Grafy

Základy algoritmizace – 7. cvičení

Zuzana Petříčková

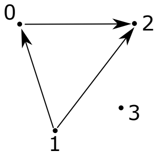
7. dubna 2020

Grafy a jejich reprezentace v programu

Graf

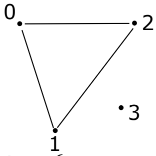
- dvojice $G = (V, E)$, kde
  - V je množina vrcholů, $|V| = n$, obvykle $V = \{0, ..., n-1\}$
  - E je množina hran, $|E| = m$
- druhy:
  - souvislý x nesouvislý
  - orientovaný x neorientovaný

orientovaný graf



$V = \{0, 1, 2, 3\}$
$E = \{(0, 2), (1, 0), (1, 2)\}$

neorientovaný graf



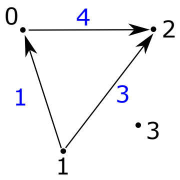
$V = \{0, 1, 2, 3\}$
$E = \{\{0, 2\}, \{1, 0\}, \{1, 2\}\}$

Grafy a jejich reprezentace v programu

Graf

- dvojice $G = (V, E)$, kde
  - V je množina vrcholů, $|V| = n$, obvykle $V = \{0, ..., n-1\}$
  - E je množina hran, $|E| = m$
- druhy:
  - hranově ohodnocený x hranově neohodnocený

hranově ohodnocený graf



$w : E \rightarrow < 0, \infty)$

w_{ij} ... váha hrany (i, j) (např. délka hrany, kapacita, průtok, cena za průchod aj.)

$V = \{0, 1, 2, 3\}$
$E' = \{(0, 2, 4), (1, 0, 1), (1, 2, 3)\}$

Grafy a jejich reprezentace v programu

Samostatná práce: nastudujte si skripta

- kapitola 2.4.1 (Graf), strana 74-76
- kapitola 3.2 (Hladový algoritmus), strana 83-88 (hlavně příklad 3.4, Hledání nejkratší cesty (Dijkstrův algoritmus))
- kapitola 8.1.1 (Rozklad grafu na komponenty), strana 232-234

Reprezentace grafu v programu

Jak můžeme grafy reprezentovat v programu?

1. matice sousednosti (adjacentní matice)
2. seznamy následníků (sousedů)
3. seznamy hran
4. matice incidence

   … a další způsoby

Jaká reprezentace je nejlepší?

- záleží na tom, jakou úlohu řešíme
- různé operace mají pro různé datové reprezentace různou časovou složitost

Reprezentace grafu v programu

1. Matice sousednosti (adjacentní matice)

- vrcholy: $0, \dots, n-1$
- hrany: matice A tvaru *nxn*

$$a_{ij} = \begin{cases} 0 & : (i,j) \notin E \\ 1 & : (i,j) \in E \end{cases}$$

Pro orientovaný graf:
$V = \{0,1,2,3\}$
$E = \{(0,2),(1,0),(1,2)\}$

$$A = \begin{pmatrix} 0 & 0 & 1 & 0 \\ 1 & 0 & 1 & 0 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \end{pmatrix}$$

Pro neorientovaný graf:
$V = \{0,1,2,3\}$
$E = \{\{0,2\},\{1,0\},\{1,2\}\}$

$$A = \begin{pmatrix} 0 & 1 & 1 & 0 \\ 1 & 0 & 1 & 0 \\ 1 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 \end{pmatrix}$$

Reprezentace grafu v programu

1. Matice sousednosti (adjacentní matice)

- vrcholy: $0, ..., n-1$
- hrany: matice A tvaru *nxn*

$$a_{ij} = \begin{cases} 0 & : (i,j) \notin E \\ w_{ij} & : (i,j) \in E \end{cases}$$

Pro orientovaný, hranově ohodnocený graf:
$V = \{0, 1, 2, 3\}$
$E' = \{(0, 2, 4), (1, 0, 1), (1, 2, 3)\}$

$$A = \begin{pmatrix} 0 & 0 & 4 & 0 \\ 1 & 0 & 3 & 0 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \end{pmatrix}$$

Reprezentace grafu v programu

2. Seznamy následníků (sousedů)

- vrcholy: $0, \ldots, n-1$
- hrany: pole délky n, na indexu i je (spojový) seznam následníků/sousedů i

Pro orientovaný graf:
$V = \{0, 1, 2, 3\}$
$E = \{(0, 2), (1, 0), (1, 2)\}$

$$\begin{bmatrix} 0: & 2 & \\ 1: & 0, & 2 \\ 2: & \emptyset & \\ 3: & \emptyset & \end{bmatrix}$$

Pro neorientovaný graf:
$V = \{0, 1, 2, 3\}$
$E = \{\{0, 2\}, \{1, 0\}, \{1, 2\}\}$

$$\begin{bmatrix} 0: & 1, & 2 \\ 1: & 0, & 2 \\ 2: & 0, & 1 \\ 3: & \emptyset & \end{bmatrix}$$

Reprezentace grafu v programu

2. Seznamy následníků (sousedů)

- vrcholy: $0, ..., n-1$
- hrany: pole délky n, na indexu *i* je (spojový) seznam následníků/sousedů *i*

Pro orientovaný, hranově ohodnocený graf:
$V = \{0, 1, 2, 3\}$
$E' = \{(0, 2, 4), (1, 0, 1), (1, 2, 3)\}$

$$\begin{bmatrix} 0: & (2,4) & \\ 1: & (0,1), & (2,3) \\ 2: & \emptyset & \\ 3: & \emptyset & \end{bmatrix}$$

Reprezentace grafu v programu

2.A Komprimované seznamy následníků (sousedů)

- vrcholy: $0, \ldots, n-1$
- hrany:
  - pro hranově ohodnocený graf: pole w délky m (váhy hran)
  - pole k délky m (koncové vrcholy hran)
  - pole z délky (n+1) (začátky hran)

následníci vrcholu i jsou v poli k na indexech $z[i], \ldots, (z[i+1]-1)$

Pro orientovaný graf:
$V = \{0, 1, 2, 3\}$
$E = \{(0, 2), (1, 0), (1, 2)\}$

$w : [1, 1, 1]$
$k : [2, 0, 2]$
$z : [0, 1, 3, 3, 3]$

Pro neorientovaný graf:
$V = \{0, 1, 2, 3\}$
$E = \{\{0, 2\}, \{1, 0\}, \{1, 2\}\}$

$w : [1, 1, 1, 1, 1, 1]$
$k : [1, 2, 0, 2, 0, 1]$
$z : [0, 2, 4, 6, 6]$

Reprezentace grafu v programu

2.A Komprimované seznamy následníků (sousedů)

- vrcholy: $0, ..., n-1$
- hrany:
  - pro hranově ohodnocený graf: pole w délky m (váhy hran)
  - pole k délky m (koncové vrcholy hran)
  - pole z délky (n+1) (začátky hran)

následníci vrcholu i jsou v poli k na indexech $z[i], ..., (z[i+1]-1)$

Pro hranově ohodnocený orientovaný graf:
$V = \{0, 1, 2, 3\}$
$E' = \{(0, 2, 4), (1, 0, 1), (1, 2, 3)\}$

$w : [4, 1, 3]$
$k : [2, 0, 2]$
$z : [0, 1, 3, 3, 3]$

Reprezentace grafu v programu

3. Seznam hran

- vrcholy: $0, \ldots, n-1$
- hrany: pole nebo (spojový) seznam hran (délky m)

Pro orientovaný graf:
$V = \{0, 1, 2, 3\}$
$E = \{(0, 2), (1, 0), (1, 2)\}$

Pro neorientovaný graf:
$V = \{0, 1, 2, 3\}$
$E = \{\{0, 2\}, \{1, 0\}, \{1, 2\}\}$

Pro orientovaný, hranově ohodnocený graf:
$V = \{0, 1, 2, 3\}$
$E' = \{(0, 2, 4), (1, 0, 1), (1, 2, 3)\}$

Reprezentace grafu v programu

4. Matice incidence

- vrcholy: $0, \ldots, n-1$
- hrany: matice B tvaru *nxm*

$$b_{ij} = \begin{cases} 1 & : \text{ i je počáteční vrchol j-té hrany} \\ -1 & : \text{ i je koncový vrchol j-té hrany} \\ 0 & : \text{ jinak} \end{cases}$$

Pro orientovaný graf:
$V = \{0, 1, 2, 3\}$
$E = \{(0, 2), (1, 0), (1, 2)\}$

$$B = \begin{pmatrix} 1 & -1 & 0 \\ 0 & 1 & 1 \\ -1 & 0 & -1 \\ 0 & 0 & 0 \end{pmatrix}$$

Pro neorientovaný graf:
$V = \{0, 1, 2, 3\}$
$E = \{\{0, 2\}, \{1, 0\}, \{1, 2\}\}$

$$B = \begin{pmatrix} 1 & 1 & 0 \\ 0 & 1 & 1 \\ 1 & 0 & 1 \\ 0 & 0 & 0 \end{pmatrix}$$

Reprezentace grafu v programu

4. Matice incidence

- vrcholy: $0, ..., n-1$
- hrany: matice A tvaru *nxn*

$$b_{ij} = \begin{cases} w_{ik} & : \text{ i je počáteční vrchol j-té hrany (i,k)} \\ -w_{ki} & : \text{ i je koncový vrchol j-té hrany (k,i)} \\ 0 & : \text{ jinak} \end{cases}$$

Pro orientovaný, hranově ohodnocený graf:
$V = \{0, 1, 2, 3\}$
$E' = \{(0, 2, 4), (1, 0, 1), (1, 2, 3)\}$

$$B = \begin{pmatrix} 4 & -1 & 0 \\ 0 & 1 & 3 \\ -4 & 0 & -3 \\ 0 & 0 & 0 \end{pmatrix}$$

Různé reprezentace grafu a složitost operací

- volba vhodné reprezentace záleží na typu grafu a na tom, jakou úlohu řešíme
- různé operace mají pro různé datové reprezentace různou časovou složitost

	matice sousednosti	seznamy následníků	seznam hran / mat. incidence
existuje hrana (i, j)?	$\Theta(1)$	$O(m_i)$	$O(m)$
změna váhy hrany (i, j)	$\Theta(1)$	$O(m_i)$	$O(m)$
najdi všechny následníky i	$\Theta(n)$	$\Theta(m_i)$	$\Theta(m)$
prostorová složitost	$\Theta(n^2)$	$\Theta(n+m)$	$\Theta(m)$ / $\Theta(nm)$
kdy je reprezentace výhodná?	m velké a n malé	n velké a m (resp. m_i) malé	m malé když není nutné graf procházet

n	počet vrcholů
m	počet hran
m_i	počet hran vedoucích z vrcholu i

Algoritmy nad grafem

- hodně algoritmů nad grafem je založeno na procházení grafem do šířky nabo do hloubky

Procházení grafem do hloubky (depth-first search, DFS)

- podobný postup, jako u procházení stromem do hloubky
- ale musíme si pamatovat, které vrcholy jsme již navštívili (jinak se zacyklíme)

$\rightarrow$ pro každý vrchol i zavedeme indikátor a_i:

$$a_i = \begin{cases} 1 & : \text{ i je tzv. „aktivní“ } (= \text{dosud nenavštívený}) \\ 0 & : \text{ jinak} \end{cases}$$

Možnosti

1. průchod do hloubky s využitím rekurze
2. průchod do hloubky s využitím zásobníku

Průchod grafem do hloubky s využitím rekurze

vstup: G=(V,E), počáteční vrchol $s \in V$
proměnná: $a_i, \forall i \in V$ ... indikátor, zda je vrchol i aktivní (dosud nenavštívený)
inicializace: $\forall i \in V$: nastav $a_i = 1$ // žádný vrchol zatím nebyl navštíven
zavoláme rekurzivní funkci: DFS(s, a)

```
DFS(vrchol u, pole a)
  a_u = 0 // právě jsme vrchol navštívili
  zpracuj u
  ∀v ∈ následnici u do
    if a_v // procházíme jen dosud nenavštívené vrcholy
      DFS(v, a)
    endif
  enddo
```

Průchod grafem do hloubky s využitím zásobníku

vstup: G=(V,E), počáteční vrchol $s \in V$
proměnné:
  $a_i, \forall i \in V$ ... indikátor, zda je vrchol i aktivní (dosud nenavštívený)
  zásobník na vrcholy
inicializace: $\forall i \in V$: nastav $a_i = 1$

```
a_s = 0 // navštívíme počáteční vrchol s
zásobník ← s // s vložíme na zásobník
while zásobník není prázdný do
  u ← zásobník
  zpracuj u
  ∀v ∈ následnici u do
    if a_v // procházíme jen dosud nenavštívené vrcholy
      a_v = 0
      zásobník ← v
    endif
  enddo
enddo
```

Složitost procházení grafem do hloubky

Samostudium: rozmyslete si, že následující vztahy platí:

	matice sousednosti	seznamy následníků	seznam hran / mat. incidence
čas	$\Theta(n^2)$	$\Theta(n+m)$	$\Theta(nm)$
prostor (graf)	$\Theta(n^2)$	$\Theta(n+m)$	$\Theta(m)$/ $\Theta(nm)$

n	počet vrcholů
m	počet hran
$\Theta(n)$	prostorová složitost navíc (pole a, hloubka rekurze / zásobník)

- každý vrchol navštívíme jednou, přitom projdeme (překontrolujeme) všechny následníky tohoto vrcholu
- záleží tedy hlavně na tom, s jakou složitostí je možné najít všechny následníky vrcholu
  - např. pro matici sousednosti nás stojí najít všechny následníky $\Theta(n)$ (musím projít celý řádek matice), celkem $n * \Theta(n) = \Theta(n^2)$

Aplikace procházení grafem do hloubky Rozklad grafu na komponenty souvislosti

- skripta, kapitola 8.1.1, strana 232-234
- chceme graf rozdělit na souvislé podgrafy
- můžeme aplikovat procházení grafem do hloubky (s využitím rekurze nebo zásobníku)

$\rightarrow$ pro každý vrchol i zavedeme indikátor c_i (k jaké komponentě vrchol patří, zároveň indikátor, zda je vrchol doposud nenavštívený):

$$c_i = \begin{cases} -1 & : \text{ i je tzv. ,,aktivní'' } (= \text{dosud nenavštívený}) \\ j & : \text{ i je navštívený, patří ke komponentě vrcholu j} \end{cases}$$

Aplikace procházení grafem do hloubky
Rozklad grafu na komponenty souvislosti

Algoritmus

1. zvolíme libovolný (doposud nenavštívený) vrchol s,
2. z s spustíme průchod grafem do hloubky,
   přitom všechny vrcholy navštívené z s přiřadíme do komponenty s (tj. $c_i = s$)
3. pokud zbyly v grafu doposud nenavštívené vrcholy, pokračujeme krokem 1, jinak konec

Rozklad grafu na komponenty souvislosti (řešení s využitím zásobníku)

vstup: G=(V,E), počáteční vrchol $s \in V$
proměnné:
$c_i, \forall i \in V$ ... indikátor, ke které komponentě vrchol i patří a zda je aktivní (dosud nenavštívený)
zásobník na vrcholy
inicializace: $\forall i \in V$: nastav $c_i = -1$

$\forall u \in V$
  if $c_u == -1$ // vrchol u bude počátečním vrcholem
    $c_u = u$ // komponenta vrcholu u
    zásobník $\leftarrow$ u
    while zásobník není prázdný **do**
      v $\leftarrow$ zásobník
      zpracuj v
      $\forall w \in$ následnici v **do**
      if $c_w == -1$
        $c_w = u$ // vrchol přiřadíme ke komponentě vrcholu u
        zásobník $\leftarrow$ w
      endif
    enddo
  endif
enddo

Rozklad grafu na komponenty souvislosti
Příklad

$V = \{0, 1, 2, 3, 4, 5\}$
$E = \{\{0, 4\}, \{1, 2\}, \{1, 3\}, \{2, 3\}, \{4, 5\}\}$

krok	u	v	c	zásobník
0			$c = [-1, -1, -1, -1, -1, -1]$	$z = []$
1	0		$c = [0, -1, -1, -1, -1, -1]$	$z = [0]$
1.1	0	0	$c = [0, -1, -1, -1, 0, -1]$	$z = [4]$
1.2	0	4	$c = [0, -1, -1, -1, 0, 0]$	$z = [5]$
1.3	0	5	$c = [0, -1, -1, -1, 0, 0]$	$z = []$
2	1		$c = [0, 1, -1, -1, 0, 0]$	$z = [1]$
2.1	1	1	$c = [0, 1, 1, 1, 0, 0]$	$z = [3, 2]$
2.2	1	3	$c = [0, 1, 1, 1, 0, 0]$	$z = [2]$
2.2	1	2	$c = [0, 1, 1, 1, 0, 0]$	$z = []$

Složitost rozkladu grafu na komponenty souvislosti

Samostudium: rozmyslete si, že následující vztahy platí:

	matice sousednosti	seznamy následníků	seznam hran / mat. incidence
inicializace	$\Theta(n)$	$\Theta(n)$	$\Theta(n)$
DFS na komponentu	$\Theta({n_i}^2)$	$\Theta(n_i + m_i)$	$\Theta(mn_i)$
čas celkem	$\Theta(n^2)$	$\Theta(n + m)$	$\Theta(nm)$
prostor (graf)	$\Theta(n^2)$	$\Theta(n + m)$	$\Theta(m)/\ \Theta(nm)$

n	počet vrcholů
m	počet hran
n_i	počet vrcholů v komponentě i
m_i	počet hran z vrcholů v komponentě i

Prostor navíc:

$\Theta(n)$	pole c
$\Theta(n)$	zásobník na vrcholy

Procházení grafem do šířky (breadth-first search, BFS)

- podobný postup, jako u procházení stromem do šířky
- **algoritmus vlny:** ,,do grafu budeme pumpovat vodu a budeme se dívat, jak postupuje vlna

Aplikace: Hledání nejkratší cesty v grafu (z nějakého počátečního vrcholu do všech ostatních)

1. hranově neohodnocený graf $\rightarrow$ obyčejné procházení grafem do šířky s využitím fronty
2. hranově ohodnocený graf $\rightarrow$ např. tzv. Dijkstrův algoritmus

Procházení grafem do šířky
Nejkratší cesta v hranově neohodnoceném grafu

- hledáme délky nejkratších cest z nějakého počátečního vrcholu s do všech ostatních vrcholů
- použijeme podobný postup, jako u procházení stromem do šířky pomocí fronty
- podobně jako u DFS si musíme pamatovat, které vrcholy jsme již navštívili (jinak se zacyklíme)

$\rightarrow$ pro každý vrchol i zavedeme indikátor d_i (vzdálenost i od s):

$$d_i = \begin{cases} -1 & : \text{ cesta z s do i (doposud) nebyla nalezena} \\ x, x \in R, x \geq 0 & : \text{ i je navštívený, jeho vzdálenost od s je x} \end{cases}$$

Procházení grafem do šířky
Nejkratší cesta v hranově neohodnoceném grafu

- pokud budeme chtít umět pro každý vrchol i zrekonstruovat cestu z s do i:

→ zavedeme indikátor p_i (předchůdce i na cestě z s do i):

$$p_i = \begin{cases} -1 & : \text{ cesta z s do i (doposud) nebyla nalezena} \\ j & : \text{ j je předchůdce i na jedné z nejkratších cest z s do i} \end{cases}$$

- nejkratších cest z s do i může být více, takto máme uchovanou jednu z nich

Procházení grafem do šířky
Nejkratší cesta v hranově neohodnoceném grafu

vstup: G=(V,E), počáteční vrchol $s \in V$
proměnné:
  d_v, $v \in V$ ... vzdálenost v od s
  p_v, $v \in V$ ... předchůdce v na cestě z s do v
  fronta na vrcholy
inicializace:
 $\forall v \in V : d_v = -1, p_v = -1$
 $d_s = 0$

fronta ← s
while fronta není prázdná **do**
 u ← fronta
 $\forall v \in$následnici u **do**
  if $d_v == -1$ // vrchol v zatím nebyl navštíven
   $d_v = d_u + 1, p_v = u$ // navštívíme v
   fronta ← v
  endif
 enddo
enddo

Nejkratší cesta v hranově neohodnoceném grafu Příklad

$V = \{0, 1, 2, 3, 4, 5\}$
$E = \{(0, 1), (0, 4), (1, 2), (1, 3), (2, 3), (5, 2), (5, 4)\}$
$s = 0$

krok	u	d	p	fronta
0		$[0, -1, -1, -1, -1, -1]$	$[-1, -1, -1, -1, -1, -1]$	$[0]$
1	0	$[0, 1, -1, -1, 1, -1]$	$[-1, 0, -1, -1, 0, -1]$	$[1, 4]$
2	1	$[0, 1, 2, 2, 1, -1]$	$[-1, 0, 1, 1, 0, -1]$	$[4, 2, 3]$
3	4	$[0, 1, 2, 2, 1, -1]$	$[-1, 0, 1, 1, 0, -1]$	$[2, 3]$
4	2	$[0, 1, 2, 2, 1, -1]$	$[-1, 0, 1, 1, 0, -1]$	$[3]$
5	3	$[0, 1, 2, 2, 1, -1]$	$[-1, 0, 1, 1, 0, -1]$	$[]$

Složitost procházení grafem do šířky

Samostudium: rozmyslete si, že následující vztahy platí:

	matice sousednosti	seznamy následníků	seznam hran / mat. incidence
inicializace	$\Theta(n)$	$\Theta(n)$	$\Theta(n)$
cyklus	$nx(O(1) + \text{následníci})$		
celkem	$\Theta(n^2)$	$\Theta(n+m)$	$\Theta(nm)$
prostor (graf)	$\Theta(n^2)$	$\Theta(n+m)$	$\Theta(m)/\ \Theta(nm)$

n počet vrcholů
m počet hran

Prostor navíc:
$\Theta(n)$ pomocná pole
$\Theta(n)$ fronta na vrcholy

Nejkratší cesta v hranově ohodnoceném grafu Dijkstrův algoritmus

- v hranově ohodnoceném grafu nebude předchozí postup fungovat správně, protože různé hrany mohou mít různé délky

$\rightarrow$ použijeme tzv. **Dijkstrův algoritmus**

Dijkstrův algoritmus

- patří mezi tzv. hladové algoritmy (skripta, kapitola 3.2, příklad 3.4)
  $\rightarrow$ v každém kroku ,,definitivně vyřešíme" právě jeden vrchol
- opět se jedná o **algoritmus vlny:** ,,do grafu budeme pumpovat vodu a budeme se dívat, jak postupuje vlna

Nejkratší cesta v hranově ohodnoceném grafu Dijkstrův algoritmus

- opět zavedeme pomocné indikátory d_i (vzdálenost i od s) a p_i (předchůdce i na nejkratší cestě z s do i)

$$d_i = \begin{cases} -1 & : \quad \text{cesta z s do i (doposud) nebyla nalezena} \\ x, x \in R, x \geq 0 & : \quad \text{i je navštívený,} \\ & \quad\;\; \text{horní odhad vzdálenosti i od s je x} \end{cases}$$

$$p_i = \begin{cases} -1 & : \quad \text{cesta z s do i (doposud) nebyla nalezena} \\ j & : \quad \text{j je předchůdce i na jedné z nejkratších cest z s do i} \end{cases}$$

- navíc indikátor a_i (zda je vrchol ,,aktivní'', tj. ještě přes něj nepřešla vlna)

Nejkratší cesta v hranově ohodnoceném grafu Dijkstrův algoritmus

- navíc indikátor a_i (zda je vrchol ,,aktivní", tj. ještě přes něj nepřešla vlna)

$$a_i = \begin{cases} 1 & : & \text{i zatím nebyl ,,definitivně vyřešen", je ,,aktivní"} \\ 0 & : & \text{i už byl ,,definitivně vyřešen", není ,,aktivní"} \end{cases}$$

Nejkratší cesta v hranově ohodnoceném grafu Dijkstrův algoritmus

Algoritmus (myšlenka)

1. začni do grafu z vrcholu s pumpovat vodu ($d_s = 0$)
2. označ $u = s$ (aktuální vrchol)
3. u označ jako „definitivně vyřešený" ($a_u = 0$)
4. přepočítej vzdálenosti d_v do dosud aktivních následníků vrcholu u
   Není cesta přes vrchol u kratší než doposud nalezená nejkratší cesta do v?
   (pokud pro aktuálně spočítanou vzdálenost d_v z s do v platí: $d_v > d_u + w_{uv}$, nastav $d_v = d_u + w_{uv}$ a $p_v = u$)
5. najdi vrchol u, přes který přejde vlna v dalším kroku (je to aktivní vrchol s minimální hodnotou $d_u \neq -1$)
6. pokud existuje u splňující předchozí podmínku, pokračuj krokem 3

Nejkratší cesta v hranově ohodnoceném grafu Dijkstrův algoritmus

vstup: G=(V,E), počáteční vrchol $s \in V$
proměnné:
d_v, $v \in V$ ... vzdálenost v od s
p_v, $v \in V$ ... předchůdce v na cestě z s do v
a_v, $v \in V$ ... zda je vrchol aktivní
inicializace:
$\forall v \in V : d_v = -1, p_v = -1, a_v = 1$
$d_s = 0$

u = s
do
$a_u = 0$
$\forall v \in$následnici u **do**
if a_v and $(d_v = -1$ or $d_u + w_{uv} < d_v)$
$d_v = d_u + w_{uv}$
$p_v = u$
endif
enddo
$u \leftarrow$ aktivní vrchol s min $d_u \neq -1$
while u nalezen

Dijkstrův algoritmus: Příklad

$V = \{0, 1, 2, 3, 4\}$
$E = \{(0, 1, 1), (0, 2, 1), (0, 3, 4), (2, 3, 2), (3, 0, 2), (4, 2, 5)\}$
$s = 0$

krok	u	d	p	a
0		$[0, -1, -1, -1, -1]$	$[-1, -1, -1, -1, -1]$	$[1, 1, 1, 1, 1]$
1	0	$[0, 1, 1, 4, -1]$	$[-1, 0, 0, 0, -1]$	$[0, 1, 1, 1, 1]$
2	1	$[0, 1, 1, 4, -1]$	$[-1, 0, 0, 0, -1]$	$[0, 0, 1, 1, 1]$
3	2	$[0, 1, 1, 3, -1]$	$[-1, 0, 0, 2, -1]$	$[0, 0, 0, 1, 1]$
4	3	$[0, 1, 1, 3, -1]$	$[-1, 0, 0, 2, -1]$	$[0, 0, 0, 0, 1]$

Složitost Dijkstrova algoritmu

Samostudium: rozmyslete si, že následující vztahy platí:

	matice sousednosti	seznamy následníků	seznam hran / mat. incidence
inicializace	$\Theta(n)$	$\Theta(n)$	$\Theta(n)$
cyklus	$nx(O(1)$	+následníci	+najdi minimum)
projdi následníky	$\Theta(n)$	$\Theta(m_i)$	$\Theta(m)$
najdi minimum	$\Theta(n)$	$\Theta(n)$	$\Theta(n)$
celkem	$\Theta(n^2)$	$\Theta(n^2)$	$\Theta(n*(n+m))$
prostor (graf)	$\Theta(n^2)$	$\Theta(n+m)$	$\Theta(m)/\ \Theta(nm)$

n	počet vrcholů
m	počet hran
m_i	počet hran z vrcholu i

Prostor navíc: $\Theta(n)$ pomocná pole d, p, a

Shrnutí ... různé reprezentace grafu a časová složitost operací

	matice sousednosti	seznam následníků	seznam hran (mat. incidence)
DFS (průchod do hloubky)	$\Theta(n^2)$	$\Theta(n+m)$	$\Theta(nm)$
BFS (průchod do šířky)	$\Theta(n^2)$	$\Theta(n+m)$	$\Theta(nm)$
Komponenty souvislosti	$\Theta(n^2)$	$\Theta(n+m)$	$\Theta(nm)$
Dijkstrův algoritmus	$\Theta(n^2)$	$\Theta(n^2)$	$\Theta(n^2+nm)$

n	počet vrcholů
m	počet hran
$\Theta(n)$	prostorová složitost navíc: fronta / zásobník / pomocná pole