Opakování reprezentace grafů, Dijkstra, Bellman-Ford, Johnson

Petr Ryšavý

18. září 2017

Katedra počítačů, FEL, ČVUT

OPAKOVÁNÍ REPREZENTACE GRAFŮ

Graf

Definice (Graf) ***Graf*** *G je uspořádaná dvojice $G = (V, E)$, kde*

- *V je množina vrcholů a*
- *$E \subseteq V \times V$ je množina hran.*

Podle toho, zda jsou hrany uspořádané/neuspořádané dvojice rozlišujeme orientované/neorientované grafy.

Příklady grafů: silniční/železniční síť, web, sociální sítě

Typicky značíme $|V| = n$ a $|E| = m$.

Příklad

Uvažujme neorientovaný graf s n vrcholy, bez paralelních hran, který je tvořený jedinou komponentou souvislosti. Jaký je minimální a maximální počet hran, které graf může mít?

1. $n-1$ a $\dfrac{n(n-1)}{2}$
2. $n-1$ a n^2
3. n a 2^n
4. n a n^n

Hustý vs. řídký graf (anglicky *dense* a *sparse*)

- Ve většině aplikací je m mezi $\Omega(n)$ a $\mathcal{O}(n^2)$.
- Pro řídký graf je m v $\mathcal{O}(n)$ nebo se mu blíží.
- Pro hustý graf je m blízko $\Theta(n^2)$.

Matice sousednosti (anglicky *adjacency matrix*)

Graf representujeme jako matici nul a jedniček o rozměrech $n \times n$ (tj. $\mathbf{A} \in \{0,1\}^{n \times n}$), kde platí

$$A_{ij} = 1 \quad \text{iff} \quad G \text{ má hrany mezi vrcholy } i \text{ a } j.$$

Varianty:

- A_{ij} je počet hran mezi vrcholy i a j,
- A_{ij} je váha (cena/délka) hrany,
- A_{ij} je ± 1 podle orientace hrany.

Příklad

Jaká je prostorová náročnost uložení matice sousednosti?

1. $\Theta(n)$
2. $\Theta(m)$
3. $\Theta(m + n)$
4. $\Theta(n^2)$

Příklad

Jaká je prostorová náročnost uložení matice sousednosti?

1. $\Theta(n)$
2. $\Theta(m)$
3. $\Theta(m+n)$
4. $\Theta(n^2)$

Kvadratická prostorová náročnost je v pořádku pro hustý graf. V řídkém grafu plýtváme prostorem.

List sousednosti (anglicky *adjacency list*)

- pole (list) vrcholů
- pole (list) hran
- každá hrana odkazuje odpovídající dvojici vrcholů
- každý vrchol odkazuje na incidentní hrany

Příklad

Jaká je prostorová náročnost uložení listu sousednosti?

1. $\Theta(n)$
2. $\Theta(m)$
3. $\Theta(m + n)$
4. $\Theta(n^2)$

Která reprezentace je lepší?

- Záleží na hustotě.
- Záleží na operacích, které chceme poskytovat
  - List sousednosti je lepší pro prohledávání grafu.
  - Například web má 10^{10} vrcholů.

DIJKSTRŮV ALGORITMUS

Single-source shortest path

Vstup:

- Orientovaný graf $G = (V, E)$
- každá hrana má nezápornou váhu
- počáteční vrchol s

Výstup:

- Pro každý vrchol $v \in V$ spočtěme

$$L(v) = \text{délka nejkratší cesty z } s \text{ do } v.$$

Předpoklady:

- (pro pohodlnost) Vše je dostupné z s.
- Délky hran jsou nezáporné.

Příklad

Jaké jsou délky nejkratších cest z vrcholu s do vrcholů s, v, w, t v grafu na tabuli?

- $0, 1, 2, 3$
- $0, 1, 4, 7$
- $0, 1, 4, 6$
- $0, 1, 3, 5$

Neumíme redukovat problém na BFS?

1. Nepočítá už BFS nejkratší cesty v lineárním čase?

Neumíme redukovat problém na BFS?

1. Nepočítá už BFS nejkratší cesty v lineárním čase?
   - Ano, ale $l_e = 1$ pro všechny hrany $e \in E$

Neumíme redukovat problém na BFS?

1. Nepočítá už BFS nejkratší cesty v lineárním čase?
   - Ano, ale $l_e = 1$ pro všechny hrany $e \in E$
2. Nešlo by nahradit celočíselné váhy cestami jednotkové délky?

Neumíme redukovat problém na BFS?

1. Nepočítá už BFS nejkratší cesty v lineárním čase?
   - Ano, ale $l_e = 1$ pro všechny hrany $e \in E$
2. Nešlo by nahradit celočíselné váhy cestami jednotkové délky?
   - Neškáluje, problém se může značně zvětšit.

Neumíme redukovat problém na BFS?

1. Nepočítá už BFS nejkratší cesty v lineárním čase?
   - Ano, ale $l_e = 1$ pro všechny hrany $e \in E$
2. Nešlo by nahradit celočíselné váhy cestami jednotkové délky?
   - Neškáluje, problém se může značně zvětšit.

Řešením je Dijkstrův algoritmus.

Pseudokód

function DIJKSTRA-ALGORITHM(*graph, s*) **returns** shortest path to each v

$X \leftarrow \{s\}$ ▷ Množina vrcholů, pro které známe $L(v)$

$A[s] = 0$ ▷ Spočtená délka cesty

$B[s] = emptypath$ ▷ Spočtená cesta

while $X \neq V$ **do**

$(v^*, w^*) \leftarrow$ hrana $(v, w) \in E$ s $v \in X$, $w \notin X$, která minimalizuje

$$A[v] + l_{(v,w)}$$

$X \leftarrow X \cup \{w^*\}$

$A[w^*] = A[v^*] + l_{(v^*,w^*)}$

$B[w^*] = B[v^*] \cup (v^*, w^*)$

end while

end function

Příklad

KOREKTNOST DIJKSTROVA ALGORITMU

Korektnost Dijkstrova algoritmu

Tvrzení (Dijkstra, 1959) *Pro každý orientovaný graf s nezápornými délkami hran Dijsktrův algoritmus spočte korektně všechny nejkratší cesty z vrcholu s, tj.*

$$\forall v \in V : A[v] = L(v).$$

Důkaz

Indukcí přes počet iterací.

IMPLEMENTACE DIJKSTROVA ALGORITMU

Příklad

Jaký je čas běhu, pokud Dijkstrův algoritmus naimplementujeme naivně podle pseudokódu?

1. $\Theta(m + n)$
2. $\Theta(n \log n)$
3. $\Theta(n^2)$
4. $\Theta(mn)$

Lepší implementace?

- Stále opakujeme operaci hledání minima.
- Nešlo by tyto opakované výpočty urychlit pomocí lepší organizace dat?

Halda

- Datová struktura co provádí operace `insert` a `extract-min` v $\mathcal{O}(\log n)$.
- Perfektně vyvážený binární strom.
- V každém vrcholu je splněná vlastnost haldy: velikost klíče je menší než velikosti klíčů potomků.
- `extract-min`- odebereme vrchol, na jeho místo vložíme poslední uzel a probubláme dolů
- `insert`- vložíme na konec a probubláme nahoru
- Dále máme možnost odebrat z prostředku (probublávání nahoru nebo dolů podle potřeby)

Dijkstra s prioritní frontou.

Invariant 1 *V haldě máme vrcholy z množiny* $V \setminus X$.

Dijkstra s prioritní frontou.

Invariant 1 *V haldě máme vrcholy z množiny* $V \setminus X$.

Invariant 2 *Pro každý* $v \notin X$ *platí, že* $\mathrm{key}[x]$ *je nejmenší Dijkstrovo hladové skóre ze všech hran* $(u, v) \in E$ *s* $u \in X$ *(popř.* ∞*, neexistuje-li taková hrana)*

Dijkstra s prioritní frontou.

Invariant 1 *V haldě máme vrcholy z množiny* $V \setminus X$.

Invariant 2 *Pro každý* $v \notin X$ *platí, že* $\mathrm{key}[x]$ *je nejmenší Dijkstrovo hladové skóre ze všech hran* $(u, v) \in E$ *s* $u \in X$ *(popř.* ∞*, neexistuje-li taková hrana)*

Pokud udržíme tyto invarianty pravdivé, pak `extract-min` vede ke správnému vrcholu w^*, který přidáme do X v dalším kroku algoritmu.

Jak udržet Invariant 2 pravdivý?

- Mění se množina hran, která přechází hranici z X do $V \setminus X$.
- Přidáním w se mohlo snížit minimální skóre.

Jak udržet Invariant 2 pravdivý?

- Mění se množina hran, která přechází hranici z X do $V \setminus X$.
- Přidáním w se mohlo snížit minimální skóre.

Když je w přidáno, provedeme následující kroky:

for each (w, v) **in** E **do**
    odeber v z haldy
    přepočítej $\mathrm{key}[v] = \min\{\mathrm{key}[v], A[w] + l_{wv}\}$
    znovu vlož v do haldy
end for

Čas běhu (list sousednosti)

- $n-1$ krát provedeme operaci `extract-min`
- Každá hrana způsobí maximálně jednu dvojici operací `delete` a `insert`.
- Čas běhu je tedy

$$\mathcal{O}\left((n-1)\log n + m\log n\right) = \mathcal{O}\left((m+n)\log n\right).$$

Čas běhu (matice sousednosti)

- Pro nalezení všech hran, co vedou z vrcholu je třeba $\Theta(n)$ práce.
- Nepotřebujeme haldu, nalezení minima stačí v $\Theta(n)$.
- Místo haldy postačuje pole.
- $n - 1$ operací `extract-min`
- Pro každou $\Theta(n)$ práce, celkem tedy

$$\Theta(n^2).$$

Vzorová implementace Dijkstrova algoritmu

- Zkuste nejprve naimplementovat Dijsktrův algoritmus sami.
- V Javě např. `http://keithschwarz.com/interesting/code/?dir=dijkstra`.
- Fibbonacciho halda je pouze pro lepší asymptotickou složitost. Poskytuje stejné operace jako klasická binární halda, pouze je v součtu rychlejší. Implementace je na `http://keithschwarz.com/interesting/code/?dir=fibonacci-heap`. Pokud použijete `PriorityQueue` z knihoven Javy, tak kód bude pomalejší.
- V C++ je implementace např. `http://www.geeksforgeeks.org/greedy-algorithms-set-6-dijkstras-shortest-path-algorithm/`. (Pozor, implementace je tentokrát pro matici sousednosti.)

Přestávka

Programovací příklady

- Jednoduchá: UVA 10986 - Sending email
- Středně těžká: UVA 10801 - Lift Hopping
- Těžší: UVA 11635 - Hotel booking

PROBLÉM HRAN ZÁPORNÉ DÉLKY

Příklad

Dijkstra na grafu se záporně ohodnocenými hranami.

Zkusme přidat kladnou konstantu ke všem hranám.

Zkusme přidat kladnou konstantu ke všem hranám.

- Problém, že nejkratší cesta nemusí být stále nejkratší.
- Cesty mají rozdílnou délku!

Závěr

- Dijkstra může spočíst špatně nejkratší cestu, pokud graf obsahuje záporně ohodnocené hrany.
- Dijkstrův algoritmus je rychlý, ale ne vždy korektní.
- Dijkstrův algoritmus se špatně distribuuje.
- Řešením je Bellman-Fordův algoritmus

Definice nejkratší cesty v grafu s cykly záporné délky

- Funguje naše dosavadní chápání nejkratší cesty pro graf s cykly záporné délky?

Definice nejkratší cesty v grafu s cykly záporné délky

- Funguje naše dosavadní chápání nejkratší cesty pro graf s cykly záporné délky?
  - Nefunguje. Nejkratší cesta občas neexistuje.

Definice nejkratší cesty v grafu s cykly záporné délky

- Funguje naše dosavadní chápání nejkratší cesty pro graf s cykly záporné délky?
  - Nefunguje. Nejkratší cesta občas neexistuje.
- Zkusme spočíst nejkratší cestu z s do v, která neobsahuje cyklus.

Definice nejkratší cesty v grafu s cykly záporné délky

- Funguje naše dosavadní chápání nejkratší cesty pro graf s cykly záporné délky?
  - Nefunguje. Nejkratší cesta občas neexistuje.
- Zkusme spočíst nejkratší cestu z s do v, která neobsahuje cyklus.
  - NP-těžký problém ...

Definice nejkratší cesty v grafu s cykly záporné délky

- Funguje naše dosavadní chápání nejkratší cesty pro graf s cykly záporné délky?
  - Nefunguje. Nejkratší cesta občas neexistuje.
- Zkusme spočíst nejkratší cestu z s do v, která neobsahuje cyklus.
  - NP-těžký problém ...
- Předpokládejme, že graf neobsahuje cyklus se zápornou délkou.
  - Chceme, aby algoritmus byl schopný takovýto cyklus objevit.

Příklad

Nechť graf G neobsahuje cyklus se zápornou délkou. Pak platí:

1. Pro všechny vrcholy existuje nejkratší cesta s nejvýše $n-1$ hranami.
2. Pro všechny vrcholy existuje nejkratší cesta s nejvýše n hranami.
3. Pro všechny vrcholy existuje nejkratší cesta s nejvýše m hranami.
4. Každá nejkratší cesta může obsahovat libovolný počet hran.

Single-source shortest path, revisited

Vstup:

- Orientovaný graf $G = (V, E)$
- každá hrana má váhu l_e
- počáteční vrchol s

Výstup:

1. Pro každý vrchol $v \in V$ spočtěme

$$L(v) = \text{délka nejkratší cesty z } s \text{ do } v.$$

nebo

2. Identifikujeme cyklus se zápornou délkou.

BELLMAN-FORDŮV ALGORITMUS

Dynamické programování

- Definujeme jak řešit větší problém pomocí známých řešení podproblémů.
- Podobné rekurzi, ale
  - jdeme odspodu, ne shora a
  - řešení podproblémů máme první.

Aplikace DP na SSSP

- Podcesta nejkratší cesty je sama o sobě nejkratší cestou.
- Omezíme problém podle počtu hran v nejkratší cestě.

Aplikace DP na SSSP

- Podcesta nejkratší cesty je sama o sobě nejkratší cestou.
- Omezíme problém podle počtu hran v nejkratší cestě.

Máme tedy jeden podproblém na

1. každý vrchol v
2. počet hran, které připouštíme na nejkratší cestě.

Lemma *Nechť $G = (V, E)$ je orientovaný graf s délkami hran l_e a počátečním vrcholem s. Pro všechny $v \in V$ a $i \in \mathbb{N}$ označme jako P nejkratší cestu z s do v s nejvýše i hranami (cykly jsou povoleny). Pak platí*

1. *pokud má P nejvýše $i - 1$ hran, pak je i nejkratší cestou z s do v s nejvýše $i - 1$ hranami;*
2. *pokud má P právě i hran s poslední hranou z w do v, pak cesta P' je nejkratší cestou z s do w s nejvýše $i - 1$ hranami.*

Důkaz

Příklad

Kolik existuje kandidátu pro optimální řešení pro podproblém zahrnující vrchol v a i hran?

1. 2
2. $1 + \text{in} - \text{degree}(v)$
3. $n - 1$
4. n

Bellman-Fordův algoritmus

- Nechť $L_{i,v}$ je délka nejkratší cesty z s do v, která připouští nejvýše i hran.
- Pak $\forall v \in V, i \in \mathbb{N}$ platí

$$L_{i,v} = \min \begin{cases} L_{i-1,v}, \\ \min_{(w,v)\in E}\{L_{i-1,v} + l_{wv}\}. \end{cases}$$

Korektnost vychází z předchozího lemmatu.

Pokud graf G neobsahuje cyklus záporné délky

Pokud graf G neobsahuje cyklus záporné délky, pak

- nejkratší cesty neobsahují cykly,
- mají nejvýše $n - 1$ hran,
- stačí uvažovat pouze i do $n - 1$.

Pseudokód

function BELLMAN-FORD(*graph*, s) **returns** shortest path to each v

$A \leftarrow$ prázdné pole indexované i a v

$A[0, s] = 0$

$\forall v \in V \setminus \{s\} : A[0, v] = \infty$

for $i = 1$ **to** $n - 1$ **do**

for each $v \in V$ **do**

$A[i, v] = \min \left\{ A[i-1, v], \min_{(w,v) \in E} \{ A[i-1, w] + l_{wv} \} \right\}$

end for

end for

end function

Příklad

Příklad

Jaký je čas běhu Bellman-Fordova algoritmu?

1. $\mathcal{O}(n^2)$
2. $\mathcal{O}(mn)$
3. $\mathcal{O}(n^3)$
4. $\mathcal{O}(m^2)$

Předčasné zastavení

- Pokud se nic nezmění pro nějaké $j < n - 1$, pak víme, že

$$\forall v \in V : A[j, v] = A[j - 1, v]$$

- Nic se nezmění ani pro $j + 1$, ani nikdy dále.
- Můžeme zastavit výpočet dříve.

DETEKCE ZÁPORNÝCH CYKLŮ

Co se stane při existenci cyklu záporné délky

Tvrzení G *nemá cyklus záporné délky*

$$\Leftrightarrow$$

pokud přidáme jednu iteraci B-F algoritmu, pak
$\forall v \in V : A[n-1, v] = A[n, v]$.

Důkaz

Vzorová implementace Bellman-Fordova algoritmu.

- Zkuste nejprve naimplementovat Bellman-Fordův algoritmus sami.
- V Javě např.
  `http://www.geekviewpoint.com/java/graph/bellman_ford_shortest_path`.
- V C++ je implementace např. `http://www.geeksforgeeks.org/dynamic-programming-set-23-bellman-ford-algorithm/`.

Programovací příklady

- Jednoduchá: UVA 558 - Wormholes
- Středně těžká: UVA 436 - Arbitrage (II) (Řešení je na dalších slidech, ale zkuste na ně přijít sami a nečíst ho.)

Arbitrage detection

- Máme převodní kurzy několika měn.
- Detekujeme cyklus, kde je součin kurzů ≥ 1.

Arbitrage detection

- Máme převodní kurzy několika měn.
- Detekujeme cyklus, kde je součin kurzů ≥ 1.
- Formulujeme jako detekci negativního cyklu po nahrazení délek hran zápornou hodnotou jejich logaritmu.

Přestávka

JOHNSONŮV ALGORITMUS

All-pairs shortest path

Vstup:

- Orientovaný graf $G = (V, E)$
- každá hrana má váhu l_e

Výstup:

1. Pro každou dvojici vrcholů $u, v \in V$ spočtěme délku nejkratší cesty z u do v.

nebo

2. Identifikujeme cyklus se zápornou délkou.

Souvislost s předchozími

APSP se redukuje na n volání SSSP.

Algoritmus	Čas běhu	Přípustná data
n volání Dijkstry	$\mathcal{O}(mn \log n)$	nezáporné délky hran
n volání Bellman-Forda	$\mathcal{O}(mn^2)$	obecný graf
Floyd-Warshall	$\mathcal{O}(n^3)$	obecný graf
Johnson[1]	$\mathcal{O}(mn \log n)$	obecný graf

[1] 1 volání B-F, n volání Dijkstry

Převážení hran

Nechť $G = (V, E)$ je orientovaný graf s obecnými délkami hran l_e. Zafixujme číslo p_v pro každý vrchol $v \in V$. Nech pro každou hranu $e = (u, v)$ grafu G je

$$l'_e = l_e + p_u - p_v.$$

Příklad

Je-li P libovolná cesta z s do t délky L, jaká je délka P', pokud převážíme hrany?

1. L
2. $L + p_s + p_t$
3. $L + p_s - p_t$
4. $L - p_s + p_t$

Důsledek

- Každá cesta z s do t se změní o stejnou hodnotu.
- Převážení zachovává uspořádání cest podle jejich délek.
- Nejkratší cesta zůstává nejkratší cestou.

Existují vždy takové váhy, aby se ceny hran staly nezáporné?

Příklad

- Přidejme nový počáteční vrchol.
- Spusťme Bellman-Fordův algoritmus a spočtěme nejkratší cestu z s do libovolného vrcholu.
- hodnoty použijme jako váhy vrcholů

$$p_v = \text{ délka nejkratší cesty z } s \text{ do } v.$$

Hrany mají po převážení nezáporné váhy

Tvrzení *Pro všechny hrany* $e = (u, v)$ *platí*

$$l'_e = l_e + p_u - p_v \geq 0.$$

Důkaz

Pseudokód

function JOHNSON-ALGORITHM(*graph*)

vytvoř G' z G přidáním vrcholu s a hran (s, v) o ceně 0 pro všechny $v \in V$

spusť Bellman-Fordův algoritmus na G' s počátečním vrcholem s $\triangleright$ zastav, pokud nalezneš cyklus záporné délky

$\forall v \in V : p_v \leftarrow$ délka nejkratší cesty z s do v v G'

$\forall e = (u, v) \in E : l'_e = l_e + p_u - p_v$

for each $v \in V$ **do**

spusť Dijkstrův algoritmus na grafu G s l'_e a s počátkem u

označ vypočtené délky jako $d'(u, v)$ pro všechny $v \in V$

end for

$\forall u, v \in V : d(u, v) \leftarrow d'(u, v) - p_u + p_v$

end function

Čas běhu

$$\mathcal{O}(n) + \mathcal{O}(mn) + \mathcal{O}(m) + \mathcal{O}(mn \log n) + \mathcal{O}(n^2) = \mathcal{O}(mn \log n).$$

To je mnohem lepší než Floyd-Warshallův algoritmus pro řídké grafy.

Vzorová implementace Johnsonova algoritmu.

- Zkuste nejprve naimplementovat Johnsonův algoritmus sami.
- V Javě např. `http://www.sanfoundry.com/java-program-to-implement-johnsons-algorithm/`.
- V C++ je implementace např. `https://gist.github.com/ashleyholman/6793360`.

Programovací příklady

- UVA 821 - Page Hopping

References

- heavily inspired by Tim Roughgarden's online courses, `http://theory.stanford.edu/~tim/videos.html`
- Robert Sedgewick and Kevin Wayne, Algorithms, `http://algs4.cs.princeton.edu/home/`, namely `http://algs4.cs.princeton.edu/44sp/`
- Halim, S., Halim, F., Skiena, S. S., & Revilla, M. A. (2013). Competitive Programming 3. Lulu Independent Publish.

Děkuji za pozornost.
Čas na otázky!