Hálózati folyamok és alkalmazásaik

ELTE-TTK, Operációkutatási Tanszék

Jüttner Alpár
alpar@cs.elte.hu

ELTE-TTK, Operációkutatási tanszék

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 $w_1, w_2, \ldots, w_p$, az érték-k pedig $c_1, c_2, \ldots, c_p$. 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”.

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

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 $w_1, w_2, \ldots, w_p$, az érték-k pedig $c_1, c_2, \ldots, c_p$. 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”.

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

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 $w_1, w_2, \ldots, w_p$, az érték-k pedig $c_1, c_2, \ldots, c_p$. 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”.

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

Definíció (Irányított 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* $[t(a)]$
- $\rho_G(v) = \rho(v) := \{(v, j) \in A\}$
- $\delta_G(v) = \delta(v) := \{(i, v) \in A\}$
- $G' = (N', A')$, *részgráf, ha* $N' \subseteq N$ *és* $A' \subseteq A$.
- $G' = (N', A')$ *feszített részgráf, ha* $N' = N$ *és* $A' \subseteq A$.
- $G' = (N', A')$ *indukált részgráf, ha* $N' \subseteq N$ *és* $A' := \{(i, j) \in A | i, j \in N'\}$.

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

Gráfok II.

Definíció (Utak, Séták)

- $\{a_1, a_2, \ldots, a_k\} \subseteq A$, *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* $\{a_1, a_2, \ldots, a_{k-1}\}$ *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.*

Gráf (hálózat) reprezentációk

- Csúcs-csúcs adjacencia mátrix
  - $M \in \{0,1\}^{N \times N}, \quad m_{i,j} := \begin{cases} 1, & \text{ha } (i,j) \in A \\ 0, & \text{egyébként} \end{cases}$
- Csúcs-él incidencia mátrix
  - $M \in \{0,1\}^{N \times A}, \quad m_{i,a} := \begin{cases} 1, & \text{ha } \exists j \in N : a = (i,j) \\ -1, & \text{ha } \exists j \in N : a = (j,i) \\ 0, & \text{egyébként} \end{cases}$
- Éllista reprezentáció (Adjacency list)
- Ki-csillag, be-csillag (Forward and Reverse Star) reprezentáció

Feladatok (dinamikus programozás)

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 kérdezünk?

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.

Reklám

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:      ∀v ∈ N-re m(v) := 0                          ▷ Minden csúcs jelöletlen (eléretlen)
3:      m(s) := 1                                    ▷ kivéve s
4:      pred(s) := 0; order(s) := 1
5:      next := 1
6:      LIST := {s}                                  ▷ Feldolgozandó csúcsok listája
7:      while LIST ≠ ∅ do
8:          Kiválasztunk egy i ∈ LIST csúcsot
9:          if ∃(i, j) ∈ δ_G(i),    m(j) = 0 then
10:             m(j) := 1                            ▷ j-t megjelöljük, mint elértet
11:             pred(j) := i
12:             next := next + 1; order(j) := next
13:             LIST := LIST ∪ {j}
14:         else
15:             LIST := LIST \ {i}
16:         end if
17:     end while
18: end procedure
```

Gráf keresés

```
1: procedure SEARCH(G, s)
2:     ∀v ∈ N-re m(v) := 0                          ▷ Minden csúcs jelöletlen (eléretlen)
3:     m(s) := 1                                    ▷ kivéve s
4:     pred(s) := 0; order(s) := 1
5:     next := 1
6:     LIST := {s}                                  ▷ Feldolgozandó csúcsok listája
7:     while LIST ≠ ∅ do
8:         Kiválasztunk egy i ∈ LIST csúcsot
9:         if ∃(i, j) ∈ δ_G(i),   m(j) = 0 then
10:            m(j) := 1                            ▷ j-t megjelöljük, mint elértet
11:            pred(j) := i
12:            next := next + 1; order(j) := next
13:            LIST := LIST ∪ {j}
14:        else
15:            LIST := LIST \ {i}
16:        end if
17:    end while
18: end procedure
```

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 : $N \longrightarrow N$ 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.

Gráf keresés II.

Állítás

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

Állítás (házi feladat 1/4.)

pred : $N \longrightarrow N$ 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.

Gráf keresés II.

Állítás

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

Állítás (házi feladat 1/4.)

pred : $N \longrightarrow N$ 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.

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 $order(j) > order(i)$.*
- *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 $order(j) > order(i)$.*
- *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 $order(j) > order(i)$.*
- *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.