Příklad algoritmu
Grafy

Redukce počtu uzlů dat z ArcGISu

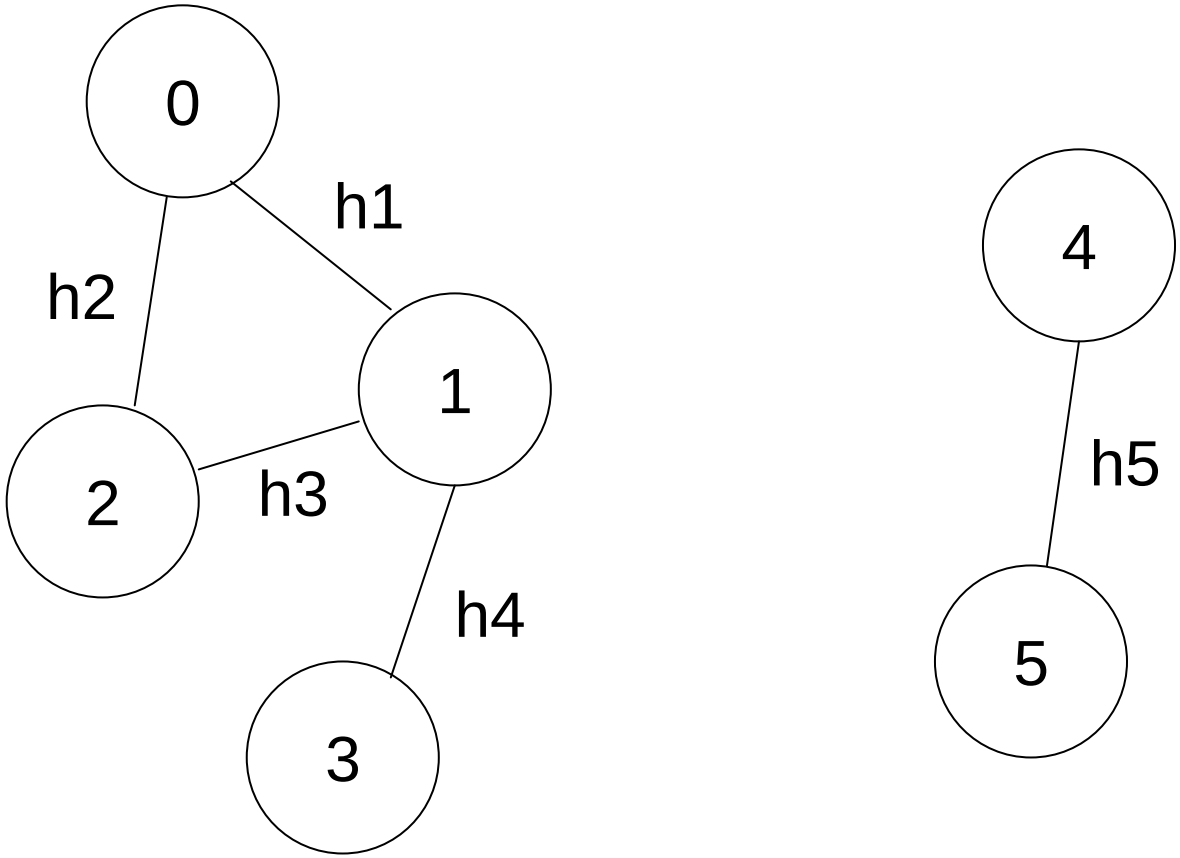
(bakalářská práce M. Stambolidis)

Graf

$G = (V,E,I)$

- V množina uzlů (vrcholů)
- E množina hran
- I incidence

- orientovaný a neorientovaný graf


0
h1
h2
1
2
h3
h4
3
4
h5
5

Reprezentace grafu v programování

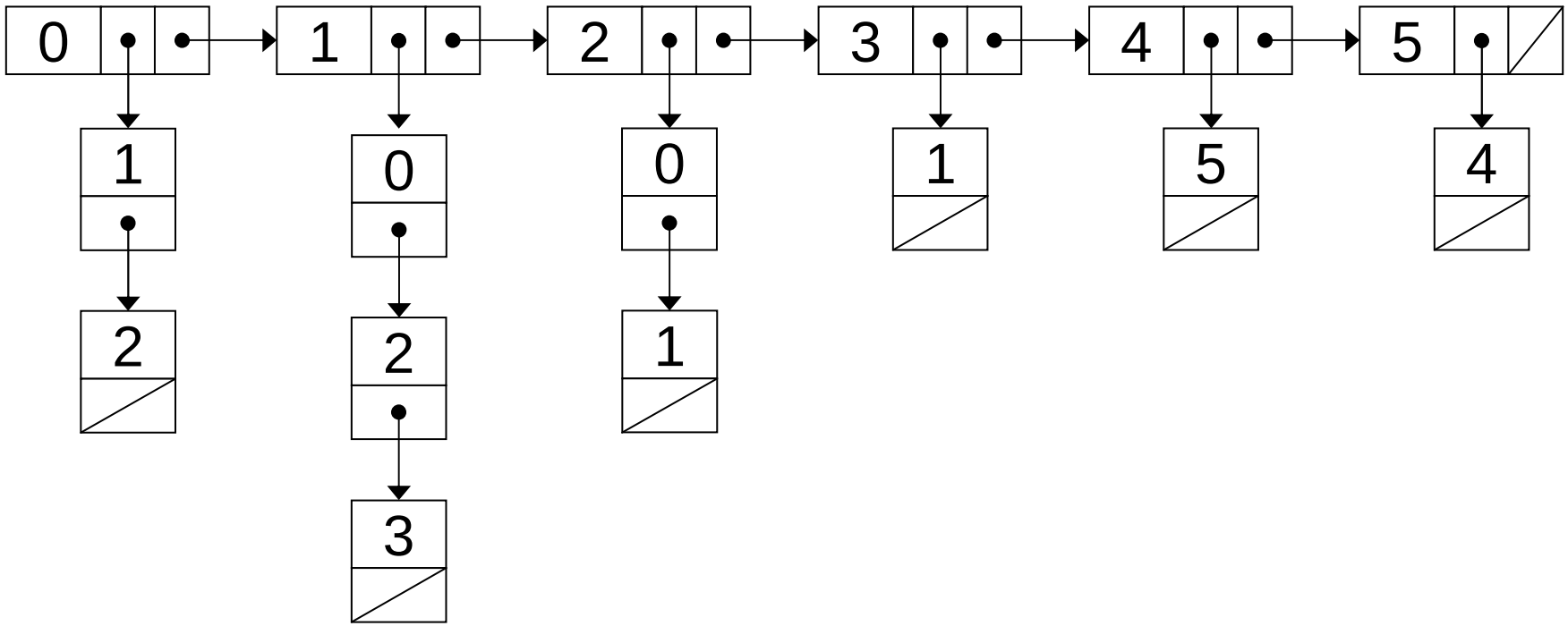
1. maticí sousednosti

	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. spojovým seznamem

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

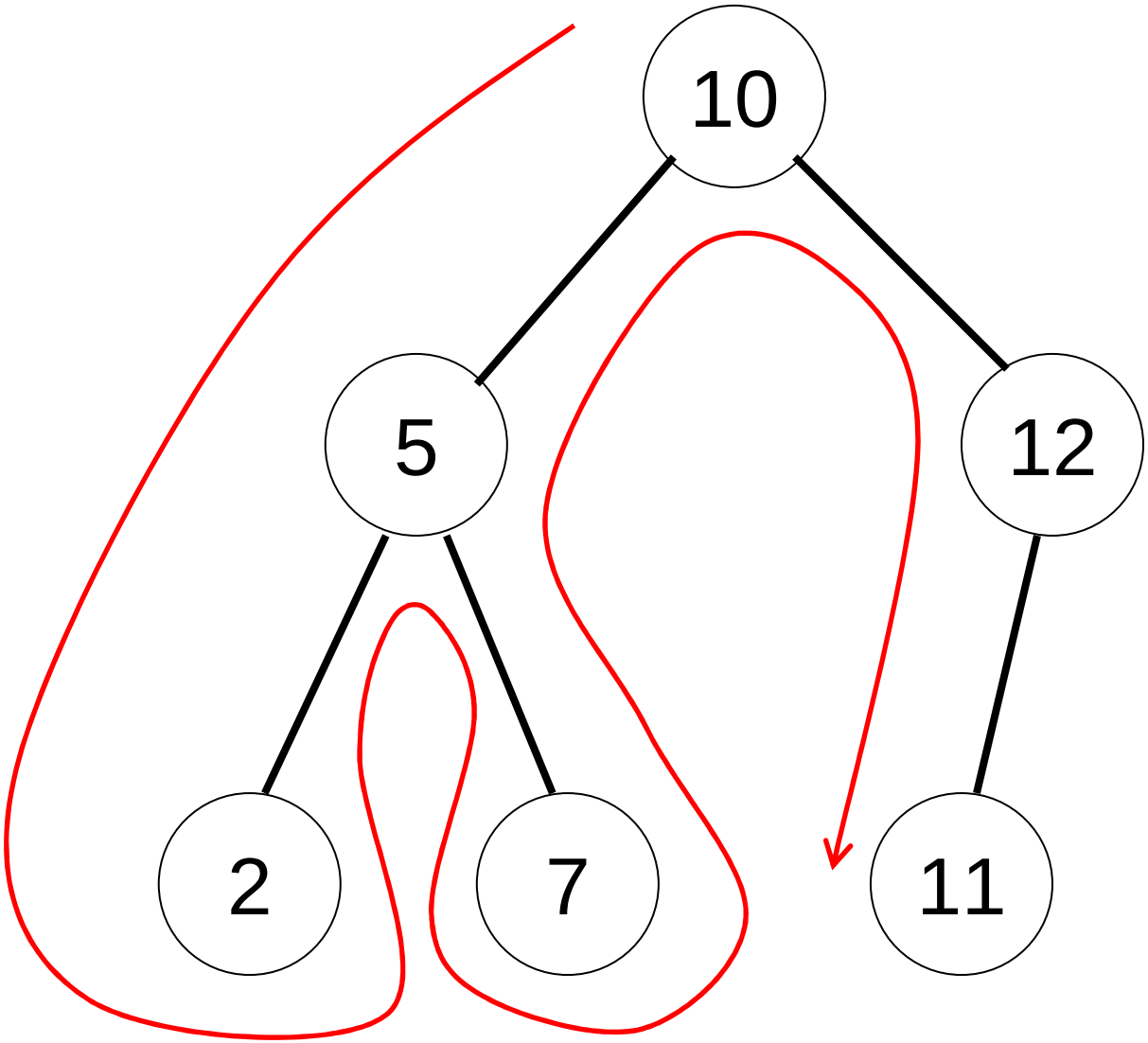

0
1
2
3
4
5
1
2
0
2
3
0
1
1
5
4

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
}
```

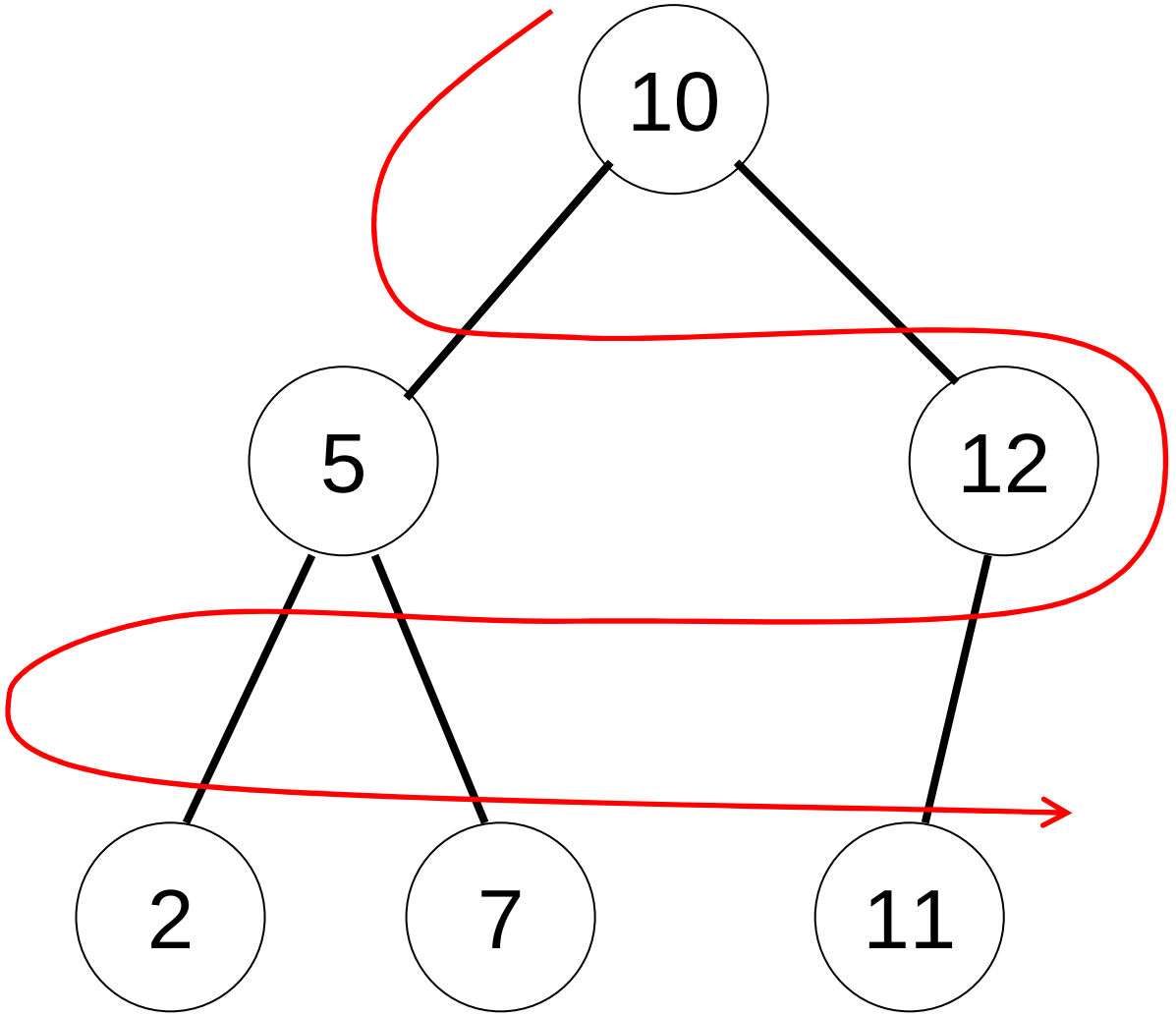

10
5
12
2
7
11

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

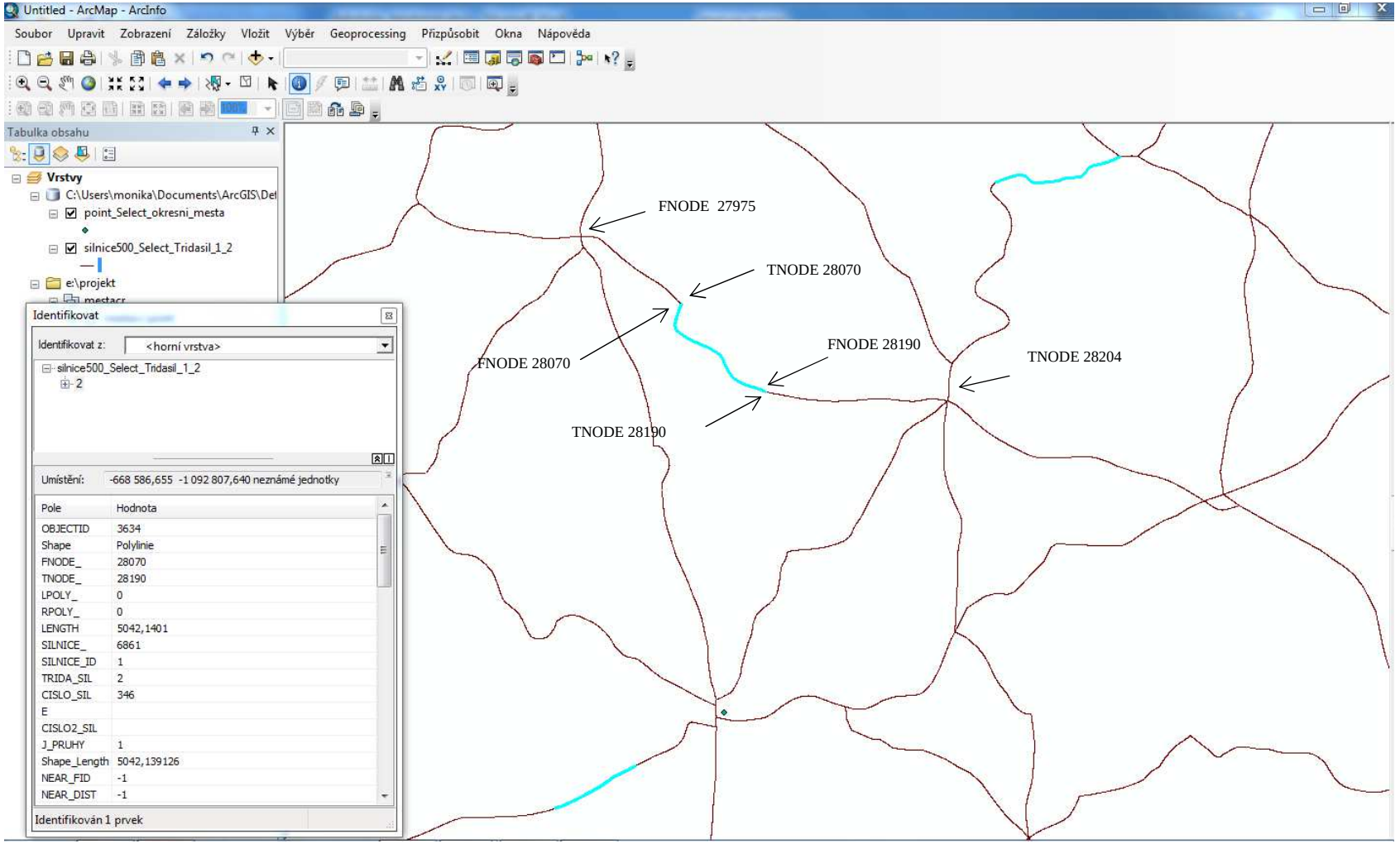
- procházení do hloubky se snadno realizuje (viz stromy), protože zásobník je realizován automaticky při volání procedur (rekurze)

2. do šířky

– místo zásobníku používám **frontu**



Reprezentace silnic



Datové soubory

- stupně uzlu

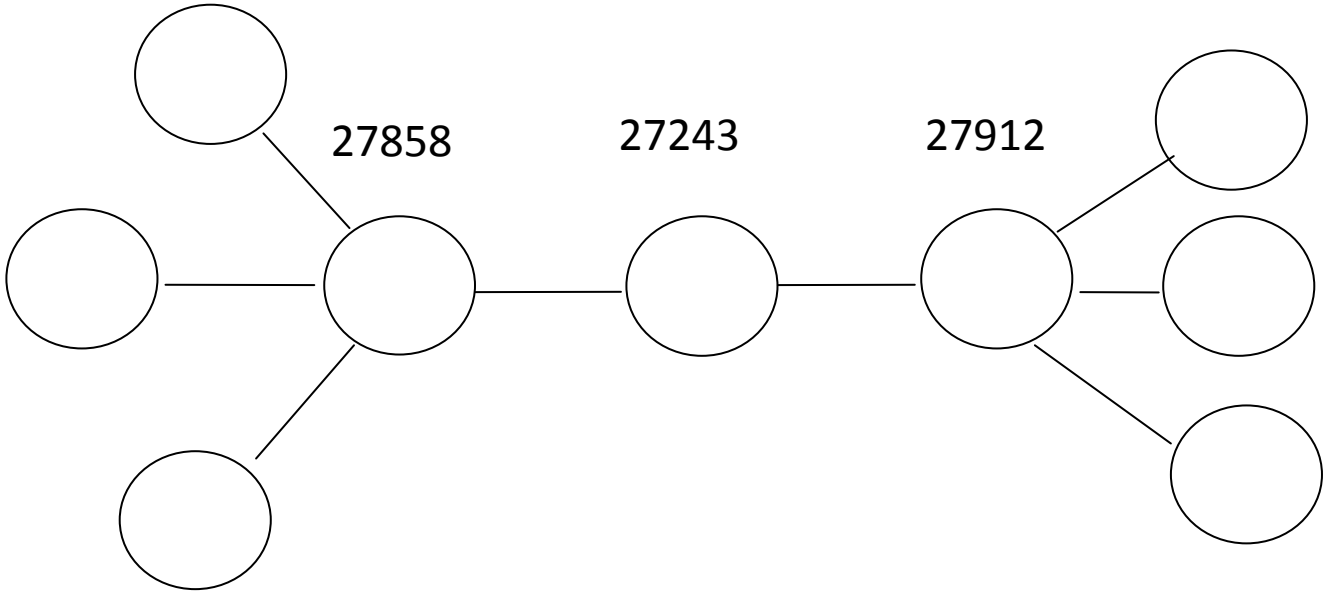
```
suma NODE
2 23785
3 23786
2 23787
2 23788
2 23789
2 23791
1 23792
2 23794
```

Datové soubory

- vzdálenosti mezi uzly

```
suma NODE
2 23785
3 23786
2 23787
2 23788
2 23789
2 23791
1 23792
2 23794
```

Graf



Princip algoritmu

- kombinuji procházení grafu do šířky a hloubky
- udržuji množinu uzlů se stupněm >= 3
  - vybírám z ní postupně uzly
  - procházím sousedy vybraného uzlu, mají-li stupeň 2 a počítám celkovou vzdálenost
    - původní stupeň >= 3 znamená křižovatku, stupeň 1 koncový uzel

```
while (množina není prázdná)
{
  zpracovávaný_uzel = první uzel z množiny Mn
  while (stupeň(zpracovávaný_uzel) > 0)
  {
    soused=vyber prvního souseda uzlu „zpracovávaný_uzel“
    celková_vzdál=vzdálenost(uzel,soused)
    vymaž uzel „soused“ ze seznamu sousedů uzlu „zpracovávaný_uzel“
    vymaž uzel „zpracovávaný_uzel“ ze seznamu sousedů uzlu „soused“
    while (původní_stupeň(soused)==2)
    {
       předchozí_uzel = soused;
      soused=vyber prvního souseda uzlu „soused“
      celková_vzdál=celková_vzdál+vzdálenost(předchozí_uzel,soused);
      vymaž uzel „soused“ ze seznamu sousedů uzlu „předchozí_uzel“
      vymaž uzel „předchozí_uzel“ ze seznamu sousedů uzlu „soused“
    }
    tiskni „uzel soused celková_vzdálenost“
    sniž stupeň uzlu „zpracovávaný_uzel“ v množině Mn
    if(původní_stupeň(soused)>=3)
      sniž stupeň uzlu „soused“ v množině Mn
  }
  vyjmi „zpracovávaný_uzel“ z množiny Mn
}
```

Použité datové typy

```
typedef struct
{
  int node;
  float vzdalenost;
} info_sousedi;

typedef struct
{
  int suma_nodu
  set<info_sousedi,mnoz_less> sousedi;
} Info_NODE;
```

Reprezentace grafu

- seznam sousedů

`map<int,Info_NODE> graf;`

<table>
<tr><td><u>Klíč (key vallue)</u> – zde je uložena proměnná
int NODE</td><td colspan="3"><u>Hodnota klíče (maped vallue)</u> – zde je uložena struktura
struct info</td></tr>
<tr><td rowspan="3">27867</td><td>int suma_nodu</td><td colspan="2">set<info_mnozina_sousedi,mnoz_less> sousedi</td></tr>
<tr><td rowspan="2">3</td><td colspan="2">struktura info_množina_sousedi</td><td>funktor</td></tr>
<tr><td>int NODE</td><td>int vzdalenost</td><td>Slouží k porovnávání prvků ve množině</td></tr>
</table>

Přístup k prvkům mapy

- pomocí klíče
  - `graf[27867]. suma_nodu=3`
- pomocí iterátoru
  - položky mapy (klíč, záznam) rozliším pomocí first, second
  - iter = graf.begin();
  - `(*iter).first` – přistupuji ke klíči
  - `(*iter).second` – přistupuji ke struktuře