Dijkstrův algoritmus

nalezení nejkratší cesty v grafu

Dominik Horký

Fakulta Informačních technologií
Vysoké učení technické v Brně

2020

Definice

- Konečný algoritmus sloužící k nalezení nejkratší cesty v grafu
- Používá se nad kladně ohodnocenými grafy

Definice

- Konečný algoritmus sloužící k nalezení nejkratší cesty v grafu
- Používá se nad kladně ohodnocenými grafy
  - neohodnocené grafy lze snadno převést na ohodnocené
  - pro záporně ohodnocené se používá např. Bellmalův-Fordův algoritmus

Definice

- Konečný algoritmus sloužící k nalezení nejkratší cesty v grafu
- Používá se nad kladně ohodnocenými grafy
  - neohodnocené grafy lze snadno převést na ohodnocené
  - pro záporně ohodnocené se používá např. Bellmalův-Fordův algoritmus
- Pojmenován podle svého autora Edsgera Dijkstry

Popis algoritmu

- Všechny vrcholy grafu G (tzv. $d[v]$) z množiny všech vrcholů V nastavíme na ∞

Popis algoritmu

- Všechny vrcholy grafu G (tzv. $d[v]$) z množiny všech vrcholů V nastavíme na ∞
- Počáteční vrchol ($d[s]$) nastavíme na hodnotu 0

Popis algoritmu

- Všechny vrcholy grafu G (tzv. $d[v]$) z množiny všech vrcholů V nastavíme na ∞
- Počáteční vrchol ($d[s]$) nastavíme na hodnotu 0
- V průběhu algoritmu si zaznamenáváme, které vrcholy jsme už navštívili (množina Z) a které nikoliv (množina N)

Popis algoritmu

- Všechny vrcholy grafu G (tzv. $d[v]$) z množiny všech vrcholů V nastavíme na ∞
- Počáteční vrchol ($d[s]$) nastavíme na hodnotu 0
- V průběhu algoritmu si zaznamenáváme, které vrcholy jsme už navštívili (množina Z) a které nikoliv (množina N)
- Algoritmus cyklí tak dlouho, dokud N není prázdnou množinou

Popis algoritmu

- Všechny vrcholy grafu G (tzv. $d[v]$) z množiny všech vrcholů V nastavíme na ∞
- Počáteční vrchol ($d[s]$) nastavíme na hodnotu 0
- V průběhu algoritmu si zaznamenáváme, které vrcholy jsme už navštívili (množina Z) a které nikoliv (množina N)
- Algoritmus cyklí tak dlouho, dokud N není prázdnou množinou
- V každém průchodu cyklu se přidá jeden vrchol z N do Z takový, který má nejmenší hodnotu $d[v]$ ze všech vrcholů v N

Popis algoritmu

- Všechny vrcholy grafu G (tzv. $d[v]$) z množiny všech vrcholů V nastavíme na ∞
- Počáteční vrchol ($d[s]$) nastavíme na hodnotu 0
- V průběhu algoritmu si zaznamenáváme, které vrcholy jsme už navštívili (množina Z) a které nikoliv (množina N)
- Algoritmus cyklí tak dlouho, dokud N není prázdnou množinou
- V každém průchodu cyklu se přidá jeden vrchol z N do Z takový, který má nejmenší hodnotu $d[v]$ ze všech vrcholů v N
- Po skončení algoritmu je délka nejkratší cesty uložena v $d[v]$

Pseudokód

Celý algoritmus lze však (mnohem lépe) pochopit z pseudokódu.

Pseudokód (jazyk Pascal)

```
function Dijkstra(E, V, s)
    for each vertex v in V do
        d[v] := ∞
        p[v] := undefined
    end for
    d[s] := 0
    N := V
    while N is not empty do
        u := extract_min(N)
        for each neighbor v of u do
            alt = d[u] + l(u, v)
            if alt < d[v] then
                d[v] := alt
                p[v] := u
            end if
        end for
    end while
```

Složitost

- Samotný algoritmus není pro implementaci příliš složitý
- Lze jej naimplementovat s asymptotickou časovou složitostí:

$$O(E + V \log V)$$

- $|V|$ je počet uzlů, $|E|$ je počet hran

Složitost - řídké grafy

- Algoritmus lze optimalizovat pro tzv. řídké grafy (ty, které mají $|E| < |V|^2$)

Složitost - řídké grafy

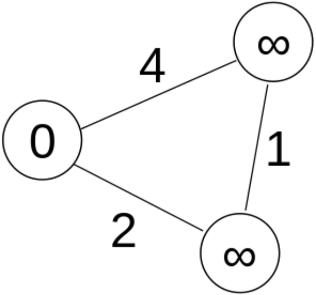
- Algoritmus lze optimalizovat pro tzv. řídké grafy (ty, které mají $|E| < |V|^2$)
  - graf se uloží podle seznamu sousedů
  - funkci *extract_min* implementujeme pomocí Fibonacciho nebo binární haldy

Složitost - řídké grafy

- Algoritmus lze optimalizovat pro tzv. řídké grafy (ty, které mají $|E| < |V|^2$)
  - graf se uloží podle seznamu sousedů
  - funkci *extract_min* implementujeme pomocí Fibonacciho nebo binární haldy
- Po těchto optimalizacích algoritmus běží v čase:

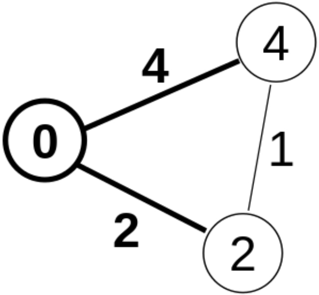
$$O((|E| + |V|) \log |V|)$$

Příklad použití algoritmu (1/3)



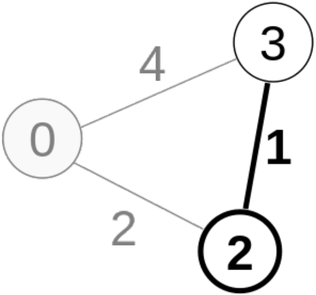
Počáteční vrchol nastavíme na hodnotu 0, ostatní na ∞

Příklad použití algoritmu (2/3)



Nastavíme hodnoty vrcholů dle ohodnocení hran vedoucích k nim

Příklad použití algoritmu (3/3)



Nejkratší cesta tohoto malého váženého grafu má hodnotu 3

Praxe, využití, závěr

- Vhodné je implementovat algoritmus spolu s prioritní frontou

Praxe, využití, závěr

- Vhodné je implementovat algoritmus spolu s prioritní frontou
- Dijkstrův algoritmus lze použít při hledání nejkratší cesty na mapách
- Algoritmus je nicméně pro reálné využití v mapách (jako např. Google Maps) příliš pomalý

Praxe, využití, závěr

- Vhodné je implementovat algoritmus spolu s prioritní frontou
- Dijkstrův algoritmus lze použít při hledání nejkratší cesty na mapách
- Algoritmus je nicméně pro reálné využití v mapách (jako např. Google Maps) příliš pomalý
- Nejčastěji je tak využitý zejména v hledání nejkratších cest grafů

Použité zdroje

- Algoritmus, pseudokód
  https://cs.wikipedia.org/wiki/Dijkstr%C5%AFv_algoritmus
- Dodatečné informace
  http://voho.eu/wiki/algoritmus-dijkstra/
- Obrázky (graf použitý v příkladu)
  https://upload.wikimedia.org/wikipedia/commons/thumb/b/be/Dijkstra%27s_algorithm.svg/100px-Dijkstra%27s_algorithm.svg.png
- Použití v praxi
  https://is.mendelu.cz/eknihovna/opory/zobraz_cast.pl?cast=19938