Algoritmuselmélet

Kruskal implementáció, Floyd algoritmus

Katona Gyula Y. / Vizer Máté

Számítástudományi és Információelméleti Tanszék
Budapesti Műszaki és Gazdaságtudományi Egyetem

Minimális költségű feszítőfák

Most egyszerű, irányítatlan gráfokkal foglalkozunk.

Definíció (minimális költségű feszítőfa)

Legyen $G = (V, E)$ egy összefüggő gráf. A G gráf egy körmentes összefüggő $F = (V, E')$ részgráfja a gráf egy feszítőfája. Legyen továbbá az éleken értelmezve egy $c : E \to \mathbb{R}$ súlyfüggvény. Ekkor a G gráf egy F feszítőfája minimális költségű, ha költsége (a benne szereplő élek súlyainak összege) minimális G összes feszítőfáját tekintve.

Probléma

Adott egy $G = (V, E)$ összefüggő irányítatlan gráf, és az élein értelmezett $c : E \to \mathbb{R}$ súlyfüggvény. Határozzuk meg a G egy minimális költségű feszítőfáját.

Például: villamosvezetékek kiépítése.

Kruskal algoritmusa

A $V(G)$ csúcshalmazon egy F körmentes részgráfot építünk élek hozzávételével:

- Kezdetben $E(F) = \emptyset$.
- A még nem vizsgált élek közül kiválasztjuk a legkisebb súlyút (pontosabban az egyik ilyet).
- Ha ez nem alkot kört a már F-beli élekkel, akkor hozzávesszük F-hez. (Ha kört alkot nem vesszük hozzá, eldobjuk.)
- Ha F-nek már $n-1$ éle van, akkor véget ér az algoritmus, F egy minimális súlyú feszítőfa.

F mindvégig egy erdőt fog alkotni, azaz minden összefüggő komponense egy fa. Amikor új élet veszünk hozzá, az két komponenst köt össze.

Tétel (BSZ2-n volt)

A Kruskal-algoritmus eredményeként végül F a G gráf egy minimális költségű feszítőfájának éleit tartalmazza.

A Kruskal algoritmus implementációi

Jobban járunk, ha a gráf éllistával van megadva, csak ezt az esetet vizsgáljuk.

Minden körben kétféle feladat van:

1. Melyik a legkisebb súlyú még nem vizsgált él?
2. Ez kört alkot-e F eddigi éleivel?

A Kruskal algoritmus implementációi

Megjegyzés

Egyszerű gráf esetén $O(\log m) \subseteq O(\log n)$.

Bizonyítás.

$m \leq \frac{n(n-1)}{2} \leq n^2$. Így $\log m \leq \log n^2 = 2 \log n$. □

A Kruskal algoritmus implementációi

A legkisebb súlyú, még nem vizsgált él kiválasztása:

- 1. módszer: Először rendezzük az éleket pl. összefésüléses rendezéssel $O(m \log m)$ lépésben, így a következő legkisebb kiválasztása $O(1)$ lépés. Ezt legfeljebb m alkalommal kell megtenni. Lépésszám:
  $O(m \log m) + O(m) \subseteq O(m \log m) \subseteq O(m \log n)$.
- 2. módszer: Építsünk kupacot az élekből $O(m)$ lépésben, így a következő kiválasztása $O(\log m)$. Ezt legfeljebb m alkalommal kell megtenni. Lépésszám: $O(m) + O(m \log m) \subseteq O(m \log n)$.

Megjegyzés: Igaz, hogy a minimális feszítőfába végül $n - 1$ él fog bekerülni, de egy él vizsgálata után nem kerül be, ha kört alkot az eddigiekkel. Ezért lehet, hogy majdnem minden élet meg kell vizsgálni.

A Kruskal algoritmus implementációi

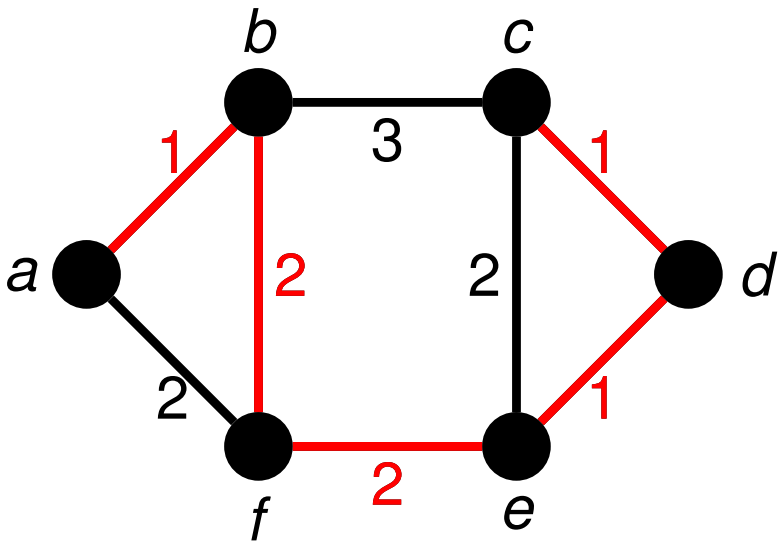
Kört alkot-e a következő él F eddigi éleivel?

Ötlet: Tartsuk nyilván F komponenseit, azaz minden v csúcsra jegyezzük fel, hogy melyik komponensben van. A komponens „neve” legyen a „legkisebb sorszámú” csúcsa. (Az éllistás megadásban a csúcsok fel vannak sorolva egy tömbben, e szerint kapnak sorszámot.) Ezt a KOMPONENS[v] tömbben tároljuk.

1. Kört alkot az (u, v) él F éleivel?
   Ha KOMPONENS[u] $\neq$ KOMPONENS[v], akkor nem, különben igen.
2. Ha nem alkot kört, akkor hozzávesszük F-hez. Ezzel viszont módosulnak a komponensek.
3. Ha KOMPONENS[u] $= x$ sorszáma kisebb, mint KOMPONENS[v] $= y$ sorszáma, akkor végigmegyek a KOMPONENS tömbön és minden y-t x-re cserélek.
4. Az algoritmus végén a KOMPONENS tömb minden eleme a legkisebb sorszámú csúcs lesz.

Példa

A pontok sorrendje legyen a, b, c, d, e, f.



a	b	c	d	e	f	új él
a	b	c	d	e	f	$\{a, b\}$
a	a	c	d	e	f	$\{c, d\}$
a	a	c	c	e	f	$\{d, e\}$
a	a	c	c	c	f	$\{b, f\}$
a	a	c	c	c	a	$\{f, e\}$
a	a	a	a	a	a	

A Kruskal lépésszáma

1. KOMPONENS[u] $\neq$ KOMPONENS[v] igaz-e? Lépésszám: $O(1)$
2. Ha nem alkot kört, akkor hozzávesszük F-hez. Lépésszám: $O(1)$.
3. KOMPONENS tömb frissítése. Lépésszám: $O(n)$.

Az ellenőrzést m-szer kell elvégezni, de mivel csak $n-1$ élet veszünk hozzá F-hez, így a frissítést csak $n-1$-szer:
Lépésszám: $O(m+n^2) \subseteq O(n^2)$.

A teljes algoritmus lépésszáma a kezdeti rendezéssel együtt:
$O(m \log n) + O(n^2)$.

Ha $m < n^2/\log n$, akkor a legtöbb idő a KOMPONENS tömb frissítése.
Ezt egy alkalmas, új adatstukatúrával felgyorsíthatjuk.

Az UNIÓ-HOLVAN adatszerkezet

Legyen adott egy véges S halmaz. (Mi a ponthalmazra alkalmazzuk.)
Ennek egy partícióját szeretnénk tárolni $\rightarrow U_1, \ldots, U_k \subseteq S$.
(A komponensek ponthalmazai.)

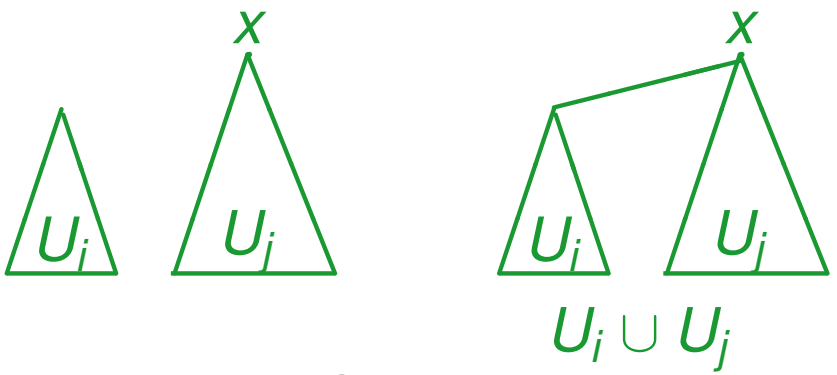
- Adott egy n elemű S halmaz, és ennek bizonyos $U_1, \ldots, U_m$ részhalmazai, melyekre $U_i \cap U_j = \emptyset$ $(i \neq j)$ és $U_1 \cup \ldots \cup U_m = S$ (vagyis az U_j részhalmazok S egy partícióját adják).
- *Műveletek:*
  UNIÓ$(U_i, U_j) = (\{U_1, \ldots, U_m\} \cup \{U_i \cup U_j\}) \setminus \{U_i, U_j\}$
  (az U_i, U_j halmazokat $U_i \cup U_j$ helyettesíti).
  HOLVAN(v) eredménye (itt $v \in S$) annak az U_i halmaznak a neve, amelynek v eleme.

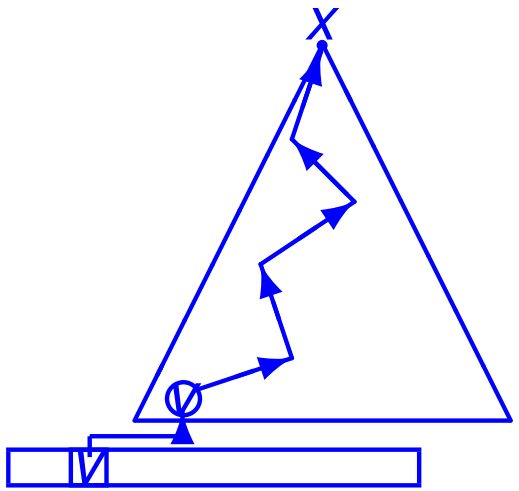
Kruskalban:
Annak megvizsgálása, hogy bevegyük-e az (u, v) élet:
Ha HOLVAN$(u) \neq$ HOLVAN(v), akkor bevesszük, különben nem.
Ha bevesszük $\Longrightarrow$ UNIÓ(HOLVAN(u), HOLVAN(v))

Implementáció fákkal

$U_j \rightarrow$ gyökeres, felfelé irányított fa
U_j elemeit a fa csúcsaiban tároljuk, egy szülőmutatóval.
Egy részhalmaz *neve* legyen az őt ábrázoló fa gyökere. A gyökérben nyilvántartjuk még a fa méretét (azaz a csúcsok számát) is.

- **UNIÓ:** $U_i \cup U_j$ fáját a következőképpen készítjük el: Tegyük fel, hogy $|U_i| \leq |U_j|$. Ekkor az U_j fa x gyökeréhez gyermekként hozzákapcsoljuk U_i gyökerét.





- **HOLVAN:** A $v \in S$ elemet tartalmazó részhalmaz nevét, azaz a megfelelő fa gyökerét a szülőkhöz menő mutatók végigkövetésével találhatjuk meg.

Az UNIÓ hívásakor az U_i és U_j halmazok a gyökerükkel adottak
$\Longrightarrow$ költség: $O(1)$
Ha egy v csúcs új gyökér alá kerül, akkor egy szinttel lesz távolabb a gyökértől, míg az új fájának a mérete legalább az eredeti duplájára változik.
$\Longrightarrow$ Egy csúcs legfeljebb $\log_2 n$-szer kerülhet új gyökér alá (különben a fa csúcsszáma nagyobb lenne, mint n).
$\Longrightarrow$ szintszám $\leq \log_2 n$.
$\Longrightarrow$ HOLVAN költsége $O(\log n)$.

Tétel

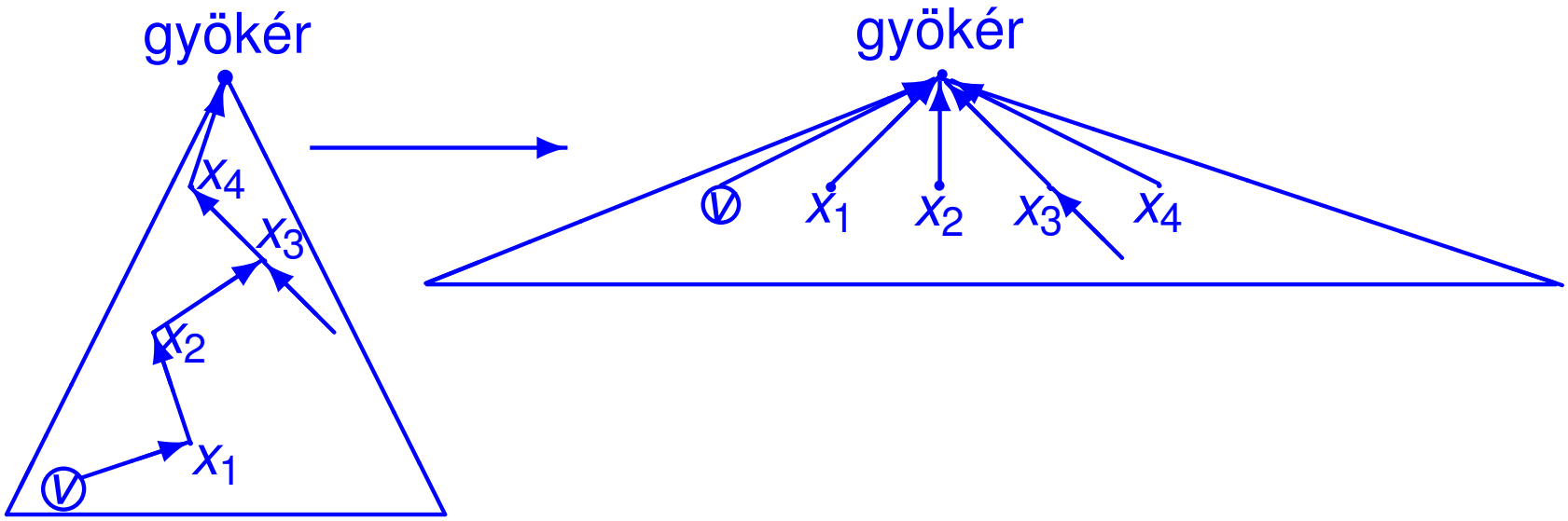
A Kruskal-algoritmus költsége $O(m \log n)$. □

Bizonyítás.

Először az élek rendezése $O(m \log n)$ lépés. Utána $2m$ HOLVAN, és $n-1$ UNIÓ műveletet jelent. Ezek időigénye
$O(m \log n) + O(m \log n) + O(n) \subseteq O(m \log n)$. □

A HOLVAN gyorsítása: útösszenyomás

Egy pontról többször is megnézzük HOLVAN, mindig $\log n$ lépés.
$\Longrightarrow$ Az első alkalommal minden ősét kössük közvetlenül a gyökérbe.



Animáció: Unió-Holvan útösszenyomással

Tétel

Legyen $|S| = n$, és tegyük fel, hogy kezdetben mindegyik U_j egyelemű. Ha egy olyan utasítássorozatot hajtunk végre útösszenyomással, melyben $n-1$ UNIÓ és $m \geq n-1$ HOLVAN szerepel, akkor ennek az időigénye $O(m\alpha(m))$.

A korlátban szereplő $\alpha : \mathbb{Z}^+ \to \mathbb{Z}^+$ függvény az *inverz Ackermann-függvény*.

Ackermann-függvény (kb.):

$$A[0] = 1+1+1 = 3 \cdot 1, \quad A[1] = 2 \cdot 2 \cdot 2 = 2^3 = 2 \uparrow 3$$

$$A[2] = 2 \uparrow (2 \uparrow 2) = 2^{2^2} = 2 \uparrow\uparrow 3 = 16$$

$$A[3] = 2 \uparrow\uparrow (2 \uparrow\uparrow 2) = 2^{2^{2^2}} = 2 \uparrow\uparrow\uparrow 3 = 65536$$

$$A[4] = 2 \uparrow\uparrow\uparrow (2 \uparrow\uparrow\uparrow 2) = 2 \uparrow\uparrow\uparrow 4 = \underbrace{2^{2^{\cdots^2}}}_{\text{65536 db kitevő}} = 2 \uparrow^4 3$$

$$A[n] = 2 \uparrow^n 3$$

Tétel

Legyen $|S| = n$, és tegyük fel, hogy kezdetben mindegyik U_j egyelemű. Ha egy olyan utasítássorozatot hajtunk végre útösszenyomással, melyben $n-1$ UNIÓ és $m \geq n-1$ HOLVAN szerepel, akkor ennek az időigénye $O(m\alpha(m))$.

$\alpha(m)$ ennek az inverz függvénye.
$\alpha(m)$ a végtelenhez tart, ha $m \to \infty$, de nagyon lassan, sokkal lassabban mint a logaritmus k-szori önmagába helyettesítésével adódó $\log\log\cdots\log m$ függvény ($k \in \mathbb{Z}^+$ tetszőleges).
Pl. $\alpha(m) \leq 4$, ha $m < 2^{65536}$
A normális méretű feladatoknál tehát $\alpha(m)$ állandónak (≤ 4) tekinthető.

Tétel

Ha az élek rendezésével kapcsolatos teendők $O(e)$ időben megoldhatók, akkor a Kruskal-algoritmus $O(e\alpha(e))$ időben megvalósítható.

Manipuláció a súlyokkal $\Longrightarrow$ Yao (1975): $O(e \log\log n)$
Véletlen módszerek $\Longrightarrow$ D. R. Karger, P. N. Klein, R. E. Tarjan, (1994): várhatóan $O(e)$

Online algoritmus: Vegyük hozzá az eddigi erdőhöz a következő élet. Ha kör keletkezik, akkor hagyjuk el a kör legnagyobb súlyú élét. Így a kiválasztott élek minden új él hozzávétele után az eddigi legkisebb súlyú körmentes részgráfot alkotják.

Floyd módszere

Feladat

Miként lehet egy irányított gráfban az összes pontpár távolságát meghatározni?

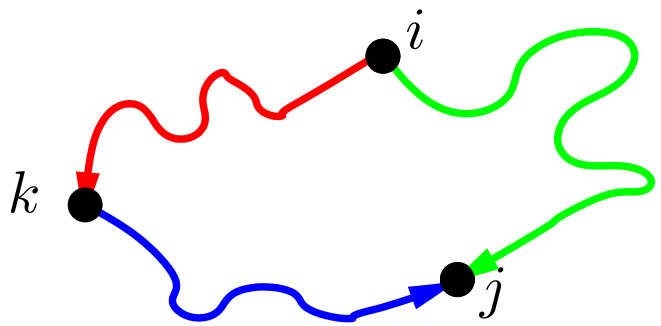
≥ 0 élsúlyok $\Longrightarrow$ ha a Dijkstra-algoritmust minden csúcsra mint forrásra lefuttatjuk $\Longrightarrow$ $nO(n^2) = O(n^3)$
Negatív súlyokkal $\Longrightarrow$ Bellman-Ford-algoritmus minden csúcsra mint forrásra $\Longrightarrow$ $nO(n^3) = O(n^4)$
Van olyan, ami nem lassabb és működik negatív élsúlyokra is, ha nincs negatív összsúlyú kör.

Feladat

Adott egy $G = (V, E)$ irányított gráf, és egy $c : E \to \mathbb{R}$ súlyfüggvény úgy, hogy a gráfban nincs negatív összsúlyú irányított kör. Határozzuk meg az összes $v, w \in V$ rendezett pontpárra a $d(v, w)$ távolságot.

Floyd módszere

A G gráf a C adjacencia-mátrixával adott.
Egy szintén $n \times n$-es F mátrixot fogunk használni a számításhoz.
Kezdetben $F_0[i,j] := C[i,j]$.
$\Longrightarrow$ ciklus $\Longrightarrow$ k-adik lefutása után $F_k[i,j]$ azon $i \rightsquigarrow j$ utak legrövidebbjeinek a hosszát tartalmazza, amelyek közbülső pontjai k-nál nem nagyobb sorszámúak
Az új $F_k[i,j]$ értékeket kiszámíthatjuk ha ismerjük $F_{k-1}[i,j]$-t $\forall i,j$-re
Egy legrövidebb $i \rightsquigarrow j$ út, melyen a közbülső pontok sorszáma legfeljebb k, vagy tartalmazza a k csúcsot vagy nem.
— igen $\Longrightarrow$ $F_k[i,j] := F_{k-1}[i,k] + F_{k-1}[k,j]$



— nem $\Longrightarrow$ $F_k[i,j] = F_{k-1}[i,j]$

Floyd algoritmusa

(1) for $i := 1$ **to** n **do**
  for $j := 1$ **to** n **do**
    $F[i,j] := C[i,j]$
(2) for $k := 1$ **to** n **do**
  for $i := 1$ **to** n **do**
    for $j := 1$ **to** n **do**
      $F[i,j] := \min\{F[i,j], F[i,k] + F[k,j]\}$

Tétel

$F[i,j]$ a (2)-beli iteráció k-adik menete után azon legrövidebb $i \rightsquigarrow j$ utak hosszát tartalmazza, amelyek belső csúcsai $1, 2, \ldots, k$ közül valók.

$k = n \implies F[i,j] = d(i,j)$

Lépésszám: n-szer megyünk végig a táblázaton, minden helyen $O(1)$ lépés $\implies O(n^3)$

Animáció: Floyd algoritmus

A legrövidebb utak nyomon-követése

Menet közben karbantartunk egy $n \times n$-es P tömböt.
Kezdetben $P[i,j] := 0$.
Ha egy $F[i,j]$ értéket megváltoztatunk, mert találtunk egy k-n átmenő rövidebb utat, akkor $P[i,j] := k$.
$\Longrightarrow$ Végül $P[i,j]$ egy legrövidebb $i \rightsquigarrow j$ út „középső" csúcsát fogja tartalmazni.
$i \rightsquigarrow j$ út összeállítása rekurzív $\Longrightarrow$

```
procedure legrövidebb út(i, j:csúcs);
var k:csúcs;
begin
        k := P[i, j];
        if k = 0 then return;
        legrövidebb út(i, k);
        kiír(k);
        legrövidebb út(k, j)
end;
```

Tranzitív lezárás

Bemenet: $G = (V, E)$ irányított gráf.

Csak arra vagyunk kíváncsiak, hogy mely pontok között vezet irányított út. Floyd $\Longrightarrow$ ha a végén $F[i, j] \neq \infty$, akkor van út, különben nincs.

Kicsit egyszerűbb korábbi algoritmus: S. Warshall

Definíció (tranzitív lezárt)

Legyen $G = (V, E)$ egy irányított gráf, A az adjacencia-mátrixa. Legyen továbbá T a következő $n \times n$-es mátrix:

$$T[i, j] = \begin{cases} 1 & \text{ha } i\text{-ből } j \text{ elérhető irányított úttal;} \\ 0 & \text{különben.} \end{cases}$$

Ekkor a T mátrixot – illetve az általa meghatározott gráfot – az A mátrix – illetve az általa meghatározott G gráf – tranzitív lezártjának hívjuk.

Feladat

Adott a (Boole-mátrixként értelmezett) A adjacencia-mátrixával a $G = (V, E)$ irányított gráf. Adjuk meg a G tranzitív lezártját.

Warhall algoritmus

(1) ciklusban a kezdőértékek beállítása helyett

$$T[i,j] := \begin{cases} 1 & \text{ha } i = j \text{ vagy } A[i,j] = 1, \\ 0 & \text{különben.} \end{cases}$$

A (2) ciklusban F értékeinek változtatása helyett (ugyanazt megfogalmazva logikai műveletekkel)

$$(+) \qquad T[i,j] := T[i,j] \vee (T[i,k] \wedge T[k,j]).$$

Bizonyítás ugyanúgy.
Lépésszám: $O(n^3)$

Alkalmazás Floyd módszerére

A súlyozott élű G irányított gráf v csúcsára legyen

$$e(v) = \max\{d(w, v) : \quad w \in V\}.$$

A v csúcs *centruma* G-nek, ha $e(v)$ minimális az összes $v \in V$ között.

Feladat

Keressük meg a G gráf centrumát.

Algoritmus centrum keresésére:

(1) Először Floyd módszerével kiszámítjuk a G-beli pontpárok közötti távolságokat. $\Longrightarrow$ $O(n^3)$ művelet

(2) Az előző lépésben kapott F mátrix minden oszlopában meghatározzuk a maximális értéket ($\Longrightarrow$ $e(v)$). $\Longrightarrow$ n-szer keresünk maximumot n elem közül $\Longrightarrow$ összesen $O(n^2)$.

(3) Végül megkeressük az n darab $e(v)$ érték minimumát. $\Longrightarrow$ $O(n)$.