Anotace

- Grafové algoritmy II

Dijkstrův algoritmus

Hledá nejkratší cestu z daného vrcholu (do všech ostatních)

Vstup: Graf s `nezáporně` ohodnocenými hranami.

- Udržujeme "frontu" vrcholů setříděnou podle dosud nejkratší cesty do nich.
- Na začátku zinicializujeme vzdálenosti do všech vrcholů kromě startovního nekonečnem (tedy dost vysokou hodnotou) a vzdálenost do startu nulou.
- Startovní vrchol přidáme do "fronty" dosažitelných vrcholů.
- Vrchol, do kterého se dostaneme nejkratší cestou z fronty odstraníme a pokusíme se cestu jdoucí z něj rozšířit do jeho sousedů.
- Toto opakuj, dokud je "fronta" neprázdná.

Rozšíření cesty

Rozšíření cesty vypadá tak, že pro vrchol v ve vzdálenosti $d(v)$ zkusíme pro každou hranu $\{v, w\}$, zda

$$d(w) > d(v) + delka(\{v, w\}).$$

Pokud ano, $d(w) := d(v) + delka(\{v, w\})$ a oprav pozici vrcholu w ve "frontě".

Analýza

- Algoritmus je popsán a dokázán v mnoha knihách (a skriptech), kupř. Kapitoly z diskrétní matematiky nebo Algebraické algoritmy (Kučera, Nešetřil)...
- Konečnost: V každé iteraci odstraníme z "fronty" jeden vrchol, který se v ní už neobjeví
  (protože mezi vrcholy ve frontě měl nejmenší vzdálenost od startu a vzdálenosti jsou nezáporné).
- Parciální správnosti pomůže invariant:
  V každém kroku evidujeme nejkratší cesty ze startu používající pouze vrcholy již odstraněné z "fronty".
- Z invariantu plyne korektnost.
- Jde jen o modifikovaný algoritmus vlny, tedy hledání do šířky!
- Složitost významně závisí na reprezentaci grafu a na reprezentaci "fronty"!

Poznámky

- Pokud jde o graf neohodnocený (délka všech hran je 1), potom se z Dijkstrova algoritmu stane obyčejný algoritmus vlny.
- Aplikace grafových algoritmů: Zde začíná teoretická informatika. :-)
- Příklady využití grafových algoritmů:
- Thesseus a Minotaurus,
- Král (či jiné figurky) na šachovnicích různých tvarů s různě pozakazovanými políčky,
- ...

Další grafové problémy/algoritmy

- Minimální kostra, aneb jak natáhnout elektrické vedení?
- Eulerovský graf, aneb lze všechny hrany grafu nakreslit jedním tahem (abychom žádnou hranou neprojeli dvakrát)?
- Hamiltonskost, aneb kružnice, která navštíví všechny vrcholy (každý právě jednou),
- Klika, aneb maximální úplný podgraf,
- Barevnost, aneb jaký je nejmenší počet barev takový, abychom mohli obarvit vrcholy grafu tak, že sousední vrcholy nedostanou stejnou barvu?
- Hranová barevnost, aneb jaký je nejmenší počet barev takový, abychom mohli obarvit hrany grafu tak, aby žádná dvojice hran přiléhající ke společnému vrcholu neměla stejnou barvu?
- ...

Grafově-optimalizační problémy

- Jak reprezentovat grafy – bylo v zimě, nyní už snad dokážete i implementovat.
- Jak grafy prohledávat – také bylo (do šířky a do hloubky), nyní umíte též implementovat.
- Hledání nejkratší cesty, minimální kostra,
- topologické uspořádání, faktorová množina (vyšetřování komponent).
- Modifikace problémů: Maximální kostra, nejdelší cesta – čím se liší?

Nejkratší cesta

- **Dijkstrův algoritmus:** Modifikované prohledávání do šířky (netvoříme frontu nalezených vrcholů, ale strukturu organizujeme podle doby, kdy se do dotyčného vrcholu dostaneme).
- **Bellman-Fordův algoritmus:** $n - 1$-krát zopakujeme: Z každého vrcholu zkus "natáhnout"cestu po všech hranách z něj vycházejících (tedy zlepši případnou nalezenou cestu).
- Oproti Dijkstrovu funguje i při záporném ohodnocení hran, nesmí ale být přítomna záporná kružnice (kružnice záporné délky).

Floyd – Warshallův algoritmus

aneb all-pairs shortest paths

- Další příklad dynamického programování!
- Pro trojici (a, u, v), kde u, v jsou vrcholy a a přirozené číslo počítáme nejkratší cestu z u do v o nejvýše a hranách.
- Postupujeme pro rostoucí a (projdeme všechny možné dvojice) tak, že natahujeme cestu o jedna kratší o "poslední" hranu.
- Jako by vybízelo k rekurzi...
- ... jenže jako obvykle velmi neefektivní.
- Tedy ji vybavíme cachí v podobě třírozměrného pole...
- ... a zjistíme, že rekurzi vůbec nepotřebujeme, že postačí čtyři cykly v sobě...

Floyd – Warshallův algoritmus pseudokód

```
for a:=1 to n-1 do
   for u in vrcholy do
      for v in vrcholy do
         for w in sousedi(v) do
            cache[a,u,v]:=min(cache[a,u,v],cache[a-1,u,w]+
                 length(w,v));
```

Minimální kostra

- Vstup: Graf s ohodnocenými hranami
- Cíl: Kostra s minimální váhou.
- Algoritmy: Kruskal, Borůvka, Jarník, Prim.
- Kruskalův: Setřiď hrany podle váhy, postupně zkoušej přidávat a koukej, zda jsme vytvořili kružnici (pokud ano, hranu nepřidej, jinak přidej).
- Borůvkův: Spojování komponent souvislosti: Vyber hranu s nejmenší vahou, která vychází z dané komponenty a přidej.
- Jarníkův (Primův): Pěstování stromu: K dosud postavenému stromu přidej hranu z něj vycházející, která má nejnižší váhu.
- Všechny algoritmy počítají to samé. Důkazy korektnosti budou příště.

Modifikace

- Hledání nejtěžší kostry...
- ... změň na každé hraně váhu z v_i na $-v_i$.
- Hledání nejdelší cesty...
- ... těžké (NP-těžké).
- Proč? Protože víme, kolik hran má kostra, ale nevíme, kolik hran má cesta.
- Tudíž i nalezení nejkratší cesty v (potenciálně záporně) ohodnoceném grafu je těžké (NP-těžký problém):
- Sedí za ním schovaná Hamiltonskost, tedy hledání cesty délky $n - 1$: