Algoritmusok és adatszerkezetek II

Gráfok témakör jegyzete

Készült Ásványi Tibor előadásai és gyakorlatai alapján

Sárközi Gergő, 2021-22-1. félév
Nincsen lektorálva!

Tartalomjegyzék

1. Egyszerű gráfok és ábrázolásaik

1.1. Definíciók

1.1.1. Gráf

- $G = (V, E)$ rendezett pár
- V a csúcsok véges halmaza
  - $V = \{\}$ üres gráf
- $E \subseteq V \times V$ $\{(u, u) | u \in V\}$ az élek halmaza
  - hurokéleket explicit, párhuzamos éleket implicit kizártuk
  - tehát gráf = egyszerű gráf a továbbiakban

1.1.2. Irányítottság

- gráf irányítatlan, ha $(u, v) \in E \Leftrightarrow (v, u) \in E$
- gráf irányított, ha nem következik az egyik a másikból
- irányított $G = (V, E)$ gráf irányítatlan megfelelője:
  - $G' = (V, E')$ ahol $E' = \{(u, v) | (u, v) \in E \wedge (v, u) \in E\}$

1.1.3. Út

- út: $< u_0, u_1, u_2, ..., u_n >$ ($u_i \in V$ és $(u_{i-1}, u_i) \in E$)
  - ekkor az út hossza n (ami az élek száma is)
- rész-út: logikusan
- kör: kezdő és végpontja azonos, $n > 0$ és minden éle különböző
  - nincs hurokél: $n \geq 2$
- körmentes út: nem tartalmaz kört (nincs olyam részútja, ami kör)
- körmentes gráf: csak körmentes utak vannak benne

1.1.4. DAG (irányított, körmentes gráf)

- Directed Acyclic Graph

1.1.5. Összefüggőség

- G irányítatlan gráf összefüggő, ha mindegyik két csúcsa között van út
- G irányított gráf összefüggő, ha az irányítatlan megfelelője összefüggő
- generátor csúcs: ha tetszőleges csúcs elérhető belőle (van út köztük)
  - csak irányított gráfoknál van értelme
  - van generátor csúcs $\implies$ összefüggő (fordítva nem)

1.1.6. Fa

- szabad fa, irányítatlan fa: irányítatlan, körmentes, összefüggő gráf
- gyökeres fa, irányított fa: irányított gráf aminek van generátor csúcsa és az irányítatlan megfelőle körmentes
  - ekkor a generátor csúcs a gyökér csúcs
- nemüres fának pontosan eggyel kevesebb éle van, mint csúcsa
- erdő: összefüggő komponensei fák (vagy egyetlen fából áll)
  - irányítatlan G gráf erdő $\Leftrightarrow$ G körmentes
  - irányított G gráf erdő $\Leftrightarrow$ G irányítatlan megfelelője körmentes és G mindegyik összefüggő komponensének van generátor csúcsa

1.1.7. Részgráf

- $G = (V, E)$ gráfnak részgráfja a $G' = (V', E')$ gráf, ha:
  - mindkét gráf irányított vagy mindkét gráf irányítatlan
  - $V' \subseteq V$ és $E' \subseteq E$
- valódi részgráf: $G \neq G'$
- diszjunkt részgráfok: nincs közös csúcsok (és ezáltal közös élük sem)
- összefüggő komponens: olyan részgráf, aminek nincs összefüggő részgráfja
  - minden gráf vagy összefüggő, vagy felbontható diszjunkt összefüggő komponensekre

1.2. Gráfábrázolás alapjai

- Legyen $V = \{v_1, v_2, ..., v_n\}$
- Számok helyett gyakran használunk betűket (a=1, b=2, ...)
- Legyen $n = |V|$ és $m = |E|$, ekkor $0 \leq m \leq n * (n-1) \leq n^2$
  - azaz $m \in O(n^2)$
  - ritka gráf: $m \in O(n)$
  - sűrű gráf: $m \in \Theta(n^2)$

1.3. Grafikus gráfábrázolás

- Csúcs: kör (bele írjuk a csúcs sorszámát)
- Irányítatlan él: vonal
- Irányított él: nyíl

1.4. Szöveges ábrázolás

- Irányítatlan gráf: $u - u_1; u_2; u_3; ...$
  - u csúcsnak szomszédai a u_1, u_2, stb. csúcsok
  - nem kell oda-vissza feltünteni (tehát lehet mindent növekvő sorrendben leírni)
- Irányított gráf: $u \rightarrow u_1; u_2; u_3; ...$
  - u csúcsból vezet él a u_1, u_2, stb. csúcsokba

1.5. Szomszédossági mátrix (adjacency matrix), avagy csúcsmátrixos ábrázolás

- $A/1 : bit[n, n]$ mátrix, ahol bit=$\{0,1\}$ és $n = |V|$
  - $A[i, j] = 1 \Leftrightarrow (v_i, v_j) \in E$
  - $A[i, j] = 0 \Leftrightarrow (v_i, v_j) \notin E$
  - i a sor, j az oszlop: sorszámból vezet oszlopszámba él
- főátlóban mindig 0-k vannak, mert csak egyszerű gráfokkal foglalkozunk
  - $A[i, i] = 0$
- irányítatlan gráf: főátlóra szimmetrikus mátrix
  - $A[i, j] = A[j, i]$
  - emiatt n^2 helyett elég $n*(n-1)/2$ hosszú tömb a mátrix ábrázolására (általában az alsó háromszögmátrixot választjuk, főátló nélkül)

1.6. Szomszédossági listás (adjacency list) ábrázolás

- hasonlít a szöveges reprezentációhoz
- az i. csúcs szomszédainak sorszámát a $A[i]$ S1L tartalmazza
  - valójában: $A/1 : Edge * [n]$, ahol $Edge$ tagjai: v, $next$
- irányítatlan gráfok esetén minden él így kétszer van tárolva
- tárigénye: n db mutató, listáknak összesen m vagy $2 * m$ elem
  - azaz mindkét esetben $\Theta(n + m)$
  - ritka gráf: $\Theta(n)$ (mert $m \in O(n)$)
  - sűrű gráf: $\Theta(n^2)$ (mert $m \in O(n^2)$)

2. Absztrakt halmaz, sorozat, gráf

2.1. Absztrakt halmaz

- T típusú véges halmaz: $T\{\}$
- $s : T\{\}$ egy üres halmaz, $s : T\{a, b, c\}$ egy 3 elemű halmaz
- kiválasztás (tetszőleges elem eltávolítása): u *from* S
  - S egy nemüres halmaz
  - S-ből eltávolítunk egy elemet és ezt u értékül kapja

2.2. Absztrakt sorozat

- T típusú véges sorozat: $T <>$
- $s : T <>$ egy üres sorozat, $s : T < a, b, c >$ egy 3 elemű sorozat
- 1-től indexeljük őket
- ha $u, v : T <>$ akkor $u + v$ a konkatenálás

2.3. Absztrakt gráf

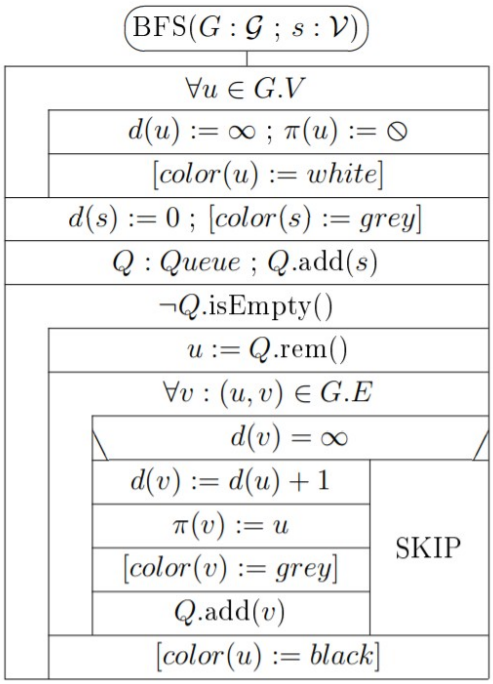
- $\mathcal{V}$: vertex (csúcs) típus
  - tetszőlegesen sok, névvel jelölt, értékkel ellátott címkét tárol
  - legyen v csúcs, *name* a címke neve, ekkor $name(v)$ a címke értéke
- $\mathcal{E}$: edge (él) típus, adattagjai: $u, v : \mathcal{V}$
- $\mathcal{G}$: élsúlyozatlan absztrakt gráf
  - adattagok: $V : \mathcal{V}\{\}$ és $E : \mathcal{E}\{\}$
  - V véges
  - $E \subseteq V \times V \setminus \{(u, u) | u \in V\}$

3. Elemi gráfalgoritmusok

- gráfok nincsenek élsúlyozva

3.1. BFS (szélességi gráfkeresés)

- irányított és irányítatlan gráfokra is értelmezett
- egy adott csúcsból (s) meghatározza az összes másik csúcsba vezető legrövidebb utat
- használt címkék:
  - d: távolság a kezdő csúcstól
  - π: "szülő", azaz a kezdő csúcs felé levő úton az első csúcs
    - $\pi(s) = \varnothing$
  - ha u egy el nem érhető csúcs: $pi(u) = \varnothing$ és $d(u) = \infty$
  - $color$: csak vizualizáció miatt
    - fehér: még nincs megtalálva
    - szürke: feldolgozásra vár
    - fekete: feldolgozva
- indeterminisztikus esetekben a kisebb címkéjű csúcsot válasszuk
- időkomplexitás
  - legyen $n = |G.V|$ és $m = |G.E|$
  - $MT(n, m) \in \Theta(n + m)$ (2. ciklus max m-szer iterál)
    - szomszédossági listás ábrázolás esetén: $\Theta(n^2)$
  - $mT(n, m) \in \Theta(n)$ (2. ciklus min egyszer iterál)

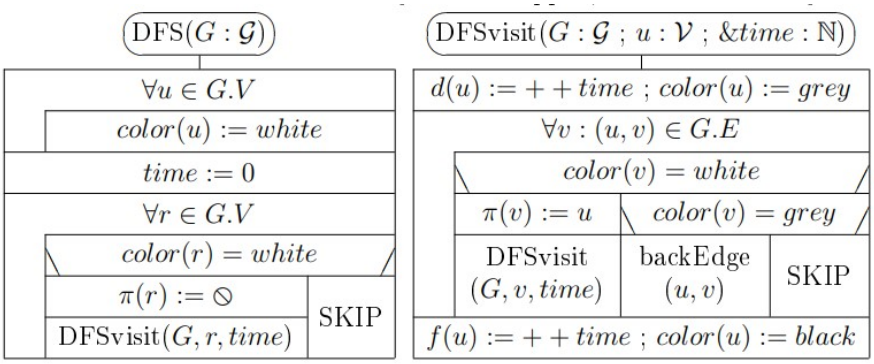


3.1.1. Szélességi fa (Breadth-first tree)

- ha $\pi(u) = v$-t úgy értelmezzük, hogy v szüleje u-nak, akkor fát kapunk
- más néven: legrövidebb utak fája

3.2. DFS (Mélységi gráfkeresés)

- egyszerű, irányított gráfokra értelmezett
- használt címkék:
  - d: elérési idő (discovery time)
  - f: befejezési idő (finishing time)
  - π: "szülő", azaz a kezdő csúcs felé levő úton az első csúcs
    - $\varnothing$, ha egy mélységi fa gyökere
  - color: (nem kihagyható)
    - fehér: érintetlen
    - szürke: belőle elérhető csúcsokat járunk be éppen
    - fekete: feldolgozva
- DFSvisit mélységi fát számol ki, DFS pedig mélységi feszítő erdőt
- time akkor lép, amikor elérünk vagy befejezünk egy csúcsot
  - az első time érték = 1
- backEdge(u,v): tetszőleges eljárás, tegyük fel, hogy $\Theta(1)$
- időkomplexitás:
  - legyen $n = |G.V|$ és $m = |G.E|$
  - DFSvisit n-szer hívódik, de a ciklusa ÖSSZESEN csak m-et iterál
  - $MT(n,m), mT(n,m) \in \Theta(n+m)$



3.2.1. Élek osztályozása

- (u,v) fa-él (tree edge): (u,v) valamelyik mélységi fa éle.
  (Ezen élek mentén járjuk be a gráfot.)
- (u,v) vissza-él (back edge): v az u őse, de nem faél.
  - Ez azt jelenti, hogy irányított kört találtunk.
- (u,v) előre-él (forward edge): v az u leszármazottja, de nem faél.
- (u,v) kereszt-él (cross edge): u és v két olyan csúcs, amelyek vagy azonos mélységi fa két különböző ágán vannak, vagy két különböző mélységi fában vannak.

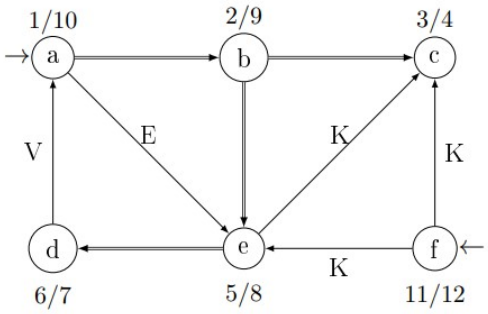
3.2.2. Élek felismerése

Mélységi bejárás során az (u,v) él feldolgozásakor:
(Ha szöveges címkét rendelnénk a feldolgozás során, így tennénk.)

- (u,v) fa-él: v csúcs még fehér
- (u,v) vissza-él: v csúcs éppen szürke
- (u,v) előre-él: v csúcs már fekete és $d(u) < d(v)$
- (u,v) kereszt-él: v csúcs már fekete és $d(u) > d(v)$

3.2.3. Bejárás szemléltetése

- Élek: dupla a fa él, többi betűvel jelölve
- csúcsnál x/y: d(u) / f(u)



3.2.4. DAG tulajdonság eldöntése

- G irányított gráf akkor Directed Acyclic Graph, ha nem tartalmaz irányított kört
- DFS fog találni vissza-élt, ha van irányított kör a gráfban
  - viszont nem fog annyi vissza-élt találni, ahány irányított kör van: egy él lehet több kör része is
- (u, v) vissza-él $\implies < u, v, ..., \pi(\pi(u)), \pi(u), u >$ egy irányított kör
  - ezt a kört az algoritmus a fordított sorrendben találná meg
- a backEdge eljárás hasznos eme infó lementése céljából

3.2.5. Topologikus rendezés

- Topologikus rendezés: gráf csúcsainak olyan sorba rendezése, hogy minden él egy későbbi csúcsba mutasson. (Pl. minden él balról jobbra.)
- Irányított gráfnak lehet 0, 1 vagy több féle topologikus rendezése.
- Irányított gráfnak pontosan akkor van topologikus rendezése, ha DAG.
- Mélységi bejárás segítségével topologikus rendezés:
  - csúcsok $f(u)$ szerint csökkenő rendezése
  - vagy csúcsokat befejezésükkör rakjuk egy verembe

4. Élsúlyozott gráfok és ábrázolásaik

4.1. Absztrakt ábrázolás

- $\mathcal{V}$: vertex (csúcs) típus, címkéket tárol, lásd normál gráf
- $\mathcal{E}$: edge (él) típus, adattagjai: $u, v : \mathcal{V}$
- $\mathcal{G}_w$: élsúlyozott absztrakt gráf
  - adattagok: $V : \mathcal{V}\{\}$, $E : \mathcal{E}\{\}$ és $w : E \to \mathbb{R}$
  - V véges
  - $E \subseteq V \times V \setminus \{(u, u) | u \in V\}$
  - w a súlyfüggvény

4.2. Emlékeztető

- $n = |V|$ és $m = |E|$
- $0 \leq m \leq n * (n - 1) \leq n^2$
- ritka gráf: $m \in O(n)$
- sűrű gráf: $m \in \Theta(n^2)$

4.3. Grafikus ábrázolás

- Csúcs: kör (bele írjuk a csúcs sorszámát)
- Irányítatlan él: vonal; irányított él: nyíl
- Élek mellé odaírjuk a súlyukat

4.4. Szöveges ábrázolás

- Irányítatlan gráf: $v - v_1, w_1; v_2, w_2; ...$
  - v csúcsnak szomszédai a v_i csúcsok, az él súlya w_i
  - nem kell oda-vissza feltünteni (tehát lehet mindent növekvő sorrendben leírni)
- Irányított gráf: $v \to v_1, w_1; v_2, w_2; ...$
  - u csúcsból vezet él a u_i csúcsokba, w_i élsúllyal

4.5. Szomszédossági mátrix, csúcsmátrix, adjacency matrix

- $A \rightarrow: \mathbb{R}_\infty[n, n]$ ahol $n = |V|$
  - $A[i, i] = 0$ (egyszerű gráf)
  - $A[i, j] = \infty \Leftrightarrow (v_i, v_j) \notin E$
  - $A[i, j] = w(v_i, v_j) \Leftrightarrow (v_i, v_j) \in E$
  - i a sor, j az oszlop: sorszámból vezet oszlopszámba él
- Irányítatlan gráf: főátlóra szimmetrikus
- Tárigény $\Theta(n^2)$ (ha egy $\mathbb{R}_\infty$ konstans hosszú)

4.6. Szomszédossági lista, adjacency list

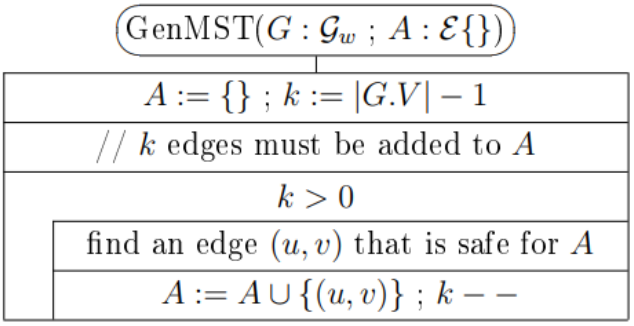
- Hasonlít a szöveges reprezentációhoz
- $A : Edge * [n]$ ahol $n = |V|$ (A egy S1L)
  - *Edge* adattagjai: $v : \mathbb{N}$; $w : \mathbb{R}$; $next : Edge*$
- Irányítatlan gráfok esetén minden él így kétszer van tárolva
- Tárigény: n db mutató, listáknak összesen m vagy $2 * m$ elem
  - ritka gráf: $\Theta(n)$ (mert $m \in O(n)$)
  - sűrű gráf: $\Theta(n^2)$ (mert $m \in O(n^2)$)

5. Minimális feszítőfák, Minimum Spanning Tree

- $G = (V, E)$ irányítatlan gráf feszítő erdeje $T = (V, F)$, ahol $F \subseteq E$
  - azaz T minden komponense fa, amik páronként diszjunktak
- Legyen $w(G)$ az E élek súlyösszege
  - T a G MST-je, ha T a G feszítőfája és bármely T' feszítőfára $w(T) \leq w(T')$
- Mohó modszerrel, $O(m * \lg n)$ időben számolható (Kruskal és Prim)

5.1. Általános algoritmus

- $A : \mathcal{E}\{\}$ kezdetben üres, végig G egy feszítőfájának élhalmazának részhalmaza
- Minden lépésben egy biztonságos (u, v) élt adunk A-hoz: olyan élt, amivel az előbbi állítás (invariáns) igaz marad
- Addig fut az algoritmus, amíg $|A| \neq |G.V| - 1$



5.1.1. Vágás (nem tűnik fontosnak)

- $G = (V, E)$ gráf, $\{\} \subset S \subset V$
- Vágás G-n: $(S, V \setminus S)$
- $(u, v) \in E$ él keresztezi a $(S, V \setminus S)$ vágást, ha $u \in S = v \in V \setminus S$
  - azaz ha $(u \in S \wedge v \in V \setminus S) \vee (u \in V \setminus S \wedge v \in S)$
- $(u, v) \in E$ könnyű él egy vágásban, ha
  - (u, v) keresztezi a vágást
  - és $\forall (p, q)$ vágást keresztező élre $w(u, v) \leq w(p, q)$
- $A \subseteq E$ élhalmaz elkerüli a vágást, ha A egyetlen éle sem keresztezi azt

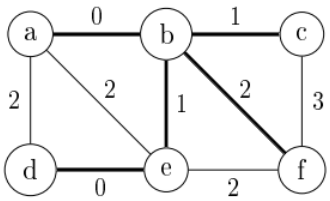
5.1.2. Tétel biztongságos élre (nem tűnik fontosnak)

- Legyen $g = (V, E)$ irányítatlan, összefüggő, élsúlyozott gráf
- A részhalmaza G valamelyik minimális feszítőfájának élhalmazának
- $(S, V \setminus S)$ vágás elkerüli az A élhalmazt
- $(u, v) \in E$ könnyű él a $(S, V \setminus S)$ vágásban
- Ekkor (u, v) biztonságos él; hozzávehető A-hoz
- Bizonyítás: nem nagyon értem, nem tűnik fontosnak, de jegyzetben

5.2. Kruskal algoritmusa

- Legyen $G = (V, E)$ összefüggő, irányítatlan, élsúlyozott gráf
- Invariáns: (V, A) a G egy feszítő erdeje és A részhalmaza G egy MST-jének élhalmazának
- Kezdetben $A = \{\}$. Az éleket súlyuk szerint növekvő sorrendben dolgozzuk fel. Minden új éllel két fát kötünk össze, így a fák száma mindig csökken. Amennyiben egy él nem volt biztonságos, eldobjuk.

Komponensek	él	él súlya	biztonságos?
a, b, c, d, e, f	(a,b)	0	+
ab, c, d, e, f	(d,e)	0	+
ab, c, de, f	(b,c)	1	+
abc, de, f	(b,e)	1	+
abcde, f	(a,d)	2	-
abcde, f	(a,e)	2	-
abcde, f	(b,f)	2	+
abcdef	-	-	-



Kruskal($G : \mathcal{G}_w$; $A : \mathcal{E}\{\}$) : $\mathbb{N}$ //O(m * log n)

$\forall v \in G.V$

makeSet(v) // a spanning forest of single vertices is formed

$A := \{\}$; $k := |G.V|$

// k is the number of components of the spanning forest

// let Q be a minimum priority queue of $G.E$ by weight $G.w$:

Q : minPrQ($G.E, G.w$) //Θ(m)

$k > 1 \wedge \neg Q$.isEmpty()

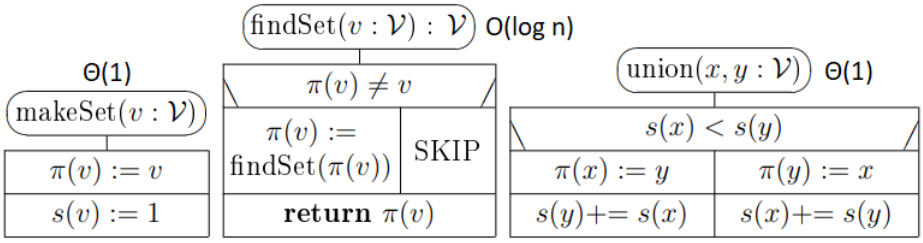
$e : \mathcal{E} := Q$.remMin() //O(log m), O(log n)

$x :=$ findSet($e.u$) ; $y :=$ findSet($e.v$) //O(log n)

$x \neq y$

$A := A \cup \{e\}$; union(x, y) ; $k - -$	SKIP

return k //komponensek száma



- Az irányítatlan feszítőerdő mellett egy irányított erdőt is kezelünk. Ez az irányított erdő a "gyökere felé mutat".
- π címke: szülő csúcsot tárolja, gyökérre viszont $\pi(r) = r$
- $s(r)$ az r gyökérhez tartozó fa mérete
- *union* mindig a kisebbet fűzi be a nagyobb-egyenlő gyökerébe közvetlenül. Így sosem lesz a fák magassága nagyobb, mint méretük $\log_2$. Ezért lesz a *findSet* hatékony.

5.3. Prim algoritmusa

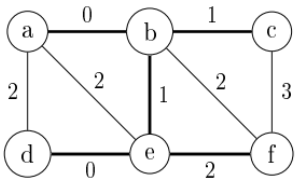
- Legyen $G = (V, E)$ összefüggő, irányítatlan, élsúlyozott gráf
- Invariáns: $T = (N, A)$ mindig G egy MST-jének része
- A-hoz mindig olyan biztonságos élet adunk hozzá, amelyik él egy A-beli és A-n kívüli csúcsot köt össze.
- Kezdetben tetszőleges csúcsból indulunk. ($N = \{r\};\ A = \{\}$)
- Minden $V \setminus N$ csúcshoz tartozik c és p címke, ha van él köztük és a N egyik csúcsa között (azaz a $(N, V \setminus N)$ vágásban)
  - $p(u)$: egy vágásban lévő él másik csúcsa; az él: $(p(u), u)$
  - $c(u)$: az él súlya; $c(u) = w(p(u), u)$
  - mindez úgy, hogy $\forall(x, u)$ vágásbeli élre $w(x, u) \geq w(p(u), u)$
- Ha nincs él u és N egy csúcsa között sem: $p(u) = \varnothing$ és $c(u) = \infty$
- A-hoz mindig a $c(u)$ szerint legkisebb élet adjuk hozzá.
  - Egy min prioritásos sorban tároljuk az u csúcsokat.
  - Egy ilyen hozzáadáskor frissíteni kell a $c(v)$ értékeket minden olyan v-re, amelyik szomszédja u-nak.
- A végső fa élei $(p(x), x)$, ahol $x \in V \setminus \{r\}$ (csúcshalmaza pedig V)

$\text{Prim}(G : \mathcal{G}_w \; ; \; r : \mathcal{V})$ O((n+m) ∗ log n)

$\forall v \in G.V$	
$c(v) := \infty \; ; \; p(v) := \oslash$ // costs and parents still undefined	
// edge $(p(v), v)$ will be in the MST where $c(v) = G.w(p(v), v)$	
$c(r) := 0$ // r is the root of the MST where $p(r)$ remaines undefined	
// let Q be a minimum priority queue of $G.V \setminus \{r\}$ by label values $c(v)$:	
Q : minPrQ$(G.V \setminus \{r\}, c)$ // $c(v)$ = cost of light edge to (partial) MST	Θ(n)
$u := r$ // vertex $u = r$ has become the first node of the (partial) MST	
$\neg Q$.isEmpty()	n iteráció
// neighbors of u may have come closer to the partial MST	
$\forall v : (u, v) \in G.E \wedge v \in Q \wedge c(v) > G.w(u, v)$ (minden élre)	össz.: O(m ∗ log n)
$p(v) := u \; ; \; c(v) := G.w(u, v) \; ; \; Q$.adjust$(v)$ (max 2x)	O(log n)
$u := Q$.remMin() // $(p(u), u)$ is a new edge of the MST	O(log n)

O(n ∗ log n) + O(m ∗ log n) = O((n+m) ∗ log n) de ha összefüggő m ≥ n−1 tehát MT ∈ O(m ∗ log n)

c értékek Q-ban						minimális feszítő-fába:	p címkék változásai					
a	b	c	d	e	f		a	b	c	d	e	f
∞	∞	∞	0	∞	∞		⊘	⊘	⊘	⊘	⊘	⊘
2	∞	∞		0	∞	d	d				d	
2	1	∞			2	e		e				e
0		1			2	b	b		b			
		1			2	a						
					2	c						
						f						
0	1	1	0	0	2	eredmény	b	e	b	⊘	d	e



6. Legrövidebb utak egy forrásból

- Adott s csúcsból minden s-ből elérhető csúcsba legrövidebb utat keresünk.
- Értelmezzük az irányítatlan gráfot olyan irányított gráfként, ahol $(u, v) \in E \implies (v, u) \in E \wedge w(u, v) = w(v, u)$
- Pontosan akkor oldható meg, ha nem létezik s-ből elérhető negatív kör
  - Negatív kör: olyan kör, amelynek összsúlya negatív
  - Irányítatlan gráfok esetén akkor is lesz ilyen "kör", ha van negatív súlyú él.
- Ha megoldható, mindegyik $v \in V \setminus \{s\}$-re két lehetőség van:
  - Létezik út s-ből v-be:
    - $d(v)$ az optimális út hossza
    - $\pi(v)$ az optimális úton v-t közvetlenül megelőző csúcs
  - Nem létezik út s-ből v-be: $d(v) = \infty$ és $\pi(v) = \varnothing$
  - s csúcsra: $d(s) = 0$ és $\pi(s) = \varnothing$
- Az algoritmusok az optimális utat folyamatosan közelítik, $d(v) = \infty$ és $\pi(v) = \varnothing$-ből indulva. Jelölje $s \rightsquigarrow v$ azt a kiszámolt utat, ami az eddigi legjobb; ekkor $d(v) \in \mathbb{R}$ és $v \neq s$ esetén $\pi(v) \neq \varnothing$. Gráf éleit folyamatosan vizsgáljuk és ha pl. (u, v)-t nézzük éppen és $s \rightsquigarrow u \to v$ rövidebb, mint $s \rightsquigarrow v$, akkor frissítjük: közelítjük (relaxation).
- Az algoritmusok közös vonása, hogy egy feldolgozandó csúcsok halmazából minden lépésben kivesznek egy csúcsot és az összes kimenő élre elvégzik a közelítést. A közelítéseket a csúcs kiterjesztésének nevezzük. A csúcs kivételét és a kiterjesztést együtt pedig a csúcs feldolgozásának.

6.1. Dijkstra algoritmusa

- Előfeltétel: mindegyik él súlya nemnegatív
- Nagyon hasonlít Prim algoritmusára, mohó
  - $MT \in O((n+m) * \lg n)$ (magyarázat: Prim algoritmusa)
  - $mT \in \Theta(n)$ (ha s-nek nincs rákövetkezője)

Dijkstra($G : \mathcal{G}_w$; $s : \mathcal{V}$)

$\forall v \in G.V$

$d(v) := \infty$; $\pi(v) := \oslash$ // distances are still ∞, parents undefined

// $\pi(v)$ = parent of v on $s \rightsquigarrow v$ where $d(v) = w(s \rightsquigarrow v)$

$d(s) := 0$ // s is the root of the shortest-path tree

// let Q be a minimum priority queue of $G.V \backslash \{s\}$ by label values $d(v)$:

Q : minPrQ($G.V \backslash \{s\}, d$)

$u := s$ // going to calculate shortest paths form s to other vertices

$d(u) < \infty \wedge \neg Q$.isEmpty()

// check, if $s \rightsquigarrow u \rightarrow v$ is shorter than $s \rightsquigarrow v$ before

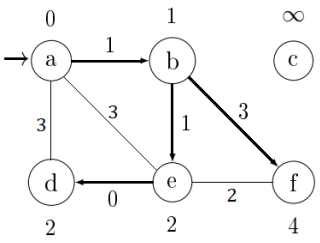
$\forall v : (u, v) \in G.E \wedge d(v) > d(u) + G.w(u, v)$

$\pi(v) := u$; $d(v) := d(u) + G.w(u, v)$; Q.adjust(v)

$u := Q$.remMin() // $s \rightsquigarrow u$ is optimal now, if it exists

d értékek Q-ban						kiter-
a	b	c	d	e	f	jesztett
0	∞	∞	∞	∞	∞	csúcs : d
	1	∞	3	3	∞	a : 0
		∞	3	2	4	b : 1
		∞	2		4	e : 2
		∞			4	d : 2
		∞				f : 4
0	1	∞	2	2	4	eredmény

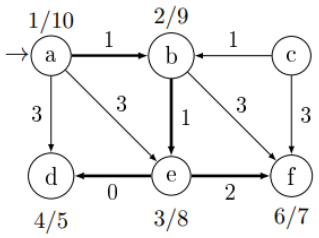
π címkék változásai					
a	b	c	d	e	f
$\oslash$	$\oslash$	$\oslash$	$\oslash$	$\oslash$	$\oslash$
	a		a	a	
				b	b
				e	
a	b	$\oslash$	e	b	b



- $u := Q.remMin()$-nél $s \rightsquigarrow u$ optimális ha $d(u) \neq \infty$. Bizonyítás indirekt módon, jegyzetben.
- Ha $u := Q.remMin()$-nél $d(u) = \infty$ akkor Q egy eleme sem érhető el s-ből.

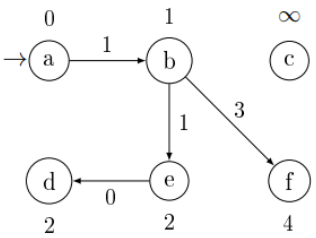
6.2. DAGshP (DAG shortest Paths)

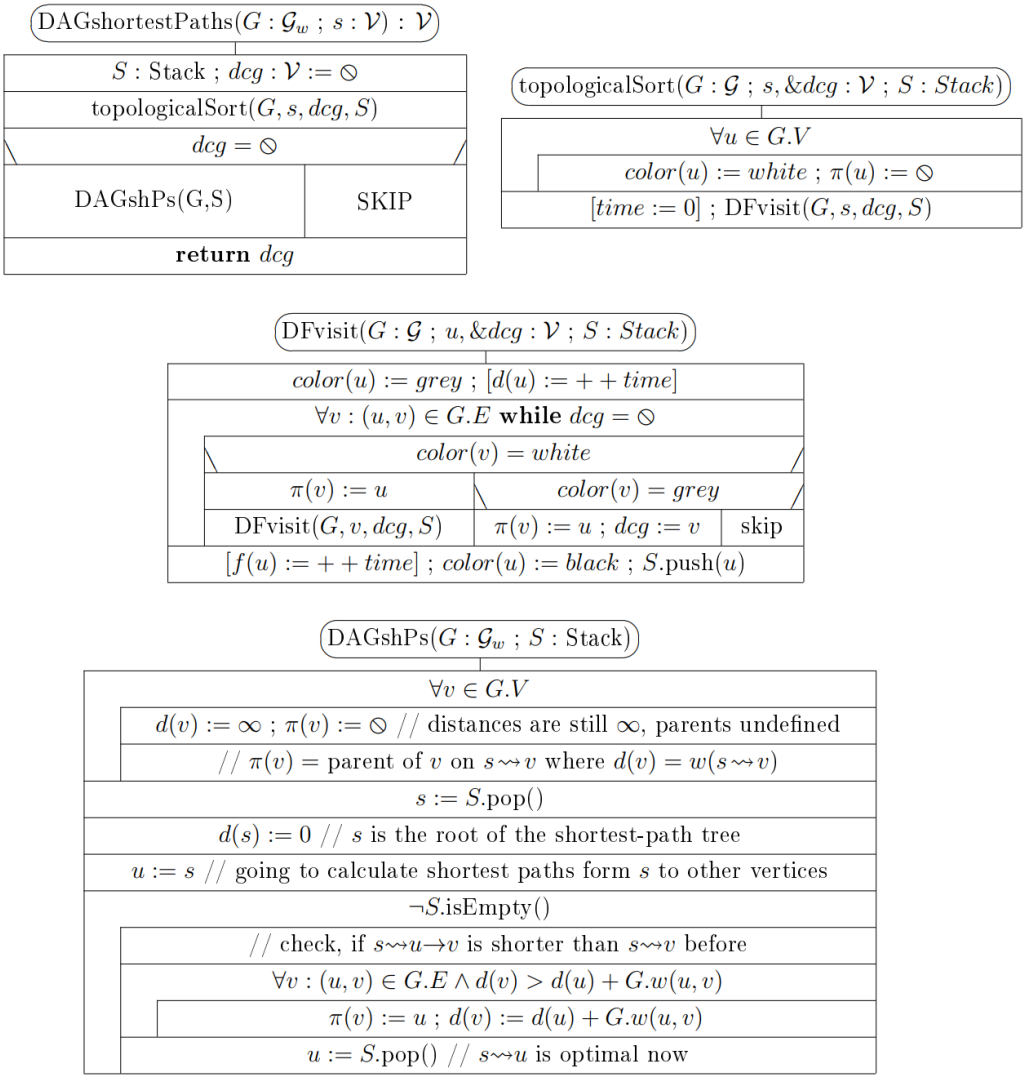
- Előfeltétel: irányított a gráf és nincs benne irányított kör
- Az algoritmus ellenőrzi az előfeltételét:
  - nincs irányított kör: $\varnothing$-tel tér vissza
  - különben a kör egy csúcsával tér vissza, ami a π címkék mentén fordított irányban bejárható
- $MT \in \Theta(n+m)$ és $mT \in \Theta(n)$
- A *topologicalSort* csak az s-ből elérhető részgráfot teszi be S-be, a megfelelő sorrendben.
- A *time* változó globális, de csak szemléltetésre van, elhagyható.



$S = \langle$ a, b, e, f, d $\rangle$

d értékek változásai						kiter-	π címkék változásai					
a	b	c	d	e	f	jesztett	a	b	c	d	e	f
0	∞	∞	∞	∞	∞	csúcs : d	$\oslash$	$\oslash$	$\oslash$	$\oslash$	$\oslash$	$\oslash$
	1		3	3		a : 0		a		a	a	
				2	4	b : 1					b	b
			2			e : 2				e		
						f : 4						
						d : 2						
0	1	∞	2	2	4	eredmény	$\oslash$	a	$\oslash$	e	b	b

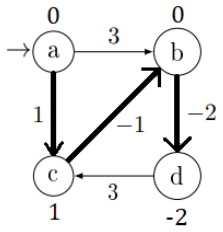



DAGshortestPaths($G : \mathcal{G}_w$; $s : \mathcal{V}$) : $\mathcal{V}$
S : Stack ; $dcg : \mathcal{V} := \oslash$
topologicalSort(G, s, dcg, S)
$dcg = \oslash$
DAGshPs(G,S)
SKIP
return dcg
topologicalSort($G : \mathcal{G}$; $s, \&dcg : \mathcal{V}$; $S : Stack$)
$\forall u \in G.V$
$color(u) := white$; $\pi(u) := \oslash$
$[time := 0]$; DFvisit(G, s, dcg, S)
DFvisit($G : \mathcal{G}$; $u, \&dcg : \mathcal{V}$; $S : Stack$)
$color(u) := grey$; $[d(u) := ++time]$
$\forall v : (u, v) \in G.E$ **while** $dcg = \oslash$
$color(v) = white$
$\pi(v) := u$
DFvisit(G, v, dcg, S)
$color(v) = grey$
$\pi(v) := u$; $dcg := v$
skip
$[f(u) := ++time]$; $color(u) := black$; S.push(u)
DAGshPs($G : \mathcal{G}_w$; S : Stack)
$\forall v \in G.V$
$d(v) := \infty$; $\pi(v) := \oslash$ // distances are still ∞, parents undefined
// $\pi(v)$ = parent of v on $s \rightsquigarrow v$ where $d(v) = w(s \rightsquigarrow v)$
$s := S$.pop()
$d(s) := 0$ // s is the root of the shortest-path tree
$u := s$ // going to calculate shortest paths form s to other vertices
$\neg S$.isEmpty()
// check, if $s \rightsquigarrow u \rightarrow v$ is shorter than $s \rightsquigarrow v$ before
$\forall v : (u, v) \in G.E \wedge d(v) > d(u) + G.w(u, v)$
$\pi(v) := u$; $d(v) := d(u) + G.w(u, v)$
$u := S$.pop() // $s \rightsquigarrow u$ is optimal now

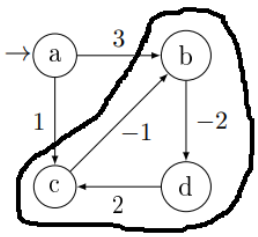
6.3. Sor-alapú Bellman-Ford algoritmus (QBF)

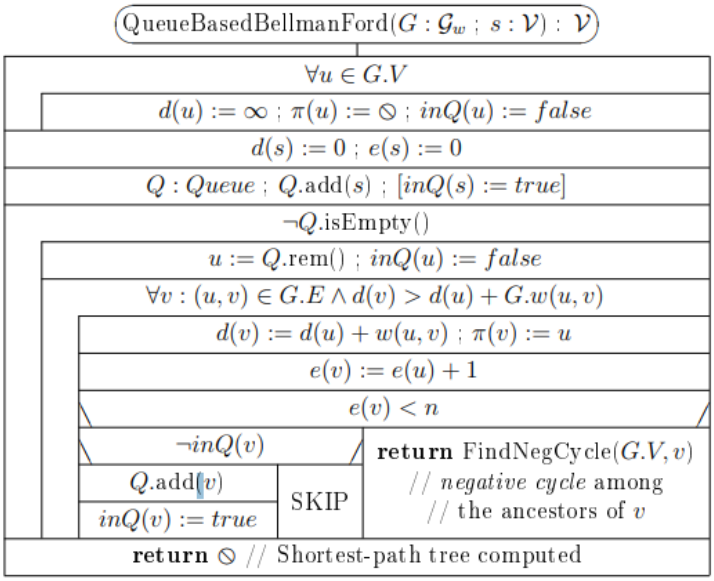
- Nem mohó
- Előfeltétel: nincs s-ből elérhető negatív kör.
- Előfeltétel ellenőrzése algoritmusban:
  - $e(v)$ címke: $s \rightsquigarrow v$ út élszáma (NEM súlya!)
  - nincs negatív kör: $e(v) < n$ és $\varnothing$ a visszatérési érték
  - $e(v) \geq n \implies v$ része egy negatív körnek, vagy $\pi(v)$-n keresztül negatív körbe érkezünk (max n lépésben megoldható), a kör egyik csúcsa a visszatérési érték
- $MT = O(n * m)$
  - gyakorlatban átlagosan $\Theta(n)$, ha ritka a gráf ($m \in O(n)$)
- Új címke: inQ (benne van-e a sorban, bit tömbbel megvalósítható)

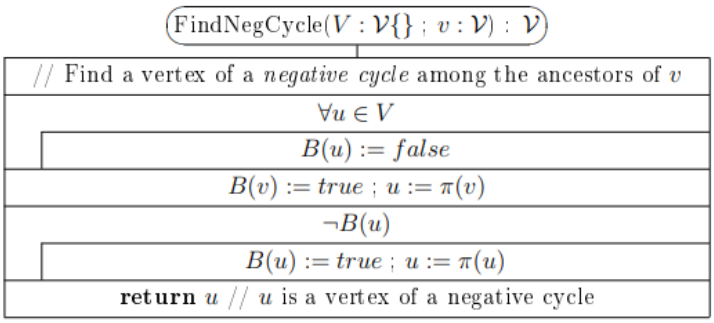
$d; e$ változásai				kiter-	Q :	π változásai			
a	b	c	d	jesztett	Queue	a	b	c	d
0; 0	∞	∞	∞	csúcs: $d; e$	⟨ a ⟩	⦸	⦸	⦸	⦸
	3; 1	1; 1		a: 0; 0	⟨ b, c ⟩		a	a	
			1; 2	b: 3; 1	⟨ c, d ⟩				b
	0; 2			c: 1; 1	⟨ d, b ⟩		c		
				d: 1; 2	⟨ b ⟩				
			−2; 3	b: 0; 2	⟨ d ⟩				b
				d: −2; 3	⟨ ⟩				
0	0	1	−2	végső d és π értékek		⦸	c	a	b



$d; e$ változásai				kiter-	Q :	π változásai			
a	b	c	d	jesztett	Queue	a	b	c	d
0; 0	∞	∞	∞	csúcs: $d; e$	⟨ a ⟩	⦸	⦸	⦸	⦸
	3; 1	1; 1		a: 0; 0	⟨ b, c ⟩		a	a	
			1; 2	b: 3; 1	⟨ c, d ⟩				b
	0; 2			c: 1; 1	⟨ d, b ⟩		c		
				d: 1; 2	⟨ b ⟩				
			−2; 3	b: 0; 2	⟨ d ⟩				b
		0; 4		d: −2; 3	⟨ c ⟩			d	





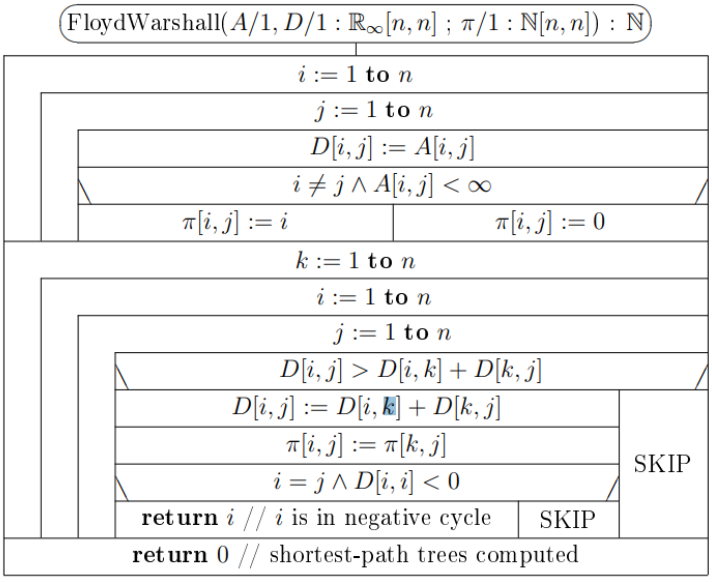


- Menet:
  - 0.: s feldolgozása
  - $(i + 1)$.: i. menet lévén sorban lévő csúcsok feldolgozása
- Ha s-ből nem érhető el negatív kör:
  - Ha $s \rightsquigarrow u$ optimális út k élből áll, akkor a k. menet elején már megtalálta az algoritmus.
  - Minden optimális út max $n - 1$ hosszú $\implies$ n-1. menet végére kiürül a sor. Innen jön $MT \in O(n * m)$.

7. Legrövidebb utak minden csúcspárra

7.1. Floyd-Warshall (FW) algoritmus

- Legyen $\mathbb{R}_\infty = \mathbb{R} \cup \{\infty\}$, ahol ∞ jelentése: nincs él/út
- Mátrixok kiolvasása, $A[i,j]$ jelentése: i-ből (sor) j-be (oszlop) valami
- Bemenet: élsúlyozott gráf egy $A/1 : \mathbb{R}_\infty[n,n]$ csúcsmátrixként
- Kimenet:
  - Utak hossza: $D/1 : \mathbb{R}_\infty[n,n]$
  - Útban a csúcs megelőzője: $\pi/1 : \mathbb{N}[n,n]$
    - Ha nincs út, vagy $i = j$ akkor: $\pi[i,j] = \varnothing$
- Előfeltétel (ellenőrzi magától): gráfban nincs negatív kör
  - Visszatér egy negatív körbeli csúccsal, ha van ilyen.



- Műveletigény: $MT(n) \in \Theta(n^3)$
  - Ha van negatív kör, akkor $mT(n) \in \Theta(n^2)$ (egyébként $\Theta(n^3)$)

- Megoldás módszere: dinamikus programozás
  - két csúcs (i és j) között az első k csúcs felhasználásával keresünk utat ($i > k$ és $j > k$ megengedett)
  - $\exists i \rightsquigarrow^{0}_{opt} j$ ha $(i,j) \in G.E$ (ekkor az út: $< i,j >$)
  - $\forall k \in 1..n : i \rightsquigarrow^{k}_{opt} j$ vagy $i \rightsquigarrow^{k-1}_{opt} j$ vagy $i \rightsquigarrow^{k-1}_{opt} k \rightsquigarrow^{k-1}_{opt} j$
  - Ha az i-ből i-be vezető optimális út súlya negatív, akkor (és csak akkor) van negatív kör és ekkor i a negatív kör része.

7.2. Visszavezetés az "utak egy forrásból" feladatra

- Floyd-Warshall algoritmus D és π mátrixának i-edik sora a "legrövidebb utak egy forrásból" feladat megoldása $s = i$ esetre, valamint fordítva.
- Fontos: a DAG és Dijkstra algoritmusok előfeltétele szigorúbb.
- A DAG algoritmus n-szer meghívva: $MT \in \Theta(n * (n + m))$
  - Ritka gráfon aszimptotikusan jobb: $\Theta(n^2)$
  - Sűrű gráfon egyenlő ($\Theta(n^3)$), gyakorlatban lassabb
- Dijkstra algoritmust n-szer meghívva: $MT \in \Theta(n * (n + m) * \log n)$
  - Ritka gráf: $\Theta(n^2 * \log n)$ (tehát jobb)
  - Sűrű gráf: $\Theta(n^3 * \log n)$ (tehát rosszabb)
- QBF algoritmust n-szer meghívva: $MT \in \Theta(n^2 * m)$
  - Ez soha nem jobb, legkedvezőbb esetben aszomptotikusan azonos

8. Gráf tranzitív lezártja (Transitive Closure)

- Gráfban honnét hova lehet eljutni (konkrét út és súly lényegtelen)
- $G = (V, E)$ tranzitív lezártja a $T \subseteq V \times V$ reláció, ahol:
  $(u, v) \in T \Leftrightarrow G$-ben az u csúcsból elérhető a v csúcs

8.1. Egyszerű algoritmus

- Bemenet: gráf egy $A/1 : \mathbb{B}[n, n]$ csúcsmátrixként
- Kimenet: $T/1 : \mathbb{B}[n, n]$ ahol $T[i, j] =$ van út i-ből (sor) j-be (oszlop)
- Megoldási módszer: dinamikus programozás, $mT = MT \in \Theta(n^3)$
  - $T^{(0)}_{i,j} = \exists i \rightsquigarrow j$ az első k csúcsot használva csak $(i, j > k$ szabad)
  - $T^{(0)}_{i,j} = A[i, j] \vee (i = j)$
  - $T^k_{i,j} = T^{k-1}_{i,j} \vee (T^{k-1}_{i,k} \wedge T^{k-1}_{k,j})$
- Floyd-Warshall kimenetből megadható: $T[i, j] = D[i, j] < \infty$
  (de FW-nek van előfeltétele és gyakorlatban nem annyira hatékony)

```
TransitiveClosure(A/1, T/1 : B[n, n])
    i := 1 to n
        j := 1 to n
            T[i, j] := A[i, j]
        T[i, i] := 1
    k := 1 to n
        i := 1 to n
            j := 1 to n
                T[i, j] := T[i, j] ∨ (T[i, k] ∧ T[k, j])
```

8.2. Visszavezetés szélességi keresésre

- A kimeneti mátrix i-edik sora a BFS $s = i$ kimenetéből kiszámolható:
  $\forall j : T[i, j] \Leftrightarrow \pi(j) \neq \varnothing$ (vagy $color(j) \neq white$ vagy $d(j) \neq \infty$)
- BFS n-szeri lefuttatása: $MT \in \Theta(n * (n + m))$
  - Ritka gráfon aszimptotikusan jobb: $\Theta(n^2)$
  - Sűrű gráfon egyenlő ($\Theta(n^3)$), gyakorlatban lassabb