Dijkstrův algoritmus

Tento algoritmus dostane orientovaný graf s hranami ohodnocenými nezápornými čísly a nalezne v něm nejkratší cestu mezi dvěma zadanými vrcholy. Ve skutečnosti dělá o malinko více: najde totiž nejkratší cesty z jednoho zadaného vrcholu do všech ostatních.

Nechť v_0 je vrchol grafu, ze kterého chceme určit délky nejkratších cest. Budeme si udržovat pole délek zatím nalezených cest z vrcholu v_0 do všech ostatních vrcholů grafu. Navíc u některých vrcholů budeme mít poznamenáno, že cesta nalezená do nich je už ta nejkratší možná. Takovým vrcholům budeme říkat *definitivní*.

Na začátku inicializujeme v poli všechny hodnoty na ∞ kromě hodnoty odpovídající vrcholu v_0, kterou inicializujeme na 0 (délka nejkratší cesty z v_0 do v_0 je 0). V každém *kroku* algoritmu pak provedeme následující: Vybereme vrchol w, který ještě není definitivní, a mezi všemi takovými vrcholy je délka zatím nalezené cesty do něj nejkratší možná (v první kroku tedy v_0). Vrchol w prohlásíme za definitivní. Dále otestujeme, zda pro nějaký sousední vrchol v vrcholu w cesta z vrcholu v_0 do w a pak po hraně z w do v není kratší, než zatím nalezená cesta z v_0 do v, a je-li tomu tak, upravíme délku zatím nalezené cesty do v. Toto provedeme pro všechny takové vrcholy v.

Celý algoritmus skončí, pokud jsou už všechny vrcholy definitivní nebo všechny vrcholy, co nejsou definitivní, mají délku cesty rovnou ∞ (v takovém případě se graf skládá z více nesouvislých částí).

Předtím než dokážeme, že právě představený algoritmus opravdu nalezne délky nejkratších cest z vrcholu v_0, se zamysleme nad jeho časovou složitostí.

Pro každý z N vrcholů si délku dosud nalezené cesty uchováme v poli. Celý algoritmus má nejvýše N kroků, protože v každém kroku nám přibude jeden definitivní vrchol. Ten vybíráme jako minimum z délky aktuální cesty přes všechny dosud nedefinitivní vrcholy, kterých je $\mathcal{O}(N)$.

V každému kroku musíme zkontrolovat tolik vrcholů v, kolik hran vede z vrcholu w. Počet takových změn pro všechny kroky dohromady je pak nejvýše $\mathcal{O}(M)$, kde M je počet hran vstupního grafu. Z toho vyjde časová složitost $\mathcal{O}(N^2 + M)$, čili $\mathcal{O}(N^2)$, jelikož M je nejvýše N^2. Tuto implementaci Dijkstrova algoritmu najdete na konci naší kuchařky.

K uchovávání délek dosud nalezených nejkratších cest můžeme ovšem použít haldu. Ta bude na začátku obsahovat N prvků a v každém kroku se počet jejích prvků sníží o jeden: nalezneme a smažeme nejmenší prvek, to zvládneme v čase $\mathcal{O}(\log N)$, a případně upravíme délky nejkratších cest do sousedů právě zpracovávaného vrcholu. To pro každou hranu trvá rovněž $\mathcal{O}(\log N)$, celkově za všechny hrany tedy $\mathcal{O}(M \log N)$. Z toho vyjde celková časová složitost algoritmu $\mathcal{O}((N+M) \log N)$, a to je pro „řídké“ grafy (tedy grafy s $M \ll N^2$) výrazně lepší.

Vraťme se nyní k důkazu správnosti Dijkstrova algoritmu. Ukážeme, že po každém kroku algoritmu platí následující tvrzení: nechť A je množina definitivních vrcholů, pak délka dosud nalezené cesty z v_0 do v (v je libovolný vrchol grafu) je

délka nejkratší cesty $v_0 v_1 \ldots v_k v$ takové, že všechny vrcholy $v_0, v_1, \ldots, v_k$ jsou v množině A.

Tvrzení dokážeme indukcí dle počtu kroků algoritmu, které již proběhly. Zřejmě to platí před a po prvním kroku algoritmu (A je prázdná, potom obsahuje jen v_0), takže je třeba ukázat platnost tvrzení pro k-tý krok za předpokladu, že platí pro krok $k-1$ a všechny předchozí.

Nechť w je vrchol, který byl v předchozím kroku prohlášen za definitivní. Uvažme nejprve nějaký vrchol v, který je definitivní. Pokud $v = w$, tvrzení je triviální. V opačném případě ukážeme, že existuje nejkratší cesta z v_0 do v přes vrcholy z A, která nepoužívá vrchol w.

Označme D délku cesty z v_0 do v přes vrcholy A bez vrcholu w. Protože v každém kroku vybíráme vrchol s nejmenším ohodnocením a ohodnocení vybraných vrcholů v jednotlivých krocích tvoří neklesající posloupnost (váhy hran jsou nezáporné!), tak délka cesty z v_0 do w přes vrcholy z A je alespoň D. Ale potom délka libovolné cesty z v_0 do v přes w používající vrcholy z A je alespoň D. Z volby D pak víme, že existuje nejkratší cesta z v_0 do v přes vrcholy z A, která nepoužívá vrchol w.

Nyní uvažme takový vrchol v, který není definitivní. Nechť $v_0 v_1 \ldots v_k v$ je nejkratší cesta z v_0 do v taková, že všechny vrcholy $v_0, v_1, \ldots, v_k$ jsou v množině A.

Pokud $v_k = w$, pak jsme ohodnocení v změnili na délku této cesty v právě proběhlém kroku. Pokud $v_k \neq w$, pak $v_0 v_1, \ldots, v_k$ je nejkratší cesta z v_0 do v_k přes vrcholy z množiny A a tedy můžeme předpokládat, že žádný z vrcholů $v_1, \ldots, v_k$ není w (podle toho, co jsme si rozmysleli na konci minulého odstavce). Potom se ale délka cesty do v rovnala správné hodnotě už před právě proběhlým krokem.

Vzhledem k tomu, že po posledním kroku množina A obsahuje právě ty vrcholy, do kterých existuje cesta z vrcholu v_0, dokázali jsme, že náš algoritmus funguje správně.

Poznámky

- Dijkstrův algoritmus je možné snadno upravit tak, aby nám kromě délky nejkratší cesty i takovou cestu našel: u každého vrcholu si v okamžiku, kdy mu měníme ohodnocení, poznamenáme, ze kterého vrcholu do něj přicházíme. Nejkratší cestu do nějakého vrcholu pak zrekonstruujeme tak, že u posledního vrcholu této cesty zjistíme, který vrchol je předposlední, u předposledního, který je předpředposlední, atd.
- Existují i jiné než binární haldy, například k-regulární haldy, v nichž má každý prvek k následníků (rozmyslete si, jaká je v takové haldě časová složitost operací a jak nastavit k v závislosti na M a N, aby byl Dijkstrův algoritmus co nejrychlejší), nebo tzv. Fibonacciho halda, která dokáže upravit hodnotu prvku v konstantním čase. S tou pak umíme hledat nejkratší cesty v čase $\mathcal{O}(M + N \log N)$.

Dan Kráľ, Martin Mareš a Petr Škoda

Implementace Dijkstrova algoritmu

```
var N: integer;     { počet vrcholů }
    vahy: array[1..MAX, 1..MAX] of integer;
                { váhy hran, -1 = hrana neexistuje }
    delky: array[1..MAX] of integer;
                { délky zatím nalezených cest, -1 = nekonečno }
    def: array[1..MAX] of boolean;
                { definitivní? }

procedure Dijkstra(odkud: integer);
var i, w, v: integer;
begin
  for i:=1 to N do begin
    def[i]:=false;  delky[i]:=-1;
  end;
  delky[odkud]:=0;
  repeat
    w:=0;
    for i:=1 to N do
      if not def[i] and ((w=0) or (delky[i]<delky[w])) then w:=i;
    if w<>0 then begin
      def[w]:=true;
      for i:=1 to N do
        if (vahy[w][i]<>-1) and (delky[w]+vahy[w][i]<delky[i]) then
          delky[i]:=delky[w]+vahy[w][i]
    end
  until w=0;
end;
```

Úloha 18-3-4: Pochoutka pro prasátko

V lese sousedícím s poklidným rybníčkem našich hrošíků se objevilo hladem šilhající prasátko. Zaslechlo totiž zvěsti o Velké Bukvici, která si tou dobou lebedila v podzemí lesíku. Začalo tedy bez rozmyslu rejdit mezi stromy, leč brzy mu došly síly – byl to už přeci jenom nějaký čas od poslední mňamky. Budete umět prasátku poradit?

Les si představme jako čtvercovou síť, v jednom políčku prasátko, v jiném bukvice. Aby to nebylo tak jednoduché, prasátko se v lese může pohybovat jen podle určitých pravidel a každé z nich stojí nějaké kladné množství námahy.

Konkrétně – na vstupu dostanete rozměry lesa a pozici prasátka a bukvice spolu s pravidly, podle kterých se prasátko může pohybovat. Každé pravidlo obsahuje trojici čísel x y z, kde x a y je povolený posun v mřížce (o kolik se změní pozice prasátka ve čtvercové síti), zatímco z je úsilí, které prasátko musí vynaložit pro daný přesun. Vaším úkolem je najít a vypsat cestu od prasátka k bukvici. Na své cestě nesmí prasátko opustit lesík. A aby milý čuník hlady nepošel, musí být vynaložené úsilí nejmenší možné.

Příklad: Les má rozměry 6×6, poloha prasátka je $[3, 3]$ a poloha bukvice $[1, 5]$. Pohyb prasátka se řídí třemi pravidly 2 2 3, 1 1 1 a -4 0 5. Potom je pro prasátko nejvýhodnější použít dvakrát pravidlo 2 ($\rightarrow [5, 5]$) a pak jednou pravidlo 3 ($\rightarrow [1, 5]$). Vynaložená námaha je pak $2 \cdot 1 + 5 = 7$.

Úloha 22-1-1: Alčina interpretace

Máme velký dům se spoustou pokojů, mezi některými z nich vedou schodiště: z jednoho pokoje do druhého se buď stoupá, nebo klesá. Jen tak z legrace hledáme cestu z jednoho pokoje do druhého tak, abychom co nejméně krát musili přestat vycházet schody a začít scházet, nebo naopak přestat scházet a začít vycházet.