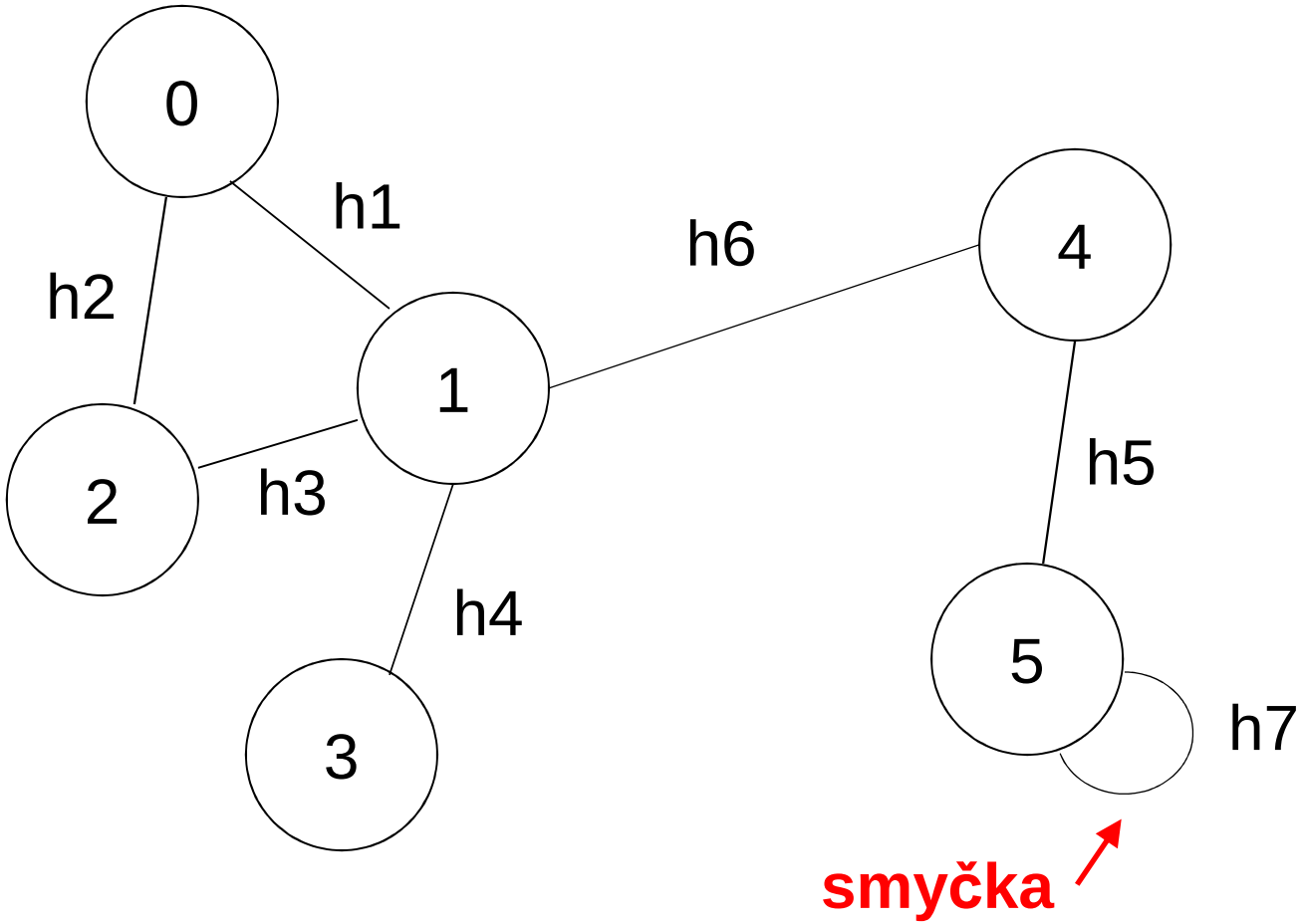
Úvod do teorie grafů

Neorientovaný graf

$G = (V,E,I)$

- V množina uzlů (vrcholů) - *vertices*
- E množina hran - *edges*
- I incidence
  - incidence je zobrazení, buď:
    - funkce: $I: E \rightarrow V \times V$
    - relace: $I \subset E \times V \times V$
      - incidence přiřadí hraně neuspořádanou dvojici uzlů
- dle českého značení též $G = (U,H,I)$

Příklad grafu



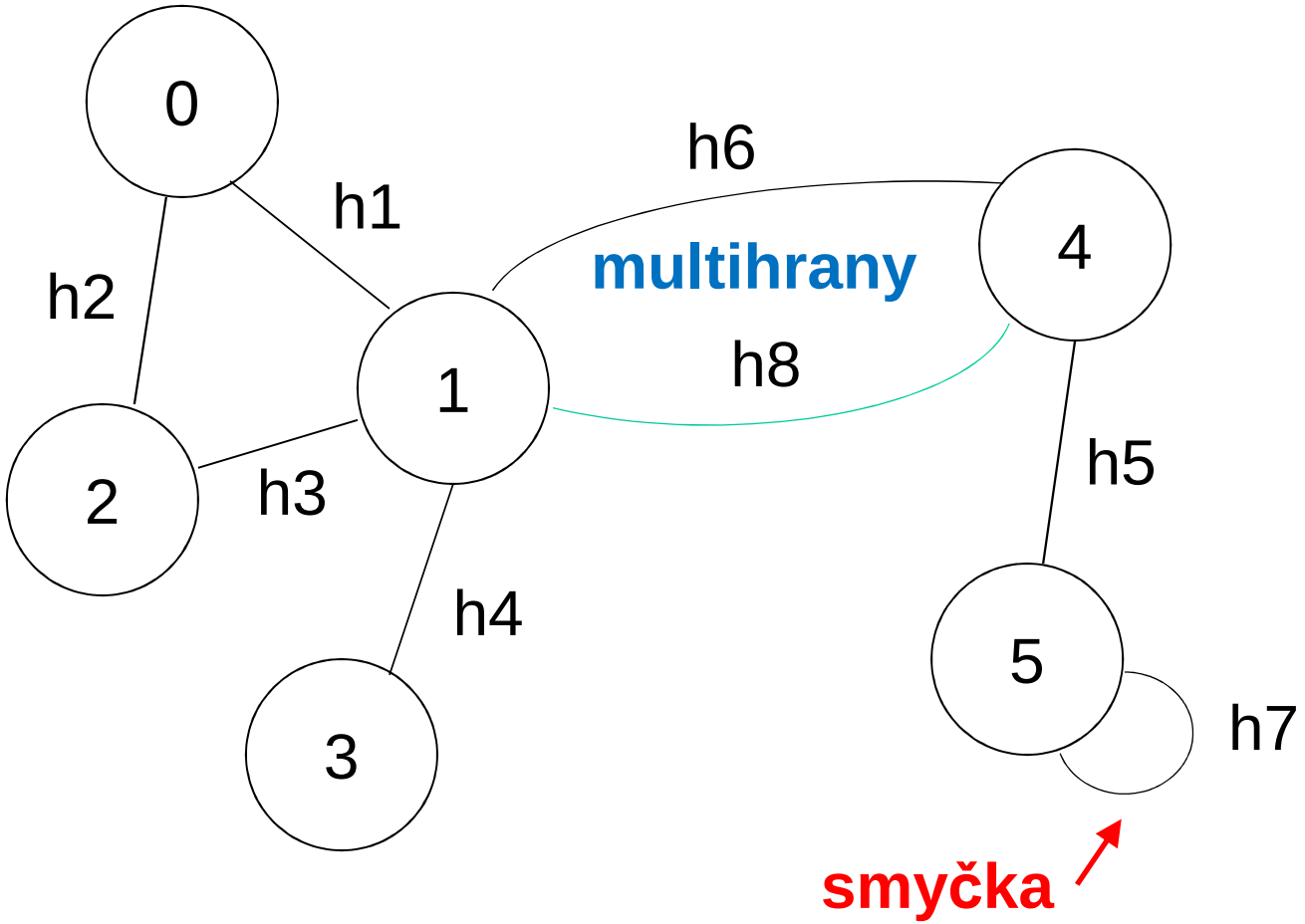
Popis grafu pomocí množin:

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

$$E = \{h_1, h_2, h_3, h_4, h_5, h_6, h_7\}$$

$$I : h_1 \to [0,1], h_2 \to [0,2], \cdots, h_7 \to [5,5]$$

Příklad grafu II



- příklad incidence jako relace:

$$V = \{0,1,2,3,4,5\}$$
$$E = \{h_1, h_2, h_3, h_4, h_5, h_6, h_7, h_8\}$$
$$I = \{(h_1,0,1),(h_2,0,2),\cdots,(h_6,1,4),(h_7,5,5),(h_8,1,4)\}$$

Pojmy

- podgraf *G'* = (*V'*,*E'*,*I'*) grafu G

$$V' \subset V, E' \subset E, \forall h \in E'\ I'(h) = I(h)$$

- stupeň uzlu *u*
  - počet hran incidujících s uzlem *u*
- sousedé uzlu u
  - množina uzlů incidujících s uzlem u
- úplný graf
  - graf, kde každý uzel je spojen s každým hranou, počet hran je $\binom{n}{2}$

Pojmy

- sled
  - posloupnost uzlů u_i a hran h_i

$$S = \{u_1, h_1, u_2, h_2, \ldots, u_n\} : I(h_i) = [u_i, u_{i+1}]$$

- tah
  - sled, kde se neopakují hrany
    - srovnejte: nakreslete obrázek jedním tahem
- cesta
  - tah, kde se neopakují uzly

Pojmy

- uzavřená cesta = kružnice
  - cesta, kde shodují první a poslední uzel

Poznámka:

- sled, resp. tah, kde se shodují první a poslední uzel se také nazývá **uzavřený**

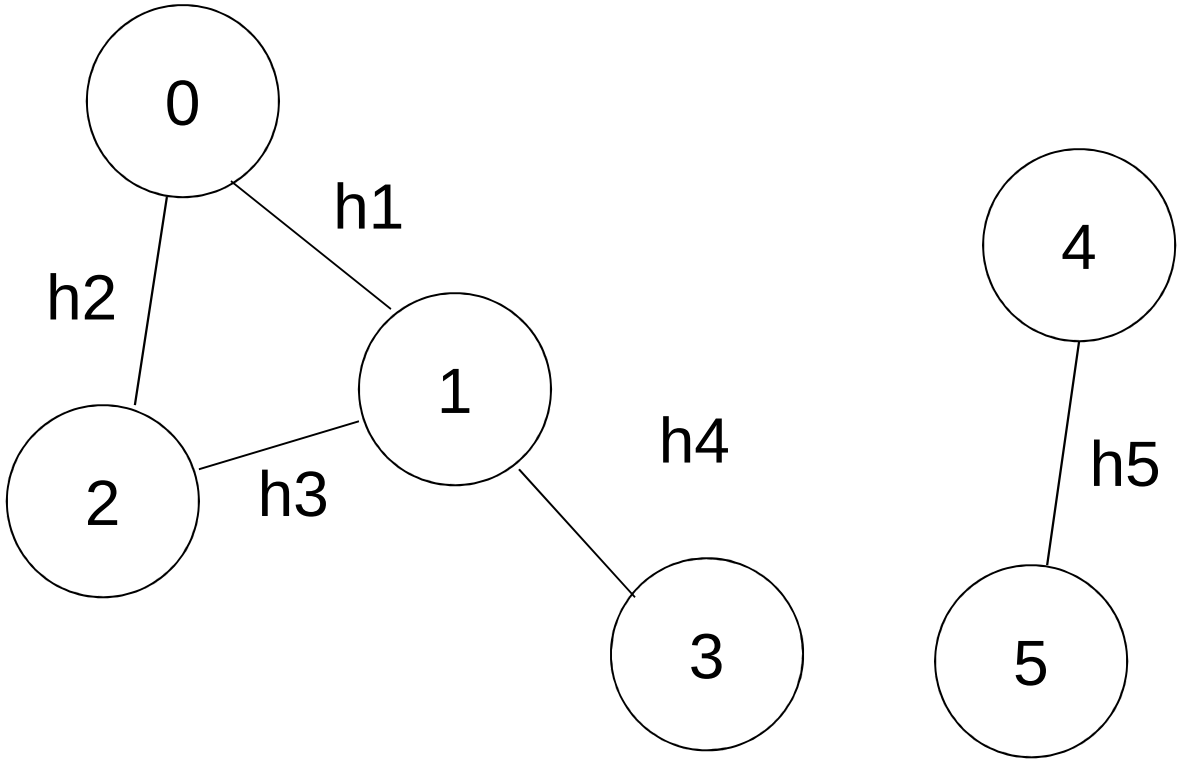
Souvislost grafu

- graf je souvislý, jestliže existuje cesta mezi každou dvojicí uzlů

Komponenta souvislosti

- maximální souvislý podgraf

Příklad



příklad: dvě komponenty souvislosti

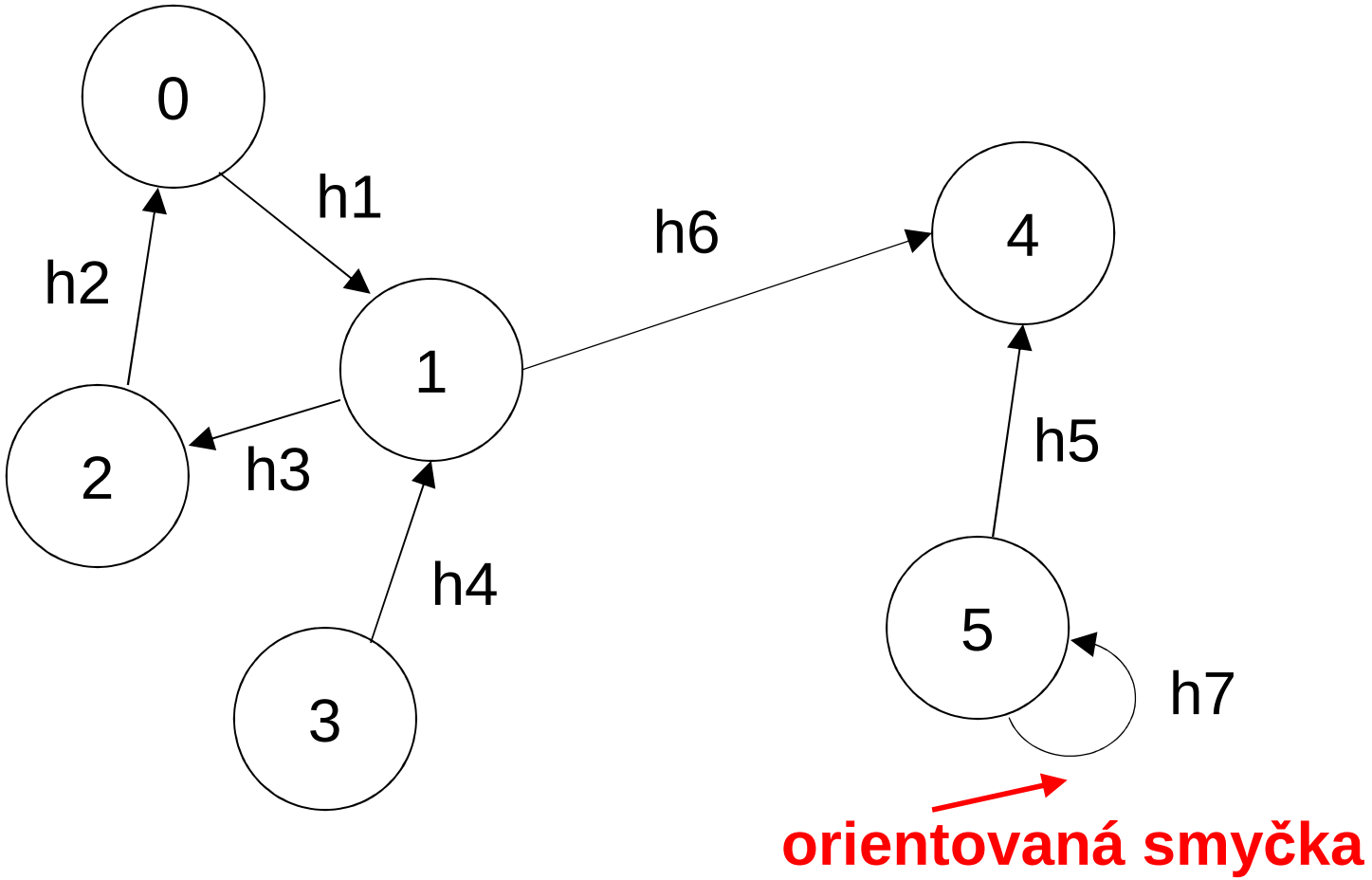
1. komponenta: {0,1,2,3}
2. komponenta: {4,5}

Orientovaný graf

$G = (V,E,I)$

- V množina uzlů (vrcholů)
- E množina hran
- I incidence
  - incidence je zobrazení, buď:
    - funkce: $I: E \rightarrow V \times V$
    - relace: $I \subset E \times V \times V$
      - incidence přiřadí hraně uspořádanou dvojici uzlů

Příklad orientovaného grafu



Popis grafu pomocí množin:

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

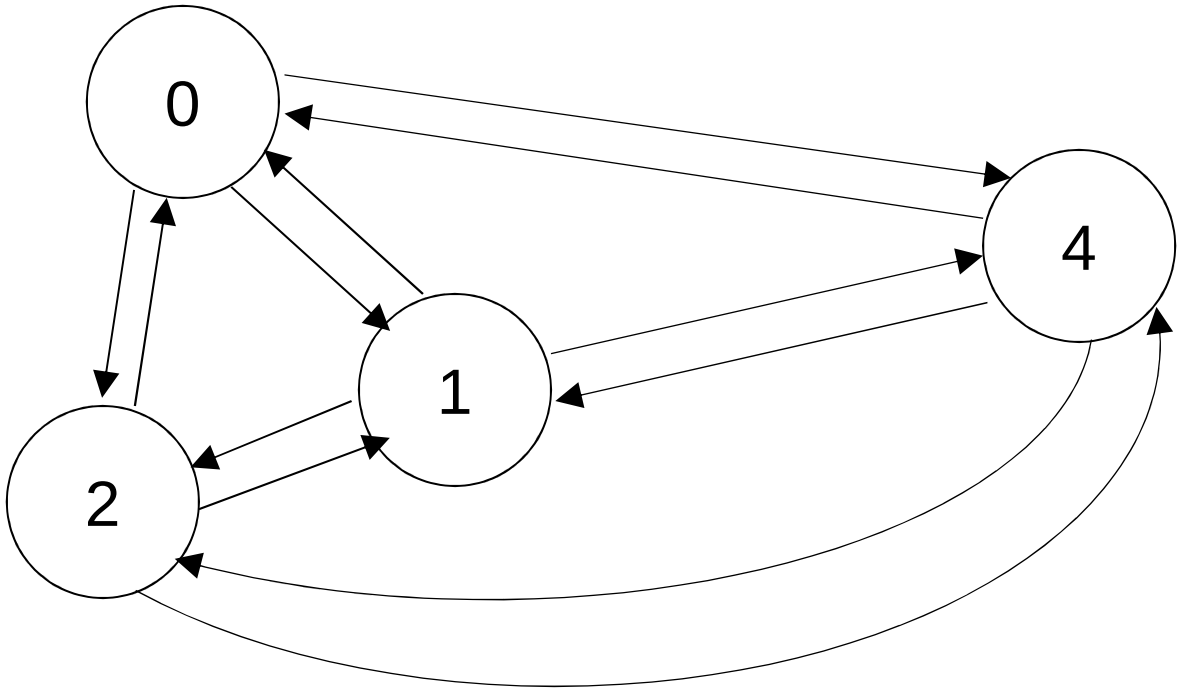
$$E = \{h_1, h_2, h_3, h_4, h_5, h_6, h_7\}$$

$$I : h_1 \to (0,1), h_2 \to (2,0), \cdots, h_7 \to (5,5)$$

Pojmy

- výstupní stupeň uzlu *u*
  - počet hran směřujících z uzlu *u*
- vstupní stupeň uzlu u
  - počet hran směřujících do uzlu *u*
- úplný symetricky orientovaný graf
  - graf, kde každá orientovaná dvojice uzlů je spojena hranou

Úplný symetricky orientovaný graf



Pojmy

- sled
  - posloupnost uzlů u_i a hran h_i, která je sledem v grafu G’ vzniklým z G po zrušení orientace
- analogicky tah, cesta
- spojení
  - posloupnost uzlů u_i a hran h_i

$$S = \{u_1, h_1, u_2, h_2, \ldots, u_n\} : I(h_i) = (u_i, u_{i+1})$$

    - uvažuji orientaci hran

Pojmy

- orientovaný tah, orientovaná cesta
  - spojení, které je tahem, resp. cestou po zrušení orientace
- cyklus = uzavřená orientovaná cesta

- silná souvislost:
  - orientovaný graf G nazýváme silně souvislým, jestliže pro libovolnou dvojici uzlů *u, v* existuje spojení z uzlu *u* do uzlu *v* a spojení z uzlu *v* do uzlu *u*

Pojmy

- ohodnocený graf (hranově):
  - je dána funkce, která každé hraně přiřadí reálné číslo
  - existuje i uzlově ohodnocený graf
    - funkce přiřadí každému uzlu reálné číslo, popř. „cokoliv“ – prvek z nějaké domény (jméno města, stav hry atd.)

Využití teorie grafů

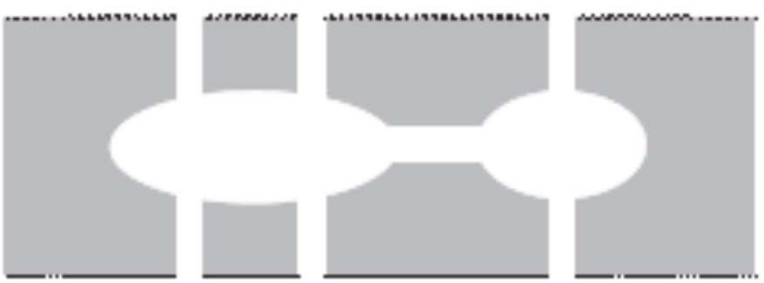
- doprava
  - modelování dopravní sítě ohodnocenými grafy (ohodnocení hran = vzdálenost)
- počítačové a datové, dopravní sítě
  - modelování propustnosti (ohodnocení hran = kapacita hrany), tzv. *toky v sítích*
- matematika, logika, informatika, teorie her
  - uzly grafu představují např. stavový prostor problému
  - prohledávání, rozhodování

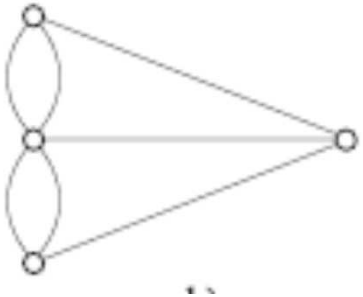
Využití teorie grafů

- vybrané úlohy a algoritmy nad grafy
  - hledání kostry grafu, resp. minimální kostry
  - hledání nejkratší cesty mezi dvěma uzly
  - hledání nejkratších cest mezi všemi uzly
  - hledání všech cest délky *n*
  - hledání maximálního toku v síti

Využití teorie grafů

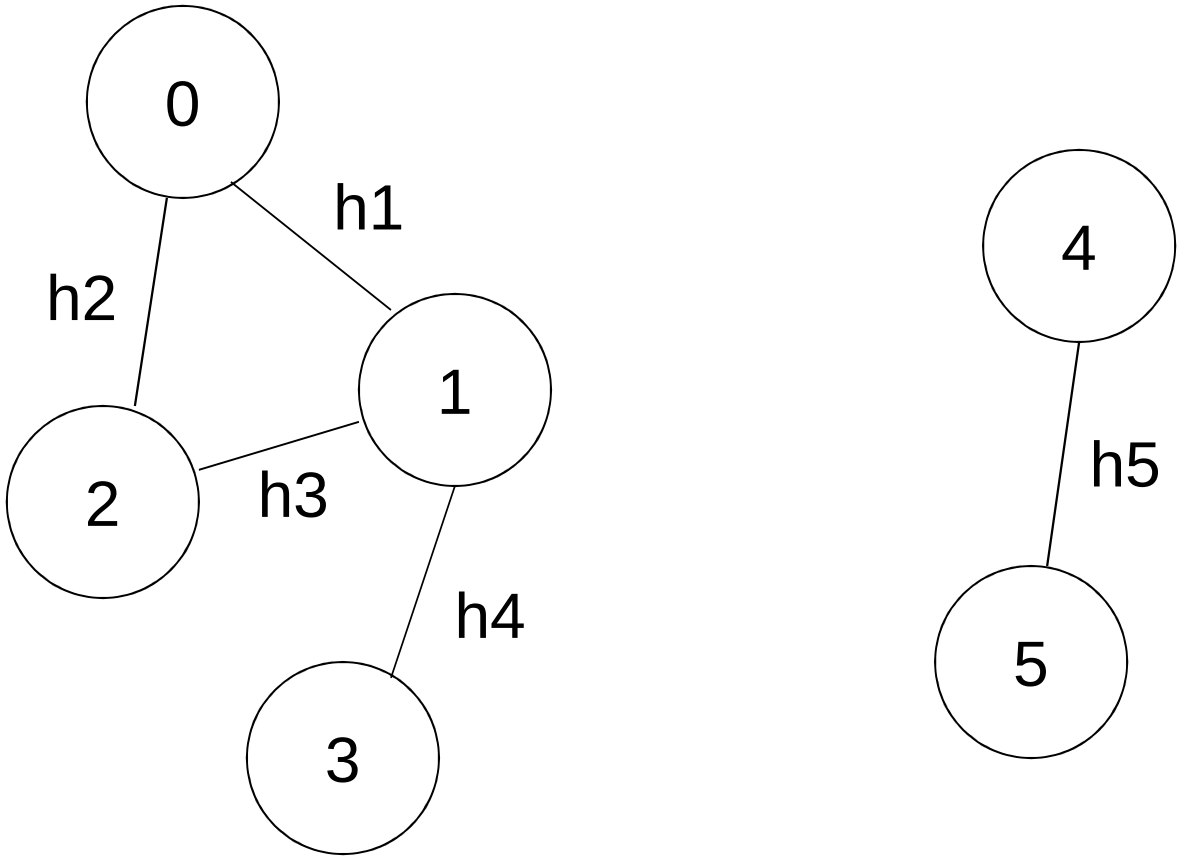
– dva příklady aplikace teorie grafů
  - problém sedmi mostů města Královce





    – hledáme pokrytí uzavřeným tahem
      » graf musí být Eulerův, tj. všechny uzlu musejí být sudého stupně, aby šel pokrýt uzavřeným tahem
      » pokud jsou max. 2 uzly lichého stupně, lze graf pokrýt otevřeným tahem
  - hra „15“
    – zamyslete se, jak využít teorii grafů této úloze

Příklad grafu a jeho reprezentace



Reprezentace grafu v programování

1. maticí sousednosti
   - matice uzel - uzel

	0	1	2	3	4	5
0	0	1	1	0	0	0
1	1	0	1	1	0	0
2	1	1	0	0	0	0
3	0	1	0	0	0	0
4	0	0	0	0	0	1
5	0	0	0	0	1	0

- pokud existuje mezi uzly *i,j* hrana, prvek matice $a_{i,j}$ = 1, jinak 0
- pro neorientované grafy je matice sousednosti **symetrická**
- pokud jsou hrany ohodnocené, zapisuje se do matice ohodnocení hran
  - problém s nulovým ohodnocením hrany
  - pak se používá místo nuly hodnota např. maxint

2. maticí incidence

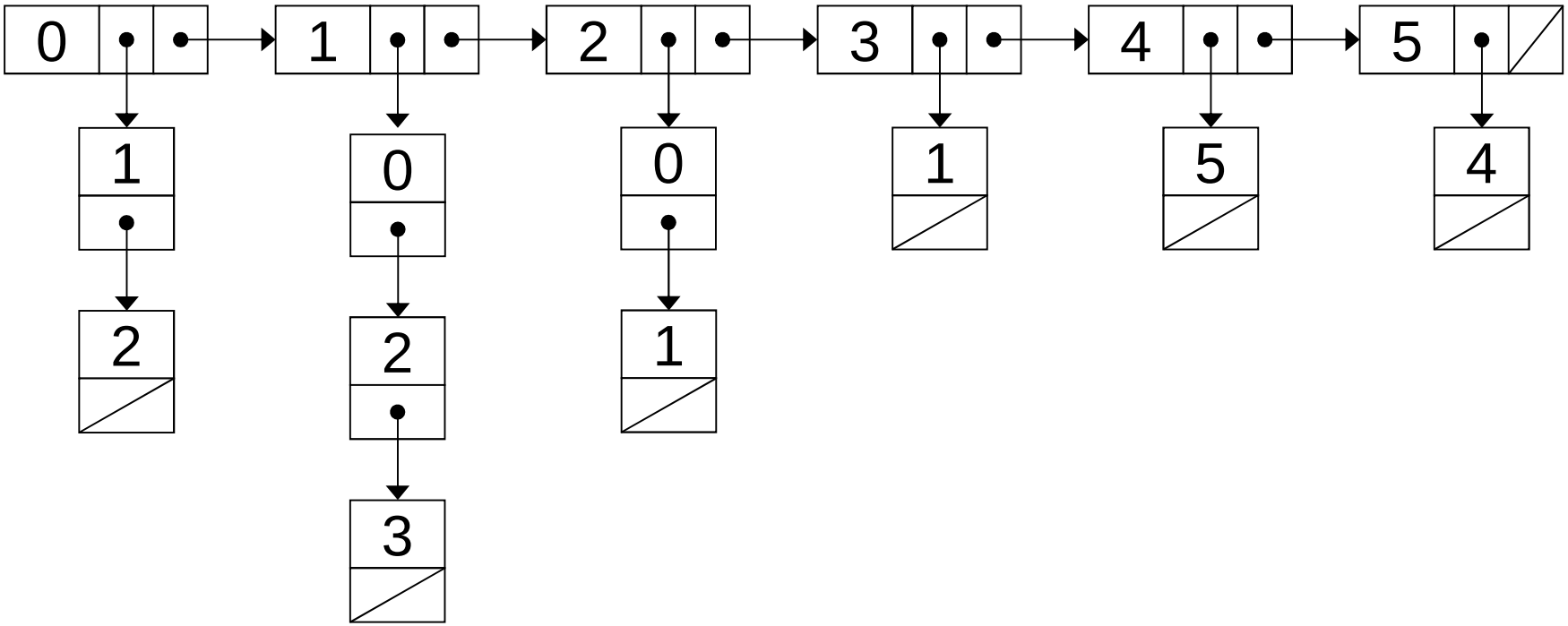
– matice uzel – hrana

- u orientovaných grafů je hodnota 1 u počátečního a -1 u koncového uzlu

	h1	h2	h3	h4	h5
0	1	1	0	0	0
1	1	0	1	1	0
2	0	1	1	0	0
3	0	0	0	1	0
4	0	0	0	0	1
5	0	0	0	0	1

3. spojovým seznamem

- spojový seznam uzlů
- každý uzel ukazuje na spojový seznam sousedů (hran)


0
1
2
3
4
5
1
2
0
2
3
0
1
1
5
4

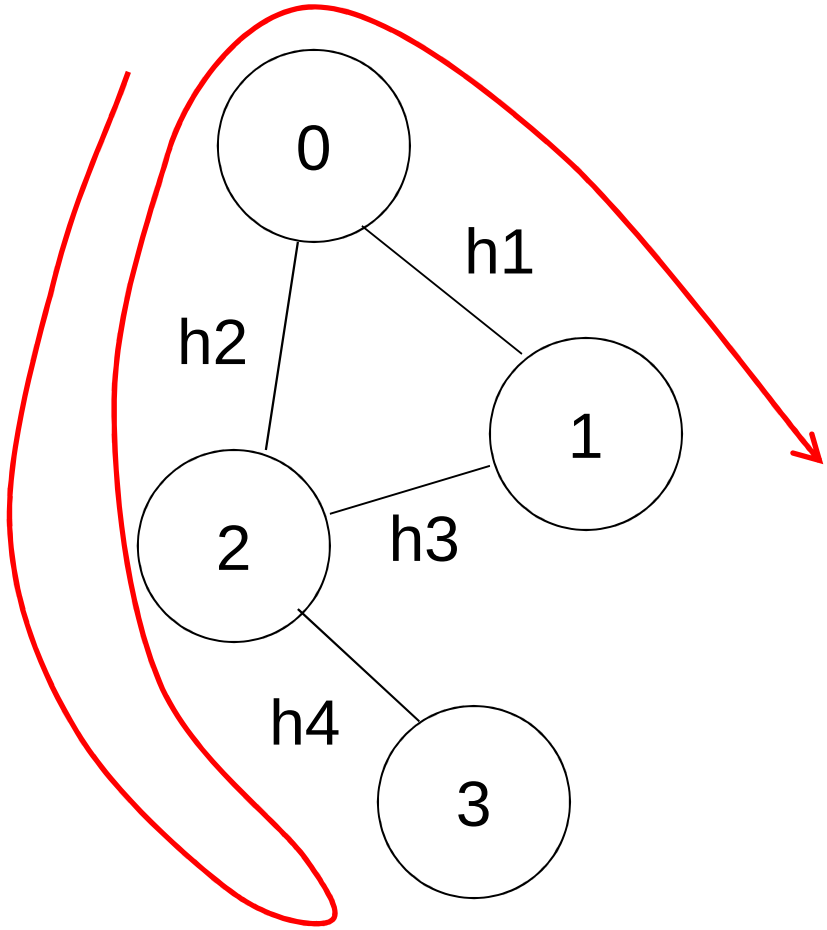
Obecné algoritmy procházení grafu

1. do hloubky

- používám **zásobník**

```
Vyber libovolný uzel
Vloz do zásobníku
while (zásobník není prázdný)
{
  Vyber uzel u ze zásobníku
  Zpracuj uzel u
  Označ uzel u jako zpracovaný
  for všechny sousedy v uzlu u
   if (v není zpracovaný) vlož v do zásobníku
}
```

Procházení grafu do hloubky

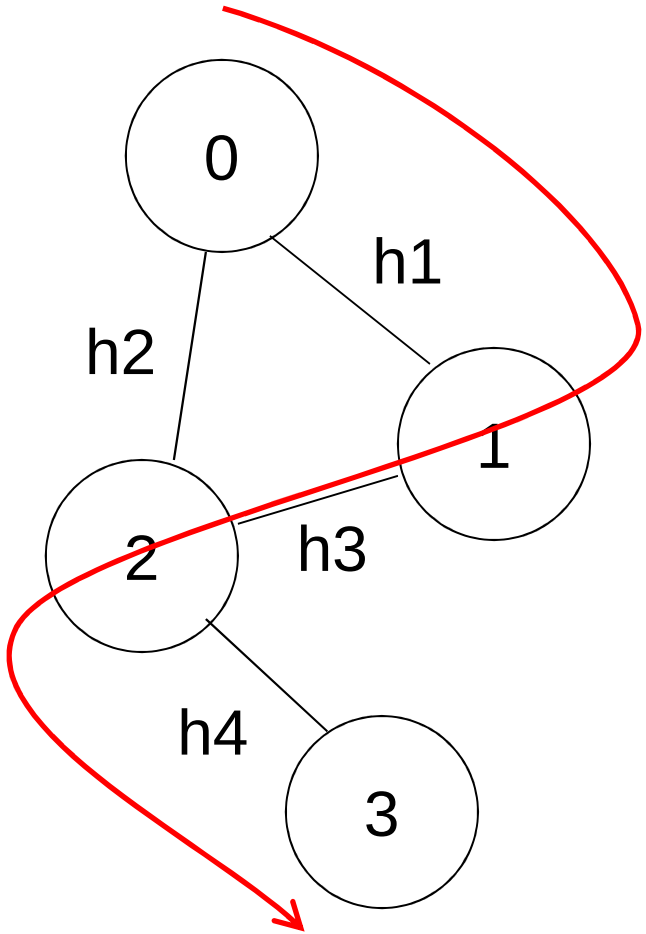


zkuste si nasimulovat ručně na papír algoritmus procházení grafu do hloubky

- procházení do hloubky se snadno realizuje (viz stromy), protože zásobník je realizován automaticky při volání procedur
  - procházení grafu do hloubky se realizuje rekurzí

2. do šířky

– místo zásobníku používám **frontu**



Procházení komponent souvislosti

V

– množina všech uzlů grafu

U

– pomocná množina uzlů, do které se ukládají prohledávané uzly

K

– množina uzlů patřící k jedné komponentě

```
počáteční stav uzlů: neprozkoumaný; U=Ø;
while (V je neprázdná) {
  K=Ø;
  u = libovolný uzel z V; V = V – {u};
  Vloz u do U; stav(u) = prozkoumaný;
  while (U je neprázdná)
  { v = libovolný uzel z U;
    for (všechny hrany e vycházející z v) {
      w = druhý vrchol hrany e = vw;
      if (stav(w)!=prozkoumaný)
      {  stav(w) = prozkoumaný;
         Vlož w do U; V = V – {w};
      }
    }
    U = U – {v};   Vlož v do K;
  }
  Vypiš komponentu souvislosti z množiny K
}
```

Datové struktury

- množina V
  - množina (charakteristický vektor)
    - odebírání prvků se složitostí O(1)
    - test na prázdnou množinu a výběr lib. prvku – O(n) – provádí se pouze (*k*+1) krát, kde *k* je počet komponent
- množina U
  - fronta
    - operace výběru libovolného prvku se realizuje se složitostí O(1),
    - pokud bych reprezentoval množinově, výběr libovolného prvku by představoval hledání v char. vektoru

- množina K
  - frontou
    - výpis není podle pořadí uzlů
    - při výpisu se fronta automaticky vyprázdní
  - množinou
- prozkoumané uzly
  - booleovský vektor – de facto množina
- graf
  - maticí sousednosti
    - načteme ji z textového souboru

```
while(!je_mnozina_prazdna(&V))
  {
    pocet_komponent++;
    u = vyber_libovolny(&V);
    vloz(&U,u); prozkoumane[u] = 1;
    while(!je_fronta_prazdna(&U))
    {
      v = vyjmi(&U);
      for(w=0;w<n;w++)
        if (matice_sous[v][w]==1 && prozkoumane[w]==0)
        {
          vloz(&U,w);
          prozkoumane[w] = 1;
          vyjmi_prvek(&V,w);
        }
       vloz(&K,v);
     }
    Vypis_komponentu(pocet_komponent,&K);
  }
```

- ideální složitost prohledávání grafu je $O(n+m)$, kde
  - n je počet uzlů grafu
  - m je počet hran grafu