

Hálózati folyamatok és alkalmazásai

ELTE-TTK, Operációkutatási Tanszék

Jüttner Alpár

alpar@cs.elte.hu

ELTE-TTK, Operációkutatási tanszék

- 1 Jelölések, alapvető ismeretek
 - Algoritmusok futásideje, polinomiális algoritmusok
 - Gráfok, gráf-reprezentációk
- 2 Gráf keresések
- 3 Gráfok reprezentálása
- 4 Legrövidebb utak
 - Általános súlyfüggvény mellett
- 5 Törtoptimalizálás
- 6 A szintező algoritmus
- 7 Megengedett folyamatok

Algoritmusok futásideje

Definíció

Algoritmus futásidején vagy

- egy adott bemeneten megtett lépések számát

vagy

- a megtett lépések számának a bemenet méretétől való függését (átlagosan, **legrosszabb esetben**, gyakorlati példákon)

értjük.

- Mi a „bemenet mérete?”
- Mi egy „lépés”?

Definíció ($O(\cdot)$ jelölés)

- $f = O(g)$, ha $\exists c \in \mathbb{R}_+, n_0 \in \mathbb{N}$, hogy $\forall n \geq n_0$ -ra $f(n) < cg(n)$
- $f = \Omega(g)$, ha $\exists c \in \mathbb{R}_+, n_0 \in \mathbb{N}$, hogy $\forall n \geq n_0$ -ra $f(n) > cg(n)$

Definíció

Egy algoritmus **polinomiális**, ha $\exists k$, hogy futásideje $O(n^k)$.

Algoritmusok futásideje

Definíció

Algoritmus futásidején vagy

- egy adott bemeneten megtett lépések számát

vagy

- a megtett lépések számának a bemenet méretétől való függését (átlagosan, **legrosszabb esetben**, gyakorlati példákon)

értjük.

- Mi a „bemenet mérete?”
- Mi egy „lépés”?

Definíció ($O(\cdot)$ jelölés)

- $f = O(g)$, ha $\exists c \in \mathbb{R}_+, n_0 \in \mathbb{N}$, hogy $\forall n \geq n_0$ -ra $f(n) < cg(n)$
- $f = \Omega(g)$, ha $\exists c \in \mathbb{R}_+, n_0 \in \mathbb{N}$, hogy $\forall n \geq n_0$ -ra $f(n) > cg(n)$

Definíció

Egy algoritmus **polinomiális**, ha $\exists k$, hogy futásideje $O(n^k)$.

Algoritmusok futásideje

Definíció

Algoritmus futásidején vagy

- egy adott bemeneten megtett lépések számát

vagy

- a megtett lépések számának a bemenet méretétől való függését (átlagosan, **legrosszabb esetben**, gyakorlati példákon)

értjük.

- Mi a „bemenet mérete?”
- Mi egy „lépés”?

Definíció ($O(\cdot)$ jelölés)

- $f = O(g)$, ha $\exists c \in \mathbb{R}_+, n_0 \in \mathbb{N}$, hogy $\forall n \geq n_0$ -ra $f(n) < cg(n)$
- $f = \Omega(g)$, ha $\exists c \in \mathbb{R}_+, n_0 \in \mathbb{N}$, hogy $\forall n \geq n_0$ -ra $f(n) > cg(n)$

Definíció

Egy algoritmus **polinomiális**, ha $\exists k$, hogy futásideje $O(n^k)$.

Dinamikus programozás

Az eredeti feladatot „nem túl sok” (polinomiális darab) egymásra épülő részfeladat kiszámítására bontjuk.

Példa (Binomiális együtthatók kiszámítása (csak összeadással))

- $\binom{n}{k} = \frac{n!}{k!(n-k)!}$
- $\binom{n}{k} = \begin{cases} 1, & \text{ha } k = 0 \text{ vagy } k = n \\ \binom{n-1}{k} + \binom{n-1}{k-1} & \text{egyébként} \end{cases}$

Házi feladat 1/1. Példa (Hátizsák feladat)

Van p darab tárgyunk amiknek súlya rendre w_1, w_2, \dots, w_p , az érték-k pedig c_1, c_2, \dots, c_p . A hátizsákunkban legfeljebb W súlyt bírunk el. E feltétel mellett szeretnénk a lehető legtöbb értéket magunkkal vinni. Tegyük fel, hogy a súlyok egész számok és W „nem túl nagy”.

$$\max \left\{ \sum_{i=1}^p c_i x_i : \sum_{i=1}^p w_i x_i \leq W \text{ és } \forall i\text{-re } x_i \in \{0, 1\} \right\} \quad (1)$$

Dinamikus programozás

Az eredeti feladatot „nem túl sok” (polinomiális darab) egymásra épülő részfeladat kiszámítására bontjuk.

Példa (Binomiális együtthatók kiszámítása (csak összeadással))

- $\binom{n}{k} = \frac{n!}{k!(n-k)!}$
- $\binom{n}{k} = \begin{cases} 1, & \text{ha } k = 0 \text{ vagy } k = n \\ \binom{n-1}{k} + \binom{n-1}{k-1} & \text{egyébként} \end{cases}$

Házi feladat 1/1. Példa (Hátizsák feladat)

Van p darab tárgyunk amiknek súlya rendre w_1, w_2, \dots, w_p , az érték-k pedig c_1, c_2, \dots, c_p . A hátizsákunkban legfeljebb W súlyt bírunk el. E feltétel mellett szeretnénk a lehető legtöbb értéket magunkkal vinni. Tegyük fel, hogy a súlyok egész számok és W „nem túl nagy”.

$$\max \left\{ \sum_{i=1}^p c_i x_i : \sum_{i=1}^p w_i x_i \leq W \text{ és } \forall i\text{-re } x_i \in \{0, 1\} \right\} \quad (1)$$

Dinamikus programozás

Az eredeti feladatot „nem túl sok” (polinomiális darab) egymásra épülő részfeladat kiszámítására bontjuk.

Példa (Binomiális együtthatók kiszámítása (csak összeadással))

- $\binom{n}{k} = \frac{n!}{k!(n-k)!}$
- $\binom{n}{k} = \begin{cases} 1, & \text{ha } k = 0 \text{ vagy } k = n \\ \binom{n-1}{k} + \binom{n-1}{k-1} & \text{egyébként} \end{cases}$

Házi feladat 1/1. Példa (Hátizsák feladat)

Van p darab tárgyunk amiknek súlya rendre w_1, w_2, \dots, w_p , az érték-k pedig c_1, c_2, \dots, c_p . A hátizsákunkban legfeljebb W súlyt bírunk el. E feltétel mellett szeretnénk a lehető legtöbb értéket magunkkal vinni. Tegyük fel, hogy a súlyok egész számok és W „nem túl nagy”.

$$\max \left\{ \sum_{i=1}^p c_i x_i : \sum_{i=1}^p w_i x_i \leq W \text{ és } \forall i\text{-re } x_i \in \{0, 1\} \right\} \quad (1)$$

Definíció (Írányított gráf)

- $G = (N, A)$, ahol N a **csúcsok** (véges) halmaza és $A \subseteq N \times N$ az **élhalmaz**
 - Jelölés: $n := |N|$ és $m := |A|$
- $a = (i, j) \in A$ élnek i a **farka** $[s(a)]$ és j a **feje** $[t(a)]$
- $\rho_G(v) = \rho(v) := \{(i, v) \in A\}$
- $\delta_G(v) = \delta(v) := \{(v, j) \in A\}$
- $G' = (N', A')$, **részgráf**, ha $N' \subseteq N$ és $A' \subseteq A$.
- $G' = (N', A')$ **fesztett részgráf**, ha $N' = N$ és $A' \subseteq A$.
- $G' = (N', A')$ **indukált részgráf**, ha $N' \subseteq N$ és $A' := \{(i, j) \in A \mid i, j \in N'\}$.

Ez a definíció nem engedi meg a párhuzamos éleket. Ettől függetlenül (szinte) minden működni fog párhuzamos éleket tartalmazó gráfokra is.

Definíció (Utak, Séták)

- $\{a_1, a_2, \dots, a_k\} \subseteq A$, **irányított séta**, ha $\forall i$ -re $t(a_i) = s(a_{i+1})$.
- Egy irányított séta **irányított út**, ha nem ismétlődik benne egyik csúcs sem.
- Egy irányított séta **irányított körséta**, ha $t(a_k) = s(a_1)$.
- Egy irányított körséta **irányított kör**, ha $\{a_1, a_2, \dots, a_{k-1}\}$ irányított út.
- **(irányítatlan) séta, (irányítatlan) út, (irányítatlan) körséta, (irányítatlan) kör**: mint az előzőek, de az élek mindkét irányban állhatnak.

Definíció (Összefüggőség)

A $G = (N, A)$ irányított gráf

- **összefüggő**, ha $\forall i, j \in N$ -re \exists egy $i \rightsquigarrow j$ út.
- **erősen összefüggő**, ha $\forall i, j \in N$ -re \exists egy $i \rightsquigarrow j$ irányított út.
- **DAG** (Directed Acyclic Graph), ha nem tartalmaz irányított kört.

- Csúcs-csúcs adjacencia mátrix

- $M \in \{0, 1\}^{N \times N}$, $m_{i,j} := \begin{cases} 1, & \text{ha } (i, j) \in A \\ 0, & \text{egyébként} \end{cases}$

- Csúcs-él incidencia mátrix

- $M \in \{0, 1\}^{N \times A}$, $m_{i,a} := \begin{cases} 1, & \text{ha } \exists j \in N : a = (i, j) \\ -1, & \text{ha } \exists j \in N : a = (j, i) \\ 0, & \text{egyébként} \end{cases}$

- Éllista reprezentáció (Adjacency list)

- Ki-csillag, be-csillag (Forward and Reverse Star) reprezentáció

Házi feladat 1/2.

Adott egy $G = (N, A)$ irányított gráf. Adjunk algoritmust, ami eldönti, hogy van-e páratlan hosszú irányított kör a gráfban.

- És ha páratlan helyett párost kérdezzük?

Házi feladat 1/3.

Adott egy $G = (N, A)$ DAG, $s \in N$. Jelölje $\alpha(i)$ a különböző $s \rightsquigarrow i$ utak számát G -ben. Adjunk $O(m)$ idejű algoritmust, ami kiszámolja $\alpha(i)$ -t minden $i \in N$ -re.



<http://lemon.cs.elte.hu>

- Nyílt forráskódú C++ library gráfokkal (hálózatokkal) kapcsolatos optimalizálási feladatok megoldására
- Cél: Olyan eszköztár létrehozása, amely egyszerre alkalmas kutatási feladatokban és ipari felhasználásra.
 - Nyílt forráskód, kereskedelmi felhasználást is biztosító licenc
 - Az eszközök elérhető leghatékonyabb implementációjára törekszünk (a leggyorsabb a piacon)
- Megbízható, kiterjedt implementáció
 - \approx 55000 erősen optimalizált kódsor
 - Windows, Unix támogatás különféle fordítókkal
- **Fejlesztő kerestetik!!!**

Gráf keresés

```
1: procedure SEARCH( $G, s$ )
2:    $\forall v \in N$ -re  $m(v) := 0$ 
3:    $m(s) := 1$ 
4:    $pred(s) := 0$ ;  $order(s) := 1$ 
5:    $next := 1$ 
6:    $LIST := \{s\}$ 
7:   while  $LIST \neq \emptyset$  do
8:     Kiválasztunk egy  $i \in LIST$  csúcst
9:     if  $\exists (i, j) \in \delta_G(i), m(j) = 0$  then
10:       $m(j) := 1$ 
11:       $pred(j) := i$ 
12:       $next := next + 1$ ;  $order(j) := next$ 
13:       $LIST := LIST \cup \{j\}$ 
14:     else
15:       $LIST := LIST \setminus \{i\}$ 
16:     end if
17:   end while
18: end procedure
```

▷ Minden csúcs jelöletlen (eléretlen)
▷ kivéve s

▷ Feldolgozandó csúcsok listája

▷ j -t megjelöljük, mint elértet

Gráf keresés

```
1: procedure SEARCH( $G, s$ )
2:    $\forall v \in N$ -re  $m(v) := 0$ 
3:    $m(s) := 1$ 
4:    $pred(s) := 0$ ;  $order(s) := 1$ 
5:    $next := 1$ 
6:    $LIST := \{s\}$ 
7:   while  $LIST \neq \emptyset$  do
8:     Kiválasztunk egy  $i \in LIST$  csúcsot
9:     if  $\exists (i, j) \in \delta_G(i), m(j) = 0$  then
10:       $m(j) := 1$ 
11:       $pred(j) := i$ 
12:       $next := next + 1$ ;  $order(j) := next$ 
13:       $LIST := LIST \cup \{j\}$ 
14:     else
15:       $LIST := LIST \setminus \{i\}$ 
16:     end if
17:   end while
18: end procedure
```

▷ Minden csúcs jelöletlen (eléretlen)
▷ kivéve s

▷ Feldolgozandó csúcsok listája

▷ j -t megjelöljük, mint elértet

Megjegyzés

Van szabadságunk abban, hogy melyik csúcsot válasszuk ki 8-nál.

Gráf keresés II.

Állítás

Futásidő: $O(m)$ (Ha a LIST-műveletek $O(1)$)

Állítás (házi feladat 1/4.)

$pred : N \rightarrow N$ egy s -gyökerű kifele irányított fát (ki-fenyőt) határoz meg, ami

- pontosan az s -ből irányított úton elérhető csúcsokat feshíti
- irányítatlan gráf esetén feshíti az s -et tartalmazó összefüggő komponenst.

Állítás

Ez a fa minden s -ből irányított úton elérhető t pontra meghatároz egy egyértelmű $s \rightsquigarrow t$ (irányított) utat.

Gráf keresés II.

Állítás

Futásidő: $O(m)$ (Ha a LIST-műveletek $O(1)$)

Állítás (házi feladat 1/4.)

$pred : N \rightarrow N$ egy s -gyökerű kifele irányított fát (ki-fenyőt) határoz meg, ami

- pontosan az s -ből irányított úton elérhető csúcsokat feshíti
- irányítatlan gráf esetén feshíti az s -et tartalmazó összefüggő komponenst.

Állítás

Ez a fa minden s -ből irányított úton elérhető t pontra meghatároz egy egyértelmű $s \rightsquigarrow t$ (irányított) utat.

Gráf keresés II.

Állítás

Futásidő: $O(m)$ (Ha a LIST-műveletek $O(1)$)

Állítás (házi feladat 1/4.)

$pred : N \rightarrow N$ egy s -gyökerű kifele irányított fát (ki-fenyőt) határoz meg, ami

- pontosan az s -ből irányított úton elérhető csúcsokat feszíti
- irányítatlan gráf esetén feszíti az s -et tartalmazó összefüggő komponenst.

Állítás

Ez a fa minden s -ből irányított úton elérhető t pontra meghatároz egy egyértelmű $s \rightsquigarrow t$ (irányított) utat.

Szélességi keresés (Breadth-First Search, BFS)

LIST legyen egy sor (queue)

Állítás

Minden t pontra a BFS-fenyő által meghatározott $s \rightsquigarrow t$ út egy legrövidebb (minimális élszámú) út G gráfban.

Házi feladat 1/5.

Bizonyítsuk be a fenti állítást.

Szélességi keresés (Breadth-First Search, BFS)

LIST legyen egy sor (queue)

Állítás

Minden t pontra a BFS-fenyő által meghatározott $s \rightsquigarrow t$ út egy legrövidebb (minimális élszámú) út G gráfban.

Házi feladat 1/5.

Bizonyítsuk be a fenti állítást.

Szélességi keresés (Breadth-First Search, BFS)

LIST legyen egy sor (queue)

Állítás

Minden t pontra a BFS-fenyő által meghatározott $s \rightsquigarrow t$ út egy legrövidebb (minimális élszámú) út G gráfban.

Házi feladat 1/5.

Bizonyítsuk be a fenti állítást.

Gráf-struktúra

- Élek

- m : int
- $sources$: int[M]
- $targets$: int[M]
- $next_out$: int[M]
- $next_in$: int[M]

- Csúcsok

- n : int
- $first_out$: int[N]
- $first_in$: int[N]

```
1: function ADDNODE
2:    $n \leftarrow n + 1$ 
3:    $first\_out[n] \leftarrow 0$ 
4:    $first\_in[n] \leftarrow 0$ 
5:   return  $n$ 
6: end function
```

```
1: function ADDEDGE( $i, j$ )
2:    $m \leftarrow m + 1$ 
3:    $sources[m] \leftarrow i$ 
4:    $targets[m] \leftarrow j$ 
5:    $next\_out[m] \leftarrow first\_out[i]$ 
6:    $next\_in[m] \leftarrow first\_in[j]$ 
7:    $first\_out[i] \leftarrow m$ 
8:    $first\_in[j] \leftarrow m$ 
9:   return  $m$ 
10: end function
```

Gráf-struktúra

- Élek

- m : int
- $sources$: int[M]
- $targets$: int[M]
- $next_out$: int[M]
- $next_in$: int[M]

- Csúcsok

- n : int
- $first_out$: int[N]
- $first_in$: int[N]

```
1: function ADDNODE
2:    $n \leftarrow n + 1$ 
3:    $first\_out[n] \leftarrow 0$ 
4:    $first\_in[n] \leftarrow 0$ 
5:   return  $n$ 
6: end function
```

```
1: function ADDEDGE( $i, j$ )
2:    $m \leftarrow m + 1$ 
3:    $sources[m] \leftarrow i$ 
4:    $targets[m] \leftarrow j$ 
5:    $next\_out[m] \leftarrow first\_out[i]$ 
6:    $next\_in[m] \leftarrow first\_in[j]$ 
7:    $first\_out[i] \leftarrow m$ 
8:    $first\_in[j] \leftarrow m$ 
9:   return  $m$ 
10: end function
```

Gráf-struktúra

- Élek

- m : int
- $sources$: int[M]
- $targets$: int[M]
- $next_out$: int[M]
- $next_in$: int[M]

- Csúcsok

- n : int
- $first_out$: int[N]
- $first_in$: int[N]

```
1: function ADDNODE
2:    $n \leftarrow n + 1$ 
3:    $first\_out[n] \leftarrow 0$ 
4:    $first\_in[n] \leftarrow 0$ 
5:   return  $n$ 
6: end function
```

```
1: function ADDEDGE( $i, j$ )
2:    $m \leftarrow m + 1$ 
3:    $sources[m] \leftarrow i$ 
4:    $targets[m] \leftarrow j$ 
5:    $next\_out[m] \leftarrow first\_out[i]$ 
6:    $next\_in[m] \leftarrow first\_in[j]$ 
7:    $first\_out[i] \leftarrow m$ 
8:    $first\_in[j] \leftarrow m$ 
9:   return  $m$ 
10: end function
```

```

1: procedure BFS(s)
2:   l : bool[m], pred : int[m]
3:    $\forall i \in \{1 \dots m\}$ -re l[i]  $\leftarrow$  false
4:   l[s]  $\leftarrow$  true
5:   pred[s]  $\leftarrow$  0
6:   LIST  $\leftarrow$  {s}
7:   while LIST  $\neq$   $\emptyset$  do
8:     Kiválasztjuk a i  $\in$  LIST csúcsot
9:     LIST  $\leftarrow$  LIST  $\setminus$  {i}
10:    e  $\leftarrow$  first_out[i]
11:    while e  $\neq$  0 do
12:      j  $\leftarrow$  targets[e]
13:      if l[j] = false then
14:        l[j]  $\leftarrow$  true
15:        pred[j]  $\leftarrow$  (i, e)
16:        LIST  $\leftarrow$  LIST  $\cup$  {j}
17:      end if
18:      e  $\leftarrow$  next_out[e]
19:    end while
20:  end while
21: end procedure

```

▷ Következő él


```

1: procedure BFS(s)
2:   l : bool[m], pred : int[m]
3:    $\forall i \in \{1 \dots m\}$ -re l[i]  $\leftarrow$  false
4:   l[s]  $\leftarrow$  true
5:   pred[s]  $\leftarrow$  0
6:   LIST : int[n], lh  $\leftarrow$  1, lt  $\leftarrow$  1
7:   LIST[lt]  $\leftarrow$  s, lt  $\leftarrow$  lt + 1
8:   while lh  $\neq$  lt do
9:     i  $\leftarrow$  LIST[lh]
10:    lh  $\leftarrow$  lh + 1
11:    e  $\leftarrow$  first_out[i]
12:    while e  $\neq$  0 do
13:      j  $\leftarrow$  targets[e]
14:      if l[j] = false then
15:        l[j]  $\leftarrow$  true
16:        pred[j]  $\leftarrow$  (i, e)
17:        LIST[lt]  $\leftarrow$  j, lt  $\leftarrow$  lt + 1
18:      end if
19:      e  $\leftarrow$  next_out[e]
20:    end while
21:  end while
22: end procedure

```

▷ Következő él

Mélységi keresés (Depth-First Search, DFS)

LIST legyen egy verem (stack)

Állítás

A DFS fenyőben

- *ha j „leszármazottja” i -nek (azaz van egy $i \rightsquigarrow j$ út a DFS fenyőben és $i \neq j$), akkor $\text{order}(j) > \text{order}(i)$.*
- *egy pont leszármazottjainak sorrendjei (order értékei) egy intervallumot alkotnak.*

Házi feladat 2/1.

Bizonyítsuk be a fenti állításokat.

Házi feladat 2/2.

Legyen G irányítatlan gráf, T a DFS által megtalált fenyő és $(k, l) \in A \setminus T$.
Ekkor vagy k leszármazottja l -nek vagy l leszármazottja k -nak T -ben.

Mélységi keresés (Depth-First Search, DFS)

LIST legyen egy verem (stack)

Állítás

A DFS fenyőben

- *ha j „leszármazottja” i -nek (azaz van egy $i \rightsquigarrow j$ út a DFS fenyőben és $i \neq j$), akkor $\text{order}(j) > \text{order}(i)$.*
- *egy pont leszármazottjainak sorrendjei (order értékei) egy intervallumot alkotnak.*

Házi feladat 2/1.

Bizonyítsuk be a fenti állításokat.

Házi feladat 2/2.

Legyen G irányítatlan gráf, T a DFS által megtalált fenyő és $(k, l) \in A \setminus T$.
Ekkor vagy k leszármazottja l -nek vagy l leszármazottja k -nak T -ben.

Mélységi keresés (Depth-First Search, DFS)

LIST legyen egy verem (stack)

Állítás

A DFS fenyőben

- *ha j „leszármazottja” i -nek (azaz van egy $i \rightsquigarrow j$ út a DFS fenyőben és $i \neq j$), akkor $\text{order}(j) > \text{order}(i)$.*
- *egy pont leszármazottjainak sorrendjei (order értékei) egy intervallumot alkotnak.*

Házi feladat 2/1.

Bizonyítsuk be a fenti állításokat.

Házi feladat 2/2.

Legyen **G irányítatlan gráf**, T a DFS által megtalált fenyő és $(k, l) \in A \setminus T$.
Ekkor vagy k leszármazottja l -nek vagy l leszármazottja k -nak T -ben.

Topologikus sorrend

Definíció

A $G = (N, A)$ gráf csúcsainak egy $order : N \rightarrow \mathbb{N}$ sorrendjét **topologikus sorrend**nek nevezzük, ha minden $(i, j) \in A$ élre $order(i) < order(j)$.

Tétel

Egy gráfnak pontosan akkor van topologikus sorrendje, ha nem tartalmaz irányított kört (DAG).

Algoritmus

```
1: next:=1
2: while  $G \neq \emptyset$  do
3:   Válasszunk ki egy  $i$  csúcsot, amibe nem megy be él ▷ Van ilyen?
4:    $order(i) := next$ ;  $next := next + 1$ 
5:   Töröljük ki  $G$ -ből  $i$ -t (és a belőle kiinduló éleket)
6: end while
```

Házi feladat 2/3.

Adjuk meg a fenti algoritmus egy hatékony ($O(m + n)$ futási idejű) megvalósítását.

Topologikus sorrend

Definíció

A $G = (N, A)$ gráf csúcsainak egy $order : N \rightarrow \mathbb{N}$ sorrendjét **topologikus sorrend**nek nevezzük, ha minden $(i, j) \in A$ élre $order(i) < order(j)$.

Tétel

Egy gráfnak pontosan akkor van topologikus sorrendje, ha nem tartalmaz irányított kört (DAG).

Algoritmus

```
1: next:=1
2: while  $G \neq \emptyset$  do
3:   Válasszunk ki egy  $i$  csúcsot, amibe nem megy be él ▷ Van ilyen?
4:    $order(i) := next$ ;  $next := next + 1$ 
5:   Töröljük ki  $G$ -ből  $i$ -t (és a belőle kiinduló éleket)
6: end while
```

Házi feladat 2/3.

Adjuk meg a fenti algoritmus egy hatékony ($O(m + n)$ futási idejű) megvalósítását.

Topologikus sorrend

Definíció

A $G = (N, A)$ gráf csúcsainak egy $order : N \rightarrow \mathbb{N}$ sorrendjét **topologikus sorrend**nek nevezzük, ha minden $(i, j) \in A$ élre $order(i) < order(j)$.

Tétel

Egy gráfnak pontosan akkor van topologikus sorrendje, ha nem tartalmaz irányított kört (DAG).

Algoritmus

- 1: $next := 1$
- 2: **while** $G \neq \emptyset$ **do**
- 3: Válasszunk ki egy i csúcsot, amibe nem megy be él ▷ Van ilyen?
- 4: $order(i) := next$; $next := next + 1$
- 5: Töröljük ki G -ből i -t (és a belőle kiinduló éleket)
- 6: **end while**

Házi feladat 2/3.

Adjuk meg a fenti algoritmus egy hatékony ($O(m + n)$ futási idejű) megvalósítását.

Topologikus sorrend

Definíció

A $G = (N, A)$ gráf csúcsainak egy $order : N \rightarrow \mathbb{N}$ sorrendjét **topologikus sorrend**nek nevezzük, ha minden $(i, j) \in A$ élre $order(i) < order(j)$.

Tétel

Egy gráfnak pontosan akkor van topologikus sorrendje, ha nem tartalmaz irányított kört (DAG).

Algoritmus

- 1: $next := 1$
- 2: **while** $G \neq \emptyset$ **do**
- 3: Válasszunk ki egy i csúcsot, amibe nem megy be él
- 4: $order(i) := next$; $next := next + 1$
- 5: Töröljük ki G -ből i -t (és a belőle kiinduló éleket)
- 6: **end while**

▷ Van ilyen?

Házi feladat 2/3.

Adjuk meg a fenti algoritmus egy hatékony ($O(m + n)$ futási idejű) megvalósítását.

DAG Példa: BKK menetrend

Menetrend gráf

$V := \{(s, t) : s \in \mathcal{S}, t \in \mathcal{T}_s\}$ $A := A_{trip} \cup A_{change}$, ahol

- \mathcal{S} a megállók halmaza,
- \mathcal{T}_s az s megállóba érkező járatok halmaza,
 - Az azonos útvonalon különböző időben haladó járművek külön járatnak számítanak.
- $A_{trip} := \{((s_1, t), (s_2, t)) : s_1 \text{ és } s_2 \text{ a } t \text{ két egymást követő megállója}\}$,
- $A_{change} := \{((s, t_1), (s, t_2)) : \text{a } t_1\text{-ről a } t_2\text{-re át lehet szállni az } s\text{-ben}\}$
 - gyaloglás éleket is hozzáadhatunk adni a közeli megállók között

Házi feladat 2/3. Írjunk utazástervező programot!

- BKK menetrend:
<https://lemon.cs.elte.hu/trac/lemon/wiki/AlkMod2012>
- Fel kell fizikailag építenünk a fenti a gráfot a memóriában?
- Célfüggvény: Minimális utazási idő, legkevesebb átszállás, legkevesebb gyaloglás (vagy ezek kombinációja.)
- Alternatív útvonaltervek keresése

DAG Példa: BKK menetrend

Menetrend gráf

$V := \{(s, t) : s \in \mathcal{S}, t \in \mathcal{T}_s\}$ $A := A_{trip} \cup A_{change}$, ahol

- \mathcal{S} a megállók halmaza,
- \mathcal{T}_s az s megállóba érkező járatok halmaza,
 - Az azonos útvonalon különböző időben haladó járművek külön járatnak számítanak.
- $A_{trip} := \{((s_1, t), (s_2, t)) : s_1 \text{ és } s_2 \text{ a } t \text{ két egymást követő megállója}\}$,
- $A_{change} := \{((s, t_1), (s, t_2)) : \text{a } t_1\text{-ről a } t_2\text{-re át lehet szállni az } s\text{-ben}\}$
 - gyaloglás éleket is hozzáadhatunk adni a közeli megállók között

Házi feladat 2/3. Írjunk utazástervező programot!

- BKK menetrend:
<https://lemon.cs.elte.hu/trac/lemon/wiki/AlkMod2012>
- Fel kell fizikailag építenünk a fenti a gráfot a memóriában?
- Célfüggvény: Minimális utazási idő, legkevesebb átszállás, legkevesebb gyaloglás (vagy ezek kombinációja.)
- Alternatív útvonaltervek keresése

Legrövidebb utak irányított gráfokban

Definíció (Legrövidebb út probléma)

Adott egy $G = (N, A)$ gráf és egy $c : A \rightarrow \mathbb{R}$ hosszfüggvény az éleken, továbbá $s, t \in N$ csúcsok.

Keressük a legrövidebb $s \rightsquigarrow t$ utat.

Előfeltételek

- G irányított gráf (\implies irányított utat keresünk).
- G nem tartalmaz negatív kört (azaz negatív összhosszúságú irányított kört)
- s -ből minden pont elérhető irányított út mentén
- (Néha) A hosszúságértékek egész számok ($c : A \rightarrow \mathbb{Z}$)

Változatok

- Keressünk utat s -ből az összes többi pontba
 - nemnegatív élhosszok esetén
 - tetszőleges élhosszokkal.
- Keressünk utat mindenhonnan mindenhova.
- Egyéb általánosítások
 - Időkorlátos legrövidebb út, max kapacitású út, fordulások büntetése, ...

Legrövidebb utak irányított gráfokban

Definíció (Legrövidebb út probléma)

Adott egy $G = (N, A)$ gráf és egy $c : A \rightarrow \mathbb{R}$ hosszfüggvény az éleken, továbbá $s, t \in N$ csúcsok.

Keressük a legrövidebb $s \rightsquigarrow t$ utat.

Előfeltételek

- **G irányított gráf** (\implies irányított utat keresünk).
- **G nem tartalmaz negatív kört** (azaz negatív összhosszúságú irányított kört)
- s-ből minden pont elérhető irányított út mentén
- (Néha) A hosszúságértékek egész számok ($c : A \rightarrow \mathbb{Z}$)

Változatok

- Keressünk utat s-ből az összes többi pontba
 - nemnegatív élhosszok esetén
 - tetszőleges élhosszokkal.
- Keressünk utat mindenhonnan mindenhova.
- Egyéb általánosítások
 - Időkorlátos legrövidebb út, max kapacitású út, fordulások büntetése, ...

Legrövidebb utak irányított gráfokban

Definíció (Legrövidebb út probléma)

Adott egy $G = (N, A)$ gráf és egy $c : A \rightarrow \mathbb{R}$ hosszfüggvény az éleken, továbbá $s, t \in N$ csúcsok.

Keressük a legrövidebb $s \rightsquigarrow t$ utat.

Előfeltételek

- **G irányított gráf** (\implies irányított utat keresünk).
- **G nem tartalmaz negatív kört** (azaz negatív összhosszúságú irányított kört)
- s -ből minden pont elérhető irányított út mentén
- (Néha) A hosszúságértékek egész számok ($c : A \rightarrow \mathbb{Z}$)

Változatok

- Keressünk utat s -ből az összes többi pontba
 - nemnegatív élhosszok esetén
 - tetszőleges élhosszokkal.
- Keressünk utat mindenhonnan mindenhova.
- Egyéb általánosítások
 - Időkorlátos legrövidebb út, max kapacitású út, fordulások büntetése, ...

Legrövidebb utak fája

Állítás

Legyen p egy $s \rightsquigarrow t$ legrövidebb út. Ekkor p minden részútja is egy legrövidebb út annak két végpontja között.

(Emlékeztető: Feltettük, hogy G -ben nincs negatív kör!)

Házi feladat 2/4. (Lemma)

Minden $s \rightsquigarrow t$ irányított séta felbontható egy $s \rightsquigarrow t$ út és irányított körök uniójára.

Tétel

Jelölje $d(i)$ a legrövidebb $s \rightsquigarrow i$ út hosszát. Ekkor

- *Minden $(i, j) \in A$ éltre $d(j) - d(i) \leq c_{ij}$*
- *Egy p $s \rightsquigarrow u$ út pontosan akkor legrövidebb, ha minden $(i, j) \in p$ éltre $d(j) - d(i) = c_{ij}$*

Házi feladat 2/5. (Legrövidebb utak fája)

Az s -ből kiinduló legrövidebb utak megválaszthatók úgy, hogy egy s gyökerű ki-fenyőt alkossanak.

Legrövidebb utak fája

Állítás

Legyen p egy $s \rightsquigarrow t$ legrövidebb út. Ekkor p minden részútja is egy legrövidebb út annak két végpontja között.

(Emlékeztető: Feltettük, hogy G -ben nincs negatív kör!)

Házi feladat 2/4. (Lemma)

Minden $s \rightsquigarrow t$ irányított séta felbontható egy $s \rightsquigarrow t$ út és irányított körök uniójára.

Tétel

Jelölje $d(i)$ a legrövidebb $s \rightsquigarrow i$ út hosszát. Ekkor

- *Minden $(i, j) \in A$ éltre $d(j) - d(i) \leq c_{ij}$*
- *Egy p $s \rightsquigarrow u$ út pontosan akkor legrövidebb, ha minden $(i, j) \in p$ éltre $d(j) - d(i) = c_{ij}$*

Házi feladat 2/5. (Legrövidebb utak fája)

Az s -ből kiinduló legrövidebb utak megválaszthatók úgy, hogy egy s gyökerű ki-fenyőt alkossanak.

Legrövidebb utak fája

Állítás

Legyen p egy $s \rightsquigarrow t$ legrövidebb út. Ekkor p minden részútja is egy legrövidebb út annak két végpontja között.

(Emlékeztető: Feltettük, hogy G -ben nincs negatív kör!)

Házi feladat 2/4. (Lemma)

Minden $s \rightsquigarrow t$ irányított séta felbontható egy $s \rightsquigarrow t$ út és irányított körök uniójára.

Tétel

Jelölje $d(i)$ a legrövidebb $s \rightsquigarrow i$ út hosszát. Ekkor

- *Minden $(i, j) \in A$ élre $d(j) - d(i) \leq c_{ij}$*
- *Egy p $s \rightsquigarrow u$ út pontosan akkor legrövidebb, ha minden $(i, j) \in p$ élre $d(j) - d(i) = c_{ij}$*

Házi feladat 2/5. (Legrövidebb utak fája)

Az s -ből kiinduló legrövidebb utak megválaszthatók úgy, hogy egy s gyökerű ki-fenyőt alkossanak.

Tétel

Egy DAG-ban a legrövidebb utak fája $O(m)$ időben meghatározható.

Bemenő éleken dolgozó algoritmus

```
1: procedure DAGSHORTESTPATH( $G, c, s$ )
2:   Rendezzük  $G$  pontjait topologikus sorrendbe
3:   Hagyjuk el az  $s$  előtti pontokat           ▷ Ezek nem érhetők el  $s$ -ből
4:    $d(s) := 0, pred(s) := 0$ 
5:   for  $\forall j \in N, j \neq s$ , topologikus sorrendben do
6:      $(i_{min}, j) := \operatorname{argmin}\{d(i) + c_{ij} : (i, j) \in \rho(j)\}$ 
7:      $d(j) := d(i_{min}) + c_{i_{min}j}$ 
8:      $pred(j) := (i_{min}, j)$ 
9:   end for
10: end procedure
```

Kimenő éleken dolgozó algoritmus

```
1: procedure DAGSHORTESTPATH( $G, c, s$ )
2:   Rendezzük  $G$  pontjait topologikus sorrendbe
3:   Hagyjuk el az  $s$  előtti pontokat           ▷ Ezek nem érhetők el  $s$ -ből
4:   for  $\forall i \in N, i \neq s$  do
5:      $d(i) := \infty$ 
6:   end for
7:    $d(s) := 0, pred(s) := 0$ 
8:   for  $\forall i \in N, i \neq s$ , topologikus sorrendben do
9:     for  $j : (i, j) \in \delta(i)$  do
10:      if  $d(i) + c_{ij} < d(j)$  then
11:         $d(j) := d(i) + c_{ij}$ 
12:         $pred(j) := (i, j)$ 
13:      end if
14:    end for
15:  end for
16: end procedure
```

Legrövidebb út nemnegatív élhosszak esetén

Dijkstra algoritmus

```
1: procedure DIJKSTRA( $G = (N, A), c, s$ )
2:    $S := \emptyset, \bar{S} := N$ 
3:   for  $\forall i \in N, i \neq s$  do
4:      $d(i) := \infty$ 
5:   end for
6:    $d(s) := 0, pred(s) := 0$ 
7:   while  $\bar{S} \neq \emptyset$  do
8:      $i := \operatorname{argmin}\{d(j) : j \in \bar{S}\}$ 
9:      $S := S \cup \{i\}, \bar{S} := \bar{S} \setminus \{i\}$ 
10:    for  $i : (i, j) \in \delta(i)$  do
11:      if  $d(i) + c_{ij} < d(j)$  then
12:         $d(j) := d(i) + c_{ij}$ 
13:         $pred(j) := (i, j)$ 
14:      end if
15:    end for
16:  end while
17: end procedure
```

Legrövidebb út nemnegatív élhosszak esetén

Dijkstra algoritmus

```
1: procedure DIJKSTRA( $G = (N, A), c, s$ )
2:    $S := \emptyset, \bar{S} := N$ 
3:   for  $\forall i \in N, i \neq s$  do
4:      $d(i) := \infty$ 
5:   end for
6:    $d(s) := 0, pred(s) := 0$ 
7:   while  $\bar{S} \neq \emptyset$  do
8:      $i := \operatorname{argmin}\{d(j) : j \in \bar{S}\}$ 
9:      $S := S \cup \{i\}, \bar{S} := \bar{S} \setminus \{i\}$ 
10:    for  $i : (i, j) \in \delta(i)$  do
11:      if  $d(i) + c_{ij} < d(j)$  then
12:         $d(j) := d(i) + c_{ij}$ 
13:         $pred(j) := (i, j)$ 
14:      end if
15:    end for
16:  end while
17: end procedure
```

Bizonyítás (Helyesség)

S méretére vonatkozó indukcióval. Indukciós hipotézis:

- (1) Minden $i \in S$ -re $d(i)$, helyes azaz a legrövidebb $s \rightsquigarrow i$ út hossza.
- (2) Minden $i \in \bar{S}$ -re $d(i) := \min\{c(p) : p \text{ egy } s \rightsquigarrow i \text{ út és } p \subseteq S \cup \{i\}\}$

Futásidő (primitív implementáció): $O(n^2 + m)$

- a $d(i)$ értékeket egy tömbben tároljuk
- Csúcs kiválasztás: $n + (n - 1) + (n - 2) + \dots + 1 = O(n^2)$
- Távolságértékek ($d(i)$ -k) frissítése: $\sum_{i \in N} |\delta(i)| = m.$

Házi feladat 2/5.

Igaz-e, hogy ha G egy DAG, akkor a Dijkstra algoritmus a csúcsokat topologikus sorrendben dolgozza fel?

- (a) Tetszőleges (nemnegatív) élsúlyok esetén
- (b) Azonosan 1 élsúlyok esetén

Házi feladat 2/6.

Tegyük fel, hogy G -ben minden él hossza 1. Mutassuk meg, hogy ekkor a Dijkstra algoritmus a BFS algoritmussal megegyező sorrendben dolgozza fel a csúcsokat.

Házi feladat 2/7.

Mutassunk példát olyan negatív éleket is (de negatív kört nem) tartalmazó gráfra, ahol a Dijkstra algoritmus

- helyes eredményt ad.
- hibás eredményt ad.

Házi feladat 2/8.

Adott $G = (N, A)$ irányított gráf, $c : A \rightarrow \mathbb{R}$ hosszfüggvény és $s_1, s_2, t_1, t_2 \in N$ csúcsok. Keressünk egy utat ami s_1 -ből vagy s_2 -ből indul, t_1 -be vagy t_2 -be érkezik és az ilyen utak között minimális költségű.

Dijkstra algoritmus – heap (kupac) implementáció

Dijkstra algoritmus

```
1: procedure DIJKSTRA( $G = (N, A), c, s$ )
2:   create-heap( $H$ )
3:   for  $\forall i \in N, i \neq s$  do
4:      $d(i) := \infty$ 
5:   end for
6:    $d(s) := 0, pred(s) := 0$ 
7:   insert( $s, H$ )
8:   while  $H \neq \emptyset$  do
9:     find-min( $i, H$ ), delete-min( $H$ )
10:    for  $i : (i, j) \in \delta(i)$  do
11:       $val := d(i) + c_{ij}$ 
12:      if  $val < d(j)$  then
13:         $d(j) := val$ 
14:         $pred(j) := (i, j)$ 
15:        if  $j \in H$  then
16:          decrease-key( $val, j, H$ )
17:        else
18:          insert( $j, H$ )
19:        end if
20:      end if
21:    end for
22:  end while
23: end procedure
```


Dijkstra algoritmus – heap (kupac) implementáció

Dijkstra algoritmus

```
1: procedure DIJKSTRA( $G = (N, A), c, s$ )
2:   create-heap( $H$ )
3:   for  $\forall i \in N, i \neq s$  do
4:      $d(i) := \infty$ 
5:   end for
6:    $d(s) := 0, pred(s) := 0$ 
7:   insert( $s, H$ )
8:   while  $H \neq \emptyset$  do
9:     find-min( $i, H$ ), delete-min( $H$ )
10:    for  $i : (i, j) \in \delta(i)$  do
11:       $val := d(i) + c_{ij}$ 
12:      if  $val < d(j)$  then
13:         $d(j) := val$ 
14:         $pred(j) := (i, j)$ 
15:        if  $j \in H$  then
16:          decrease-key( $val, j, H$ )
17:        else
18:          insert( $j, H$ )
19:        end if
20:      end if
21:    end for
22:  end while
23: end procedure
```

Tétel (Dijkstra futásideje heap adatstruktúrával)

A Dijkstra algoritmus futásideje $O(nT_M + mT_K)$, ahol T_M egy min. kulcsú elem kiválasztásának, T_K egy kulcsnövelésnek az ideje.

Dijkstra algoritmus

Futásidő (primitív implementáció): $O(n^2 + m)$

- Min. csúcs kiválasztás: $O(n)$
- Kulcscsökkentés: $O(1)$.

Dijkstra algoritmus – Dial implementáció

$$c : A \longrightarrow \{0, 1, 2, \dots, C\}$$

Dial-féle implementáció

- Csinálunk $nC + 1$ darab tárolót: $B_k := \{i \in N : d(i) = k\}$.
- Nyilvántartjuk a legkisebb index d_{min} indexet, amire $B_{d_{min}}$ nem üres.
- Minden iterációban frissítjük B_k -kat és d_{min} -t.

Tétel (Futásidő)

A fenti implementáció futásideje $O(nC + m)$.

Állítás

Ha egy iterációban az $i \in N$ csúcsot dolgozzuk fel, akkor minden $j \in N$, $d(j) < \infty$ csúcsra $d(j) \leq d(i) + C$.

Tétel (Ciklikus tárolók)

Elegendő $C + 1$ tárolót fenntartani $(B_{d_{min}}, \dots, B_{d_{min}+C})$

Dijkstra algoritmus – Dial implementáció

$$c : A \longrightarrow \{0, 1, 2, \dots, C\}$$

Dial-féle implementáció

- Csinálunk $nC + 1$ darab tárolót: $B_k := \{i \in N : d(i) = k\}$.
- Nyilvántartjuk a legkisebb index d_{min} indexet, amire $B_{d_{min}}$ nem üres.
- Minden iterációban frissítjük B_k -kat és d_{min} -t.

Tétel (Futásidő)

A fenti implementáció futásideje $O(nC + m)$.

Állítás

Ha egy iterációban az $i \in N$ csúcsot dolgozzuk fel, akkor minden $j \in N$, $d(j) < \infty$ csúcsra $d(j) \leq d(i) + C$.

Tétel (Ciklikus tárolók)

Elegendő $C + 1$ tárolót fenntartani ($B_{d_{min}}, \dots, B_{d_{min}+C}$)

Dijkstra algoritmus – Dial implementáció

$$c : A \longrightarrow \{0, 1, 2, \dots, C\}$$

Dial-féle implementáció

- Csinálunk $nC + 1$ darab tárolót: $B_k := \{i \in N : d(i) = k\}$.
- Nyilvántartjuk a legkisebb index d_{min} indexet, amire $B_{d_{min}}$ nem üres.
- Minden iterációban frissítjük B_k -kat és d_{min} -t.

Tétel (Futásidő)

A fenti implementáció futásideje $O(nC + m)$.

Állítás

Ha egy iterációban az $i \in N$ csúcsot dolgozzuk fel, akkor minden $j \in N$, $d(j) < \infty$ csúcsra $d(j) \leq d(i) + C$.

Tétel (Ciklikus tárolók)

Elegendő $C + 1$ tárolót fenntartani ($B_{d_{min}}, \dots, B_{d_{min}+C}$)

Dijkstra algoritmus – Dial implementáció

$$c : A \longrightarrow \{0, 1, 2, \dots, C\}$$

Dial-féle implementáció

- Csinálunk $nC + 1$ darab tárolót: $B_k := \{i \in N : d(i) = k\}$.
- Nyilvántartjuk a legkisebb index d_{min} indexet, amire $B_{d_{min}}$ nem üres.
- Minden iterációban frissítjük B_k -kat és d_{min} -t.

Tétel (Futásidő)

A fenti implementáció futásideje $O(nC + m)$.

Állítás

Ha egy iterációban az $i \in N$ csúcsot dolgozzuk fel, akkor minden $j \in N$, $d(j) < \infty$ csúcsra $d(j) \leq d(i) + C$.

Tétel (Ciklikus tárolók)

Elegendő $C + 1$ tárolót fenntartani ($B_{d_{min}}, \dots, B_{d_{min}+C}$)

Radix heap

Hasonló a Dial implementációhoz, de az egyes tárolókban nem egyetlen távolságértékhez tartozó csúcsokat tárolunk.

Inicializáláskor

- $K := \lceil \log(nC) \rceil$
- $R_0 := [0], R_1 := [1], R_2 := [2, 3], R_3 := [4, 7], \dots, R_K := [2^{K-1}, 2^K - 1]$.
Invariáns tulajdonság: R_k intervallumban 2^{k-1} darab szám van, ha $k > 0$.
- $B_k := \{i \in N : d(i) \in R_k\}$
- k_{min} az első nemüres tároló indexe

Műveletek

- Min. csúcs kiválasztása
 - Ha $B_{k_{min}} = \emptyset$, akkor $k_{min} := k_{min} + 1$
 - Ha $|B_{k_{min}}| = 1$, akkor feldolgozzuk a benne levő csúcsot és $k_{min} := k_{min} + 1$
 - Ha $|B_{k_{min}}| > 1$, akkor
 - $R_{k_{min}}$ -t felosztjuk $R_1, \dots, R_{k_{min}-1}$ között, $R_{k_{min}} := \emptyset$
 - $B_{k_{min}}$ elemeti szétosztjuk $B_1, \dots, B_{k_{min}-1}$ -be, $|B_{k_{min}}| := \emptyset$
 - $k_{min} := 0$
- Kulcsnövekedés: **Visszafele haladva** megkeressük a megfelelő tárolót

Radix heap

Hasonló a Dial implementációhoz, de az egyes tárolókban nem egyetlen távolságértékhez tartozó csúcsokat tárolunk.

Inicializáláskor

- $K := \lceil \log(nC) \rceil$
- $R_0 := [0], R_1 := [1], R_2 := [2, 3], R_3 := [4, 7], \dots, R_K := [2^{K-1}, 2^K - 1]$.
Invariáns tulajdonság: R_k intervallumban 2^{k-1} darab szám van, ha $k > 0$.
- $B_k := \{i \in N : d(i) \in R_k\}$
- k_{min} az első nemüres tároló indexe

Futásidő („amortizált futásidő” becslés)

- Kulcscsökkentés **összesen**: $O(nK)$
- Tároló újraosztás: $O(K)$, összesen: $O(nK)$
- Élek feldolgozása: $O(m)$

Tétel

A radix heap teljes futásideje $O(m + n \log(nC))$.

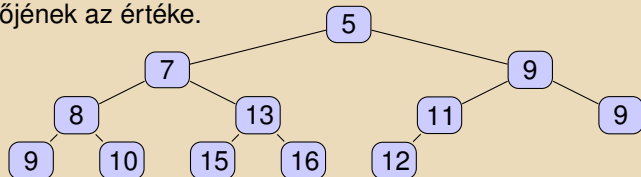
Házi feladat 3/1.

A radix heap teljes futásideje $O(m + n \log(C))$. (Elég $1 + \lceil \log C \rceil$ tároló.)

Bináris és d-Kupac

Egy kiegyensúlyozott teljes bináris (vagy d-leszármazottas) fa a következő invariáns tulajdonsággal:

- A fa minden csúcsában levő érték legalább akkora, mint a szülőjének az értéke.



Tétel

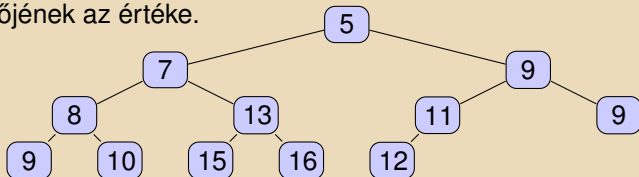
- Bináris kupacban az `insert()`, `decrease-min()` és `delete-min()` műveletek $O(\log n)$ idejűek.
- d-kupacban kupacban az `insert()` és `decrease-min()` műveletek $O(\log_d n)$ idejűek, a `delete-min()` pedig $O(d \log_d n)$.

Egy ilyen adatstruktúra tömbben tárolható!

Bináris és d-Kupac

Egy kiegyensúlyozott teljes bináris (vagy d-leszármazottas) fa a következő invariáns tulajdonsággal:

- A fa minden csúcsában levő érték legalább akkora, mint a szülőjének az értéke.



Tétel

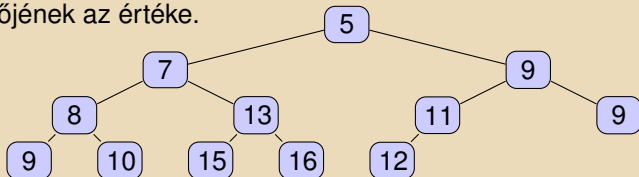
- *Bináris kupacban az `insert()`, `decrease-min()` és `delete-min()` műveletek $O(\log n)$ idejűek.*
- *d-kupacban kupacban az `insert()` és `decrease-min()` műveletek $O(\log_d n)$ idejűek, a `delete-min()` pedig $O(d \log_d n)$.*

Egy ilyen adatstruktúra tömbben tárolható!

Bináris és d-Kupac

Egy kiegyensúlyozott teljes bináris (vagy d-leszármazottas) fa a következő invariáns tulajdonsággal:

- A fa minden csúcsában levő érték legalább akkora, mint a szülőjének az értéke.



Tétel

- *Bináris kupacban az `insert()`, `decrease-min()` és `delete-min()` műveletek $O(\log n)$ idejűek.*
- *d-kupacban kupacban az `insert()` és `decrease-min()` műveletek $O(\log_d n)$ idejűek, a `delete-min()` pedig $O(d \log_d n)$.*

Egy ilyen adatstruktúra tömbben tárolható!

Tétel (Dijkstra futásideje heap adatstruktúrával)

A Dijkstra algoritmus futásideje $O(nT_M + mT_K)$, ahol T_M egy min. kulcsú elem kiválasztásának, T_K egy kulcsnövelésnek az ideje.

- Bináris kupac ($T_M = O(\log n)$, $T_K = O(\log n)$) esetén: $O(m \log n)$
- d -kupac ($T_M = O(n \log_d n)$, $T_K = O(\log_d n)$) esetén $O(m \log_d n + nd \log_d n)$
 - $d := \max\{2, \lceil m/n \rceil\}$ választással: $O(m \log_d n)$
 - Ritka gráfra ($m = O(n)$): $O(n \log n)$
 - Sűrű gráfra ($m = \Omega(n^{1+\epsilon})$): $O(m)$
- Fibonacci kupac ($T_M = O(\log n)$, $T_K = O(1)$) esetén: $O(m + n \log n)$
- Johnson kupac ($T_M = T_K = O(\log \log C)$) esetén: $O(m \log \log C)$

Dijkstra algoritmus

```
1: procedure DIJKSTRA( $G = (N, A), c, s$ )
2:   create-heap( $H$ )
3:   for  $\forall i \in N, i \neq s$  do
4:      $d(i) := \infty$ 
5:   end for
6:    $d(s) := 0, pred(s) := 0$ 
7:   insert( $s, H$ )
8:   while  $H \neq \emptyset$  do
9:     find-min( $i, H$ ), delete-min( $H$ )
10:    for  $i : (i, j) \in \delta(i)$  do
11:       $val := d(i) + c_{ij}$ 
12:      if  $val < d(j)$  then
13:         $d(j) := val$ 
14:         $pred(j) := (i, j)$ 
15:        if  $j \in H$  then
16:          decrease-key( $val, j, H$ )
17:        else
18:          insert( $j, H$ )
19:        end if
20:      end if
21:    end for
22:  end while
23: end procedure
```

Dijkstra algoritmus kulccsökentés nélkül

```
1: procedure DIJKSTRA( $G = (N, A), c, s$ )
2:   create-heap( $H$ )
3:   for  $\forall i \in N, i \neq s$  do
4:      $d(i) := \infty$ 
5:   end for
6:    $d(s) := 0, pred(s) := 0$ 
7:   insert( $s, H$ )
8:   while  $H \neq \emptyset$  do
9:     find-min( $i, H$ ), delete-min( $H$ )
10:    if  $pred(i) \neq 0$  then
11:      for  $i : (i, j) \in \delta(i)$  do
12:         $val := d(i) + c_{ij}$ 
13:        if  $val < d(j)$  then
14:           $d(j) := val$ 
15:           $pred(j) := (i, j)$ 
16:          insert( $j, H$ )
17:        end if
18:      end for
19:    end if
20:  end while
21: end procedure
```

Legrövidebb utak általános esetben

Tétel (Legrövidebb út optimalitási feltétel)

Legyen adott $s \in N$, és minden $j \in N$ csúcsra p_j legyen egy tetszőleges $s \rightsquigarrow j$ út, hosszát pedig jelölje $d(j)$. A p_j utak pontosan akkor legrövidebbek, ha

$$d(j) \leq d(i) + c_{ij} \quad (2)$$

teljesül minden $(i, j) \in A$ élre.

Tétel

Legyen $d : N \rightarrow \mathbb{R}$ függvény és $c_{ij}^d := c_{ij} + d(i) - d(j)$. Ekkor

- Minden W irányított körre (körsétára) $\sum_{(i,j) \in W} c_{ij}^d = \sum_{(i,j) \in W} c_{ij}$
- Ha P egy $k \rightsquigarrow l$ út (séta), akkor $\sum_{(i,j) \in P} c_{ij}^d = \sum_{(i,j) \in P} c_{ij} + d(k) - d(l)$
- Ha $d(\cdot)$ éppen a legrövidebb $s \rightsquigarrow \cdot$ utak hossza, akkor $c_{ij}^d \geq 0$ minden $(i, j) \in A$ élre.

Legrövidebb utak általános esetben

Tétel (Legrövidebb út optimalitási feltétel)

Legyen adott $s \in N$, és minden $j \in N$ csúcsra p_j legyen egy tetszőleges $s \rightsquigarrow j$ út, hosszát pedig jelölje $d(j)$. A p_j utak pontosan akkor legrövidebbek, ha

$$d(j) \leq d(i) + c_{ij} \quad (2)$$

teljesül minden $(i, j) \in A$ élre.

Tétel

Legyen $d : N \rightarrow \mathbb{R}$ függvény és $c_{ij}^d := c_{ij} + d(i) - d(j)$. Ekkor

- Minden W irányított körre (körsétára) $\sum_{(i,j) \in W} c_{ij}^d = \sum_{(i,j) \in W} c_{ij}$
- Ha P egy $k \rightsquigarrow l$ út (séta), akkor $\sum_{(i,j) \in P} c_{ij}^d = \sum_{(i,j) \in P} c_{ij} + d(k) - d(l)$
- Ha $d(\cdot)$ éppen a legrövidebb $s \rightsquigarrow \cdot$ utak hossza, akkor $c_{ij}^d \geq 0$ minden $(i, j) \in A$ élre.

Cimkejavító algoritmusok

Általános cimkejavító algoritmus

```
1: procedure LABELCORRECTING( $G, c, s$ )  
2:    $d(s) := 0, pred(s) := 0$   
3:    $d(j) := \infty \quad \forall j \in N$   
4:   while  $\exists (i, j) \in A, d(j) > d(i) + c_{ij}$  do  
5:      $d(j) := d(i) + c_{ij}$   
6:      $pred(j) := i$   
7:   end while  
8: end procedure
```

Állítás

Amennyiben az algoritmus megáll, $d(j)$ a legrövidebb $s \rightsquigarrow j$ út hossza lesz minden $j \in N$ -re.

Tétel

Az algoritmus futása legfeljebb $2n^2 C$ iteráció után véget ér.

Cimkejavító algoritmusok

Általános cimkejavító algoritmus

```
1: procedure LABELCORRECTING( $G, c, s$ )
2:    $d(s) := 0, pred(s) := 0$ 
3:    $d(j) := \infty \quad \forall j \in N$ 
4:   while  $\exists (i, j) \in A, d(j) > d(i) + c_{ij}$  do
5:      $d(j) := d(i) + c_{ij}$ 
6:      $pred(j) := i$ 
7:   end while
8: end procedure
```

Állítás

Amennyiben az algoritmus megáll, $d(j)$ a legrövidebb $s \rightsquigarrow j$ út hossza lesz minden $j \in N$ -re.

Tétel

Az algoritmus futása legfeljebb $2n^2 C$ iteráció után véget ér.

Cimkejavító algoritmusok

Általános cimkejavító algoritmus

```
1: procedure LABELCORRECTING( $G, c, s$ )
2:    $d(s) := 0, pred(s) := 0$ 
3:    $d(j) := \infty \quad \forall j \in N$ 
4:   while  $\exists (i, j) \in A, d(j) > d(i) + c_{ij}$  do
5:      $d(j) := d(i) + c_{ij}$ 
6:      $pred(j) := i$ 
7:   end while
8: end procedure
```

Állítás

Amennyiben az algoritmus megáll, $d(j)$ a legrövidebb $s \rightsquigarrow j$ út hossza lesz minden $j \in N$ -re.

Tétel

Az algoritmus futása legfeljebb $2n^2 C$ iteráció után véget ér.

Cimkejavító algoritmusok

Módosított cimkejavító algoritmus

```
1: procedure MODIFIEDLABELCORRECTING( $G, c, s$ )
2:   Legyen
3:    $d(s) := 0, \text{pred}(s) := 0$ 
4:    $d(j) := \infty \quad \forall j \in N$ 
5:    $LIST = \{s\}$ 
6:   while  $LIST \neq \emptyset$  do
7:     Legyen  $i \in LIST, LIST := LIST - i$ 
8:     for  $(i, j) \in \delta(i)$  do
9:       if  $d(j) > d(i) + c_{ij}$  then
10:         $d(j) := d(i) + c_{ij}$ 
11:         $\text{pred}(j) := i$ 
12:        if  $j \notin LIST$  then
13:           $LIST := LIST + j$ 
14:        end if
15:      end if
16:    end for
17:  end while
18: end procedure
```

Cimkejavító algoritmusok

Módosított cimkejavító algoritmus

```
1: procedure MODIFIEDLABELCORRECTING( $G, c, s$ )
2:   Legyen
3:    $d(s) := 0, \text{pred}(s) := 0$ 
4:    $d(j) := \infty \quad \forall j \in N$ 
5:    $LIST = \{s\}$ 
6:   while  $LIST \neq \emptyset$  do
7:     Legyen  $i \in LIST, LIST := LIST - i$ 
8:     for  $(i, j) \in \delta(i)$  do
9:       if  $d(j) > d(i) + c_{ij}$  then
10:         $d(j) := d(i) + c_{ij}$ 
11:         $\text{pred}(j) := i$ 
12:        if  $j \notin LIST$  then
13:           $LIST := LIST + j$ 
14:        end if
15:      end if
16:    end for
17:  end while
18: end procedure
```

Tétel

Az algoritmus futása $O(nmC)$ lépésben véget ér.

Tétel

A fenti algoritmusok futása tetszőleges valós távolságok esetén is véges ($O(2^n)$) lépésben véget ér.

Házi feladat 3/4.

Mutass példát, amin elképzelhető, hogy az fenti algoritmusok $\Omega(2^n)$ iterációt végeznek.

Erősen polinomiális ($O(nm)$) cimkejavító algoritmus

Minden k -ra és i -re számoljuk ki a legrövidebb legfeljebb k élből álló utat.

Általános cimkejavító algoritmus

```
1: procedure LABELCORRECTING( $G, c, s$ )  
2:    $d(s) := 0, pred(s) := 0$   
3:    $d(j) := \infty \quad \forall j \in N$   
4:   while  $\exists (i, j) \in A, d(j) > d(i) + c_{ij}$  do  
5:      $d(j) := d(i) + c_{ij}$   
6:      $pred(j) := i$   
7:   end while  
8: end procedure
```

Erősen polinomiális ($O(nm)$) címkejavító algoritmus

Minden k -ra és i -re számoljuk ki a legrövidebb legfeljebb k élből álló utat.

Bellman-Ford algoritmus

```
1: procedure BELLMANFORD( $G, c, s$ )
2:    $d(s) := 0, pred(s) := 0$ 
3:    $d(j) := \infty \quad \forall j \in N$ 
4:   for  $k \in \{1, \dots, n\}$  do
5:     for  $(i, j) \in A$  do
6:       if  $d(j) > d(i) + c_{ij}$  then
7:          $d(j) := d(i) + c_{ij}$ 
8:          $pred(j) := i$ 
9:       end if
10:    end for
11:  end for
12: end procedure
```


Erősen polinomiális ($O(nm)$) címkejavító algoritmus

Minden k -ra és i -re számoljuk ki a legrövidebb legfeljebb k élből álló utat.

Bellman-Ford algoritmus

```
1: procedure BELLMANFORD( $G, c, s$ )
2:    $d(s) := 0, pred(s) := 0$ 
3:    $d(j) := \infty \quad \forall j \in N$ 
4:   for  $k \in \{1, \dots, n\}$  do
5:     for  $(i, j) \in A$  do
6:       if  $d(j) > d(i) + c_{ij}$  then
7:          $d(j) := d(i) + c_{ij}$ 
8:          $pred(j) := i$ 
9:       end if
10:    end for
11:  end for
12: end procedure
```

Tétel

- Az algoritmus futásaideje: $O(nm)$
- a futása végén a $d(\cdot)$ értékek a legrövidebb utak hosszai lesznek.

Erősen polinomiális ($O(nm)$) címkejavító algoritmus

Minden k -ra és i -re számoljuk ki a legrövidebb legfeljebb k élből álló utat.

Bellman-Ford algoritmus II.

A módosított címkejavító algoritmusban a *LIST* adatszerkezet legyen egy **FIFO sor** (queue).

Erősen polinomiális ($O(nm)$) címkejavító algoritmus

Minden k -ra és i -re számoljuk ki a legrövidebb legfeljebb k élből álló utat.

Bellman-Ford algoritmus II.

A módosított címkejavító algoritmusban a *LIST* adatszerkezet legyen egy **FIFO sor** (queue).

Tétel

- *Az algoritmus „ugyanolyan sorrendben” dolgozza fel az éleket, mint az eredeti változat, így a futásideje $O(nm)$.*

Erősen polinomiális ($O(nm)$) címkejavító algoritmus

Minden k -ra és i -re számoljuk ki a legrövidebb legfeljebb k élből álló utat.

Bellman-Ford algoritmus II.

A módosított címkejavító algoritmusban a *LIST* adatszerkezet legyen egy **FIFO sor** (queue).

Tétel

- *Az algoritmus „ugyanolyan sorrendben” dolgozza fel az éleket, mint az eredeti változat, így a futásideje $O(nm)$.*

Házi feladat 3/5.

Adott $G = (N, A)$ irányított gráf, $c : A \rightarrow \mathbb{R}$ hosszfüggvény az éleken. Adjunk (erősen polinomiális) algoritmust annak eldöntésére, hogy a gráf tartalmaz-e negatív kört.

Legrövidebb út minden pontpár között

Nemnegatív élek esetén

Futtassuk le minden pontból a Dijkstra algoritmust.

Futásidő: $O(nm + n^2 \log n)$

Általános esetben

Futtassuk le minden pontból a Bellman-Ford algoritmust.

Futásidő: $O(n^2 m)$

Legrövidebb út minden pontpár között

Nemnegatív élek esetén

Futtassuk le minden pontból a Dijkstra algoritmust.

Futásidő: $O(nm + n^2 \log n)$

Általános esetben

Futtassuk le minden pontból a Bellman-Ford algoritmust.

Futásidő: $O(n^2 m)$

Legrövidebb út minden pontpár között

Nemnegatív élek esetén

Futtassuk le minden pontból a Dijkstra algoritmust.

Futásidő: $O(nm + n^2 \log n)$

Általános esetben

Futtassuk le minden pontból a Bellman-Ford algoritmust.

Futásidő: $O(n^2 m)$

Általános esetben: Johnson algoritmus

- Futtassuk le egy pontból a Bellman-Ford algoritmust.
Ekkor a $c_{ij}^d := c_{ij} + d(i) - d(j)$ nemnegatív
- Futtassuk le a többi pontból a c^d hosszfüggvényre a Dijkstra algoritmust.

Futásidő: $O(nm + n^2 \log n)$

Floyd-Warshall Algoritmus

Dinamikus programozás (ez még nem a Floyd-Warshall)

$d^k(i, j)$ a legrövidebb legfeljebb k élből álló $i \rightsquigarrow j$ út hossza. Ezt számítsuk ki minden k -ra.

- Futásidő: $O(n^2 m)$
- „Mátrix-szorzás”: $O(n^3 \log n)$

Floyd-Warshall Algoritmus

Floyd-Warshall Algoritmus

Legyen $d^k(i, j)$ a legrövidebb olyan $i \rightsquigarrow j$ út hossza, ami belső pontként csak az $\{1, \dots, k-1\}$ csúcsokat tartalmazza.

```
1: procedure FLOYDWARSHALL( $G, c$ )
2:    $d(i, j) := \infty$  és  $pred(i, j) := 0 \quad \forall i, j \in N$ -re
3:    $d(i, i) := 0 \quad \forall i \in N$ -re
4:   for  $(i, j) \in A$  do
5:      $d(i, j) := c_{ij}$  és  $pred(i, j) := i$ 
6:   end for
7:   for  $k \in \{1, \dots, n\}$  do
8:     for  $(i, j) \in N \times N$  do
9:       if  $d(i, j) > d(i, k) + d(k, j)$  then
10:         $d(i, j) := d(i, k) + d(k, j)$ 
11:         $pred(i, j) := pred(k, j)$ 
12:      end if
13:    end for
14:  end for
15: end procedure
```

Futásidő: $O(n^3)$

Házi feladat 4/1.

Adott $G = (N, A)$ irányított gráf $c_1, c_2 : A \rightarrow \mathbb{Z}(\mathbb{R})$ súlyfüggvény az éleken, és $s, t \in N$ pontok. Tegyük fel továbbá, hogy nincs c_1 szerint negatív kör a gráfban.

Feladat: Keressünk a c_1 -minimális $s \rightsquigarrow t$ utak között egy c_2 minimálisat. Azaz keressük

$$\min\{c_2(p) : p \in \mathcal{P}_{c_1}\}-t,$$

ahol

$$\mathcal{P}_{c_1} := \{p : p \text{ egy } c_1\text{-minimális } s \rightsquigarrow t \text{ út}\}.$$

Feladatok (legrövidebb utak)

Házi feladat 4/2.

Adott egy $G = (N, E)$ irányítatlan (vagy irányított) gráf és egy $c : E \rightarrow \mathbb{R}$ súlyfüggvény. Egy út szélességén az út minimális élsúlyát értjük, azaz

$$\text{szélesség}(P) := \min\{c(e) : e \in P\}$$

Feladat: keressük a maximális szélességű utat

- egy adott $s, t \in N$ pontpár között
- egy adott $s \in N$ pontból az összes többi pontba
- minden pontpárra

Házi feladat 4/3.

Adott egy $G = (N, E)$ irányított gráf és $c : E \rightarrow \mathbb{R}$ súlyfüggvény az éleken. Tegyük fel, hogy nincs a gráfban negatív kör. Egy $(i, j) \in E$ élre jelölje f_{ij} azt a maximális értéket, amivel az (i, j) él súlyát csökkentve a gráfban továbbra sem lesz negatív kör.

Adjunk hatékony algoritmust az összes f_{ij} érték meghatározására.

Legrövidebb utak „térképeken”

Kétirányú Dijkstra algoritmus

Párhuzamosan futtatjuk a Dijkstra algoritmust s -ből és t -ből visszafele. Megállunk amikor a két futtatás összeér.

Házi feladat 4/4. Meddig futtassuk az algoritmust?

Legyen S_s az s -ből, S_t pedig a t -ből visszafele indított Dijkstra algoritmus által feldolgozott pontok halmaza és tegyük fel, hogy $S_s \cap S_t = \{u\}$. Legyen $p_{s,u}$ és $p_{u,t}$ a legrövidebb $s \rightsquigarrow u$ ill. $u \rightsquigarrow t$ út. Igaz-e, hogy $p_{s,u}p_{u,t}$ egy legrövidebb $s \rightsquigarrow t$ út?

A* algoritmus

Legyen $c'_{u,v} := c_{u,v} - \pi(u) + \pi(v)$, ahol $\pi(u)$ az $u \rightarrow t$ távolság légvonalban.

Állítás

$$c'_{u,v} \geq 0 \quad \forall (u,v) \in A$$

Futtassuk a Dijkstra algoritmust a c' súlyfüggvénnyel!

Legrövidebb utak „térképeken”

Kétirányú Dijkstra algoritmus

Párhuzamosan futtatjuk a Dijkstra algoritmust s -ből és t -ből visszafele. Megállunk amikor a két futtatás összeér.

Házi feladat 4/4. Meddig futtassuk az algoritmust?

Legyen S_s az s -ből, S_t pedig a t -ből visszafele indított Dijkstra algoritmus által feldolgozott pontok halmaza és tegyük fel, hogy $S_s \cap S_t = \{u\}$. Legyen $p_{s,u}$ és $p_{u,t}$ a legrövidebb $s \rightsquigarrow u$ ill. $u \rightsquigarrow t$ út. Igaz-e, hogy $p_{s,u}p_{u,t}$ egy legrövidebb $s \rightsquigarrow t$ út?

A* algoritmus

Legyen $c'_{u,v} := c_{u,v} - \pi(u) + \pi(v)$, ahol $\pi(u)$ az $u \rightarrow t$ távolság légvonalban.

Állítás

$$c'_{u,v} \geq 0 \quad \forall (u, v) \in A$$

Futtassuk a Dijkstra algoritmust a c' súlyfüggvénnyel!

Törtoptimalizálás

Minimális átlagú körök

Feladat

Adott egy $G = (N, A)$ irányított gráf, ami tartalmaz irányított kört és adott egy tetszőleges $c : A \rightarrow \mathbb{R}$ súlyozás az éleken.

- Határozzunk meg a minimális körátlagot, azaz

$$\lambda_1^* := \min \left\{ \frac{c(C)}{|C|} : C \subseteq A \text{ egy irányított kör} \right\}$$

- Határozzuk meg azt a minimális λ_2^* értéket, amivel minden él súlyát egységesen csökkentve még éppen nincs negatív kör. (λ_2^* lehet negatív is...)

Állítás

$$\lambda_1^* = \lambda_2^*$$

Minimális átlagú körök

Ötlet

(Először csökkentjük le minden él költségét annyival, hogy bizonyosan legyen negatív kör)

- Keressünk egy negatív kört
- Növeljük meg az élek súlyát ennek a körnek az átlagával
- Ismételjük a fenti lépéseket addig, amíg van negatív kör

Állítás

A fenti eljárás véges, de exponenciális sok iterációra lehet szükség.

Minimális átlagú körök

Ötlet

(Először csökkentjük le minden él költségét annyival, hogy bizonyosan legyen negatív kör)

- Keressünk egy negatív kört
- Növeljük meg az élek súlyát ennek a körnek az átlagával
- Ismételjük a fenti lépéseket addig, amíg van negatív kör

Állítás

A fenti eljárás véges, de exponenciális sok iterációra lehet szükség.

Minimális átlagú körök

Ötlet

(Először csökkentjük le minden él költségét annyival, hogy bizonyosan legyen negatív kör)

- Keressünk egy **minimális súlyú** kört
- Növeljük meg az élek súlyát ennek a körnek az átlagával
- Ismételjük a fenti lépéseket addig, amíg van negatív kör

Állítás

- *A fenti n iteráció után véget ér.*
- *A minimális súlyú kör keresése \mathcal{NP} -teljes feladat*

Minimális átlagú körök

Ötlet

(Először csökkentjük le minden él költségét annyival, hogy bizonyosan legyen negatív kör)

- Keressünk egy minimális súlyú **legfeljebb n hosszú körsétát**
- Növeljük meg az élek súlyát ennek a körnek az átlagával
- Ismételjük a fenti lépéseket addig, amíg van negatív kör

Állítás

- *A fenti n iteráció után véget ér.*
- *A minimális súlyú legfeljebb n hosszú körsétát dinamikus programozással polinom időben tudunk keresni.*
- *Futásidő: $O(n^2 m)$*

Minimális átlagú körök

Ötlet

(Először csökkentjük le minden él költségét annyival, hogy bizonyosan legyen negatív kör)

- Keressünk egy minimális súlyú **legfeljebb n hosszú körsétát**
- Növeljük meg az élek súlyát ennek a körnek az átlagával
- Ismételjük a fenti lépéseket addig, amíg van negatív kör

Állítás

- *A fenti n iteráció után véget ér.*
- *A minimális súlyú legfeljebb n hosszú körsétát dinamikus programozással polinom időben tudunk keresni.*
- *Futásidő: $O(n^2 m)$*

Házi feladat 4/4.

A minimális körátlagot közvetlenül is meg tudjuk dinamikus programozással határozni.

Házi feladat 5/1. Segítség az előzőhöz

Adott egy $G = (N, A)$ irányított gráf és egy $c : E \rightarrow \mathbb{R}$ súlyfüggvény az éleken. Tegyük fel, hogy s -ből minden pont elérhető irányított úton. Jelölje a $d^k(j)$ érték a minimális súlyú pontosan k élből álló $s \rightsquigarrow j$ séta hosszát. Tegyük fel, hogy $d^k(j)$ ismert minden $j \in N$ csúcsra és $k = 1, \dots, n$ -re. Határozzuk meg a minimális körátlagot ennek segítségével.

Minimális súlyozott átlagú körök

Feladat

Adott egy $G = (N, A)$ irányított gráf, amit tartalmaz irányított kört és adott egy tetszőleges $c : A \rightarrow \mathbb{R}$ és egy pozitív $w : A \rightarrow \mathbb{R}_+$ súlyozás az éleken.

- Határozzunk meg a minimális súlyozott átlagú kört, azaz

$$\lambda_1^* := \min \left\{ \frac{c(C)}{w(C)} : C \subseteq A \text{ egy irányított kör} \right\}$$

- Határozzuk meg a

$$\lambda_2^* := \max \{ \lambda \in \mathbb{R} : \forall C \subseteq A \text{ irányított körre } c(C) - \lambda w(C) \geq 0 \}$$

értéket

Állítás

$$\lambda_1^* = \lambda_2^*$$

Algoritmus ötlet

Legyen $c_\lambda(e) := c(e) - \lambda w(e)$.

1. eset Van egy c_λ negatív C_λ kör. Ekkor $\lambda > \lambda^*$
2. eset Nincs c_λ negatív kör, de van 0 hosszú C_λ . Ekkor $\lambda = \lambda^*$, és ez a kör a súlyozott átlag szerint minimális.
3. eset Minden kör c_λ szerinti hossza pozitív. Ekkor $\lambda < \lambda^*$

Következmény

Egy adott λ értékre el tudjuk dönteni, hogy az a min. súlyozott átlagnál kisebb, nagyobb, vagy egyenlő-e vele. Ez utóbbi esetben megkapjuk a min súlyozott átlagú kört is.

Algoritmus ötletek

- Szekvenciális keresés: Ha $\lambda_i > \lambda^*$, akkor legyen $\lambda_{i+1} := \frac{c(C_{\lambda_i})}{w(C_{\lambda_i})}$.
- Bináris keresés (intervallum-felezés)

Tétel

Legyen c és w egészértékű, ekkor a fenti bináris kereséssel polinom időben meghatározhatjuk a súlyozott körátlagot.

Algoritmus ötlet

Legyen $c_\lambda(e) := c(e) - \lambda w(e)$.

1. eset Van egy c_λ negatív C_λ kör. Ekkor $\lambda > \lambda^*$
2. eset Nincs c_λ negatív kör, de van 0 hosszú C_λ . Ekkor $\lambda = \lambda^*$, és ez a kör a súlyozott átlag szerint minimális.
3. eset Minden kör c_λ szerinti hossza pozitív. Ekkor $\lambda < \lambda^*$

Következmény

Egy adott λ értékre el tudjuk dönteni, hogy az a min. súlyozott átlagnál kisebb, nagyobb, vagy egyenlő-e vele. Ez utóbbi esetben megkapjuk a min súlyozott átlagú kört is.

Algoritmus ötletek

- Szekvenciális keresés: Ha $\lambda_i > \lambda^*$, akkor legyen $\lambda_{i+1} := \frac{c(C_{\lambda_i})}{w(C_{\lambda_i})}$.
- Bináris keresés (intervallum-felezés)

Tétel

Legyen c és w egészértékű, ekkor a fenti bináris kereséssel polinom időben meghatározhatjuk a súlyozott körátlagot.

Algoritmus ötlet

Legyen $c_\lambda(e) := c(e) - \lambda w(e)$.

1. eset Van egy c_λ negatív C_λ kör. Ekkor $\lambda > \lambda^*$
2. eset Nincs c_λ negatív kör, de van 0 hosszú C_λ . Ekkor $\lambda = \lambda^*$, és ez a kör a súlyozott átlag szerint minimális.
3. eset Minden kör c_λ szerinti hossza pozitív. Ekkor $\lambda < \lambda^*$

Következmény

Egy adott λ értékre el tudjuk dönteni, hogy az a min. súlyozott átlagnál kisebb, nagyobb, vagy egyenlő-e vele. Ez utóbbi esetben megkapjuk a min súlyozott átlagú kört is.

Algoritmus ötletek

- Szekvenciális keresés: Ha $\lambda_i > \lambda^*$, akkor legyen $\lambda_{i+1} := \frac{c(C_{\lambda_i})}{w(C_{\lambda_i})}$.
- Bináris keresés (intervallum-felezés)

Tétel

Legyen c és w egészértékű, ekkor a fenti bináris kereséssel polinom időben meghatározhatjuk a súlyozott körátlagot.

Feladat

Adott egy \mathcal{X} alaphalmaz valamint adott egy tetszőleges $c : \mathcal{X} \rightarrow \mathbb{R}$ és egy pozitív $w : \mathcal{X} \rightarrow \mathbb{R}_+$ költségfüggvény.

- Határozzunk meg a

$$\lambda_1^* := \min \left\{ \frac{c(X)}{w(X)} : X \in \mathcal{X} \right\}$$

értéket

- Határozzuk meg a

$$\lambda_2^* := \max \{ \lambda \in \mathbb{R} : \forall X \in \mathcal{X} \text{-re } c(X) - \lambda w(X) \geq 0 \}$$

értéket

Állítás

$$\lambda_1^* = \lambda_2^*$$

Algoritmus ötlet

Legyen $c_\lambda(e) := c(e) - \lambda w(e)$.

1. eset Van egy c_λ negatív $X_\lambda \in \mathcal{X}$. Ekkor $\lambda > \lambda^*$
2. eset Nincs c_λ negatív, de van 0 hosszú $X_\lambda \in \mathcal{X}$. Ekkor $\lambda = \lambda^*$, és ez a súlyozott átlag szerint minimális.
3. eset Minden $X \in \mathcal{X}$ c_λ szerinti súlya pozitív. Ekkor $\lambda < \lambda^*$

Következmény

Egy adott λ értékre el tudunk dönteni, hogy az a min. súlyozott átlagnál kisebb, nagyobb, vagy egyenlő-e vele. Ez utóbbi esetben megkapjuk a min súlyozott átlagú $X \in \mathcal{X}$ -et is.

Algoritmus ötletek

- Szekvenciális keresés: Ha $\lambda_i > \lambda^*$, akkor legyen $\lambda_{i+1} := \frac{c(X_{\lambda_i})}{w(X_{\lambda_i})}$.
- Bináris keresés (intervallum-felezés)

Tétel

Legyen c és w egészértékű, ekkor a fenti bináris kereséssel polinom időben meghatározhatjuk a minimális súlyozott átlagot.

Algoritmus ötlet

Legyen $c_\lambda(e) := c(e) - \lambda w(e)$.

1. eset Van egy c_λ negatív $X_\lambda \in \mathcal{X}$. Ekkor $\lambda > \lambda^*$
2. eset Nincs c_λ negatív, de van 0 hosszú $X_\lambda \in \mathcal{X}$. Ekkor $\lambda = \lambda^*$, és ez a súlyozott átlag szerint minimális.
3. eset Minden $X \in \mathcal{X}$ c_λ szerinti súlya pozitív. Ekkor $\lambda < \lambda^*$

Következmény

Egy adott λ értékre el tudunk dönteni, hogy az a min. súlyozott átlagnál kisebb, nagyobb, vagy egyenlő-e vele. Ez utóbbi esetben megkapjuk a min súlyozott átlagú $X \in \mathcal{X}$ -et is.

Algoritmus ötletek

- Szekvenciális keresés: Ha $\lambda_i > \lambda^*$, akkor legyen $\lambda_{i+1} := \frac{c(X_{\lambda_i})}{w(X_{\lambda_i})}$.
- Bináris keresés (intervallum-felezés)

Tétel

Legyen c és w egészértékű, ekkor a fenti bináris kereséssel polinom időben meghatározhatjuk a minimális súlyozott átlagot.

Algoritmus ötlet

Legyen $c_\lambda(e) := c(e) - \lambda w(e)$.

1. eset Van egy c_λ negatív $X_\lambda \in \mathcal{X}$. Ekkor $\lambda > \lambda^*$
2. eset Nincs c_λ negatív, de van 0 hosszú $X_\lambda \in \mathcal{X}$. Ekkor $\lambda = \lambda^*$, és ez a súlyozott átlag szerint minimális.
3. eset Minden $X \in \mathcal{X}$ c_λ szerinti súlya pozitív. Ekkor $\lambda < \lambda^*$

Következmény

Egy adott λ értékre el tudunk dönteni, hogy az a min. súlyozott átlagnál kisebb, nagyobb, vagy egyenlő-e vele. Ez utóbbi esetben megkapjuk a min súlyozott átlagú $X \in \mathcal{X}$ -et is.

Algoritmus ötletek

- Szekvenciális keresés: Ha $\lambda_i > \lambda^*$, akkor legyen $\lambda_{i+1} := \frac{c(X_{\lambda_i})}{w(X_{\lambda_i})}$.
- Bináris keresés (intervallum-felezés)

Tétel

Legyen c és w egészértékű, ekkor a fenti bináris kereséssel polinom időben meghatározhatjuk a minimális súlyozott átlagot.

Algoritmus ötlet I.

Feltevés I.

Egy adott $\lambda \in \mathbb{R}$ -re meg tudjuk határozni

$$\min\{c(X) - \lambda w(X) : X \in \mathcal{X}\} \quad (3)$$

előjelét.

Szekvenciális keresés

Adott egy $X \in \mathcal{X}$. Döntsük el, hogy optimális-e, ha nem, akkor keressünk egy $X' \in \mathcal{X}$ -et, amire

$$\frac{c(X')}{w(X')} < \frac{c(X)}{w(X)}. \quad (4)$$

Azaz keressünk egy $X' \in \mathcal{X}$ -et, amire

$$c(X') - \frac{c(X)}{w(X)} w(X') < 0 \quad (5)$$

És ezt iteráljuk amíg el nem érjük az optimumot.

Algoritmus ötlet I.

Feltevés I.

Egy adott $\lambda \in \mathbb{R}$ -re meg tudjuk határozni

$$\min\{c(X) - \lambda w(X) : X \in \mathcal{X}\} \quad (3)$$

előjelét.

Szekvenciális keresés

Adott egy $X \in \mathcal{X}$. Döntsük el, hogy optimális-e, ha nem, akkor keressünk egy $X' \in \mathcal{X}$ -et, amire

$$\frac{c(X')}{w(X')} < \frac{c(X)}{w(X)}. \quad (4)$$

Azaz keressünk egy $X' \in \mathcal{X}$ -et, amire

$$c(X') - \frac{c(X)}{w(X)} w(X') < 0 \quad (5)$$

És ezt iteráljuk amíg el nem érjük az optimumot.

Algoritmus ötlet I.

Feltevés I.

Egy adott $\lambda \in \mathbb{R}$ -re meg tudjuk határozni

$$\min\{c(X) - \lambda w(X) : X \in \mathcal{X}\} \quad (3)$$

előjelét.

Szekvenciális keresés

Adott egy $X \in \mathcal{X}$. Döntsük el, hogy optimális-e, ha nem, akkor keressünk egy $X' \in \mathcal{X}$ -et, amire

$$\frac{c(X')}{w(X')} < \frac{c(X)}{w(X)}. \quad (4)$$

Azaz keressünk egy $X' \in \mathcal{X}$ -et, amire

$$c(X') - \frac{c(X)}{w(X)} w(X') < 0 \quad (5)$$

És ezt iteráljuk amíg el nem érjük az optimumot.

Algoritmus ötlet II.

Feltevés II.

Egy adott $\lambda \in \mathbb{R}$ -re meg tudjuk határozni a

$$\min\{c(X) - \lambda w(X) : X \in \mathcal{X}\} \quad (6)$$

értéket és egy minimumhelyet.

Javított szekvenciális keresés

Adott egy $X_i \in \mathcal{X}$. Allapítsuk meg a

$$H = \min\{c(X') - \frac{c(X_i)}{w(X_i)} w(X') : X' \in \mathcal{X}\} \quad (7)$$

értéket és egy $X' \in \mathcal{X}$ minimumhelyet. Ha $H < 0$ (azaz X_i nem optimális), akkor $X_{i+1} := X'$ -vel iteráljuk az eljárást.

Algoritmus ötlet II.

Feltevés II.

Egy adott $\lambda \in \mathbb{R}$ -re meg tudjuk határozni a

$$\min\{c(X) - \lambda w(X) : X \in \mathcal{X}\} \quad (6)$$

értéket és egy minimumhelyet.

Javított szekvenciális keresés

Adott egy $X_i \in \mathcal{X}$. Allapítsuk meg a

$$H = \min\{c(X') - \frac{c(X_i)}{w(X_i)} w(X') : X' \in \mathcal{X}\} \quad (7)$$

értéket és egy $X' \in \mathcal{X}$ minimumhelyet. Ha $H < 0$ (azaz X_i nem optimális), akkor $X_{i+1} := X'$ -vel iteráljuk az eljárást.

Definíció

$$h(\lambda) := \min \{c(x) - \lambda w(x) : x \in \mathcal{X}\}$$

$$x_\lambda := \operatorname{argmin}\{c(x) - \lambda w(x) : x \in \mathcal{X}\}$$

$$c_\lambda(x) := c(x) - \lambda w(x)$$

Newton-Dinkelbach eljárás

Definíció

$$h(\lambda) := \min \{c(x) - \lambda w(x) : x \in \mathcal{X}\}$$

$$x_\lambda := \operatorname{argmin}\{c(x) - \lambda w(x) : x \in \mathcal{X}\}$$

$$c_\lambda(x) := c(x) - \lambda w(x)$$

Állítás

$h(\lambda)$ szigorúan monoton fogyó, szakaszonként lineáris, konváv függvény.

Newton-Dinkelbach eljárás

Definíció

$$h(\lambda) := \min \{c(x) - \lambda w(x) : x \in \mathcal{X}\}$$

$$x_\lambda := \operatorname{argmin}\{c(x) - \lambda w(x) : x \in \mathcal{X}\}$$

$$c_\lambda(x) := c(x) - \lambda w(x)$$

Állítás

$h(\lambda)$ szigorúan monoton fogyó, szakaszonként lineáris, konváv függvény.

Feladat

Keressük a $h(\lambda)$ nullhelyét, azaz a

$$\lambda^* := \min\{\lambda \in \mathbb{R} : h(\lambda) \leq 0\}$$

értéket és a hozzá tartozó x_{λ^*} elemet.

Newton-Dinkelbach eljárás

Definíció

$$h(\lambda) := \min \{c(x) - \lambda w(x) : x \in \mathcal{X}\}$$

$$x_\lambda := \operatorname{argmin}\{c(x) - \lambda w(x) : x \in \mathcal{X}\}$$

$$c_\lambda(x) := c(x) - \lambda w(x)$$

- 1: **procedure** NEWTONDINKELBACH(c, w, \mathcal{X})
- 2: $\lambda_1 \leftarrow \infty$ (vagy $\lambda_1 \leftarrow \frac{c(X_1)}{w(X_1)}$ egy tetszőleges $X_1 \in \mathcal{X}$ -re)
- 3: $i \leftarrow 0$
- 4: **repeat**
- 5: $i \leftarrow i + 1$
- 6: $x_i \leftarrow x_{\lambda_i} = \operatorname{argmin}\{c(x) - \lambda_i w(x) : x \in \mathcal{X}\}$
- 7: $c_i \leftarrow c(x_i)$
- 8: $w_i \leftarrow w(x_i)$
- 9: $h_i \leftarrow h(\lambda_i) = c_i - \lambda_i w_i$
- 10: $\lambda_{i+1} \leftarrow \frac{c_i}{w_i}$
- 11: **until** $h_i > 0$
- 12: **end procedure**

Newton-Dinkelbach eljárás

```
1: procedure NEWTONDINKELBACH( $c, w, \mathcal{X}$ )
2:    $\lambda_1 \leftarrow \infty$  (vagy  $\lambda_1 \leftarrow \frac{c(x_1)}{w(x_1)}$  egy tetszőleges  $x_1 \in \mathcal{X}$ -re)
3:    $i \leftarrow 0$ 
4:   repeat
5:      $i \leftarrow i + 1$ 
6:      $x_i \leftarrow x_{\lambda_i} = \operatorname{argmin}\{c(x) - \lambda_i w(x) : x \in \mathcal{X}\}$ 
7:      $c_i \leftarrow c(x_i)$ 
8:      $w_i \leftarrow w(x_i)$ 
9:      $h_i \leftarrow h(\lambda_i) = c_i - \lambda_i w_i$ 
10:     $\lambda_{i+1} \leftarrow \frac{c_i}{w_i}$ 
11:  until  $h_i > 0$ 
12: end procedure
```

Newton-Dinkelbach eljárás

```
1: procedure NEWTONDINKELBACH( $c, w, \mathcal{X}$ )
2:    $\lambda_1 \leftarrow \infty$  (vagy  $\lambda_1 \leftarrow \frac{c(x_1)}{w(x_1)}$  egy tetszőleges  $x_1 \in \mathcal{X}$ -re)
3:    $i \leftarrow 0$ 
4:   repeat
5:      $i \leftarrow i + 1$ 
6:      $x_i \leftarrow x_{\lambda_i} = \operatorname{argmin}\{c(x) - \lambda_i w(x) : x \in \mathcal{X}\}$ 
7:      $c_i \leftarrow c(x_i)$ 
8:      $w_i \leftarrow w(x_i)$ 
9:      $h_i \leftarrow h(\lambda_i) = c_i - \lambda_i w_i$ 
10:     $\lambda_{i+1} \leftarrow \frac{c_i}{w_i}$ 
11:  until  $h_i > 0$ 
12: end procedure
```

Állítás

- $h_1 < h_2 < \dots < h_t = 0$
- $\lambda_1 > \lambda_2 > \dots > \lambda_t = \lambda^*$
- $w_1 > w_2 > \dots > w_t > 0$

Tétel

$$\frac{h_{i+1}}{h_i} + \frac{w_{i+1}}{w_i} \leq 1$$

Következmény

$$\frac{h_{i+1} w_{i+1}}{h_i w_i} \leq \frac{1}{4}$$

Állítás

Minden $i = 1, 2, \dots, t-2$ és $\lambda \leq \lambda_{i+2}$ -re

$$c_i - \lambda w_i \geq -h_{i+1}$$

Tétel

$$\frac{h_{i+1}}{h_i} + \frac{w_{i+1}}{w_i} \leq 1$$

Következmény

$$\frac{h_{i+1} w_{i+1}}{h_i w_i} \leq \frac{1}{4}$$

Állítás

Minden $i = 1, 2, \dots, t-2$ és $\lambda \leq \lambda_{i+2}$ -re

$$c_i - \lambda w_i \geq -h_{i+1}$$

Tétel

$$\frac{h_{i+1}}{h_i} + \frac{w_{i+1}}{w_i} \leq 1$$

Következmény

$$\frac{h_{i+1} w_{i+1}}{h_i w_i} \leq \frac{1}{4}$$

Állítás

Minden $i = 1, 2, \dots, t-2$ és $\lambda \leq \lambda_{i+2}$ -re

$$c_i - \lambda w_i \geq -h_{i+1}$$

Feladat

Mutassunk olyan $c \in \mathbb{R}^p$ -t, és egy „hosszú” $y_1, y_2, \dots, y_q \in \{-1, 0, 1\}^p$ sorozatot, amire

$$cy_1 \geq cy_2 \geq \dots \geq cy_q$$

Tétel ((Goemans))

Legyen $c \in \mathbb{R}^p$ és $y_1, y_2, \dots, y_q \in \{-1, 0, 1\}^p$ olyanok, hogy minden $i \in \{1, \dots, q-1\}$ -re $0 < y_{i+1}c \leq \frac{1}{2}y_ic$ teljesül. Ekkor $q = O(p \log p)$.

Lineáris eset

Legyen E egy véges alaphalmaz $m := |E|$ valamint tegyük fel, hogy $X \subseteq \{0, 1\}^E$ és c illetve w lineáris függvények, azaz

$$c(x) := \sum_{e \in E} c_e x_e \quad \text{és} \quad w(x) := \sum_{e \in E} w_e x_e$$

valamilyen $c \in \mathbb{R}^E$ és $w \in \mathbb{R}^E$ vektorokra.

Példák

Minimális tört-optimalis utak, körök, feszítőfák, fenyők, párosítások, vágások stb.

Feltéve, hogy a megfelelő paraméteres feladat megoldható!

Tétel (Radzik)

Ebben az esetben a Newton-Dinkelbach eljárás $O(n^2 \log n)$ iteráció után véget ér.

Lineáris eset

Legyen E egy véges alaphalmaz $m := |E|$ valamint tegyük fel, hogy $X \subseteq \{0, 1\}^E$ és c illetve w lineáris függvények, azaz

$$c(x) := \sum_{e \in E} c_e x_e \quad \text{és} \quad w(x) := \sum_{e \in E} w_e x_e$$

valamilyen $c \in \mathbb{R}^E$ és $w \in \mathbb{R}^E$ vektorokra.

Tétel (Radzik)

Ebben az esetben a Newton-Dinkelbach eljárás $O(n^2 \log n)$ iteráció után véget ér.

Bizonyítás

Tekintsük az $s_i := -h_{2i} w_{2i-1}$ számsorozatot. Erre

- $s_{i+1} < \frac{1}{4} s_i$

- $s_i = c_{2i} w_{2i-1} - c_{2i-1} w_{2i}$

Alkalmazzuk Goemans lemmáját

Lineáris eset

Legyen E egy véges alaphalmaz $m := |E|$ valamint tegyük fel, hogy $X \subseteq \{0, 1\}^E$ és c illetve w lineáris függvények, azaz

$$c(x) := \sum_{e \in E} c_e x_e \quad \text{és} \quad w(x) := \sum_{e \in E} w_e x_e$$

valamilyen $c \in \mathbb{R}^E$ és $w \in \mathbb{R}^E$ vektorokra.

Tétel (Radzik)

Ebben az esetben a Newton-Dinkelbach eljárás $O(n^2 \log n)$ iteráció után véget ér.

Bizonyítás

Tekintsük az $s_i := -h_{2i} w_{2i-1}$ számsorozatot. Erre

- $s_{i+1} < \frac{1}{4} s_i$

- $s_i = c_{2i} w_{2i-1} - c_{2i-1} w_{2i}$

Alkalmazzuk Goemans lemmáját

Tört-legrövidebb út DAG-ban

Tétel

Legyen $G = (N, E)$ egy DAG és $s, t \in N$. Ekkor Newton-Dinkelbach eljárás a $s \rightsquigarrow t$ tört-legrövidebb út problémára alkalmazva $O(m \log n)$ iteráció után megáll.

Definíció

Egy $e \in E$ él **fontos az i -ik iterációban**, ha $e \in x_i \cup x_{i+1} \cup \dots$

Lemma

Végezzünk el $k := \lceil \log n \rceil + 1$ iterációt az algoritmussal. Ekkor a fontos élek halmaza szigorúan csökken.

Házi feladat 6/2. Kereskedelmi hajó útvonaltervezés

Adott egy $G = (N, E)$ irányított gráf, egy $p : E \rightarrow \mathbb{R}$ profit és egy $t : E \rightarrow \mathbb{R}$ utazási idő függvény.

- Fentieket felhasználva adjunk erősen polinomiális algoritmust a $\frac{p(C)}{t(C)}$ értéket maximalizáló C irányított kör megtalálására a gráfban.
- Mi a helyzet a férjezett/nős hajóskapitányokkal? Azaz, ha adott egy $f \in N$ férj/feleség, és az olyan C körökön keressük a maximális $\frac{p(C)}{t(C)}$ értéket, amelyek áthaladnak az f ponton.

Tört-legrövidebb út DAG-ban II. (Megidő módszere)

Ötlet

Próbáljuk lefuttatni a DAG legrövidebb út algoritmust a $c - \lambda^* w$ költségekkel anélkül, hogy λ^* értékét ismernénk, szimbolikusan számolva vele.

$$(\lambda^* := \max \{ \lambda \in \mathbb{R} : \forall X \in \mathcal{X} \text{-re } c(X) - \lambda w(X) \geq 0 \})$$

Tört-legrövidebb út DAG-ban II. (Megiddo módszere)

Kimenő éleken dolgozó algoritmus

```
1: procedure DAGSHORTESTPATH( $G, c, s$ )
2:   for  $\forall i \in N, i \neq s$  do
3:      $d(i) := \infty$ 
4:   end for
5:    $d(s) := 0, pred(s) := 0$ 
6:   for  $\forall i \in N, i \neq s$ , topologikus sorrendben do
7:     for  $j : (i, j) \in \delta(i)$  do
8:       if  $d(i) + c_{ij} < d(j)$  then
9:          $d(j) := d(i) + c_{ij}$ 
10:         $pred(j) := (i, j)$ 
11:       end if
12:     end for
13:   end for
14: end procedure
```

Tört-legrövidebb út DAG-ban II. (Megiddo módszere)

Kimenő éleken dolgozó algoritmus

```
1: procedure DAGSHORTESTPATH( $G, c, s$ )
2:   for  $\forall i \in N, i \neq s$  do
3:      $d(i) := \infty$ 
4:   end for
5:    $d(s) := 0, pred(s) := 0$ 
6:   for  $\forall i \in N, i \neq s$ , topologikus sorrendben do
7:     for  $j : (i, j) \in \delta(i)$  do
8:       if  $d(i) + (c_{ij} - \lambda * w_{ij}) < d(j)$  then
9:          $d(j) := d(i) + (c_{ij} - \lambda * w_{ij})$ 
10:         $pred(j) := (i, j)$ 
11:       end if
12:     end for
13:   end for
14: end procedure
```

Tört-legrövidebb út DAG-ban II. (Megiddo módszere)

Kimenő éleken dolgozó algoritmus

```
1: procedure DAGSHORTESTPATH( $G, c, s$ )
2:   for  $\forall i \in N, i \neq s$  do
3:      $d(i) := \infty$ 
4:   end for
5:    $d(s) := 0, pred(s) := 0$ 
6:   for  $\forall i \in N, i \neq s$ , topologikus sorrendben do
7:     for  $j : (i, j) \in \delta(i)$  do
8:       if  $d(i) + (c_{ij} - \lambda^* w_{ij}) < d(j)$  then
9:          $d(j) := d(i) + (c_{ij} - \lambda^* w_{ij})$ 
10:         $pred(j) := (i, j)$ 
11:       end if
12:     end for
13:   end for
14: end procedure
```

- A $d(i)$ tárolókban $a + \lambda^* b$ alakú kifejezéseket (azaz (a, b) számpárokat) tárolunk. Ezekkel szimbolikusan tudunk számolni.
- Összehasonlítás: $a_1 + \lambda^* b_1 < a_2 + \lambda^* b_2 \Leftrightarrow \lambda^* > \frac{a_1 - a_2}{b_1 - b_2}$, ha $b_1 > b_2$.

Tört-legrövidebb út DAG-ban II. (Megiddo módszere)

Házi feladat 6/3.

- Gondoljuk végig a fenti eljárás működését és lássuk be, hogy algoritmus végén kapott $s \rightsquigarrow t$ út tört-minimális lesz.
- Mennyi az algoritmus futási ideje?
- ★ Ötletek futásidő javításra?

Megjegyzések

- Csak a gyengébbik feltevést használtuk, azaz azt, hogy egy adott $\lambda \in \mathbb{R}$ -re meg tudjuk határozni $\min\{c(X) - \lambda w(X) : X \in \mathcal{X}\}$ előjelét.
- Ez egy általános módszer, az algoritmusról azt kell feltenni, hogy „nem szoroz”.

Házi feladat 6/4.

Tekintsük a következő lineáris programozási feladatot:

$$\begin{aligned} \min \quad & \sum_{(i,j) \in A} c_{ij} x_{ij} \\ \sum_{j:(i,j) \in A} x_{ij} - \sum_{j:(j,i) \in A} x_{ji} &= 0 \quad \forall i \in N \\ \sum_{(i,j) \in A} w_{ij} x_{ij} &= 1 \\ x_{ij} &\geq 0 \quad \forall (i,j) \in A \end{aligned}$$

Bizonyítsuk be, hogy, hogy ennek egy optimális megoldása meghatározó körök egy halmazát, amelyek súlyozott átlaga ($\frac{c(C)}{w(C)}$ értéke) egyenlő.

Adjunk ennek segítségével eljárást, ami a fenti LP-feladat optimális megoldásának felhasználásával megad egy minimális súlyozott átlagú kört.

Irányítás fokszám előírással

Feladat

Adott egy $G = (N, E)$ irányítatlan gráf és egy $d : N \rightarrow \mathbb{N}$ „befok előírás” a csúcsokon. Irányítsuk meg G éleit úgy, hogy minden i csúcsba legalább $d(i)$ él fusson be.

Definíció

Egy $i \in N$ csúcs

- *hiányos*, ha $\varrho(i) < d(i)$,
- *pontos*, ha $\varrho(i) = d(i)$ és
- *többletes*, ha $\varrho(i) > d(i)$

Házi feladat 7/1.

Vezessük vissza a fenti feladatot a maximális folyam feladatra!

Irányítás fokszám előírással

Feladat

Adott egy $G = (N, A)$ **irányított** gráf és egy $d : N \rightarrow \mathbb{N}$ „befok előírás” a csúcsokon. **Fordítsuk meg G néhány élét** úgy, hogy minden i csúcsba legalább $d(i)$ él fusson be.

Definíció

Egy $i \in N$ csúcs

- **hiányos**, ha $\varrho(i) < d(i)$,
- **pontos**, ha $\varrho(i) = d(i)$ és
- **többletes**, ha $\varrho(i) > d(i)$

Házi feladat 7/1.

Vezessük vissza a fenti feladatot a maximális folyam feladatra!

Irányítás fokszám előírással

Feladat

Adott egy $G = (N, A)$ **irányított** gráf és egy $d : N \rightarrow \mathbb{N}$ „befok előírás” a csúcsokon. **Fordítsuk meg G néhány élét** úgy, hogy minden i csúcsba legalább $d(i)$ él fusson be.

Definíció

Egy $i \in N$ csúcs

- **hiányos**, ha $\varrho(i) < d(i)$,
- **pontos**, ha $\varrho(i) = d(i)$ és
- **többletes**, ha $\varrho(i) > d(i)$

Házi feladat 7/1.

Vezessük vissza a fenti feladatot a maximális folyam feladatra!

Irányítás fokszám előírással

Tétel

Pontosan akkor létezik a befok előírásoknak megfelelő irányítás, ha minden $X \subseteq N$ részhalmazra

$$|\{(i, j) \in E : \{i, j\} \cap X \neq \emptyset\}| \geq \sum_{i \in X} d(i)$$

Algoritmus

Keressünk egy hiányos pontból egy többletesbe vezető utat és fordítsuk meg az út éleit. Ismételjük ezt addig, amíg

(a) nincs több hiányos pont vagy

(b) nincs hiányosból többletesbe vezető irányított út

Házi feladat 7/2.

Mekkora a futásideje ennek az algoritmusnak?

Szintező algoritmus

- Minden $i \in N$ pontra nyilvántartunk egy $\ell(i) \in \mathbb{N}$ szintet

Invariáns tulajdonság

Minden $(i, j) \in A$ élre $\ell(i) \leq \ell(j) + 1$, azaz minden él „legfeljebb egyet mehet lefele”.

Szintező algoritmus az irányítási feladatra

```
1: procedure ORIENT( $G, d$ )
2:    $\ell(i) \leftarrow 0 \quad \forall i \in N$ 
3:   while  $\exists i \in N : d(i) < d(i)$  do                                ▷ Hogyan találunk ilyen?
4:     if  $\exists (i, j) \in A : \ell(j) = \ell(i) - 1$  then                    ▷ Hogyan találunk ilyen?
5:       Fordítsuk meg az  $(i, j)$  élel!
6:     else
7:        $\ell(i) \leftarrow \ell(i) + 1$ 
8:     end if
9:   end while
10: end procedure
```

Szintező algoritmus

- Minden $i \in N$ pontra nyilvántartunk egy $\ell(i) \in \mathbb{N}$ szintet

Invariáns tulajdonság

Minden $(i, j) \in A$ élre $\ell(i) \leq \ell(j) + 1$, azaz minden él „legfeljebb egyet mehet lefele”.

Szintező algoritmus az irányítási feladatra

```
1: procedure ORIENT( $G, d$ )
2:    $\ell(i) \leftarrow 0 \quad \forall i \in N$ 
3:   while  $\exists i \in N : d(i) < d(i)$  do
4:     if  $\exists (i, j) \in A : \ell(j) = \ell(i) - 1$  then
5:       Fordítsuk meg az  $(i, j)$  élet!
6:     else
7:        $\ell(i) \leftarrow \ell(i) + 1$ 
8:     end if
9:   end while
10: end procedure
```

▷ Hogyan találunk ilyeneket?
▷ Hogyan láthatunk ilyeneket?

Szintező algoritmus

- Minden $i \in N$ pontra nyilvántartunk egy $\ell(i) \in \mathbb{N}$ szintet

Invariáns tulajdonság

Minden $(i, j) \in A$ élre $\ell(i) \leq \ell(j) + 1$, azaz minden él „legfeljebb egyet mehet lefele”.

Szintező algoritmus az irányítási feladatra

```
1: procedure ORIENT( $G, d$ )
2:    $\ell(i) \leftarrow 0 \quad \forall i \in N$ 
3:   while  $\exists i \in N : \varrho(i) < d(i)$  do                                ▷ Hogyan találunk ilyen?
4:     if  $\exists (i, j) \in A : \ell(j) = \ell(i) - 1$  then                    ▷ Hogyan találunk ilyen?
5:       Fordítsuk meg az  $(i, j)$  élet!
6:     else
7:        $\ell(i) \leftarrow \ell(i) + 1$ 
8:       if  $\ell(i) = n$  then STOP (Nincs megoldás)
9:     end if
10:  end while
11: end procedure
```

Szintező algoritmus

Szintező algoritmus az irányítási feladatra

```
1: procedure ORIENT( $G, d$ )
2:    $\ell(i) \leftarrow 0 \quad \forall i \in N$ 
3:   while  $\exists i \in N : \varrho(i) < d(i)$  do
4:     if  $\exists (i, j) \in A : \ell(j) = \ell(i) - 1$  then
5:       Fordítsuk meg az  $(i, j)$  élet!
6:     else
7:        $\ell(i) \leftarrow \ell(i) + 1$ 
8:       if  $\ell(i) = n$  then STOP (Nincs megoldás)
9:     end if
10:  end while
11: end procedure
```

▷ Hogyan találunk ilyen?

▷ Hogyan találunk ilyen?

Tétel

Amennyiben egy pont eléri az n . szintet, akkor az irányítási feladatnak nincs megoldása.

Tétel

Az algoritmus $O(nm)$ iteráció alatt véget ér.

Szintező algoritmus II.

Szintező algoritmus az irányítási feladatra

```
1: procedure ORIENT( $G, d$ )
2:    $\ell(i) \leftarrow 0 \quad \forall i \in N$ 
3:   while  $\exists i \in N : \varrho(i) < d(i)$  do
4:     while  $\varrho(i) < d(i)$  és  $\exists (i, j) \in A : \ell(j) = \ell(i) - 1$  do
5:       Fordítsuk meg az  $(i, j)$  élet!
6:     end while
7:     if  $\varrho(i) < d(i)$  then
8:        $\ell(i) \leftarrow \ell(i) + 1$ 
9:       if  $\ell(i) = n$  then STOP (Nincs megoldás)
10:      end if
11:    end if
12:  end while
13: end procedure
```

Házi feladat 7/3.

Ha mindig a legmagasabban levő hiányos pontot választjuk ki (hogyan?), akkor teljes algoritmus futási ideje $O(n^2 m)$.

Szintező algoritmus II.

Szintező algoritmus az irányítási feladatra

```
1: procedure ORIENT( $G, d$ )
2:    $\ell(i) \leftarrow 0 \quad \forall i \in N$ 
3:   while  $\exists i \in N : \varrho(i) < d(i)$  do
4:     while  $\varrho(i) < d(i)$  és  $\exists (i, j) \in A : \ell(j) = \ell(i) - 1$  do
5:       Fordítsuk meg az  $(i, j)$  élet!
6:     end while
7:     if  $\varrho(i) < d(i)$  then
8:        $\ell(i) \leftarrow \ell(i) + 1$ 
9:       if  $\ell(i) = n$  then STOP (Nincs megoldás)
10:      end if
11:    end if
12:  end while
13: end procedure
```

Házi feladat 7/3.

Ha mindig a legmagasabban levő hiányos pontot választjuk ki (hogyan?), akkor teljes algoritmus futási ideje $O(n^2m)$.

Házi feladat 7/4.

Adott egy $G = (A, B, E)$ páros gráf. Vezessük vissza a teljes (vagy: A -fedő) párosítás feladatot a fenti fokszámelőírási irányítási feladatra.

Feladat

Adott egy $G = (N, E)$ irányítatlan gráf és egy $d : N \rightarrow \mathbb{N}$ függvény a csúcsokon. Irányítsuk meg G éleit úgy, hogy minden i csúcsba **legfeljebb** $d(i)$ él fusson be.

Házi feladat 7/5.

Adott egy $G = (N, E)$ irányítatlan gráf és két $d_{min}, d_{max} : N \rightarrow \mathbb{N}$ függvény a csúcsokon. Irányítsuk meg G éleit úgy, hogy minden i csúcsba legalább $d_{min}(i)$, de legfeljebb $d_{max}(i)$ él fusson be.

Házi feladat 7/6.

Programozzuk be a javítóutas és a szintező algoritmust. Melyik gyorsabb a gyakorlatban?

Házi feladat Verseny.

Kié a leggyorsabb algoritmus az irányítás feladatra?

Szintező algoritmus a párosítás feladatra

Feladat

Adott egy $G = (A, B, E)$ páros gráf. Keressünk A -t fedő párosítást!

Definíció

Tekintsünk egy $m : A \rightarrow E$ függvényt. (Rendeljünk minden $a \in A$ ponthoz egy $m(a) = (a, m_b(a)) \in E$ élel.

Egy $b \in B$ csúcs

- **hiányos**, ha $\nexists a \in A : m_b(a) = b$,
- **pontos**, ha $\exists! a \in A : m_b(a) = b$, és
- **többletes**, ha $|\{a \in A : m_b(a) = b\}| > 1$

Szintfüggvény

- Minden $b \in B$ pontra nyilvántartunk egy $\ell(b) \in \mathbb{N}$ szintet

Invariáns tulajdonság

Minden $(a, b) \in E$ élre $\ell(b) \geq \ell(m_b(a)) - 1$.

Szintező algoritmus az A -t fedő párosítás feladatra

Szintező algoritmus az A -t fedő párosítás feladatra

```
1: procedure MATCHING( $G = (A, B, E)$ )
2:    $m(a) \leftarrow (a, x) \in E \quad \forall a \in A$ 
3:    $\ell(b) \leftarrow 0 \quad \forall b \in B$ 
4:   while  $\exists b \in B$  többletes do
5:     if  $\exists a \in A, (a, b^*) \in E : m_b(a) = b \wedge \ell(b^*) = \ell(b) - 1$  then
6:        $m(a) \leftarrow (a, b^*)$ 
7:     else
8:        $\ell(b) \leftarrow \ell(b) + 1$ 
9:       if  $\ell(b) = |B|$  then STOP (Nincs megoldás)
10:      end if
11:    end if
12:  end while
13: end procedure
```

Tétel

Amennyiben egy pont eléri az $|B|$. szintet, akkor nincs A -t fedő párosítás.

Szintező algoritmus az A -t fedő párosítás feladatra

Szintező algoritmus az A -t fedő párosítás feladatra

```
1: procedure MATCHING( $G = (A, B, E)$ )
2:    $m(a) \leftarrow (a, x) \in E \quad \forall a \in A$ 
3:    $\ell(b) \leftarrow 0 \quad \forall b \in B$ 
4:   while  $\exists b \in B$  többletes do
5:     if  $\exists a \in A, (a, b^*) \in E : m_b(a) = b \wedge \ell(b^*) = \ell(b) - 1$  then
6:        $m(a) \leftarrow (a, b^*)$ 
7:     else
8:        $\ell(b) \leftarrow \ell(b) + 1$ 
9:       if  $\ell(b) = |B|$  then STOP (Nincs megoldás)
10:      end if
11:    end if
12:  end while
13: end procedure
```

Tétel

Amennyiben egy pont eléri az $|B|$. szintet, akkor nincs A -t fedő párosítás.

Házi feladat 7/8.

Adjunk szintező algoritmust a B -t fedő párosítások keresésére a fentiek alapján.

Segítség: $m : A \rightarrow E$ és $\ell : B \rightarrow \mathbb{N}$ mint az előbb, de most a *hiányos* pontokat próbáljuk megszüntetni ill. emelni. Mi lesz a jó invariáns tulajdonság?

Házi feladat 7/8.

Programozzuk be a javítóutas és a szintező (mindkét változat) algoritmust a párosítás feladatra. Melyik a leggyorsabb a gyakorlatban?

Házi feladat Verseny.

Kié a leggyorsabb párosítás algoritmus?

Házi feladat 8/1.

Mennyi a fenti párosítás algoritmusok futási ideje a legrosszabb esetben?

Házi feladat 8/2. b -matching

Adott egy $G = (A, B, E)$ páros gráf és egy $b : A \cup B \rightarrow \mathbb{N}$ fokszám előírás. Keressünk egy $M \subseteq E$ részalmazt, ami minden $i \in A \cup B$ pontra pontosan $b(i)$ darab i -vel szomszédos élet tartalmaz.

Házi feladat 8/3.

Adott egy $G = (A, B, E)$ páros gráf és egy $b : A \cup B \rightarrow \mathbb{N}(\mathbb{R})$ függvény. Keressünk egy $x \in \mathbb{R}^E$ vektort, amire $\sum_{j:(i,j) \in A} x(i) = b(i)$ teljesül minden $i \in A \cup B$ pontra.

Megengedett folyam feladatok

F értékű folyam

Adott egy $G = (N, A)$ irányított gráf, $u : A \rightarrow \mathbb{N}$ kapacitásfüggvény, $s, t \in N$ pontok (forrás és nyelő) és egy $F \in \mathbb{N}$ folyamérték. Keressünk egy $x : A \rightarrow \mathbb{N}$ élsúlyozást, amire

$$\sum_{j:(i,j) \in A} x_{ij} - \sum_{j:(j,i) \in A} x_{ji} = \begin{cases} F & , \text{ ha } i = s \\ -F & , \text{ ha } i = t, \text{ egyébként} \\ 0 & \end{cases} \quad \forall i \in N$$
$$0 \leq x_{ij} \leq u_{ij} \quad \forall (i, j) \in A$$

b -folyam

Adott egy $G = (N, A)$ irányított gráf, $u : A \rightarrow \mathbb{N}$ kapacitásfüggvény, és $b : N \rightarrow \mathbb{N}$ függvény. Keressünk egy $x : A \rightarrow \mathbb{N}$ élsúlyozást, amire

$$\sum_{j:(i,j) \in A} x_{ij} - \sum_{j:(j,i) \in A} x_{ji} = b(i) \quad \forall i \in N$$
$$0 \leq x_{ij} \leq u_{ij} \quad \forall (i, j) \in A$$

b -folyam felső korlát nélkül

Megengedett folyam feladatok

F értékű folyam

Adott egy $G = (N, A)$ irányított gráf, $u : A \rightarrow \mathbb{N}$ kapacitásfüggvény, $s, t \in N$ pontok (forrás és nyelő) és egy $F \in \mathbb{N}$ folyamérték. Keressünk egy $x : A \rightarrow \mathbb{N}$ élsúlyozást, amire

$$\sum_{j:(i,j) \in A} x_{ij} - \sum_{j:(j,i) \in A} x_{ji} = \begin{cases} F & , \text{ ha } i = s \\ -F & , \text{ ha } i = t, \text{ egyébként} \\ 0 & \end{cases} \quad \forall i \in N$$
$$0 \leq x_{ij} \leq u_{ij} \quad \forall (i, j) \in A$$

b -folyam

Adott egy $G = (N, A)$ irányított gráf, $u : A \rightarrow \mathbb{N}$ kapacitásfüggvény, és $b : N \rightarrow \mathbb{N}$ függvény. Keressünk egy $x : A \rightarrow \mathbb{N}$ élsúlyozást, amire

$$\sum_{j:(i,j) \in A} x_{ij} - \sum_{j:(j,i) \in A} x_{ji} = b(i) \quad \forall i \in N$$
$$0 \leq x_{ij} \leq u_{ij} \quad \forall (i, j) \in A$$

Megengedett folyam feladatok

b-folyam

Adott egy $G = (N, A)$ irányított gráf, $u : A \rightarrow \mathbb{N}$ kapacitásfüggvény, és $b : N \rightarrow \mathbb{N}$ függvény. Keressünk egy $x : A \rightarrow \mathbb{N}$ élsúlyozást, amire

$$\sum_{j:(i,j) \in A} x_{ij} - \sum_{j:(j,i) \in A} x_{ji} = b(i) \quad \forall i \in N$$
$$0 \leq x_{ij} \leq u_{ij} \quad \forall (i,j) \in A$$

b-folyam felső korlát nélkül

Adott egy $G = (N, A)$ irányított gráf és $b : N \rightarrow \mathbb{N}$ függvény. Keressünk egy $x : A \rightarrow \mathbb{N}$ élsúlyozást, amire

$$\sum_{j:(i,j) \in A} x_{ij} - \sum_{j:(j,i) \in A} x_{ji} = b(i) \quad \forall i \in N$$
$$0 \leq x_{ij} \quad \forall (i,j) \in A$$

Megengedett folyam feladatok

b -folyam

Adott egy $G = (N, A)$ irányított gráf, $u : A \rightarrow \mathbb{N}$ kapacitásfüggvény, és $b : N \rightarrow \mathbb{N}$ függvény. Keressünk egy $x : A \rightarrow \mathbb{N}$ élsúlyozást, amire

$$\sum_{j:(i,j) \in A} x_{ij} - \sum_{j:(j,i) \in A} x_{ji} = b(i) \quad \forall i \in N$$
$$0 \leq x_{ij} \leq u_{ij} \quad \forall (i,j) \in A$$

b -folyam felső korlát nélkül

Adott egy $G = (N, A)$ irányított gráf és $b : N \rightarrow \mathbb{N}$ függvény. Keressünk egy $x : A \rightarrow \mathbb{N}$ élsúlyozást, amire

$$\sum_{j:(i,j) \in A} x_{ij} - \sum_{j:(j,i) \in A} x_{ji} = b(i) \quad \forall i \in N$$
$$0 \leq x_{ij} \quad \forall (i,j) \in A$$

Megengedett folyam feladatok alsó korláttal

F értékű folyam alsó korláttal

Adott egy $G = (N, A)$ irányított gráf, $l, u : A \rightarrow \mathbb{N}$ kapacitásfüggvények, $s, t \in N$ pontok (forrás és nyelő) és egy $F \in \mathbb{N}$ folyamérték. Keressünk egy $x : A \rightarrow \mathbb{N}$ élsúlyozást, amire

$$\sum_{j:(i,j) \in A} x_{ij} - \sum_{j:(j,i) \in A} x_{ji} = \begin{cases} F & , \text{ ha } i = s \\ -F & , \text{ ha } i = t, \text{ egyébként} \\ 0 & \end{cases} \quad \forall i \in N$$
$$l_{ij} \leq x_{ij} \leq u_{ij} \quad \forall (i, j) \in A$$

b -folyam alsó korláttal

Adott egy $G = (N, A)$ irányított gráf, $l, u : A \rightarrow \mathbb{N}$ kapacitásfüggvények, és $b : N \rightarrow \mathbb{N}$ függvény. Keressünk egy $x : A \rightarrow \mathbb{N}$ élsúlyozást, amire

$$\sum_{j:(i,j) \in A} x_{ij} - \sum_{j:(j,i) \in A} x_{ji} = b(i) \quad \forall i \in N$$
$$l_{ij} \leq x_{ij} \leq u_{ij} \quad \forall (i, j) \in A$$

Megengedett folyam feladatok alsó korláttal

F értékű folyam alsó korláttal

Adott egy $G = (N, A)$ irányított gráf, $l, u : A \rightarrow \mathbb{N}$ kapacitásfüggvények, $s, t \in N$ pontok (forrás és nyelő) és egy $F \in \mathbb{N}$ folyamérték. Keressünk egy $x : A \rightarrow \mathbb{N}$ élsúlyozást, amire

$$\sum_{j:(i,j) \in A} x_{ij} - \sum_{j:(j,i) \in A} x_{ji} = \begin{cases} F & , \text{ ha } i = s \\ -F & , \text{ ha } i = t, \text{ egyébként} \\ 0 & \end{cases} \quad \forall i \in N$$
$$l_{ij} \leq x_{ij} \leq u_{ij} \quad \forall (i, j) \in A$$

b -folyam alsó korláttal

Adott egy $G = (N, A)$ irányított gráf, $l, u : A \rightarrow \mathbb{N}$ kapacitásfüggvények, és $b : N \rightarrow \mathbb{N}$ függvény. Keressünk egy $x : A \rightarrow \mathbb{N}$ élsúlyozást, amire

$$\sum_{j:(i,j) \in A} x_{ij} - \sum_{j:(j,i) \in A} x_{ji} = b(i) \quad \forall i \in N$$
$$l_{ij} \leq x_{ij} \leq u_{ij} \quad \forall (i, j) \in A$$

Megengedett folyam feladatok alsó korláttal

b -folyam alsó korláttal

Adott egy $G = (N, A)$ irányított gráf, $l, u : A \rightarrow \mathbb{N}$ kapacitásfüggvények, és $b : N \rightarrow \mathbb{N}$ függvény. Keressünk egy $x : A \rightarrow \mathbb{N}$ élsúlyozást, amire

$$\sum_{j:(i,j) \in A} x_{ij} - \sum_{j:(j,i) \in A} x_{ji} = b(i) \quad \forall i \in N$$
$$l_{ij} \leq x_{ij} \leq u_{ij} \quad \forall (i, j) \in A$$

Megengedett áram feladat

Adott egy $G = (N, A)$ irányított gráf és $l, u : A \rightarrow \mathbb{N}$ kapacitásfüggvények. Keressünk egy $x : A \rightarrow \mathbb{N}$ élsúlyozást, amire

$$\sum_{j:(i,j) \in A} x_{ij} - \sum_{j:(j,i) \in A} x_{ji} = 0 \quad \forall i \in N$$
$$l_{ij} \leq x_{ij} \leq u_{ij} \quad \forall (i, j) \in A$$

Megengedett folyam feladatok alsó korláttal

b -folyam alsó korláttal

Adott egy $G = (N, A)$ irányított gráf, $l, u : A \rightarrow \mathbb{N}$ kapacitásfüggvények, és $b : N \rightarrow \mathbb{N}$ függvény. Keressünk egy $x : A \rightarrow \mathbb{N}$ élsúlyozást, amire

$$\sum_{j:(i,j) \in A} x_{ij} - \sum_{j:(j,i) \in A} x_{ji} = b(i) \quad \forall i \in N$$
$$l_{ij} \leq x_{ij} \leq u_{ij} \quad \forall (i, j) \in A$$

Megengedett áram feladat

Adott egy $G = (N, A)$ irányított gráf és $l, u : A \rightarrow \mathbb{N}$ kapacitásfüggvények. Keressünk egy $x : A \rightarrow \mathbb{N}$ élsúlyozást, amire

$$\sum_{j:(i,j) \in A} x_{ij} - \sum_{j:(j,i) \in A} x_{ji} = 0 \quad \forall i \in N$$
$$l_{ij} \leq x_{ij} \leq u_{ij} \quad \forall (i, j) \in A$$

Altalános megengedett folyamok

$\leq b$ -folyam alsó korláttal

Adott egy $G = (N, A)$ irányított gráf, $l, u : A \rightarrow \mathbb{N}$ kapacitásfüggvények, és $b : N \rightarrow \mathbb{N}$ függvény. Keressünk egy $x : A \rightarrow \mathbb{N}$ élsúlyozást, amire

$$\sum_{j:(i,j) \in A} x_{ij} - \sum_{j:(j,i) \in A} x_{ji} \leq b(i) \quad \forall i \in N$$
$$l_{ij} \leq x_{ij} \leq u_{ij} \quad \forall (i, j) \in A$$

$b_l - b_u$ -folyam alsó korláttal (legáltalánosabb alak)

Adott egy $G = (N, A)$ irányított gráf, $l, u : A \rightarrow \mathbb{N}$ kapacitásfüggvények, és $b_l, b_u : N \rightarrow \mathbb{N}$ függvények. Keressünk egy $x : A \rightarrow \mathbb{N}$ élsúlyozást, amire

$$b_l(i) \leq \sum_{j:(i,j) \in A} x_{ij} - \sum_{j:(j,i) \in A} x_{ji} \leq b_u(i) \quad \forall i \in N$$
$$l_{ij} \leq x_{ij} \leq u_{ij} \quad \forall (i, j) \in A$$

Megengedett folyamok

Tétel (HF8/4))

A fenti folyamfeladatok egymással ekvivalensek.

Tétel

Adott egy $G = (N, A)$ irányított gráf, $l, u : A \rightarrow \mathbb{N}$ kapacitásfüggvények és tegyük fel, hogy $l_{ij} \leq u_{ij}$ minden $(i, j) \in A$ élre. E feltevések mellett pontosan akkor létezik megengedett áram, ha minden $S \subseteq N$ halmazra

$$\sum_{(i,j) \in A[\bar{S}, S]} l_{ij} \leq \sum_{(i,j) \in A[S, \bar{S}]} u_{ij}$$

Házi feladat 8/5.

Adjunk meg hasonló szükséges és elégséges feltételeket az áramfeladat általánosításaira (b -folyam, $\leq b$ -folyam ill. $b_l - b_u$ -folyam alsó illetve felső korláttal)

Szintező algoritmus az megengedett áram feladatra

- Minden $i \in N$ pontra nyilvántartunk egy $\ell(i) \in \mathbb{N}$ szintet

Invariáns tulajdonság

- Minden $(i, j) \in A$ élre, ha $x_{ij} < u_{ij}$, akkor $\ell(i) \leq \ell(j) + 1$, azaz minden él „legfeljebb egyet mehet lefele”.
- Minden $(i, j) \in A$ élre, ha $l_{ij} < x_{ij}$, akkor $\ell(j) \leq \ell(i) + 1$, azaz minden él „legfeljebb egyet mehet felfele”.

Szintező algoritmus az megengedett áram feladatra

Szintező algoritmus az megengedett áram feladatra

```
1: procedure FEASIBLECIRCULATION( $G, l, u$ )
2:    $\ell(i) \leftarrow 0 \quad \forall i \in N$ 
3:    $x_{ij} \leftarrow l_{ij} \quad \forall (i, j) \in A$ 
4:   while  $\exists i \in N : \Delta(i) := \sum_{j:(j,i) \in A} x_{ji} - \sum_{j:(i,j) \in A} x_{ij} > 0$  do
5:     if  $\exists (i, j) \in A : \ell(j) = \ell(i) - 1$  és  $x_{ij} < u_{ij}$  then
6:        $x_{ij} \leftarrow \min(u_{ij}, x_{ij} + \Delta(i))$  ▷ Push operáció
7:     else if  $\exists (j, i) \in A : \ell(j) = \ell(i) - 1$  és  $l_{ij} < x_{ij}$  then
8:        $x_{ij} \leftarrow \max(l_{ij}, x_{ij} - \Delta(i))$  ▷ Push operáció
9:     else
10:       $\ell(i) \leftarrow \ell(i) + 1$  ▷ Emelés
11:      if  $\ell(i) = n$  then STOP (Nincs megoldás)
12:      end if
13:    end if
14:  end while
15: end procedure
```

Házi feladat 8/6.

A fenti algoritmus futási ideje $O(n^2m)$.

Házi feladat 8/7.

Ha mindig a legmagasabban levő hiányos pontot választjuk ki (hogyan?), akkor teljes algoritmus futási ideje $O(n^3)$.

A maximális folyam feladat

Definíció (Folyam)

Adott egy $G = (N, A)$ irányított gráf, $u : A \rightarrow \mathbb{N}$ kapacitásfüggvény, $s, t \in N$ pontok. Egy $x : A \rightarrow \mathbb{N}$ élsúlyozást **$s - t$ folyam**nak nevezünk, ha

$$\sum_{j:(i,j) \in A} x_{ij} - \sum_{j:(j,i) \in A} x_{ji} = 0 \quad \forall i \in N \setminus \{s, t\}$$
$$0 \leq x_{ij} \leq u_{ij} \quad \forall (i, j) \in A$$

Egy folyam **értékén** a

$$\sum_{j:(s,j) \in A} x_{sj} - \sum_{j:(j,s) \in A} x_{js}$$

számot értjük.

Maximális folyam feladat

Adott egy $G = (N, A)$ irányított gráf, $u : A \rightarrow \mathbb{N}$ kapacitásfüggvény, $s, t \in N$ pontok. Keressünk egy maximális értékű $s - t$ folyamot.

Rezidulális gráf a hagyományos folyamfeladatra

Megállapodás

Ha $(i, j) \in A$, akkor $(j, i) \in A$ is teljesül.

Definíció (Reziduális kapacitás)

- $r_{ij} := u_{ij} - x_{ij} + x_{ji}$
- *Reziduális gráf:* $G_x := (N, A_x)$, ahol $A_x := \{(i, j) \in A : r_{ij} > 0\}$.

Egy folyam algoritmus megadásakor dolgozhatunk a folyamértékek helyett a reziduális kapacitásokkal is.

Tétel

Egy x $s - t$ -folyam pontosan akkor maximális, ha $G(x)$ -ben nincs $s \rightsquigarrow t$ út.

Tétel

A maximális $s - t$ -folyam értéke megegyezik a minimális $s - t$ vágás értékével.

Távolság címke

Definíció (Távolság címke)

Egy $d : N \rightarrow \mathbb{N}$ függvényt *távolság címkének* nevezünk, ha

- $d(t) := 0$
- $d(i) \leq d(j) + 1 \quad \forall (i, j) \in A_x.$

Állítás

$d(i)$ egy alsó becslés a minimális $i \rightsquigarrow t$ útra G_x -ben.

Állítás

Ha $d(s) \geq n$, akkor nincs $s \rightsquigarrow t$ út G_x -ben, azaz x maximális $s - t$ -folyam.

Definíció

Egy $(i, j) \in A_x$ élel *megengedettnek* (precízebben *d-megengedettnek*) nevezünk, ha $d(i) = d(j) + 1$.

Definíció (Előfolyam)

Adott egy $G = (N, A)$ irányított gráf, $u : A \rightarrow \mathbb{N}$ kapacitásfüggvény, $s, t \in N$ pontok. Egy $x : A \rightarrow \mathbb{N}$ élsúlyozást **$s - t$ előfolyamnak** nevezünk, ha

$$\begin{aligned} e(i) &\geq 0 & \forall i \in N \setminus \{s, t\} \\ 0 \leq x_{ij} &\leq u_{ij} & \forall (i, j) \in A, \end{aligned}$$

ahol

$$e(i) := \sum_{j:(j,i) \in A} x_{ji} - \sum_{j:(i,j) \in A} x_{ij}$$

az i pont **többlete**. Egy előfolyam **nagyságán** az $e(t)$ értéket értjük.

Állítás

A maximális előfolyam magysága megegyezik a maximális folyamértékkel.

Szintező algoritmus maximális folyam feladatra

- Minden $i \in N$ pontra nyilvántartunk egy $\ell(i) \in \mathbb{N}$ szintet

Invariáns tulajdonság

- Minden $(i, j) \in A$ élre, ha $x_{ij} < u_{ij}$, akkor $\ell(i) \leq \ell(j) + 1$, azaz minden él „legfeljebb egyet mehet lefele”.
- Minden $(i, j) \in A$ élre, ha $0 < x_{ij}$, akkor $\ell(j) \leq \ell(i) + 1$, azaz minden él „legfeljebb egyet mehet felfele”.

Szintező algoritmus maximális folyam feladatra

Szintező algoritmus az maximális folyam feladatra

```
1: procedure PREFLOWPUSH( $G, l, u$ )
2:    $\ell(s) \leftarrow n; \ell(i) \leftarrow 0 \quad \forall i \in N \setminus \{s\}$ 
3:    $x_{sj} \leftarrow u_{sj} \quad \forall (s, j) \in A$ 
4:    $x_{ij} \leftarrow 0 \quad \forall (i, j) \in A, i \neq s$ 
5:   while  $\exists i \in N : e(i) > 0$  do
6:     if  $\exists (i, j) \in A : \ell(j) = \ell(i) - 1$  és  $x_{ij} < u_{ij}$  then
7:        $x_{ij} \leftarrow \min(u_{ij}, x_{ij} + e(i))$  ▷ Push operáció
8:     else if  $\exists (j, i) \in A : \ell(j) = \ell(i) - 1$  és  $0 < x_{ij}$  then
9:        $x_{ij} \leftarrow \max(0, x_{ij} - e(i))$  ▷ Push operáció
10:    else
11:       $\ell(i) \leftarrow \ell(i) + 1$  ▷ Nem figyeljük, hogy elérünk-e egy szintet!
12:    end if
13:  end while
14: end procedure
```

Szintező algoritmus maximális folyam feladatra

- Minden $i \in N$ pontra nyilvántartunk egy $\ell(i) \in \mathbb{N}$ értéket

Invariáns tulajdonság

ℓ legyen távolság címke.

Szintező algoritmus maximális folyam feladatra

Szintező algoritmus az maximális folyam feladatra

```
1: procedure PREFLOWPUSH( $G, l, u$ )
2:    $\ell(s) \leftarrow n; \ell(i) \leftarrow 0 \quad \forall i \in N \setminus \{s\}$ 
3:    $x_{sj} \leftarrow u_{sj} \quad \forall (s, j) \in A$ 
4:    $x_{ij} \leftarrow 0 \quad \forall (i, j) \in A, i \neq s$ 
5:   while  $\exists i \in N : e(i) > 0$  do
6:     if  $\exists (i, j) \in A_x$   $\ell$ -megengedett then
7:       Push( $i, j, \min(e(i), r_{ij})$ ).
8:     else
9:        $\ell(i) \leftarrow \min\{d(j) + 1 : (i, j) \in A_x\}$ 
10:    end if
11:  end while
12: end procedure
13: procedure PUSH( $i, j, \delta$ )
14:    $r_{ij} \leftarrow r_{ij} - \delta$ 
15:    $r_{ji} \leftarrow r_{ji} + \delta$ 
16: end procedure
```

▷ Lehet okosabban!

▷ Emelés

Helyesség és futásidő

Tétel

Amikor az algoritmus megáll, akkor x egy maximális $s - t$ -folyam lesz.

Tétel

Az algoritmus futásideje $O(n^2m)$.

Bizonyítás

Definíció

*Egy push operáció **telítő**, ha éppen r_{ij} értéket pusholunk.*

- *Az $\ell(i)$ értékek nem nőhetnek $2n$ fölé.*
- *Legfeljebb nm telítő push lehet.*
- *Legfeljebb $O(n^2m)$ nem telítő push lehet.*
 - *Tekintsük $\Phi := \sum_{i \in I} \ell(i) - t$, ahol $I := \{i \in N : e(i) > 0\}$*

Pusholható él keresése

„Aktuális élek”

- Minden pontra lerögzítjük a kimenő élek egy sorrendjét
- Az éleket ebben a sorrendben vizsgáljuk
- Minden pontra megjegyezzük melyik élet vizsgáltuk utoljára (**aktuális él**)
 - Ha egy pontot feldolgozunk, az aktuális éltől (és nem előlről) kezdjük vizsgálni az éleken.
 - *Ha a lista végére értünk, akkor nincs push-olható él*
 - Emeléskor visszaállítjuk az aktuális élet a lista elejére

Állítás

Az aktuális él előtti éleken nem lehet (nem keletkezhet új) megengedett él.

Következmény

Minden élet legfeljebb $2n$ -szer vizsgálunk át.

Pusholható él keresése

„Aktuális élek”

- Minden pontra lerögzítjük a kimenő élek egy sorrendjét
- Az éleket ebben a sorrendben vizsgáljuk
- Minden pontra megjegyezzük melyik élet vizsgáltuk utoljára (**aktuális él**)
 - Ha egy pontot feldolgozunk, az aktuális éltől (és nem előlről) kezdjük vizsgálni az éleken.
 - *Ha a lista végére értünk, akkor nincs push-olható él*
 - Emeléskor visszaállítjuk az aktuális élet a lista elejére

Állítás

Az aktuális él előtti éleken nem lehet (nem keletkezhet új) megengedett él.

Következmény

Minden élet legfeljebb $2n$ -szer vizsgálunk át.

Pusholható él keresése

„Aktuális élék”

- Minden pontra lerögzítjük a kimenő élek egy sorrendjét
- Az éleket ebben a sorrendben vizsgáljuk
- Minden pontra megjegyezzük melyik élet vizsgáltuk utoljára (**aktuális él**)
 - Ha egy pontot feldolgozunk, az aktuális éltől (és nem előlről) kezdjük vizsgálni az éleken.
 - *Ha a lista végére értünk, akkor nincs push-olható él*
 - Emeléskor visszaállítjuk az aktuális élet a lista elejére

Állítás

Az aktuális él előtti éleken nem lehet (nem keletkezhet új) megengedett él.

Következmény

Minden élet legfeljebb $2n$ -szer vizsgálunk át.

Tétel

Ha mindig a legmagasabban levő hiányos pontot választjuk ki (hogyan?), akkor teljes algoritmus futási ideje $O(n^3)$.

Házi feladat 9/1. Legfontosabb él

Adott egy $G = (V, A)$ irányított gráf, $u : E \rightarrow \mathbb{R}_+$ kapacitásfüggvény és $s, t \in G$ pontok. Egy élet **legfontosabb**nak nevezzünk, ha törölve a maximális $s - t$ folyam értéke a legnagyobb mértékben csökken (az összes lehetséges éltörléshez képest). Bizonyítsd vagy ellenpéldával cáfold az alábbi állításokat:

- 1 Egy legfontosabb él u_{ij} értéke maximális (az összes él között).
- 2 Egy legfontosabb él x_{ij} értéke maximális (az összes él között) valamely x maximális folyamára nézve.
- 3 Egy legfontosabb él egy minimális vágás maximális u_{ij} értékkel bíró éle.
- 4 Ha egy él nincs benne egy minimális vágásban, akkor nem lehet legfontosabb.
- 5 Egy gráfban több legfontosabb él is lehet.

Házi feladat 9/2. Legkevésbé fontos él

Adott egy $G = (V, A)$ irányított gráf, $u : E \rightarrow \mathbb{R}_+$ kapacitásfüggvény és $s, t \in G$ pontok. Egy él **legkevésbé fontos**nak nevezzünk, ha törölve a maximális $s - t$ folyam értéke a legkisebb mértékben csökken (az összes lehetséges éltörléshez képest). Bizonyítsd vagy ellenpéldával cáfold az alábbi állításokat:

- 1 Ha $x_{ij} = 0$ valamely x maximális folyamra nézve, akkor (i, j) legkevésbé fontos.
- 2 Ha $x_{ij} = \min\{x_{uv} : (u, v) \in A\}$ valamely x maximális folyamra nézve, akkor (i, j) legkevésbé fontos.
- 3 Ha egy él benne van egy minimális vágásban, akkor nem lehet legkevésbé fontos.

Házi feladat 9/3.

- 1 Egy él **felfele kritikus**, ha kapacitását megnövelve nő a maximális folyamérték. Van minden hálózatban felfele kritikus él? Adjunk algoritmust a felfele kritikus élek megkeresésére. (Legyen hatékonyabb, mint m maximális folyam megkeresésé.
- 2 Egy él **lefele kritikus**, ha kapacitását (bármilyen picit) csökkentve csökken a maximális folyamérték. Ugyanazok-e a felfele és a lefele kritikus élek? Ha nem akkor adjunk algoritmust a lefele kritikus élek megkeresésére.

Házi feladat 9/4.

Bizonyítsd vagy ellenpéldával cáfold az alábbi állításokat:

- 1 Ha valamennyi kapacitás egy α érték többszöröse^a, akkor minden maximális folyamán az élkapacitások szintén a α többszörösei.
- 2 Ha minden él kapacitását α -val megnöveljük, akkor a maximális folyamérték α többszörösével fog nőni.
- 3 Legyen v^* a maximális folyamérték és v' a t -be bemenő éleken az aktuális folyamérték a PreflowPush algoritmus egy fázisában. Ekkor
$$v^* - v' \leq \sum_{i \in N \setminus \{s, t\}} e(i).$$
- 4 Ha minden $i \in N$ pontra $d(i)$ egy alsó becslés a legrövidebb $i \rightsquigarrow t$ út hosszára a reziduális gráfban, akkor $d(i)$ egy távolság címke.

^aaz egyszeres is többszörös

Házi feladat 9/5.

Adott egy $G = (V, A)$ irányított gráf, $u : E \rightarrow \mathbb{N}$ kapacitásfüggvény és $s, t \in G$ pontok. Adjunk algoritmust a legkevesebb élt tartalmazó minimális $s - t$ vágás megtalálására!

Házi feladat 9/6.

Egy valós szám **kerekítésén** az alsó vagy a felső egészrészét értjük. Bizonyítsuk be, hogy egy mátrix elemeit lehet úgy kerekíteni, hogy valamennyi sorösszeg és valamennyi oszlopösszeg is az eredeti érték kerekítése legyen (azaz a sorösszegek és az oszlopösszegek kevesebb, mint 1-gyel változnak). Adjunk algoritmust egy ilyen kerekítés megtalálására! Elérhető-e az, hogy az eddigieken felül a mátrix összes elemének az összege is az eredeti érték kerekítése legyen?

Házi feladat 9/7.

Egy hajózási társaságnak k romlandó áru szállítását kell lebonyolítani a $s_1 - t_1, s_2 - t_2, \dots, s_k - t_k$ kikötők között. Ismeretes a hajózási idő az egyes kikötők között rakománnyal teli, illetve üres hajók esetén is. Adott továbbá minden i árura, hogy mikor kell a szállítóhajónak az s_i kikötőbe az áru berakodására megérkeznie (előbb érkezhet, de nem késhet). Meg kell határoznunk, hogy minimálisan hány hajóra van szükség ezeknek a szállításoknak a lebonyolítására. Vezessük vissza ezt a problémát egy folyamfeladat megoldására!

Házi feladat 9/8.

Adott k darab „modul” (számítógépen végrehajtandó feladat), amit egy számítógép két processzorának segítségével szeretnénk megoldani. A processzorok különbözők, ennek megfelelően az i . feladat végrehajtási költsége az első processzoron α_i , míg a másikon β_i . Emellett a modulok kommunikálnak is egymással: Ha az i . és a j . modul nem ugyanazon a processzoron fut, az c_{ij} kommunikációs költséget okoz. (Ha ugyanazon a processzoron futnak, akkor a kommunikációs költség 0.)

A feladatunk a modulokat úgy hozzárendelni az egyes processzorokhoz, hogy minimalizáljuk a végrehajtási és kommunikációs költségek összegét.

Vezessük vissza ezt a problémát egy folyamfeladat megoldására!