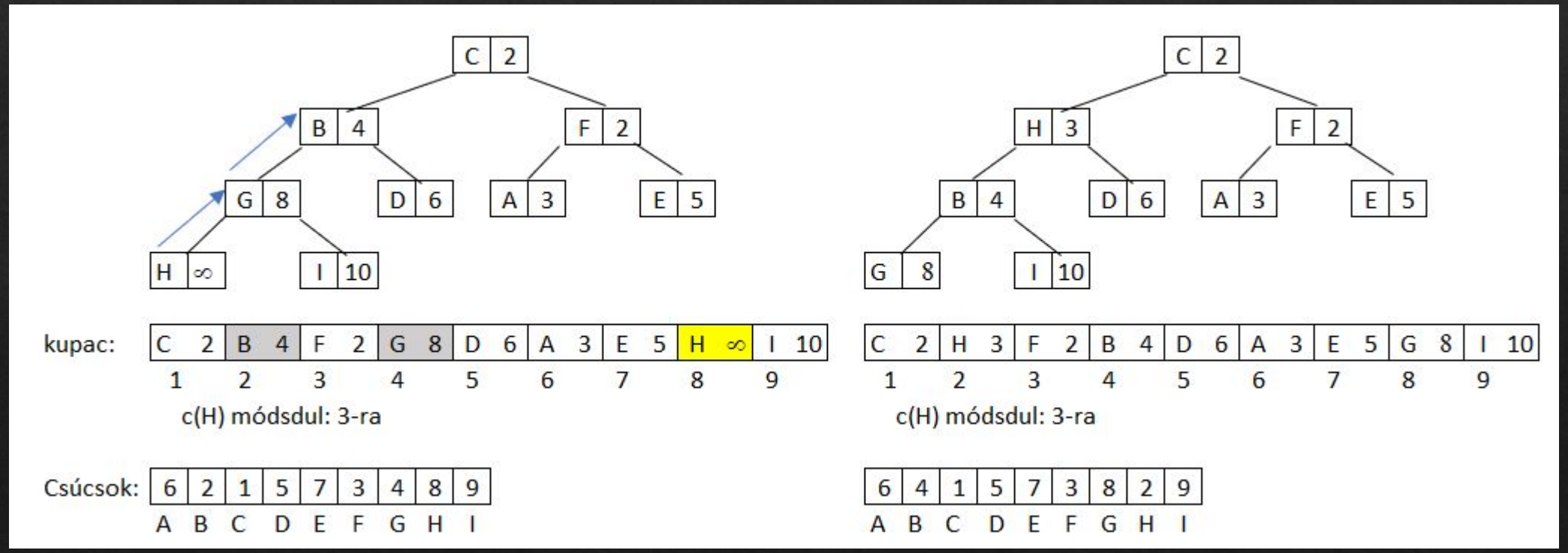






Mit takar a Q.adjust(v) művelet?

- Kupaccal ábrázolt prioritásos sor esetén:
  ha csökken egy adott kulcs -> felfelé kell a kupacban mozgatni az elemet.



Implementáció, műveletigény

Prim(G:$\mathcal{G}_w$; r:$\mathcal{V}$)

- $\forall$ v $\in$ G.V
  - c(v):= ∞ ; p(v):=0
- c(r):=0
- Q: minPrQueue(G.V\{r},c)
- u:=r
- $\neg$ Q.isEmpty()
  - $\forall$ v : (u,v) $\in$ G.E $\wedge$ v $\in$ Q
    - c(v) > G.w(u,v)
      - c(v) := G.w(u,v)
      - p(v):= u
      - *Q.adjust(v)*
      - skip
  - u:=Q.remMin()

	Ritka gráf	**Sűrű gráf**
ábrázolás:	szomszédossági lista	szomszédossági mátrix
minPrQueue:	kupac	nincs, felt,min.ker c()-n
	$\Theta(n)$	$\Theta(n)$
	$\Theta(1)$	$\Theta(1)$
	$\Theta(n)$	nincs
	$\Theta(1)$	$\Theta(1)$
"n-szer"		
"m-szer"	$O(m)$	$O(n^2)$
	$O(m)$	$O(m)$
	$O(m*\log n)$	nincs
	$O(n*\log n)$	$\Theta(n^2)$
Összesítés:	$\mathbf{O((m+n)*\log n)}$	$\mathbf{m \in \Theta(n^2)}$
	$\mathbf{m \in \Theta(n)}$	$\mathbf{\Theta(n^2)}$
	$\mathbf{O(m*\log n)}$	

Kruskal algoritmus

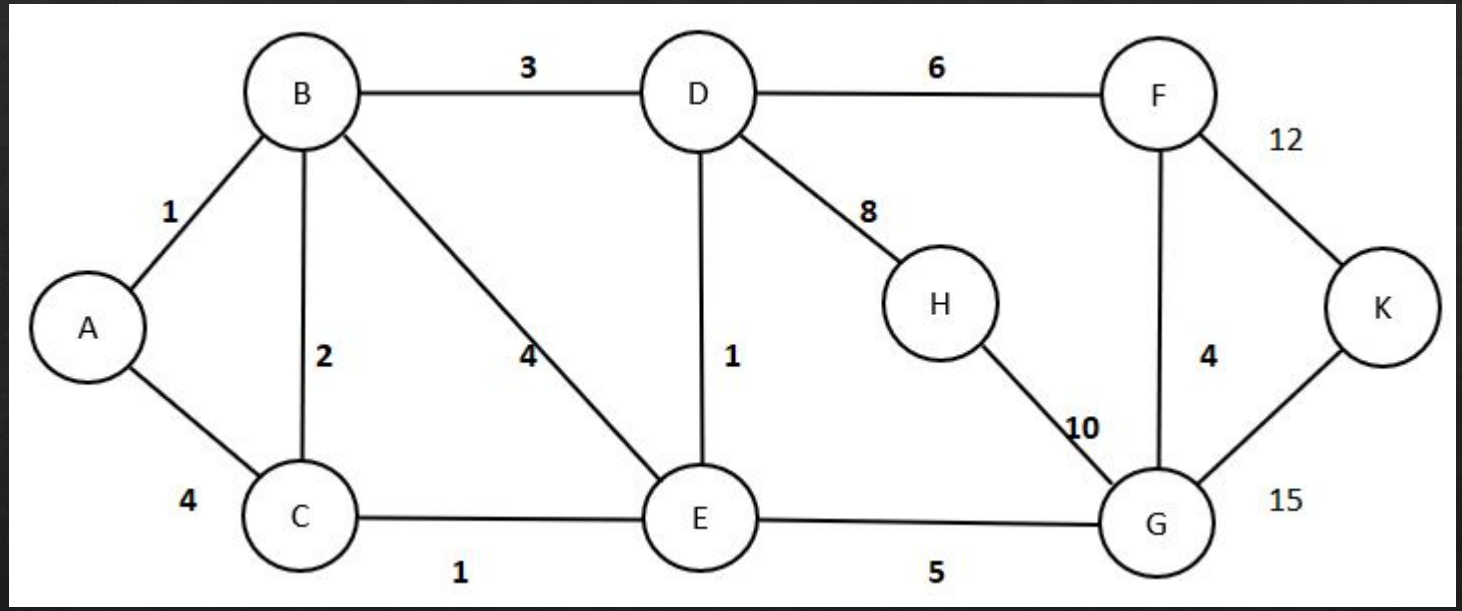
- Csúcsokból halmazokat készítünk.
- A halmazok mindig részei az –egyik lehetséges- minimális feszítő fának.
- Ha a következő él két halmazt összekötő él, akkor hozzávesszük a megoldáshoz, és összevonjuk a halmazokat. Ilyenkor két feszítőfa részletet vonunk egybe.
- Ha a következő él két végpontja ugyanabban a halmazban van, akkor kört hoz létre, nem vesszük be a megoldásba.
- Ha n-1 kék élt találtunk (|A|=n-1), kész vagyunk, az algoritmus leállhat.

Kruskal(G:$\mathcal{G}_w$): A: E{}

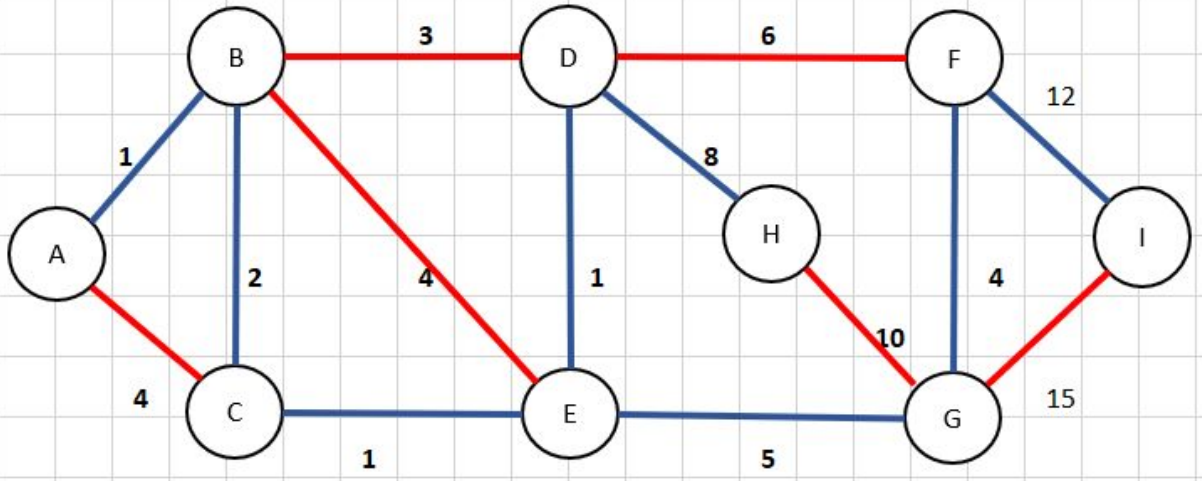
```
G.E rendezése w szerint növekvő sorrendbe
∀ v ∈ G.V
    halmazKészít(v)
A:={ }
∀ (u,v) ∈ G.E súly szerint növekvő sorrendben
    x:=holvan(u)
    y:=holvan(v)
    x ≠ y
        // (u,v) kék él
        A:=A ∪{(u,v)}
        unió(x,y)
    // (u,v) piros él
return A
```

Kruskal algoritmus lejátszása

- Mutassuk be az algoritmus működését a mellékelt gráfon.
- Lejátszás: Kruskal.xlsx



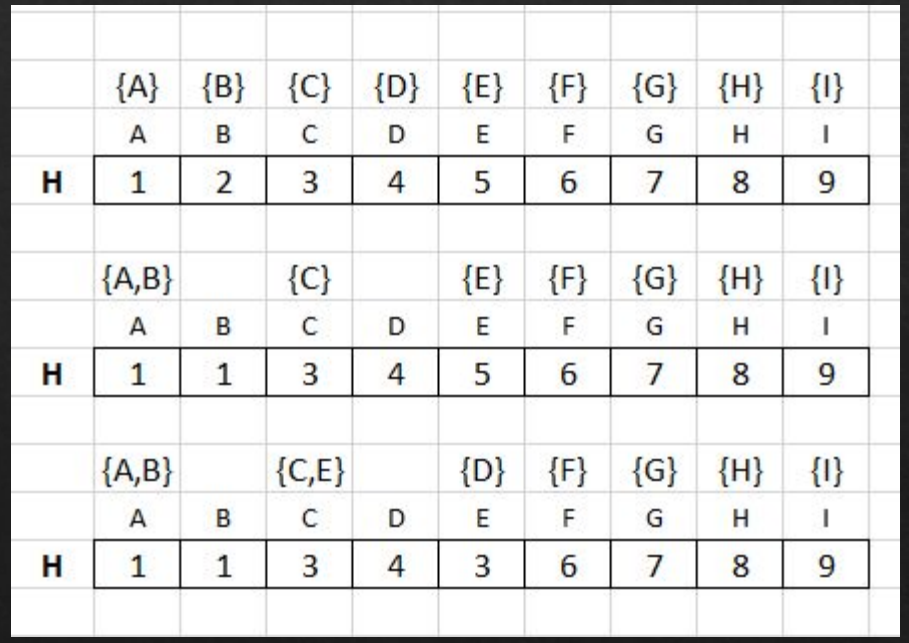
Megoldás



			{A}	{B}	{C}	{D}	{E}	{F}	{G}	{H}	{I}
(A,B)	1	k	{A,B}		{C}	{D}	{E}	{F}	{G}	{H}	{I}
(C,E)	1	k	{A,B}		{C,E}		{D}	{F}	{G}	{H}	{I}
(D,E)	1	k	{A,B}		{C,D,E}			{F}	{G}	{H}	{I}
(B,C)	2	k	{A,B,C,D,E}					{F}	{G}	{H}	{I}
(B,D)	3	p									
(A,C)	4	p									
(B,E)	4	p									
(F,G)	4	k	{A,B,C,D,E}					{F,G}		{H}	{I}
(E,G)	5	k	{A,B,C,D,E,F,G}							{H}	{I}
(D,F)	6	p									
(D,H)	8	k	{A,B,C,D,E,F,G,H}								{I}
(G,H)	10	p									
(F,I)	12	k	{A,B,C,D,E,F,G,H,I}								
(G,I)	15	p									

Hogyan ábrázoljuk a halmazokat?

- Milyen műveletnek kell hatékonynak lennie:
  - holvan(u) – melyik halmazban van az u csúcs
  - unió(x,y) – két halmaz uniójának létrehozása
- Naiv ábrázolás H egész típusú tömbbel:
  holvan: Θ(1)
  unió: O(n) – költséges!
  n-1 szer fog végrehajtódni!
- Hatékony adatszerkezet:
  „unió_holvan fa”

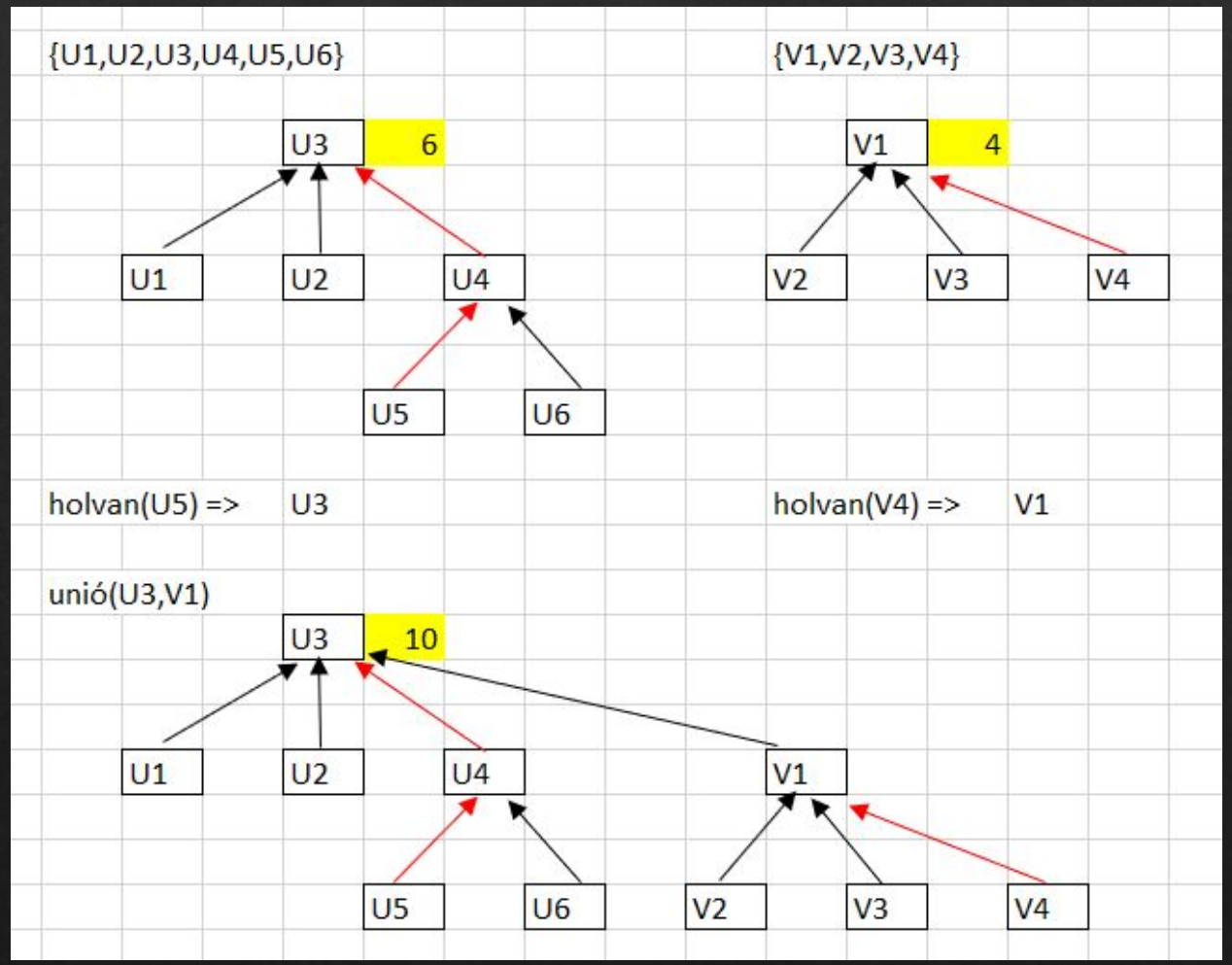


	{A}	{B}	{C}	{D}	{E}	{F}	{G}	{H}	{I}
	A	B	C	D	E	F	G	H	I
H	1	2	3	4	5	6	7	8	9

	{A,B}		{C}		{E}	{F}	{G}	{H}	{I}
	A	B	C	D	E	F	G	H	I
H	1	1	3	4	5	6	7	8	9

	{A,B}		{C,E}		{D}	{F}	{G}	{H}	{I}
	A	B	C	D	E	F	G	H	I
H	1	1	3	4	3	6	7	8	9

Az unió-holvan adatszerkezet



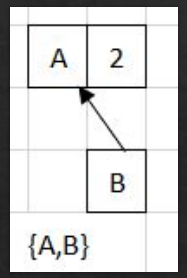
- A halmazokat fával ábrázoljuk.
- Csak a szülő irányába mutató pointerre van szükség.
- A gyökér reprezentálja a halmazt, azt is tárolja, hogy hány eleme van a halmaznak.
- Holvan művelet: fellépegetünk a fa gyökerébe: ez a fa magasságával arányos lépésszámot jelent.
- Unió: a nagyobbik fa gyökere alá befűzzük a kisebbik fát, az elemszámot módosítjuk: ez konstans lépésszámú művelet.

Rajzoljuk le unió-holvan fával a példában kapott halmazokat

{A}	{B}	{C}	{D}	{E}	{F}	{G}	{H}	{I}

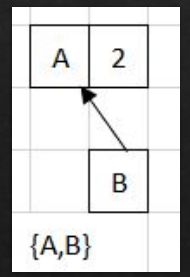
A	1	B	1	C	1	D	1	E	1	F	1	G	1	H	1	I	1

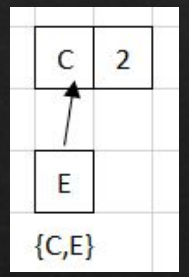
{A,B}	{C}	{D}	{E}	{F}	{G}	{H}	{I}



C	1	D	1	E	1	F	1	G	1	H	1	I	1

{A,B}	{C,E}	{D}	{F}	{G}	{H}	{I}



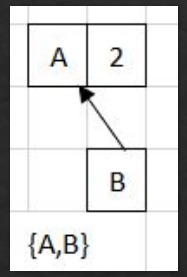


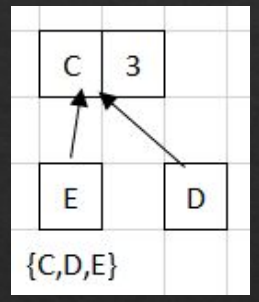
D	1

F	1	G	1	H	1	I	1

Rajzoljuk le unió-holvan fával a palában kapott halmazokat

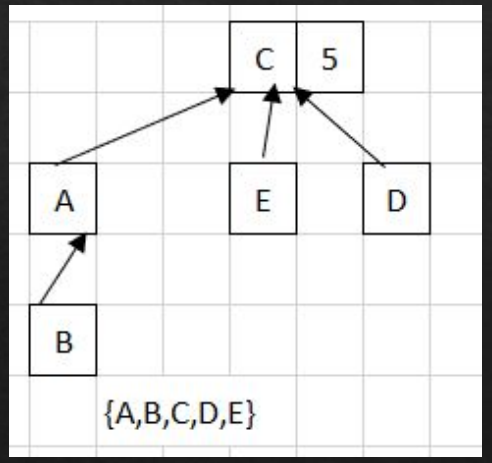
{A,B}	{C,D,E}		{F}	{G}	{H}	{I}





F	1		G	1		H	1		I	1

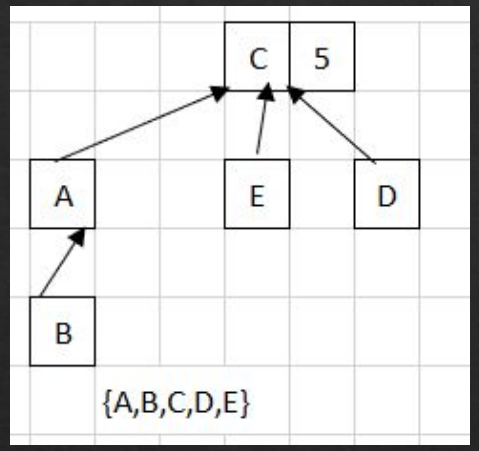
{A,B,C,D,E}			{F}	{G}	{H}	{I}

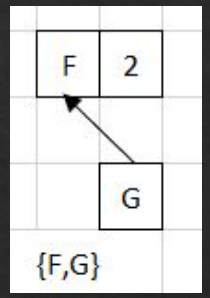


F	1		G	1		H	1		I	1

Rajzoljuk le unió-holvan fával a palában kapott halmazokat

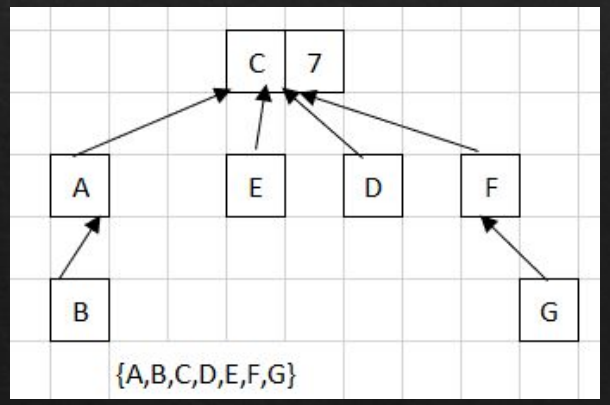
{A,B,C,D,E}	{F,G}	{H}	{I}





H	1	I	1

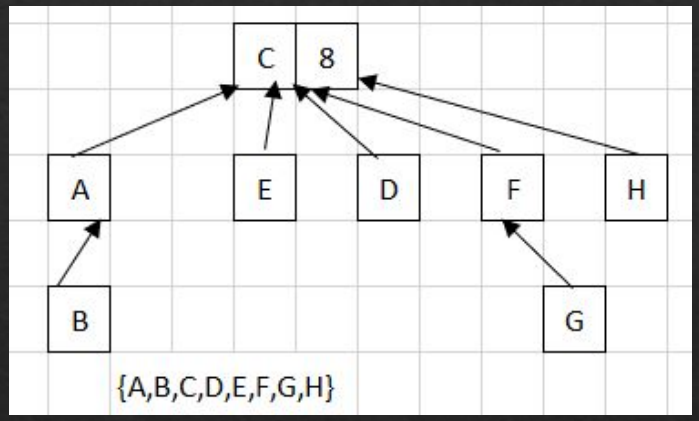
{A,B,C,D,E,F,G}	{H}	{I}



H	1	I	1

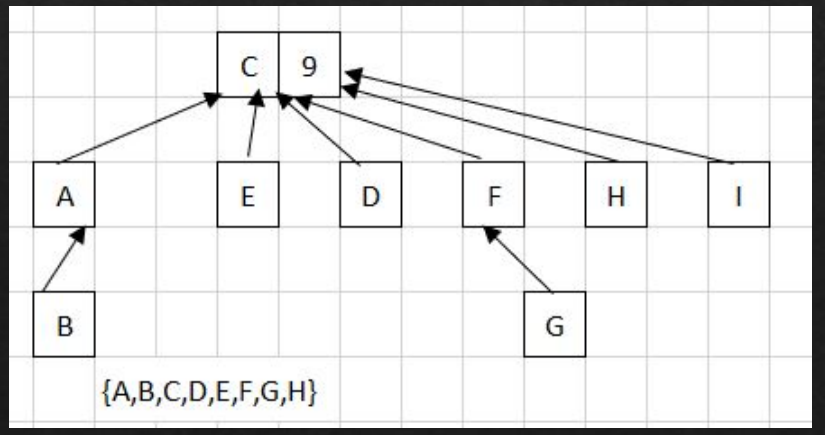
Rajzoljuk le unió-holvan fával a palában kapott halmazokat

{A,B,C,D,E,F,G,H}	{I}



I	1

{A,B,C,D,E,F,G,H,I}

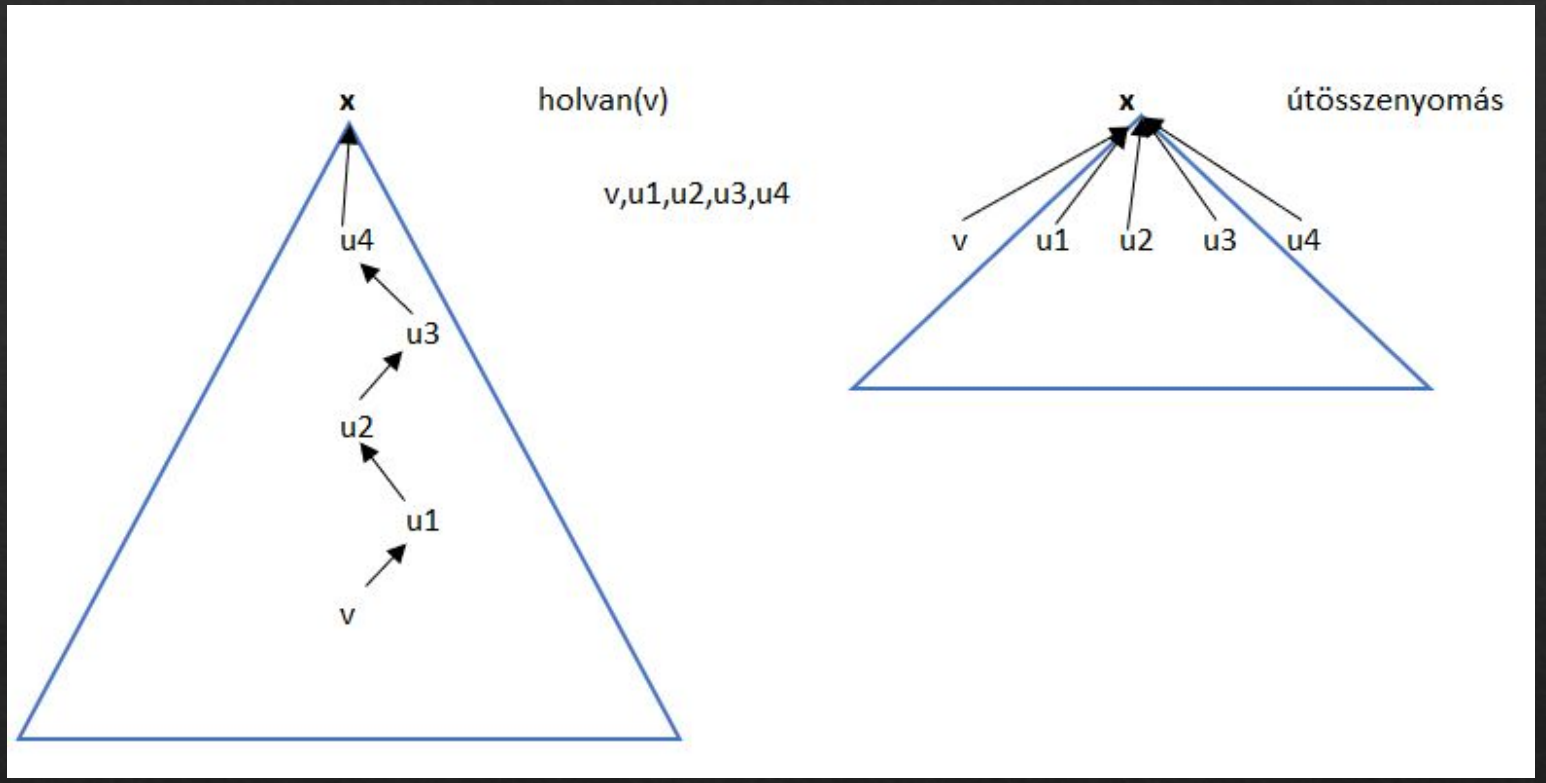


Milyen magas lehet a fa?

- Láttuk, hogy inkább terebélyes, mint magas a halmazokat ábrázoló fa.
- Minél szélesebb, annál előnyösebb, mert a holvan műveletnél gyorsabban felérünk a gyökérhez. Legideálisabb azaz alak, amikor közvetlenül a gyökér alá van bekötve valamennyi csúcs.
- Vizsgáljuk meg milyen magasságú lehet a fa.
- Unió műveletnél nő a magasság, mindig eggyel. A kisebbik fa bekötődik a nagyobb alá, így legrosszabb esetben is megduplázódik a kisebb elemszámú fa. Ez legfeljebb log2 n-szer történhet meg => a fa magassága legfeljebb log2n lehet.
- Hogyan érhető el, hogy minél szélesebb fákat kapjunk?
- A holvan művelet részeként végrehajtunk egy úgynevezett „Útösszenyomás” lépést is.

Útösszenyomás

- Holvan művelet közben az úton érintett csúcsokat megjegyezzük (például egy sorban)
- A gyökérhez felérve az úton érintett csúcsokat a sorból kivéve, bekötjük közvetlenül a gyökér alá.

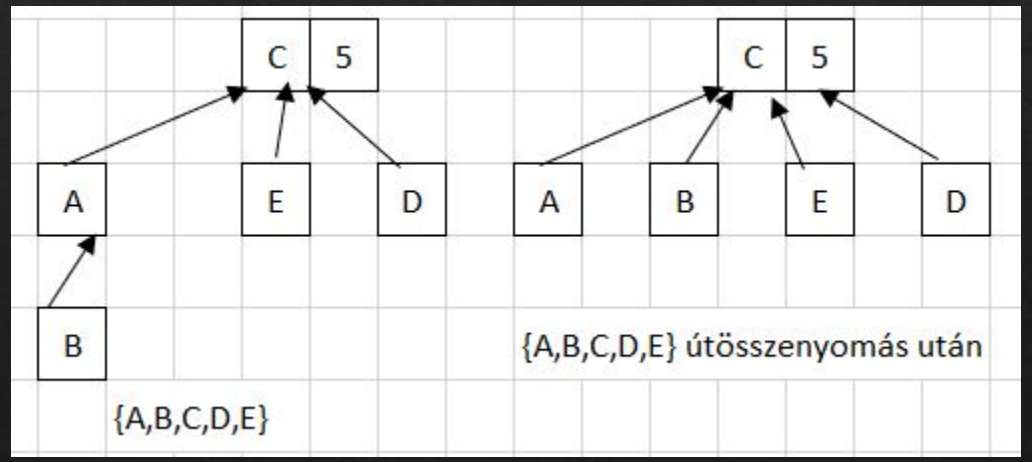


Útösszenyomást használva, igen nagy méretű gráfok esetén is a fák magassága 4 körül tartható! (Forrás: Rónyai-Ivanyos-Szabó: Algoritmusok)

Történne-e útösszenyomás a példában, ha igen hol?

			{A}	{B}	{C}	{D}	{E}	{F}	{G}	{H}	{I}
(A,B)	1	k	{A,B}		{C}	{D}	{E}	{F}	{G}	{H}	{I}
(C,E)	1	k	{A,B}		{C,E}		{D}	{F}	{G}	{H}	{I}
(D,E)	1	k	{A,B}		{C,D,E}			{F}	{G}	{H}	{I}
(B,C)	2	k	{A,B,C,D,E}					{F}	{G}	{H}	{I}
(B,D)	3	p	B-csúcsra								
(A,C)	4	p									
(B,E)	4	p									
(F,G)	4	k	{A,B,C,D,E}					{F,G}		{H}	{I}
(E,G)	5	k	{A,B,C,D,E,F,G}							{H}	{I}
(D,F)	6	p									
(D,H)	8	k	{A,B,C,D,E,F,G,H}								{I}
(G,H)	10	p	G-csúcsra								
(F,I)	12	k	{A,B,C,D,E,F,G,H,I}								
(G,I)	15	p									

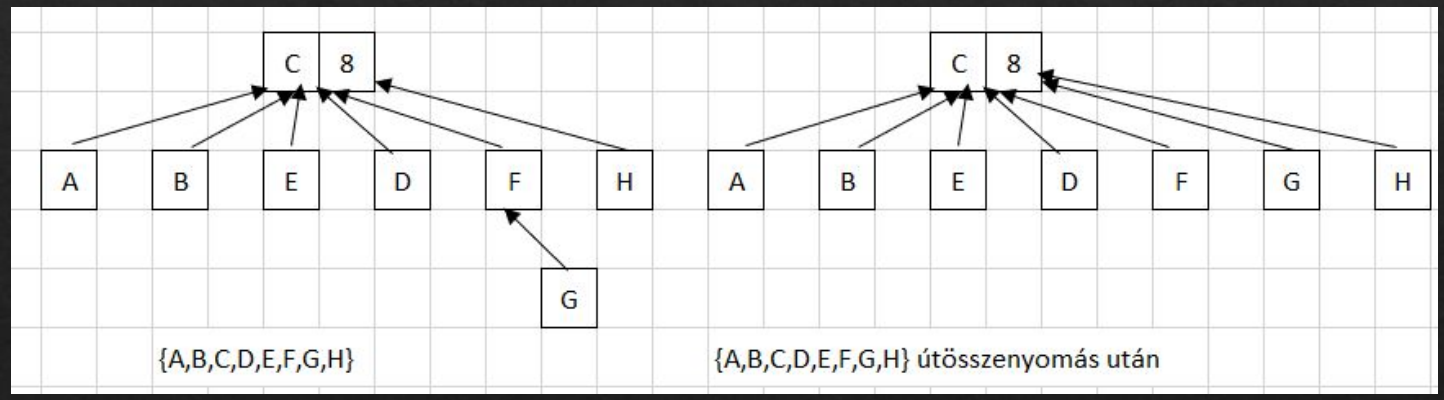
- {A,B,C,D,E} halmaznál a (B,D) él feldolgozásakor:
  holvan(B) után B közvetlenül a gyökér alá fűződik át



Történne-e útösszenyomás a példában, ha igen hol?

			{A}	{B}	{C}	{D}	{E}	{F}	{G}	{H}	{I}
(A,B)	1	k	{A,B}		{C}	{D}	{E}	{F}	{G}	{H}	{I}
(C,E)	1	k	{A,B}		{C,E}		{D}	{F}	{G}	{H}	{I}
(D,E)	1	k	{A,B}		{C,D,E}			{F}	{G}	{H}	{I}
(B,C)	2	k	{A,B,C,D,E}					{F}	{G}	{H}	{I}
(B,D)	3	p	B-csúcsra								
(A,C)	4	p									
(B,E)	4	p									
(F,G)	4	k	{A,B,C,D,E}					{F,G}		{H}	{I}
(E,G)	5	k	{A,B,C,D,E,F,G}							{H}	{I}
(D,F)	6	p									
(D,H)	8	k	{A,B,C,D,E,F,G,H}								{I}
(G,H)	10	p	G-csúcsra								
(F,I)	12	k	{A,B,C,D,E,F,G,H,I}								
(G,I)	15	p									

- {A,B,C,D,E,F,G,H} halmaznál a (G,H) él feldolgozásakor:
  holvan(G) után G közvetlenül a gyökér alá fűződik át



Kruskal műveletigény

Kruskal(G:$\mathcal{G}_w$): A: E{}

G.E rendezése w szerint növekvő sorrendbe	Rendezés: O(m*log m), de lehet lineáris (radix): Θ(m)
∀ v ∈ G.V	
halmazKészít(v)	Θ(n)
A:={ }	Θ(1)
∀ (u,v) ∈ G.E súly szerint növekvő sorrendben	legfeljebb "m-szer"
x:=holvan(u)	O(m*log n útösszenyomással: O(m)
y:=holvan(v)	
x ≠ y	
// (u,v) kék él / A:=A ∪{(u,v)} / unió(x,y)	unió: n-1 szer hajtódik végre: Θ(n)
// (u,v) piros él	
return A	Összesítve: mT(n,m)=Θ(n+m), ez ritka gráfon: Θ(n)

Szorgalmi házi feladat

- Készítsük el a Prim algoritmust szomszédossági mátrixszal ábrázolt gráfra. Felhasznált adatszerkezetek:
  - A gráfot az A/1:R[n,n] mátrix ábrázolja.
  - c/1:R[n] tömb a csúcsokhoz tartozó értékeket tárolja.
  - p/1:N[n] tömb a szülő értékeket tárolja.
  - in/1:L[n] logikai tömb
    - a prioritásos sort nem ábrázoljuk külön minPrQueue-val, hanem a c tömbben lévő értékek, és az in logikai tömb együttesen ábrázolják,
    - in[i]=true, akkor az *i* csúcs a sorban van, ha in[i]=false, akkor már nincs,
    - a remMin() műveletet egy feltételes minimum kereséssel valósítjuk meg: a sorban lévő csúcsok c értékeinek minimumát választjuk ki. Használjuk ehhez a tanult feltételes minimum keresés programozási tételét.
  - Számítsuk ki az elkészült algoritmus műveletigényét.