Algoritmusok és adatszerkezetek II

Boda Bálint

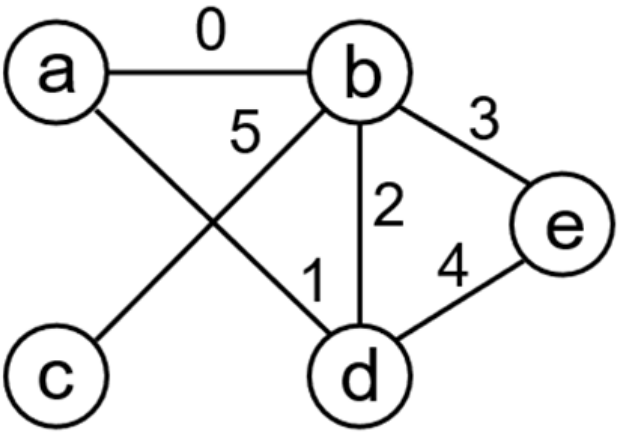
2022. őszi félév

1. Minimális feszítőfák(MST)

1.1. Kruskal algoritmus

a) A Kruskal algoritmus egy tetszőleges G gráf egy minimális feszítőerdejét állítja elő. Az algoritmus továbbá visszaad egy számot ami, ha 1 a gráf összefüggő. Nyilván ekkor a feszítőerdő egyetlen egy komponensből áll, azaz egy feszítőfa.

b)



a	b	c	d	e	él	él súlya	biztonságos
1	2	3	4	5			

c)

Tétel. Ha a $G = (V, E)$ irányítatlan, összefüggő, élsúlyozott gráfon

1. az A élhalmaz részhalmaza G valamelyik minimális feszítőfája élhalmazának
2. az $(S, V \setminus S)$ vágás elkerüli az A élhalmazt
3. $(u, v) \in E$ könnyű él az $(S, V \setminus S)$ vágásban,

akkor az (u, v) él biztonságosan hozzávehető az A élhalmazhoz.

Definíció. A $G = (V, E)$ irányítatlan összefüggő gráf feszítőfája a $T = (V, F)$ gráf, ha $F \subseteq E$ és T fa.

Definíció. A G irányítatlan összefüggő élsúlyozott gráf **minimális feszítőfája** (angolul: minimum spanning tree) T, ha T feszítőfája G-nek és G bármely T' feszítőfája esetén:

$$w(T) \leq W(T')$$

Definíció. Legyen $G = (V, E)$ egy gráf. Ha $\emptyset \neq S \subset V$, akkor az $(S, V \setminus S)$ gráfot vágásnak nevezzük.

Definíció. A $G = (V, E)$ gráf, egy (u, v) éle keresztezi a $(S, V \setminus S)$ vágást, ha

$$(u \in S \wedge v \in V \setminus S) \vee (u \in V \setminus S \wedge v \in S)$$

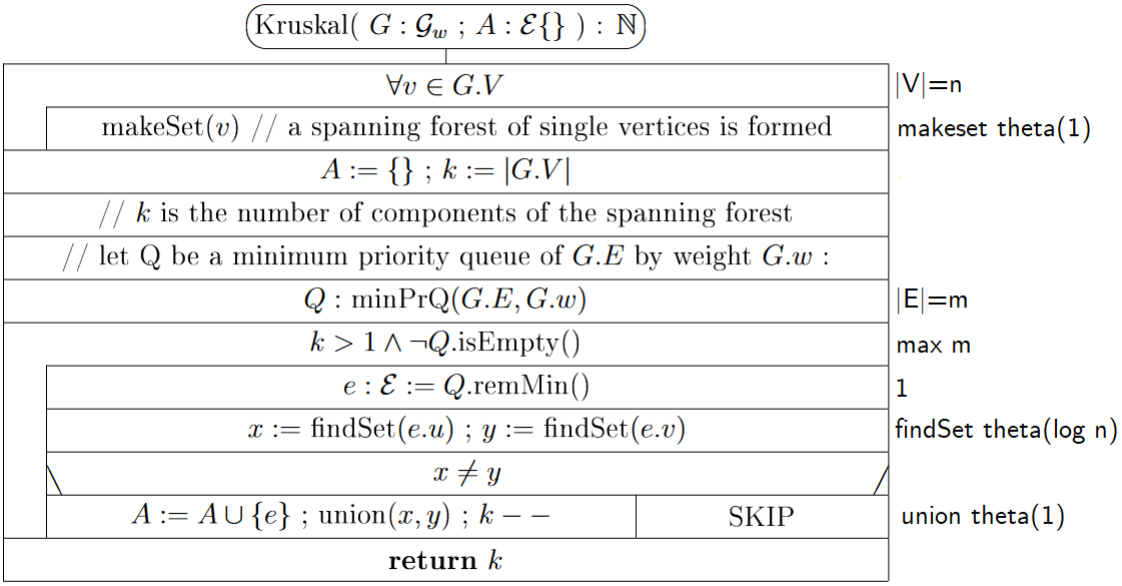
Definíció. A $G = (V, E)$ gráfban az $(S, V \setminus S)$ vágás elkerüli az $A \subseteq E$ élhalmazt, ha A egyetlen éle sem keresztezi a vágást.

Definíció. A $G = (V, E)$ élsúlyozott gráf egy $(u, v) \in E$ élét könnyű élnek nevezzük, ha keresztezi az $(S, V \setminus S)$ vágást, és költsége kisebb vagy egyenlő mint bármely más a vágást keresztező élé.

Definíció. Tegyük fel, hogy $G = (V, E)$ élsúlyozott, irányítatlan, összefüggő gráf és A részhalmaza G valamely minimális feszítőfája élhalmazának. Ekkor az $(u, v) \in E$ él biztonságosan hozzávehető az A élhalmazhoz, ha $(u, v) \notin A$ és $A \cup \{(u, v)\}$ részhalmaza G valamely minimális feszítőfája élhalmazának.

d) A Kruskal algoritmus invariánsa miatt teljesülnek a tétel feltételei, ezáltal egy adott él az algoritmus futása során, A-hoz csak biztonságos éleket veszünk.

e)

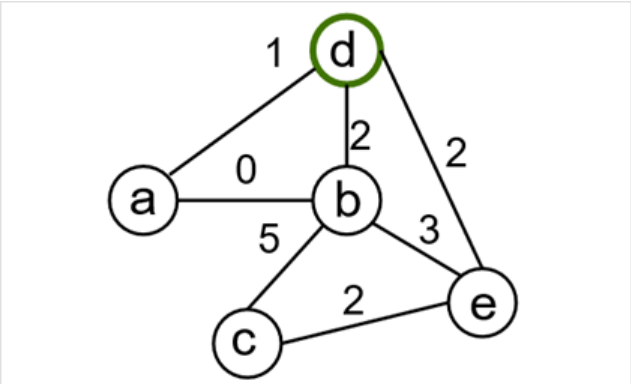


Így a műveletigény: $(n + m + m \cdot \log n) \in \Theta(m \cdot \log n)$, ha feltesszük, hogy a `makeSet` és `union` műveletek műveletigénye $\Theta(1)$ a `findSet`-é pedig $\Theta(\log n)$, illetve a minimum prioritásos sor inicializálása $\Theta(m)$, a `remMin()` metódus költsége pedig max. $\Theta(\log m)$.

1.2. Prim algoritmus

a) A Prim algoritmus egy összefüggő élsúlyozott irányítatlan gráf egy minimális feszítőfáját adja meg.

b)



	c					cúcs	p				
	A	B	C	D	E		A	B	C	D	E

c)

Tétel. Ha a $G = (V, E)$ irányítatlan, összefüggő, élsúlyozott gráfon

1. az A élhalmaz részhalmaza G valamelyik minimális feszítőfája élhalmazának
2. az $(S, V \setminus S)$ vágás elkerüli az A élhalmazt
3. $(u, v) \in E$ könnyű él az $(S, V \setminus S)$ vágásban,

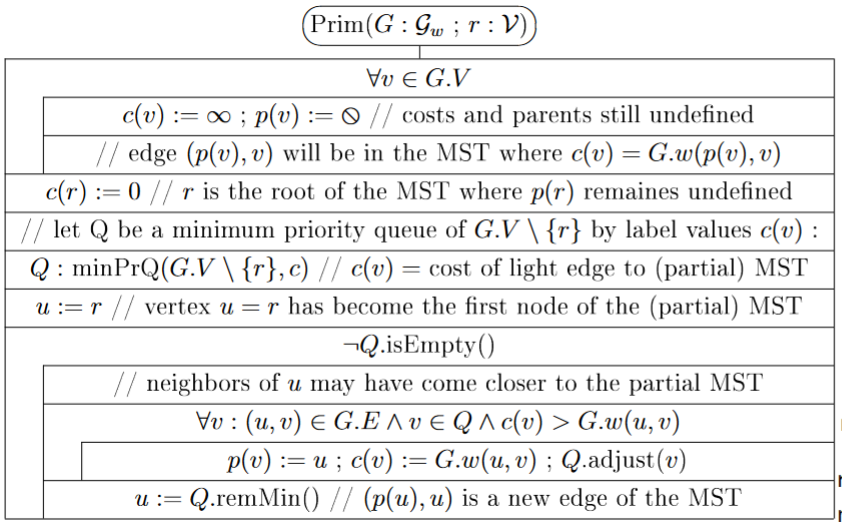
akkor az (u, v) él biztonságosan hozzávehető az A élhalmazhoz.

Definíció. A $G = (V, E)$ élsúlyozott gráf egy $(u, v) \in E$ élét könnyű élnek nevezzük, ha keresztezi az $(S, V \setminus S)$ vágást, és költsége kisebb vagy egyenlő mint bármely más a vágást keresztező élé.

Definíció. Legyen $G = (V, E)$ egy gráf. Ha $\emptyset \neq S \subset V$, akkor az $(S, V \setminus S)$ gráfot vágásnak nevezzük.

d) Az algoritmus egy tetszőleges csúcsból indulva elkezdi építeni a (V, F) minimális feszítőfát a $T = (N, A)$ kezdetben egyetlen egy csúcsból álló fából. Minden lépésben T-hez egy újabb biztonságos élt és az ahhoz tartozó csúcsot adjuk. Így az algoritmus futása során végig igaz marad az $N \subseteq V \wedge A \subseteq F$ invariáns. Ehhez minden lépésben egy könnyű élt választunk ki az $(N, V \setminus N)$ vágásban, ami a tétel miatt biztonságos él.

e)



$$MT_{prim} \in \underbrace{\Theta(n)}_{\text{inicializáló ciklus}} + \underbrace{\Theta(n)}_{\text{minPrQ inicializálása}} + \underbrace{O(n \cdot \log n)}_{\text{külső ciklus}} + \underbrace{O(m \cdot \log n)}_{\text{belső ciklus}} \in O((n+m) \cdot logn)$$

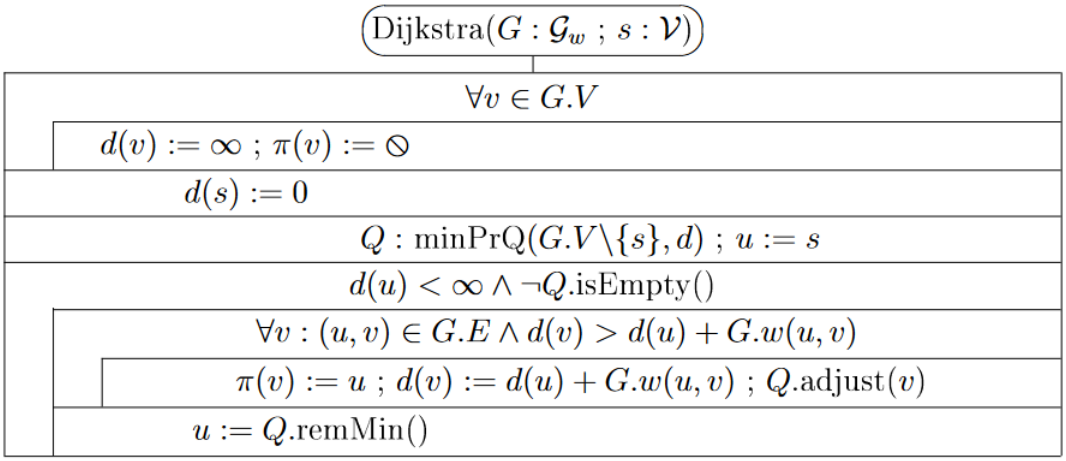
2. Legrövidebb utak

2.1. Dijkstra algoritmus

a) Egy élsúlyozott G gráf tetszőleges s csúcsából optimális utat ad meg G minden s-ből elérhető csúcsára. Minden élnek pozitív élsúlyúnak kell lennie.

b)

c)



d) Ha a prioritásos sort minimum kupacként ábrázoljuk. Ekkor az a kupac inicializálása $\Theta(n)$, az `adjust()` $\in \Theta(\log n)$

e)

f) Ekkor a `remMin()` eljárás műveletigénye $\Theta(n)$. Ez az eljárás a fő ciklus minden iterációjában lefut, figyelembe véve még a belső ciklust is $MT_{Dijkstra} \in O((n + m) \cdot n)$ műveletigény adódik. A minimális esetben egyetlen egyszer fut le a külső és egyszer sem a belső ciklus így abban az esetben az minimum prioritásos sor és a d és π tömbök inicializálása lesz meghatározó, ami $mT_{Dijkstra} \in \Theta(n)$ műveletigényt eredményez.

2.2. DAG legrövidebb utak algoritmus

2.3. Soralapú Bellman-Ford algoritmus

3. Mintaillesztés

3.1. Brute-force

Nem túl hatékony:

$$mT_{BF} \in \Theta(n)$$
$$MT_{BF} \in \Theta(n \cdot m), \text{ ami kellően nagy } m \text{ esetén } \Theta(n^2)$$

ahol n a szöveg m pedig a minta hossza.

3.2. Quicksearch

a) A Quicksearch algoritmus, egy $T/1 : \Sigma[n]$ szövegben a $P/1 : \Sigma[m]$ minta összes érvényes eltolását, azaz az

$$\{s \in [0..(n-m)] \mid T[s+1..s+n] = P[1..m]\}$$

halmazt adja meg. ($[0..(n-m)]$ intervallumot míg $T[s+1..s+n]$ részsztringet jelöl)

b)

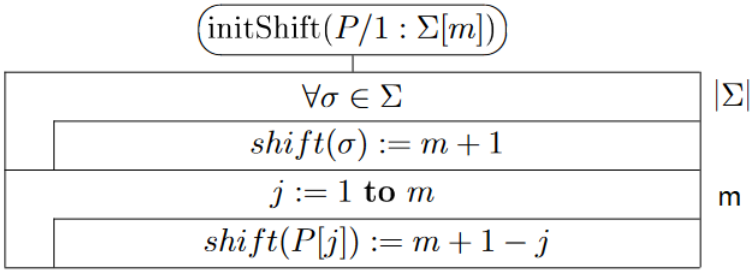
initial: mintahossz + 1				
σ	A	B	C	D
initial shift(σ)	8	8	8	8
A				
B				
A				
C				
A				
B				
A				
final shift(σ)				

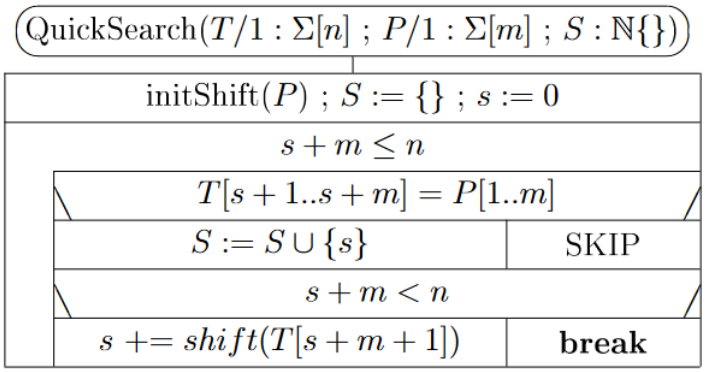
c)

initShift(P); S={}; s:=0

i	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23
T[i]	A	B	A	B	A	C	A	B	A	C	A	B	A	D	A	B	A	C	A	B	A	B	A

d)





e)

$$mT(n,m) \in \Theta\left(\frac{n}{m+1} + m\right) \quad \text{(pl. ha } T[1..n] \text{ és } P[1..m] \text{ diszjunktak)}$$

$$MT(n,m) \in \Theta\big((n-m+2)\cdot m\big) \quad \text{(pl. ha } T = AA...A \text{ és } P = A...A)$$

Definíció. Legyen $G = (V, E)$ egy gráf. Ha $\emptyset \neq S \subset V$, akkor az $(S, V \setminus S)$ gráfot vágásnak nevezzük.

f) Maximális futási ideje rosszabb mint a többi algoritmusnak. Átlagos futási ideje rosszabb mint a KMP algoritmusnak és a visszalépések miatt nem használható szekvenciális fájlokon egy ideiglenes tárhely bevezetése nélkül.

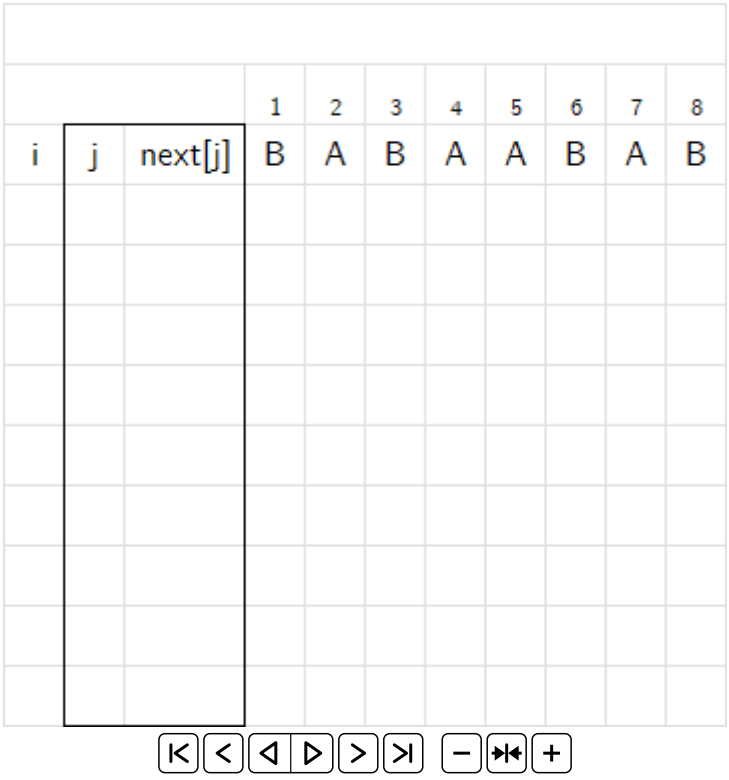
3.3. KMP

a) A KMP algoritmus, egy $T/1 : \Sigma[n]$ szövegben a $P/1 : \Sigma[m]$ minta összes érvényes eltolását, azaz az

$$\{s \in [0..(n-m)] \mid T[s+1..s+n] = P[1..m]\}$$

halmazt adja meg. ($[0..(n-m)]$ intervallumot míg $T[s+1..s+n]$ részsztringet jelöl)

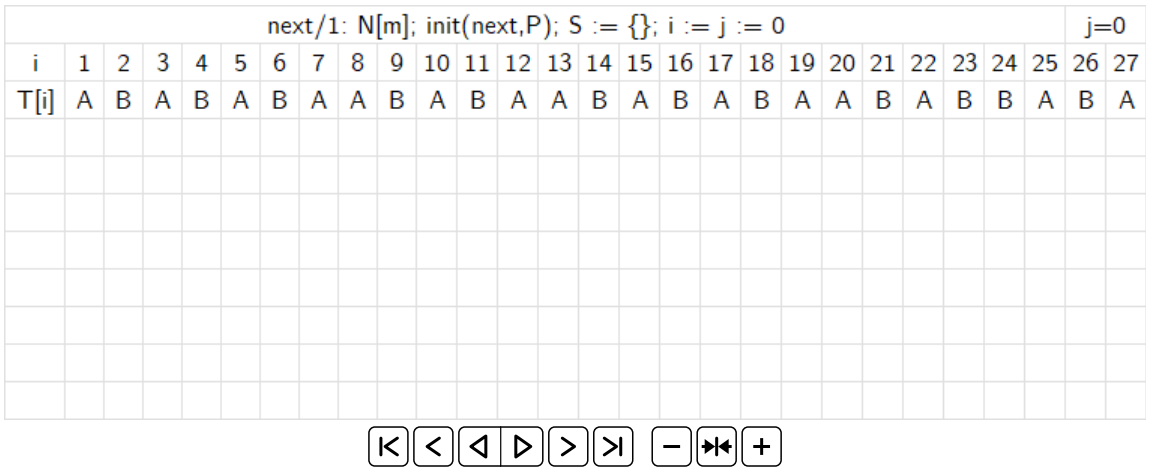
b)



Így a *next* tömb a következő:

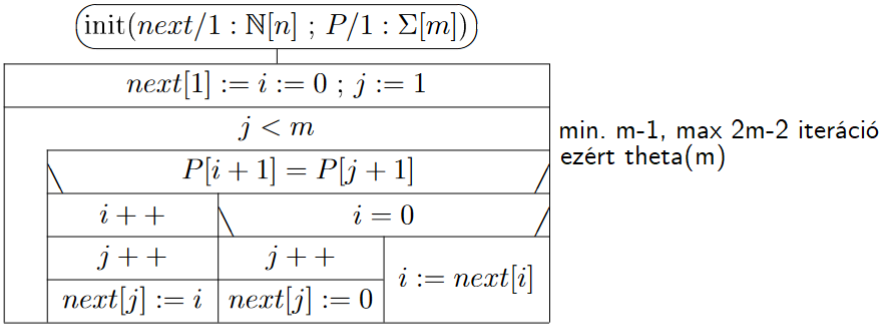
$P[j]$	B	A	B	A	A	B	A	B
j	1	2	3	4	5	6	7	8
$next[j]$	0	0	1	2	0	1	2	3

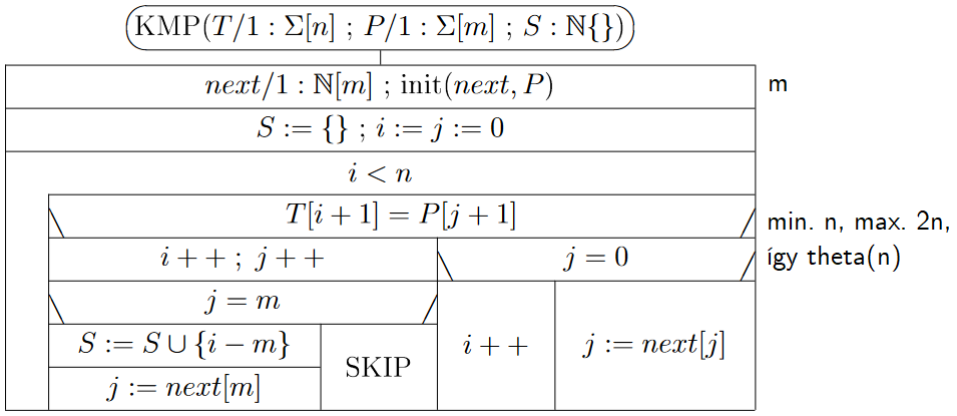
c)



Így $S = \{3, 8, 15\}$.

d)





e) Az `init()` függvény ciklusa minimum $m - 1$, maximum $2m - 2$ alkalommal iterál így műveletigénye $\Theta(m)$. A fő eljárás ciklusa legalább n-szer, legfeljebb $2n$-szer iterál, továbbá egyszer lefut az inicializáló függvény, így $m_{KMP} \in \Theta(m + n) = \Theta(n)$.

f) A KMP algoritmus műveletigénye jobb, továbbá, mivel az algoritmus során nem lépünk vissza a T szövegben, ezért szekvenciális inputfájlokkal is használható átmeneti tároló bevezetése nélkül.