

Algoritmuselmélet 8. előadás

Katona Gyula Y.
Budapesti Műszaki és Gazdaságtudományi Egyetem
Számítástudományi Tsz.
I. B. 137/b
kiskat@cs.bme.hu

2002 Március 12.

Dijkstra animációk

Java animáció: Dijkstra-algoritmus, másík, harmadik

Dijkstra algoritmusa éllistával

Ha kevés él van \implies gráfot éllistával tároljuk.

$V \setminus \text{KÉSZ}$ csúcsait *kupacba rendezve* tartjuk a $D[]$ érték szerint.

Kupacépítés $O(n)$ költség, (2) ciklusban minimumkeresést $O(\log n)$ költségű

MINTÖR

A $D[]$ értékek újraszámolását és a kupac-tulajdonság helyreállítását csak a választott csúcs szomszédaira kell elvégezni.

Minden csúcsot pontosan egyszer választunk ki, és a szomszédok számának összege e . \implies **Összidőigény $O((n + e) \log n)$.**

Java animáció: Dijkstra-algoritmus

Sűrű gráfokra $\implies d$ -kupac.

$\implies O(n + nd \log_d n + e \log_d n)$

Ha $n^{1,5} \leq e \leq n^2$ és legyen $d = \lceil e/n \rceil \implies d \geq \sqrt{n} \implies \log_d n \leq 2$.

\implies

$O(n + nd \log_d n + e \log_d n) = O(n + nd + e) = O(n + n \cdot e/n + e) = O(e)$.

Dijkstra irányítatlan gráfokra is működik.

A legrövidebb utak nyomon-követése

Minden pontra tárolunk és karbantartunk egy $P[x]$ csúcsot is, ami megadja egy az eddig ismert hozzá vezető legrövidebb úton az utolsó előtti csúcsot.

Kezdetben $P[v] := s$ minden $v \in V$ -re.

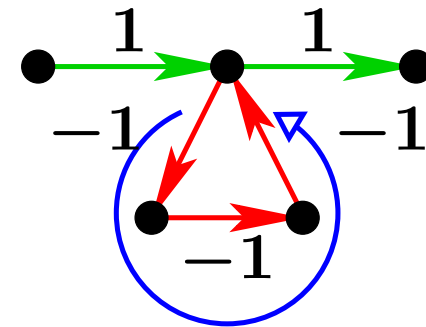
\implies (3) ciklus belsejében, ha egy külső w csúcs $D[w]$ értékét megváltoztatjuk, akkor $P[w] := x$.

Lépésszám: $O(n^2)$

A Bellman—Ford-módszer

Feladat. Egy pontból induló legrövidebb utak (hosszának) meghatározására, ha *bizonyos élsúlyok negatívak*. De feltesszük, hogy G *nem tartalmaz negatív összhosszúságú irányított kört*.

Ha van negatív összhosszúságú irányított kör, akkor a legrövidebb út ∞ .



Legyen a $G = (V, E)$ súlyozott élű irányított gráf a C adjacencia-mátrixával adott az i, j helyzetű elem a $c(i, j)$ élsúly, ha $i \rightarrow j$ éle G -nek, a többi elem pedig ∞).

Legyen $V = \{1, 2, \dots, n\}$ és $s = 1 \iff s$ -ből induló utakat keressük

Módszer \implies egy $T[1 : n - 1, 1 : n]$ táblázat sorról sorra haladó kitöltése.

(*) $T[i, j] =$ a legrövidebb olyan $1 \rightsquigarrow j$ irányított utak hossza, melyek legfeljebb i élből állnak.

$\implies T[n - 1, j]$ a legrövidebb $1 \rightsquigarrow j$ utak hossz

A $T[1, j]$ sor kitöltése $\implies T[1, j] = C[1, j]$

Tegyük fel ezután, hogy az i -edik sort már kitöltöttük

$\implies T[i, 1], T[i, 2], \dots, T[i, n]$ értékekre (*) igaz. \implies

(**) $T[i + 1, j] := \min\{T[i, j], \min_{k \neq j} \{T[i, k] + C[k, j]\}\}$

\Leftarrow Ugyanis egy legfeljebb $i + 1$ élből álló $\pi = 1 \rightsquigarrow j$ út kétféle lehet:

(a) Az útnak kevesebb, mint $i + 1$ éle van. Ekkor ennek a hossza szerepel $T[i, j]$ -ben.

(b) Az út éppen $i + 1$ élből áll. Legyen l a π út utolsó előtti pontja. Ekkor a π út $1 \rightsquigarrow l$ szakasza i élből áll, és minimális hosszúságú a legfeljebb i élű $1 \rightsquigarrow l$ utak között $\implies \pi$ hossza $T[i, l] + C[l, j]$.

Lépésszám: Egy érték (**) szerinti számítása $n - 1$ összeadás és ugyanennyi összehasonlítás (minimumkeresés n elem közül)

$\implies O(n^3)$ műveletet

Java animáció: Bellman-Ford algoritmus

Floyd módszere

Feladat. Miként lehet egy irányított gráfban az összes pontpár távolságát meghatározni?

≥ 0 élsúlyok \implies ha a Dijkstra-algoritmust minden csúcsra mint forrásra lefuttatjuk $\implies nO(n^2) = O(n^3)$

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 $c : E \rightarrow \mathbb{R}$ 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.

A G gráf a C adjacencia-mátrixával adott.

Egy szintén $n \times n$ -es F mátrixot fogunk használni a számításhoz.

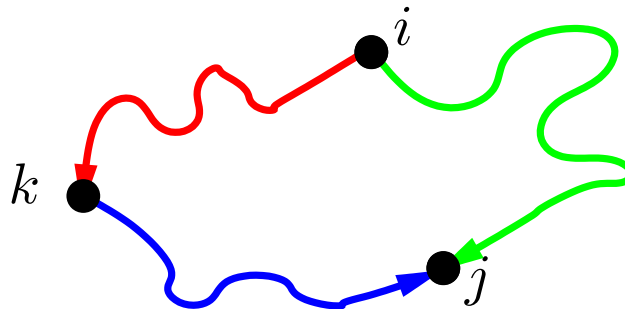
Kezdetben $F[i, j] := C[i, j]$.

\implies ciklus \implies k -adik lefutása után $F[i, j]$ 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 csúcst vagy nem.

— igen $\implies F[i, j] := F[i, k] + F[k, j]$



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

FLOYD algoritmus

(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 $1, 2, \dots, k$ közül valók.

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

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

Java 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 $P[i, j] := k$.

\implies Végül $P[i, j]$ egy legrövidebb $i \rightsquigarrow j$ út „középső” csúcsát fogja tartalmazni.

$i \rightsquigarrow j$ út összeállítása rekurzív \implies

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 \implies 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)

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

Bizonyítás ugyanúgy.

Lépésszám: $O(n^3)$

Alkalmazás Floyd módszerére

A súlyozott élű G irányított gráf v csúcsára legyen

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

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. $\implies O(n^3)$ művelet
- (2) Az előző lépésben kapott F mátrix minden oszlopában meghatározzuk a maximális értéket ($\implies e(v)$). $\implies n$ -szer keresünk maximumot n elem közül \implies összesen $O(n^2)$.
- (3) Végül megkeressük az n darab $e(v)$ érték minimumát. $\implies O(n)$.

Mélységi bejárás

Gráf bejárás \implies minden pontot felsorolunk, bejárunk

Mélységi bejárás (depth-first-search, DFS), Szélességi bejárás (breadth-first-search, BFS)

Pl. lámpagyújtogató algoritmus

Mélységi keresés

Mohó menetelés, addig megyünk előre, amíg tudunk, csak aztán fordulunk vissza.

Java animáció: Mélységi keresés

$G = (V, E)$ egy irányított gráf, ahol $V = \{1, \dots, n\}$.

$L[v]$ a v csúcs éllistája ($1 \leq v \leq n$).

bejárva[1 : n] logikai vektor \implies jártunk-e már ott

Mélységi keresés algoritmus

procedure bejár (* elvégzi a G irányított gráf mélységi bejárását *)

begin

for $v := 1$ **to** n **do**

bejárva[v] := *hamis*;

for $v := 1$ **to** n **do**

if bejárva[v] = *hamis* **then**

mb(v)

end

procedure mb (v : csúcs)

var

w : csúcs;

begin

(1) bejárva[v] := *igaz*; (* meggyújtjuk a lámpát *)

for minden $L[v]$ -beli w csúcsra **do**

(2) **if** bejárva[w] = *hamis* **then**

mb(w) (* megyünk a következő még sötét lámpához *)

end

Lépésszám: $O(n + e)$ Mélységi számok és befejezési számok

Definíció. *[mélységi számozás]* A G irányított gráf csúcsainak egy **mélységi számozása** a gráf v csúcsához azt a sorszámot rendeli, mely megadja, hogy az **mb** eljárás (1) sorában a csúcsok közül hányadikként állítottuk bejárva[v] értékét igaz-ra. A v csúcs mélységi számát **mszám[v]** jelöli.

Definíció. *[befejezési számozás]* A G irányított gráf csúcsainak egy **befejezési számozása** a gráf v csúcsához azt a sorszámot rendeli, mely megadja, hogy a csúcsok közül hányadikként ért véget az $mb(v)$ hívás. A v csúcs befejezési számát **bszám[v]** jelöli.

Előző algoritmus kis módosítással:

procedure

begin $msz := 0$; $bsz := 0$;

for $v := 1$ **to** n **do**

$bejárva[v] := hamis$; $mszám[v] := 0$; $bszám[v] := 0$;

for $v := 1$ **to** n **do**

if $bejárva[v] = hamis$ **then**

$mb(v)$

end

procedure $mb(v: csúcs)$

var

$w: csúcs$;

begin

(1) $bejárva[v] := igaz$; $msz := msz + 1$; $mszám[v] := msz$;

for minden $L[v]$ -beli w csúcsra **do**

(2) **if** $bejárva[w] = hamis$ **then**

$mb(w)$;

$bsz := bsz + 1$; $bszám[v] := bsz$;

end

Java animáció: Mélységi és befejezési számok

Mélységi feszítő erdő

Definíció. [*faél*] A $G = (V, E)$ irányított gráf $v \rightarrow w$ éle **faél** (az adott mélységi bejárásra vonatkozóan), ha megvizsgálásakor a (2) sorban a ($bejárva[w] = hamis$) feltétel teljesül.

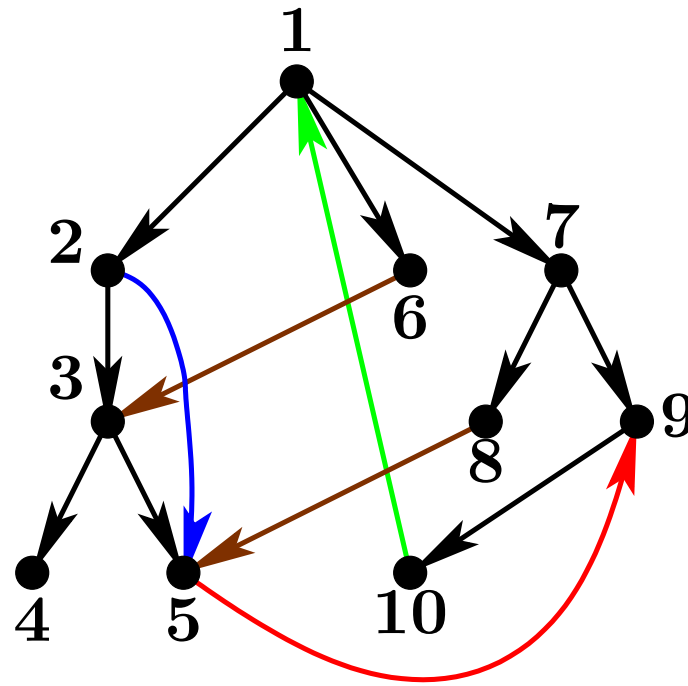
Jelölje T azt a gráfot, amelynek csúcshalmaza V , élei pedig a faélek.

\implies ez erdő

Definíció. [*mélységi feszítő erdő, feszítőfa*] Az előbb meghatározott T gráfot a G gráf egy **mélységi feszítő erdejének** nevezzük. Ha T csak egy komponensből áll, akkor **mélységi feszítőfáról** beszélünk.

Definíció. [*élek osztályozása*] Tekintsük a G irányított gráf egy mélységi bejárását és a kapott T mélységi feszítő erdőt. (Ezen bejárás szerint) G egy $x \rightarrow y$ éle

- faél,** ha $x \rightarrow y$ éle T -nek;
- előreél,** ha $x \rightarrow y$ nem faél, y leszármazottja x -nek T -ben, és $x \neq y$;
- visszaél,** ha x leszármazottja y -nak T -ben (a hurokél is ilyen);
- keresztél,** ha x és y nem leszármazottai egymásnak.



faél
előreél
visszaél
keresztél
ilyen nincs

faél, előre él: **kisebb mélységi számúból nagyobb mélységi számúba mutat**
visszaél, keresztél: **nagyobb mélységi számúból kisebb mélységi számúba mutat**

visszaél: **kisebb befejezési számúból nagyobb befejezési számúba mutat**

$x \rightarrow y$ egy	ha az él vizsgálatakor
- faél	$mszám[y] = 0$
- visszaél	$mszám[y] \leq mszám[x]$ és $bszám[y] = 0$
- előreél	$mszám[y] > mszám[x]$
- keresztél	$mszám[y] < mszám[x]$ és $bszám[y] > 0$.

Tétel. *A G irányított gráf mélységi bejárása – beleértve a mélységi, a befejezési számozást és az élek osztályozását is – $O(n + e)$ lépésben megtehető.*