Dijkstra revisited, Bellman-Ford, Floyd-Warshall

Petr Ryšavý

24. dubna 2025

Katedra počítačů, FEL, ČVUT

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.

Pseudokód

function DIJKSTRA-ALGORITHM(*graph, s*) **returns** nejkratší cesty z s

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

dist$[s] = 0$ ▷ Spočtená délka cesty

path$[s] = emptypath$ ▷ Spočtená cesta

while *VISITED* $\neq V$ **do**

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

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

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

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

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

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.*

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

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?

Lepší implementace?

- Stále opakujeme operaci hledání minima.
- Nešlo by tyto opakované výpočty urychlit pomocí lepší organizace dat?
- Ano - šlo, pomocí *haldy*.

Dijkstra s prioritní frontou.

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

Dijkstra s prioritní frontou.

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

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

Dijkstra s prioritní frontou.

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

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

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

Jak udržet Invariant 2 pravdivý?

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

Jak udržet Invariant 2 pravdivý?

- Mění se množina hran, která přechází hranici z $VISITED$ do $V \setminus VISITED$.
- 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 $\mathrm{key}[v] = \min\{\mathrm{key}[v], dist[w] + l_{wv}\}$
    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

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

HRANY ZÁPORNÝCH DÉLEK

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

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. $1 + \text{indegree}(v)$
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
$\forall v \in V : A[n-1, v] = A[n, v]$.

Arbitrage detection

- Máme převodní kurzy několika měn.
- Detekujeme cyklus, kde je součin kurzů ≥ 1.
- UVA 104 - Arbitrage

Arbitrage detection

- Máme převodní kurzy několika měn.
- Detekujeme cyklus, kde je součin kurzů ≥ 1.
- UVA 104 - Arbitrage
- Formulujeme jako detekci negativního cyklu po nahrazení délek hran zápornou hodnotou jejich logaritmu.

FLOYD-WARSHALLŮV 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

Proč další algoritmus pro hledání nejkratších cest?

- Floyd-Warshall funguje i pro záporné délky hran v čase $\mathcal{O}(n^3)$
- Lepší než Bellman-Ford spuštěný n-krát
- V hustých grafech srovnatelný s n běhy Dijkstry
- Dodnes je otevřenou otázkou, zda lze dosáhnout v hustých grafech výrazně lepšího času než $\mathcal{O}(n^3)$

Rekurzivní struktura

- Podobně jako u Bellman-Forda omezíme vrcholy, přes které cesta z s do t může procházet
- Omezíme se na konkrétní výběr vrcholů

Hlavní myšlenka

Lemma *Nechť G neobsahuje cyklus záporné délky a vrcholy $V = \{1, 2, \ldots, n\}$ jsou očíslované a nechť $V^{(k)} = \{1, 2, \ldots, k\}$. Zafixujme počátek $i \in V$ a cíl $j \in V$. Nechť P je nejkratší cesta (bez cyklu) z i do j taková, že všechny vnitřní vrcholy jsou z $V^{(k)}$. Pak platí jedna z následujících možností.*

- *Pokud není k na cestě P, pak P je nejkratší cesta z i do j s vnitřními vrcholy z $V^{(k-1)}$.*
- *Pokud je k vnitřním vrcholem cesty P, pak je zde jednou a P lze rozdělit na dvě podcesty P_1 z i do k a P_2 z k do j. Přitom platí že obě dvě cesty mají vnitřní vrcholy z $V^{(k-1)}$ a jsou nejkratšími cestami bez cyklu mezi odpovídajícími vrcholy.*

Příklad

Nechť A je trojrozměrné pole indexované $A[i, j, k]$ uchovávající délky nejkratších cest z i do j přes vrcholy z $\{1, 2, \ldots, k\}$. Jak bude inicializované $A[i, j, 0]$ pro případy, kdy $i = j$, $(i, j) \in E$ a $(i, j) \notin E$ (v tomto pořadí):

1. $0, 0, \infty$
2. $0, c_{i,j}, c_{i,j}$
3. $0, c_{i,j}, \infty$
4. $\infty, c_{i,j}, \infty$
5. $\infty, c_{i,j}, 0$

Pseudokód

function FLOYD-WARSHALL(*graph*) **returns** minimal path between all pairs of nodes

$A \leftarrow$ EMPTY-3D-ARRAY(n, n, n)

$$A[i,j,0] = \begin{cases} 0 & \text{if } i = j, \\ c_{ij} & \text{if } (i,j) \in \text{EDGES}(\textit{graph}), \\ \infty & \text{otherwise.} \end{cases}$$

for $k = 1$ **to** n **do**
  for $i = 1$ **to** n **do**
    for $j = 1$ **to** n **do**

$$A[i,j,k] = \min\{A[i,j,k-1], A[i,k,k-1] + A[k,j,k-1]\}$$

    end for
  end for
end for
return $A[.,.,n]$
end function

Detekce cyklu záporné délky

- Co se stane, máme-li v grafu cyklus záporné délky?

Detekce cyklu záporné délky

- Co se stane, máme-li v grafu cyklus záporné délky?
- Alespoň jedno $A[i, i, n]$ bude záporné.

Rekonstrukce cesty

- Stačí si zapamatovat pole $B[i, j]$ udávající nejvyšší vrchol na cestě z i do j
- Víme, že daný vrchol na cestě, je, můžeme ji rozdělit
- při druhém případu, kdy volíme $A[i, j, k] = A[i, k, k-1] + A[k, j, k-1]$ nastavujeme $B[i, j]$ na k

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!