

Szukanie drzew spinających o minimalnej średnicy

BFS

Wejście: graf G oraz dowolny wierzchołek $s \in V(G)$;

Wyjście: drzewo spinające, którego średnica jest minimalna.

Algorytm BreadthFirstSearch(G, s)

inicjalnie tylko wierzchołek s jest oznaczony;

niech kolejka L zawiera s ;

while L nie jest pusta **do begin**

$v := \text{remove}(L)$;

dla każdego nieoznaczonego sąsiada x wierzchołka v **do begin**

 oznacz x ;

$\text{add}(L, x)$

 dodaj krawędź $\{x, v\}$ do drzewa BFS;

end;

end;

return znalezione drzewo BFS;

Idea algorytmu

Rozważmy procedurę, która dla każdego wierzchołka v grafu G buduje drzewo BFS biorąc v w charakterze wierzchołka startowego s . Przez d oznaczmy szukaną minimalną średnicę.

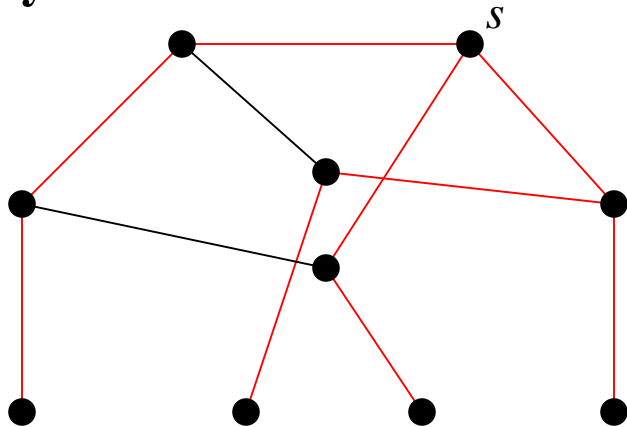
Lemat. *Jeśli T jest drzewem BFS z korzeniem s , to dla każdego wierzchołka v ścieżka łącząca s i v w T jest najkrótszą (w sensie liczby krawędzi) ścieżką pomiędzy v i s w G .*

Wniosek *Jeśli d jest parzyste, to istnieje drzewo BFS o średnicy d .*

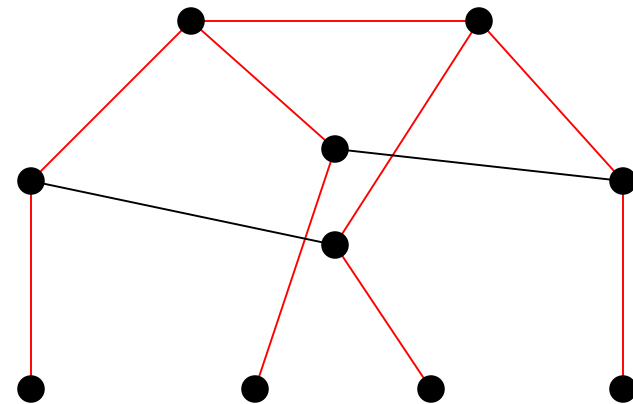
Uwaga *Jeśli d nie jest parzyste, to jest możliwe, że żadne z drzew BFS nie ma średnicy d (patrz następny przykład). Istnieje jednak drzewo BFS o średnicy $d + 1$.*

Idea algorytmu

Przykład



drzewo BFS



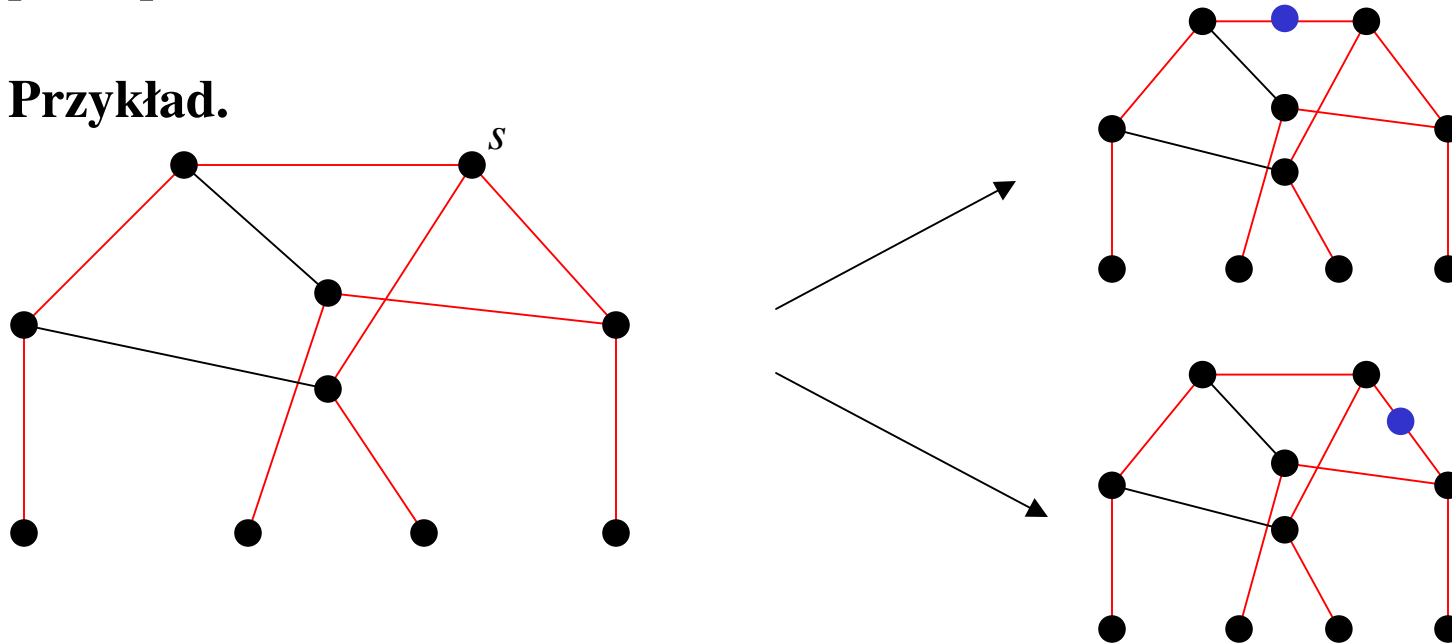
drzewo o minimalnej średnicy

Wniosek Niech T będzie drzewem BFS o najmniejszej średnicy. Jeśli d jest nieparzyste, to T jest drzewem o minimalnej średnicy dla G . Jeśli d jest parzyste, to drzewo optymalne dla G ma średnicę równą d lub $d - 1$.

Idea algorytmu

Aby poradzić sobie z przypadkami, gdy drzewo BFS o najkrótszej średnicy ma średnicę parzystą rozszerzamy nasz algorytm poprzez wstawienie wierzchołka kolejno na każdej krawędzi łączącej s z najgłębszym poddrzewem i kolejne wywołanie BFS dla nowego wierzchołka. Jeśli wówczas BFS zwróci drzewo tej samej wysokości co przed podziałem, to udało się zredukować średnicę o 1.

Przykład.



Algorytm

Algorytm MinimumDiameterSpanningTree(G)

for each $v \in V(G)$ **do begin**

$T := \text{BreadthFirstSearch}(G, v);$

if $\text{diam}(T) \bmod 2 = 0$ **then**

for each potomek u wierzchołka v w T , tż. wysokość poddrzewa
zakorzonego w u jest równa $\text{diam}(T)/2 - 1$ **do begin**

wstaw wierzchołek w na krawędzi $\{u, v\}$;

$T := \text{BreadthFirstSearch}(G, w);$

end;

end;

return drzewo o najkrótszej średnicy spośród wszystkich drzew T
wyznaczonych podczas działania algorytmu;

Szukanie drzew spinających o minimalnym stopniu

Szkic algorytmu

- Sformułowanie problemu: dany jest graf prosty G , znajdź drzewo spinające T , którego maksymalny stopień $\Delta(T)$ jest najmniejszy możliwy spośród stopni wszystkich drzew spinających. Przez Δ^* oznaczamy szukany minimalny stopień.
- Złe wieści: problem jest NP-trudny.
- z powyższego powodu ograniczamy się do podania algorytmu przybliżonego

Algorytm MinimumDegreeSpanningTree(G)

niech T będzie dowolnym drzewem spinającym G ;

while $\{u,v\} \in E(G)$ **and** ścieżka P łącząca u,v w T zawiera wierzchołek y taki, że $\deg_T(y) > \max\{ \deg_T(u), \deg_T(v) \} + 1$ **do begin**

 dodaj $\{u,v\}$ do T ;

 usuń dowolną z krawędzi incydentną do w i należącą do P ;

end;

Oszacowanie dolne

Tw. Niech G będzie grafem prostym oraz niech W będzie dowolnym podzbiorem $V(G)$. Jeśli $G - W$ posiada k składowych spójności, to

$$\Delta^* \geq \frac{|W| + k - 1}{|W|}.$$

Dowód: Wierzchołki w zbiorze W oraz podgrafy w $G - W$ traktujemy jako składowe spójności, których jest $|W| + k$. Skoro graf G jest spójny, potrzebujemy co najmniej $|W| + k - 1$ krawędzi, aby ‘spiąć’ te składowe spójności. Każda z tych krawędzi jest incydentna do pewnego wierzchołka z W , więc suma stopni wierzchołków z W wynosi co najmniej $|W| + k - 1$. Zatem średnia wartość stopnia wierzchołka w W to

$$\frac{|W| + k - 1}{|W|},$$

więc istnieje wierzchołek w W o co najmniej takim stopniu. W każdym drzewie spinającym k nie może być mniejsze, bo kończy dowód.

Pomocniczy lemat

Def. Jeśli T jest drzewem, to niech S_i oznacza podzbiór wierzchołków, których stopień w T wynosi co najmniej i .

Lemat Niech T, Δ będą odpowiednio drzewem spinającym znalezionym przez nasz algorytm oraz stopniem tego drzewa. Istnieje indeks i z zakresu $\Delta - \log_2 n, \dots, \Delta$ taki, że $|S_{i-1}| \leq 2|S_i|$.

Dowód: Mamy, że $|S_\Delta| \geq 1$. Przypuśćmy, że teza nie zachodzi. Wówczas

$$1 \leq |S_\Delta| < \frac{|S_{\Delta-1}|}{2} < \frac{|S_{\Delta-2}|}{4} < \dots < \frac{|S_{\Delta-\log_2 n}|}{2^{\log_2 n}} = \frac{|S_{\Delta-\log_2 n}|}{n}$$

skąd dostajemy

$$n < |S_{\Delta-\log_2 n}|$$

co nie jest możliwe, gdyż zbiory S_i są podzbiorem $V(T)$.

Oszacowanie górne

- poniżej zakładamy, że indeks i jest dobrany w taki sposób, aby spełniał tezę poprzedniego lematu;
- zmierzamy do podania górnego oszacowania na Δ wyrażonego jako funkcja Δ^* oraz n ;
- poniżej przez stopnie dotyczą drzewa T oraz wszystkie krawędzie o których mowa to elementy $E(T)$;
- zauważmy, że suma stopni wierzchołków w S_i wynosi co najmniej $i|S_i|$;
- pomiędzy wierzchołkami w S_i jest co najwyżej $2(|S_i| - 1)$ krawędzi;
- oznacza to, że krawędzi o jednym ‘końcu’ w zbiorze S_i jest co najmniej $i|S_i| - 2(|S_i| - 1)$;
- jeśli usuniemy wierzchołki w S_i z drzewa T to otrzymamy co najmniej $i|S_i| - 3(|S_i| - 1)$ składowe spójności;

Oszacowanie górne

- zauważmy, że każda ścieżka zawarta w G i łącząca dowolne dwie składowe spójności $T - S_i$ zawiera wierzchołek o stopniu $i - 1$, gdyż w przeciwnym razie warunek pętli while algorytmu nie jest fałszywy;
- oznacza to, że $G - S_{i-1}$ zawiera co najmniej $i|S_i| - 3(|S_i| - 1)$ składowe spójności;
- użyjemy wcześniejszego lematu przyjmując S_{i-1} w charakterze zbioru W oraz $i|S_i| - 3(|S_i| - 1)$ jako liczbę k :

$$\begin{aligned}
 \Delta^* &\geq \frac{|S_{i-1}| + |S_i|(i-3) + 3 - 1}{|S_{i-1}|} \\
 &= 1 + (i-3) \frac{|S_i|}{|S_{i-1}|} \\
 &\geq 1 + (i-3) \frac{1}{2} \\
 &= \frac{i-1}{2}
 \end{aligned}$$

Oszacowanie górne

- otrzymaliśmy zatem $\Delta^* \geq (i - 1)/2$;
- indeks i był dobrany tak, że $i \geq \Delta - \log_2 n$;
- podstawiając mamy

$$\Delta^* \geq \frac{\Delta - \log_2 n - 1}{2},$$

co ostatecznie daje twierdzenie:

Tw. *Algorytm MinimumDegreeSpanningTree dla dowolnego grafu G znajduje jego drzewo spinające o stopniu Δ takim, że*

$$\Delta \leq 2\Delta^* + \log_2 n + 1,$$

gdzie Δ^ to szukana wartość optymalna.*