

Korteste vej algoritmer, herunder Bellman-Ford og Dijkstra

Mads Ohm Larsen

6. april 2008

1 Definition af problemet

Givet en vægtet orienteret graf søger vi den korteste vej fra en knude til alle andre i grafen.

1.1 Optimal substruktur

Vi lader P være den korteste vej fra knuden u til knuden v . Et sted på denne vej ligger knuden z . P_1 må være den korteste vej fra u til z , og P_2 må ligeledes være den korteste vej fra z til v . Kunne vi skifte en af disse veje ud med noget kortere strider det mod vores antagelse at P var den korteste vej fra u til v .

2 Generelle relaxeringsalgoritme

Vi starter med at sætte alles forgænger til NIL og alles start vægt til ∞ . Til sidst sætter vi start knudens vægt til 0. Så kører vi alle knuderne igennem og ser om dens vægt, er større end en knude, der fører til den, plus den vægt den kant har. Altså om $d[v] > d[u] + w(u, v)$. Er den det, sætter vi v 's vægt til at være dette istedet, og laver dens forgænger om til u .

2.1 Egenskaber

1. Trekants ulighed: For alle kanter $(u, v) \in E$ har vi at $\delta(s, v) \leq \delta(s, u) + w(u, v)$
2. Øvregrænse-egenskab: Vi har altid $d[v] \geq \delta(s, v)$ for alle $v \in V$. Bliver $d[v] = \delta(s, v)$ skifter den ikke igen.
3. Ingen-vej-egenskab: Er der ikke en vej fra s til v har vi $d[v] = \delta(s, v) = \infty$
4. Convergens-egenskab: Samme som øvregrænse.
5. Vej-relaxerings-egenskab: Hvis $p = v_0, v_1, v_2, \dots, v_k$ er den korteste vej fra $s = v_0$ til v_k , og kanterne i p bliver relaxeret i rækkefølge, så vil $d[v_k] = \delta(s, v_k)$
6. Forgænger-subgraf-egenskab: Når $d[v] = \delta(s, v)$ for alle $v \in V$, så er forgænger subgrafen et korteste vej træ med rod i s .

3 Bellman-Ford algoritmen

I Bellman-Ford algoritmen må der gerne være negative kanter. Er der negative cykler returnere den FALSE.

Den fungerer ved at vi først initialisere alle knuderne. Derefter kører vi alle knuder igennem og for hver knude kører alle kanter igennem og relaxerer dem ($|V| - 1$ gange). Når den har gjort det prøver den en gang til at se om der er en hurtigere vej fra en knude til en anden. Er der det, må der være en negativ cyklus. Tiden er $O(V \cdot E)$, da initialiseringen tager $O(V)$ og vi derefter kører alle kanterne igennem for hver knude, $O(V \cdot E)$. I alt $O(V \cdot E)$.

4 Korteste vej i acykliske grafter

Sorter knuderne topologisk. Kør dem igennem og relaxer alle kanter præcis én gang!

4.1 Køretid

En topologisk sortering kan gøres i $\Theta(V + E)$ tid. Da vi kører alle kanter, der udgår fra hver knude, igennem præcis én gang tager dette $O(|E|)$ tid. Altså $O(V + E)$ tid ialt.

5 Dijkstra's algoritme

Der kan ikke være negative kanter. Lad S være en mængde af knuder for hvilke den korteste vej fra s allerede er fundet. Til at starte med er $S = \emptyset$, $d[s] = 0$ og $d[u] = \infty$ for $u \neq s$. Ved hver iteration vælger vi $u \notin S$ således at $d[u]$ er den mindste. Så relaxere vi alle kanter fra u og tilføjer u til S . Dette ligner meget bredde-først-søgning.

5.1 Korrekthed

Vi skal vise at når en knude, u , bliver tilføjet S , så er $d[u] = \delta(s, u)$. Dette vil vi bevise ved modstrid. Lad $u \neq s$ være den første knude der bliver tilføjet S med $d[u] > \delta(s, u)$.

Lad nu P være den korteste vej fra s til u .

Lad x være den sidste knude på P som er i S . Det er muligt at $x = s$.

Lad nu (x, y) være en kant på P . Det er muligt at $y = u$.

Da x blev tilføjet til S , blev (x, y) relaxeret. Derfor

$$\begin{aligned} d[y] &= \delta(s, x) + w(x, y) \\ &= \delta(s, y) \\ &\leq \delta(s, u) \\ &\leq d[u] \end{aligned}$$

Da vi allerede havde tilføjet u til S , havde vi at $d[u] \leq d[y]$, altså må $\delta(s, u) = d[u]$, altså en modstrid.

5.2 Køretid

Ved at implementere det som en min-hob, tager hver iteration $O(\lg V)$ tid, for at finde den mindste knude. Vi har også $|E|$ **DECREASE-KEY** operationer (Relax i løkken), der hver tager $O(\lg V)$ tid. Det tager $O(V)$ at bygge hoben. I alt $O((V + E) \lg V)$, som er $O(E \lg V)$ hvis alle knuder kan nås fra kildeknuden.