Zaprojektować i zaimplementować algorytm dokładny znajdujący najtańszą ścieżkę rozpinającą w grafie pełnym oparty o strategię dziel i zwyciężaj o złożoności $O(4^n n^{logn})$

Anna Bekas
Bartosz Woźniak

1 października 2017

1 Przedstawienie problemu

Projekt ma na celu opisanie oraz zaimplementowanie algorytmu rozwiązującego problem znalezienia najtańszej ścieżki rozpinającej w grafie przy zachowaniu złożoności $O(4^n n^{logn})$, gdzie $n = |V|$ - liczba wierzchołków w grafie.
Problem znalezienia najtańszej ścieżki rozpinającej, inaczej ścieżki Hamiltona, w grafie polega na znalezieniu ścieżki o najmniejszym koszcie przechodzącej przez wszystkie wierzchołki grafu dokładnie raz.

2 Opis rozwiązania

W celu znalezienia rozwiązania problemu, stosować będziemy algorytm rekurencyjny oparty o strategię dziel i zwyciężaj.
Danymi potrzebnymi do wywołania rekurencji są: podzbiór zbioru wierzchołków grafu G, wierzchołek startowy dla ścieżki oraz wierzchołek końcowy.
Algorytm polega na rekurencyjnym podziale grafu na dwie równe części aż do momentu w którym wybór najtańszej ścieżki jest trywialny. Podział następuje przez wybór wierzchołka dzielącego oraz wyznaczenie rozłącznych zbiorów A oraz B niezawierających wierzchołka startowego, końcowego ani rozważanego wierzchołka dzielącego. Następnie do zbioru A dodawany jest wierzchołek startowy oraz dzielący, a do zbioru B wierzchołek dzielący i końcowy.
Zakładamy, że prosty przypadek występuje gdy w grafie znajdują się trzy wierzchołki lub mniej. Wówczas zwracana zostaje najtańsza ścieżka złożona z tych wierzchołków. Po powrocie z rekurencji, otrzymujemy dwie ścieżki Hamiltona powstałe w wyniku podziału aktualnie rozważanego zbioru wierzchołków o ustalonym początku oraz końcu.
Wynik rozważanego wywołania funkcji uzyskamy dzięki połączeniu znalezionych

ścieżek za pomocą wierzchołka dzielącego.
Jeżeli znalezione rozwiązanie jest gorsze od dotychczas najlepszego, należy je odrzucić i rozważyć inny podział wierzchołków grafu.

3 Pseudokod

$BestPath \leftarrow null$

for all $(Start, End), Start, End \in V, Start \neq End$ **do**
    $path \leftarrow HPA(V, Start, End)$
    if $path.Cost < BestPath.Cost$ **then**
        $BestPath \leftarrow path$
    end if
end for

function HPA(G, Start, Finish)
    if $n \leqslant 3$ **then**
        $hpa \leftarrow$ Znajdź najtańszą ścieżkę między dwoma lub trzema wierzchołkami
        return
    end if

    $localBestPath \leftarrow null$

    if $n > 3$ **then**
        for all Center in $V - \{Start, Finish\}$ **do**
            $U \leftarrow V - \{Start, Center, Finish\}$
            for all A in U, $|A| = \left\lfloor \frac{|U|}{2} \right\rfloor$ **do**
                $B \leftarrow U - A$
                $A' \leftarrow$ podgraf grafu G zawierający $A \cup \{Start, Center\}$
                $B' \leftarrow$ podgraf grafu G zawierający $B \cup \{Center, Finish\}$

                $aHPA \leftarrow HPA(A', Start, Center)$
                $bHPA \leftarrow HPA(B', Center, Finish)$
                $hpa \leftarrow$ Połączona ścieżka $aHPA$ oraz $bHPA$

                if $hpa.Cost < localBestPath.Cost$ **then**
                    $localBestPath \leftarrow hpa$
                end if
            end for
        end for
    end if
return $localBestPath$
end function

4 Analiza poprawności

Pokażemy, że podany pseudokod jest realizacją algorytmu dokładnego - analizuje każdą możliwą do utworzenia ścieżkę oraz zwraca najlepsze ze znalezionych rozwiązań.
Algorytm rozważa każdą parę wierzchołków traktując je odpowiednio jako końcowy oraz startowy i dla każdej takiej dwójki wywołuje rekurencyjną funkcje znajdowania ścieżki Hamiltona między wybranymi punktami.
Każde z wywołań rekurencji sprawdza wszystkie ścieżki, które mogą powstać w rozważanym grafie G będącym podgrafem pierwotnego grafu, o początku i końcu zdefiniowanym przez zmienne $Start$ i $Finish$. Dzieje się to przez sprawdzenie wszystkich wyborów wierzchołka środkowego oraz wszystkich podziałów grafu na dwie części.
W każdej z części, będącej wywołaniem omawianej fukcji, obliczana jest najtańsza ścieżka bazująca na podanych do metody parametrach. W związku z tym, w omawianym wywołaniu, otrzymujemy dwie ścieżki $aHPA$ oraz $bHPA$, które są najkrótrzymi drogami w zadanych podgrafach. Posiadając informacje o wierzchołku $Start$, $Center$ oraz $Finish$ oraz wiedząc, że $Start$ jest początkiem ścieżki $aHPA$, a $Center$ jej końcem, będąc zarazem początkiem ścieżki $bHAP$, kończącej się w wierzchołku $Finish$, możemy utworzyć nową ścieżkę.
Budujemy więc ścieżkę będącą złączeniem wcześniej obliczonych fragmentów. Dzieje się tak poprzez złączenie wierzchołka końcowego $aHPA$ oraz wierzchołka początkowego $bHAP$ - wierzchiołka $Center$. Wówczas koszt nowo powstałej ścieżki hpa jest równy sumie złączonych ścieżek.
Algorytm HPA zwróci więc najtańszą znalezioną ścieżkę - znajdzie prawidłowe rozwiązanie problemu.

5 Analiza złożoności czasowej

Złożoność czasowa algorytmu $T(n)$ jest rzędu $O(4^n n^{logn})$.

Dowód. Przedstawiony algorytm jest algorytmem rekurencyjnym, zatem każde wywołanie funkcji posiada koszt obliczeń wykonywanych przez funkcje oraz koszt wywołania rekurencji. Możemy zatem zapisać:

$$T(n) = x(n) \cdot T\left(\frac{n}{2} + 1\right),$$

gdzie $x\left(n\right)$ jest kosztem operacji wykonywanych w danym wywołaniu funkcji.
Pokażemy, że można go ograniczyć przez $O\left(2^{n+bl}\right)$, gdzie $l = \log n$, b - pewna stała.

W każdym wywołaniu rekurencyjnym następuje:

- wybór wierzchołka dzielącego na $(n-2)$ sposoby
- dla każdego wierzchołka dzielącego: wyznaczenie zbiorów A oraz B na $\binom{n-3}{\frac{n-3}{2}}$ sposoby.

Złożonośc tych operacji można zatem ograniczyć korzystając ze wzoru Stirlinga:

$$n! \approx \left(\frac{n}{e}\right)^n \sqrt{2n\pi}$$

Przyjmując $k = n - 3$ otrzymujemy:

$$\binom{k}{\frac{k}{2}} = \frac{k!}{\frac{k}{2}! \cdot \frac{k}{2}!} \approx \frac{\left(\frac{k}{e}\right)^k \sqrt{2k\pi}}{\left(\left(\frac{\frac{k}{2}}{e}\right)^{\frac{k}{2}} \sqrt{2\frac{k}{2}\pi}\right)^2} = 2^k \frac{1}{\sqrt{\frac{k\pi}{2}}} = 2^{n-3} \frac{1}{\sqrt{\frac{(n-3)\pi}{2}}} = O(2^n).$$

Zatem koszt czasowy operacji wewnątrz funkcji reukrencyjnej

$$x(n) = O((n-3)2^n) = O(2^{n+b\log n}) = O(2^{n+bl})$$

Koszt znalezienia ścieżki Hamiltona można wyznaczyć w następujący sposób:

$$\begin{aligned} T(n) &= x(n) \cdot T\left(\frac{n}{2}+1\right) = C \cdot x(n) \cdot x\left(\frac{n}{2}+1\right) \cdot \ldots \cdot x(3) = \\ &= C \cdot 2^{n+b\log n} \cdot 2^{\frac{n}{2}+1+b(\log\frac{n}{2}+1)} \cdot \ldots \cdot 1 = \\ &= C \cdot 2^{n+b\log n+\frac{n}{2}+1+b(\log\frac{n}{2}+1)+\ldots+1} \leqslant C \cdot 2^{2n+b\log n\log n} = \\ &= 2^{2n+b\log n\log n+\log C} \leqslant 2^{2n+\log C} \cdot 2^{b\log n\log n} = O(2^{2n} \cdot n^{\log n}) \end{aligned}$$

6 Opis wejścia i wyjścia

6.1 Dane wejściowe

Daną wejściową jest graf pełny G, w którym V- zbiór wierzchołków grafu, $n = |V|$- liczba wierzchołków w grafie.

6.2 Dane wyjściowe

Algorytm zwraca znalezioną ścieżkę Hamiltona wraz z jej długością/kosztem, gdzie ścieżka jest ciągiem kolejnych wierzchołków $\{v_1, v_2, \ldots, v_n\}$.