Algorytmiczna teoria grafów

Przeszukiwanie grafów, drzewa spinające, drzewa wyrażeń arytmetycznych

dr Hanna Furmańczyk

Wykład 3

$*$ **17.03.2018** $*$

Twierdzenie

Niech T będzie grafem o n wierzchołkach. Wówczas następujące warunki są równoważne:

1. T jest drzewem.

Twierdzenie

Niech T będzie grafem o n wierzchołkach. Wówczas następujące warunki są równoważne:

1. T jest drzewem.
2. T nie zawiera cykli i ma $n-1$ krawędzi.

Twierdzenie

Niech T będzie grafem o n wierzchołkach. Wówczas następujące warunki są równoważne:

1. T jest drzewem.
2. T nie zawiera cykli i ma $n-1$ krawędzi.
3. T jest spójny i ma $n-1$ krawędzi.

Drzewa

Twierdzenie

Niech T będzie grafem o n wierzchołkach. Wówczas następujące warunki są równoważne:

1. T jest drzewem.
2. T nie zawiera cykli i ma $n - 1$ krawędzi.
3. T jest spójny i ma $n - 1$ krawędzi.
4. T jest spójny, ale usunięcie dowolnej krawędzi e rozspaja T (każda krawędź jest mostem).

Twierdzenie

Niech T będzie grafem o n wierzchołkach. Wówczas następujące warunki są równoważne:

1. T jest drzewem.
2. T nie zawiera cykli i ma $n-1$ krawędzi.
3. T jest spójny i ma $n-1$ krawędzi.
4. T jest spójny, ale usunięcie dowolnej krawędzi e rozspaja T (każda krawędź jest mostem).
5. Dowolne dwa wierzchołki grafu T połączone są dokładnie jedną drogą.

Twierdzenie

Niech T będzie grafem o n wierzchołkach. Wówczas następujące warunki są równoważne:

1. T jest drzewem.
2. T nie zawiera cykli i ma $n-1$ krawędzi.
3. T jest spójny i ma $n-1$ krawędzi.
4. T jest spójny, ale usunięcie dowolnej krawędzi e rozspaja T (każda krawędź jest mostem).
5. Dowolne dwa wierzchołki grafu T połączone są dokładnie jedną drogą.
6. T nie zawiera cykli, lecz dodanie dowolnej nowej krawędzi tworzy dokładnie jeden cykl.

Twierdzenie

W każdym nietrywialnym drzewie istnieją przynajmniej dwa liście.

Dowód

Przez sprzeczność:
$2(|V|-1)+1 \leq \sum_{v \in V(T)} \deg(v) = 2|E| = 2(|V|-1)$

Definicja

Niech $G = (V(G), E(G))$ będzie grafem i niech $H = (V(H), E(H))$ będzie jego podgrafem. Mówimy, że H jest podgrafem spinającym grafu G, jeśli $V(H) = V(G)$.

Definicja

Niech $G = (V(G), E(G))$ będzie grafem i niech $H = (V(H), E(H))$ będzie jego podgrafem. Mówimy, że H jest podgrafem spinającym grafu G, jeśli $V(H) = V(G)$.

Definicja

Drzewo spinające grafu $G = (V, E)$ jest to podgraf spinający będący drzewem.

Definicja

Niech $G = (V(G), E(G))$ będzie grafem i niech $H = (V(H), E(H))$ będzie jego podgrafem. Mówimy, że H jest podgrafem spinającym grafu G, jeśli $V(H) = V(G)$.

Definicja

Drzewo spinające grafu $G = (V, E)$ jest to podgraf spinający będący drzewem.

Kilka własności

* Każdy spójny graf zawiera drzewo spinające.
* Niech $G = (V(G), E(G))$ będzie grafem, $T = (V(T), E(T))$ jego drzewem spinającym i niech $e \in E(G) \setminus E(T)$. Wówczas $T + e = (V(T), E(T) \cup \{e\})$ zawiera jeden cykl.

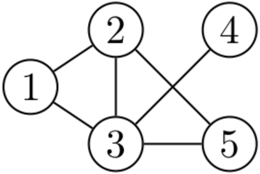
Algorytmy konstrukcji drzewa spinającego

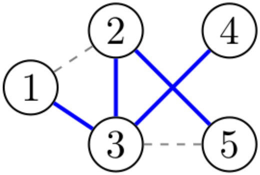
1. Algorytm zachłanny
2. Drzewo przeszukiwania w głąb (ang. *deep first search*, DFS)
3. Drzewo przeszukiwania wszerz (ang. *breadth-first search*, BFS)
4. Minimalne drzewo spinające (ang. *minimum spanning tree*)
   - Algorytm Kruskala (Kruskal 1956)
   - Algorytm Prima (Jarník 1930, Prim 1957, Dijkstra 1959)
   - ...
5. ...

ALGORYTM ZACHŁANNY

Niech $G = (V, E)$ będzie spójnym (multi)grafem.

1. Dopóki (multi)graf nie jest drzewem, usuń dowolną krawędź dowolnego cyklu.





Przykład

- Rozważamy cykl o $(1, 2, 5, 3)$ i usuwamy np. krawędź $\{1, 2\}$.
- Rozważamy cykl $(2, 3, 5)$ i usuwamy np. krawędź $\{3, 5\}$.
- W otrzymanym grafie nie ma już cykli.

Otrzymane drzewo spinające $T = (V, E')$:

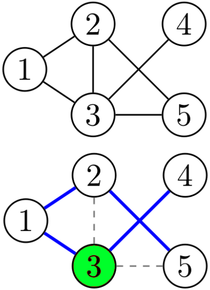
$$V = \{1, 2, 3, 4, 5\} \text{ oraz } E' = \{\{1, 3\}, \{2, 3\}, \{2, 5\}, \{3, 4\}\}.$$

ALGORYTM DFS

Niech $G = (V, E)$ będzie danym grafem spójnym, a $v \in V$ wierzchołkiem początkowym.

1. Odwiedzamy wierzchołek v (zaznaczamy go jako odwiedzony) i wkładamy go na STOS.
2. Dopóki STOS nie jest pusty, powtarzamy:
   - Jeżeli v jest wierzchołkiem na wierzchu STOSU, to sprawdzamy, czy istnieje wierzchołek sąsiedni z v, który nie był jeszcze odwiedzony.
   - Jeżeli u jest takim wierzchołkiem, to odwiedzamy u (zaznaczamy jako odwiedzony) i wkładamy go na STOS.
   - Jeżeli takiego u nie ma, to zdejmujemy v ze STOSU.

Przykład



v	STOS	zbiór krawędzi drzewa DFS
$\underline{3}$	3	$\emptyset$
$\underline{1}$	3,1	$\{\{1,3\}\}$
$\underline{2}$	3,1,2	$\{\{1,3\},\{1,2\}\}$
$\underline{5}$	3,1,2,5	$\{\{1,3\},\{1,2\},\{2,5\}\}$
2	3,1,2	$\{\{1,3\},\{1,2\},\{2,5\}\}$
1	3,1	$\{\{1,3\},\{1,2\},\{2,5\}\}$
3	3	$\{\{1,3\},\{1,2\},\{2,5\}\}$
$\underline{4}$	1,4	$\{\{1,3\},\{1,2\},\{2,5\},\{1,4\}\}$
3	3	$\{\{1,3\},\{1,2\},\{2,5\},\{1,4\}\}$
–	$\emptyset$	$\{\{1,3\},\{1,2\},\{2,5\},\{1,4\}\}$

Kolejność DFS wierzchołków: $3, 1, 2, 5, 4$.

Otrzymane drzewo spinające DFS $T = (V, E')$:

$$V = \{1, 2, 3, 4, 5\} \text{ oraz } E' = \{\{1, 3\}, \{1, 2\}, \{2, 5\}, \{1, 4\}\}.$$

Uwagi do algorytmu DFS

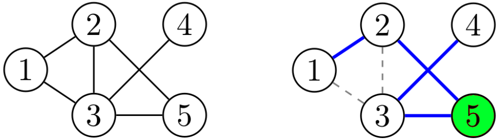
1. Jeśli jest kilka wierzchołków do wyboru, to wybieramy zgodnie z ustalonym porządkiem.
2. Wierzchołki na STOSIE w dowolnym kroku tworzą ścieżkę od korzenia do wierzchołka aktualnie odwiedzanego.
3. Jeśli w powyższej procedurze w kroku 2, w którym odwiedzamy wierzchołek u, do początkowo pustego zbioru E' krawędzi dodawać będziemy krawędź $\{v, u\}$, to otrzymamy drzewo spinające DFS.
4. Dowolne dwa wierzchołki w drzewie DFS, jeśli są sąsiednie w grafie, to w drzewie znajdują się na jednej ścieżce do korzenia (wierzchołka startowego) — mamy zatem porządek częściowy.

ALGORYTM BFS

Niech $G = (V, E)$ będzie danym grafem spójnym, a $v \in V$ wierzchołkiem początkowym.

1. Odwiedzamy wierzchołek v (zaznaczamy go jako odwiedzony) i wstawiamy go do KOLEJKI.
2. Dopóki KOLEJKA nie jest pusta, powtarzamy:
    - Bierzemy wierzchołek v z początku KOLEJKI.
    - Odwiedzamy wszystkie do tej pory jeszcze nie odwiedzone wierzchołki sąsiednie z v (zaznaczamy je jako odwiedzone) i wstawiamy je na koniec KOLEJKI.

Przykład



v	odwiedzane	KOLEJKA	zbiór krawędzi drzewa DFS
5	5	5	$\emptyset$
5	2,3	2,3	$\{\{2,5\},\{3,5\}\}$
2	1	3,1	$\{\{2,5\},\{3,5\},\{1,2\}\}$
3	4	1,4	$\{\{2,5\},\{3,5\},\{1,2\},\{3,4\}\}$
1	–	4	$\{\{2,5\},\{3,5\},\{1,2\},\{3,4\}\}$
4	–	$\emptyset$	$\{\{2,5\},\{3,5\},\{1,2\},\{3,4\}\}$

Kolejność BFS wierzchołków: $5, 2, 3, 1, 4$.

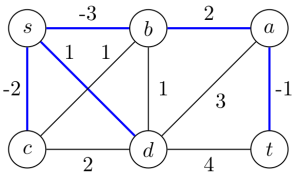
Otrzymane drzewo spinające BFS $T = (V, E')$:

$$V = \{1,2,3,4,5\} \text{ oraz } E' = \{\{2,5\},\{3,5\},\{1,2\},\{3,4\}\}.$$

Uwagi do algorytmu BFS

1. Wierzchołki wstawiamy do KOLEJKI np. w kolejności uporządkowania etykiet.
2. Wierzchołki przeszukiwane są w kolejności leżących najbliżej korzenia.
3. Jeśli w powyższej procedurze w kroku 2, w którym odwiedzamy wszystkie nieodwiedzone jeszcze wierzchołki sąsiednie do v, do początkowo pustego zbioru E' krawędzi dodawać będziemy odpowiednie krawędzie $\{v, u\}$, to otrzymamy drzewo spinające BFS.
4. Dowolne dwa wierzchołki w drzewie BFS, jeśli są sąsiednie w grafie, to w drzewie ich odległości od korzenia (wierzchołka startowego) różnią się o co najwyżej 1.

Niech $G = (V, E, w)$ będzie spójnym grafem ważonym, tzn. każdej krawędzi $e \in E$ przyporządkowana jest pewna waga $w(e) \in \mathbb{R}$. Problem minimalnego drzewa spinającego (*drzewo MST*) definiujemy jako problem wyznaczenia drzewa spinającego $T = (V, E')$ w grafie G o najmniejszej sumie ważonej $\sum_{e \in E'} w(e)$.



- Zastosowanie np. przy wyznaczeniu „najtańszej" sieci dróg, torów kolejowych, itp., która łączy/spina dane miasta.

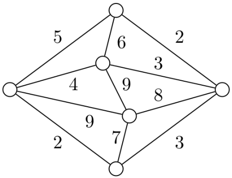
ALGORYTM KRUSKALA (1956)

Niech $G = (V, E, w)$ będzie spójnym grafem ważonym z funkcją wagową $w\colon E \to \mathbb{R}$.

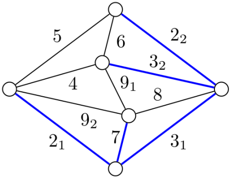
1. $T := (V, E')$, gdzie $E' := \emptyset$.
2. Posortuj krawędzie grafu G w kolejności niemalejących wag.
3. Dla każdej krawędzi $e \in E$:
   - Jeśli dodanie rozważanej krawędzi e nie utworzy cyklu w T, wówczas $E' := E' \cup \{e\}$.

Przykład

- Dla ułatwienia utożsamiamy wagi krawędzi z samymi krawędziami.
- Posortowany ciąg krawędzi: $2_1, 2_2, 3_1, 3_2, 4, 5, 6, 7, 8, 9_1, 9_2$.



krawędź	cykl?	MST
2_1	–	2_1
2_2	–	$2_1, 2_2$
3_1	–	$2_1, 2_2, 3_1$
3_2	–	$2_1, 2_2, 3_1, 3_2$
4	$\surd$	$2_1, 2_2, 3_1, 3_2$
5	$\surd$	$2_1, 2_2, 3_1, 3_2$
6	$\surd$	$2_1, 2_2, 3_1, 3_2$
7	–	$2_1, 2_2, 3_1, 3_2, 7$
8	$\surd$	$2_1, 2_2, 3_1, 3_2, 7$
9_1	$\surd$	$2_1, 2_2, 3_1, 3_2, 7$
9_2	$\surd$	$2_1, 2_2, 3_1, 3_2, 7$



- Powyższy graf ma 6 wierzchołków, a z definicji każde jego drzewo spinające ma 5 krawędzi, wykonywanie algorytmu można było już przerwać po dodaniu 5-tej krawędzi o wadze 7.

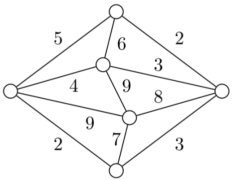
ALGORYTM PRIMA (1957)

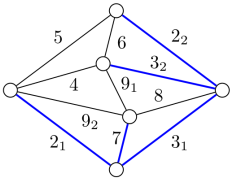
Niech $G = (V, E, w)$ będzie spójnym grafem ważonym z funkcją wagową $w\colon E \to \mathbb{R}$.

1. $T := (V, E')$, gdzie $E' := \emptyset$.
2. Wybierz krawędź $e_1 \in E$ o najmniejszej wadze $w(e_1)$.
3. Jeśli $e_1, e_2, e_3, \ldots, e_i$ są krawędziami do tej pory wybranymi, to wybierz $e_{i+1} \in E \setminus \{e_1, e_2, \ldots, e_i\}$ taką, że:
    - $|V(G[\{e_1, \ldots, e_i\}]) \cap e_{i+1}| = 1$;
    - waga $w(e_{i+1})$ jest najmniejsza z możliwych.
4. Zatrzymaj się, jeśli krok 2 jest niemożliwy.

Przykład

» Dla ułatwienia utożsamiamy wagi krawędzi z samymi krawędziami.





krawędź e_i	MST
2_1	2_1
3_1	$2_1, 3_1$
2_2	$2_1, 3_1, 2_2$
3_2	$2_1, 3_1, 2_2, 3_2$
7	$2_1, 3_1, 2_2, 3_2, 7$

Uwagi o algorytmach i zastosowania

- Minimalne drzewa rozpinające zwracane przez algorytm Kruskala i Prima nie muszą być takie same.
  - » Kiedy i dlaczego są takie same?
- **Algorytm Kruskala**
  Złożoność czasowa rzędu $O(|E| \log |V|)$
  - » w oparciu o struktury zbiorów rozłącznych.
- **Algorytm Prima**
  Złożoność czasowa rzędu $O(|E| + |V| \log |V|)$
  - » w oparciu o kopce Fibonacci'ego i listę sąsiedztwa.
- **Problem Komiwojażera** /problem NP-trudny/
  3/2-przybliżony Algorytm Christofidesa (1976) dla grafów ważonych spełniających nierówność trójkąta.

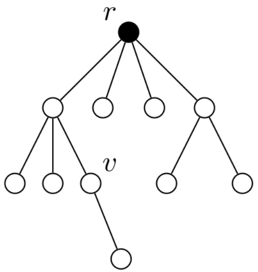
Drzewa ukorzenione

Definicje

Drzewo ukorzenione posiada wierzchołek wyróżniony zwany korzeniem. Dowolny wierzchołek może mieć dziecko/syna (relacja ojciec-syn), ale – za wyjątkiem korzenia – każdy wierzchołek jest synem dokładnie jednego innego wierzchołka. Wierzchołki nie posiadające synów zwane są liśćmi. Wysokość/głębokość drzewa to długość najdłuższej ścieżki od korzenia do liścia.

Przykład

Drzewo o korzeniu r posiadającym czterech synów. Wierzchołek v posiada tylko jednego syna będącego liściem. Głębokość drzewa wynosi 3.

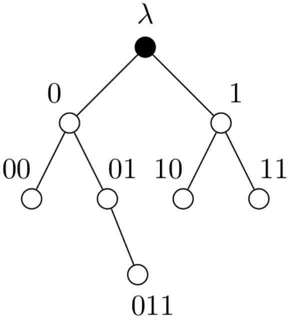


Definicja

Ukorzenione drzewo, w którym każdy wierzchołek ma co najwyżej dwóch synów, nazywamy drzewem binarnym.

W drzewie binarnym wierzchołki można etykietować ciągami złożonymi z 0 i 1. Wówczas korzeń drzewa oznaczony jest przez λ, a jeśli jakiś wierzchołek oznaczony jest przez x, to jego lewego syna etykietujemy $x0$, a prawego $x1$. Przy takim etykietowaniu wierzchołków kolejne bity wierzchołka wyznaczają ścieżkę od korzenia do tegoż wierzchołka:

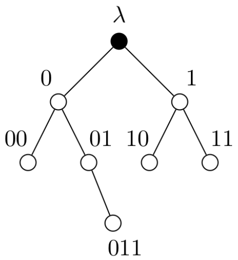
- 0 – wchodzimy do lewego syna,
- 1 – wchodzimy do prawego syna.



Przeszukiwanie drzewa w kolejności postorder

Aby przeszukać (pod)drzewo mające swój korzeń w wierzchołku x:

1. Przeszukujemy jego lewe poddrzewo (z korzeniem w $x0$).
2. Przeszukujemy jego prawe poddrzewo (z korzeniem w $x1$).
3. Odwiedzamy wierzchołek x (korzeń drzewa).

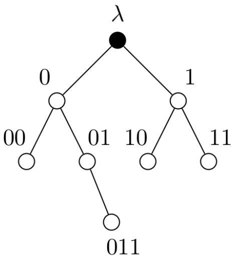


Kolejność postorder: $00, 011, 01, 0, 10, 11, 1, \lambda$.

Przeszukiwanie drzewa w kolejności inorder

Aby przeszukać (pod)drzewo mające swój korzeń w wierzchołku x:

1. Przeszukujemy jego lewe poddrzewo (z korzeniem w $x0$).
2. Odwiedzamy wierzchołek x (korzeń drzewa).
3. Przeszukujemy jego prawe poddrzewo (z korzeniem w $x1$).

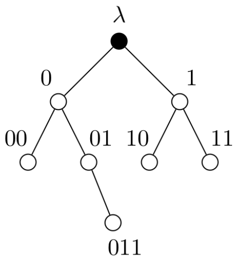


Kolejnośś inorder: $00, 0, 01, 011, \lambda, 10, 1, 11$.

Przeszukiwanie drzewa w kolejności preorder

Aby przeszukać (pod)drzewo mające swój korzeń w wierzchołku x:

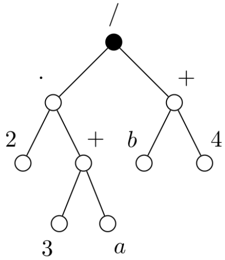
1. Odwiedzamy wierzchołek x (korzeń drzewa).
2. Przeszukujemy jego lewe poddrzewo (z korzeniem w $x0$).
3. Przeszukujemy jego prawe poddrzewo (z korzeniem w $x1$).



Kolejność preorder: $\lambda, 0, 00, 01, 011, 1, 10, 11.$

Zastosowanie drzew binarnych

1. Drzewiaste struktury danych (Algorytmy i Struktury Danych)
2. Drzewa wyrażeń arytmetycznych



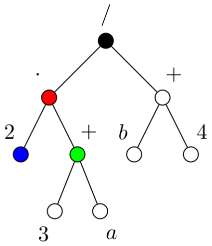
3. ...

Definicja

Drzewo wyrażeń arytmetycznych jest to drzewo binarne, w którym każdy wierzchołek ma albo dwóch synów albo wcale. W takim drzewie liście etykietowane są stałymi albo zmiennymi. Pozostałe wierzchołki etykietowane są operacjami arytmetycznymi.

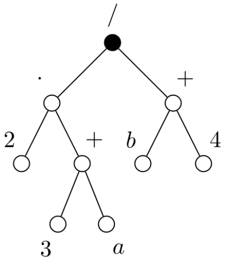
Każdemu wierzchołkowi x w drzewie możemy przypisać wyrażenie arytmetyczne $W(x)$ według zasady:

- dla liści wyrażeniami są etykiety tych liści (stałe lub zmienne);
- jeżeli x ma etykietę op, a jego synom $x0$ i $x1$ przypisano odpowiednio wyrażenia $W(x0)$ i $W(x1)$, to W(x)=W(x0) op W(x1).



$$
\begin{aligned}
W(x) &= W(x0) \cdot W(x1) \\
&= 2 \cdot (3 + a)
\end{aligned}
$$

Postacie wyrażeń arytmetycznych



(postać pre- jak i postfixowa nie wymagają nawiasowania)

- Notacja **infixowa**: $((2 \cdot (a + 3))/(b + 4)$;
- Notacja **prefixowa**: $/ \ \cdot \ 2 \ + \ a \ 3 \ + \ b \ 4$;
- Notacja **postfixowa**: $2 \ a \ 3 \ + \ \cdot \ b \ 4 \ + \ /$.

Mając drzewo wyrażenia arytmetycznego, aby otrzymać postać postfixową/infixową/prefixową tego wyrażenia, należy przeszukać to drzewo odpowiednio metodą postorder/inorder/preorder i wypisać po kolei etykiety odwiedzanych wierzchołków.

- W celu otrzymania postaci infixowej przy przeszukiwaniu inorder za każdym pójściem w lewo wstawiamy nawias otwierający '(', a przy powrocie z prawej i wyjściu z wierzchołka – nawias zamykający ')'.

Algorytm obliczania wartości wyrażenia w postaci postfixowej

Dla kolejnych elementów zapisu wyrażania powtarzamy:

1. Jeżeli element jest stałą albo zmienną, to wkładamy jego wartość (po podstawieniu) na stos.
2. Jeżeli element jest znakiem operacji, to zdejmujemy dwie wartości ze stosu, wykonujemy operację na tych wartościach, a następnie obliczoną wartość wkładamy na wierzch stosu.
3. Po przejściu całego wyrażenia wartość znajduje się na stosie.

Przykład

Wartość wyrażenia „2 a 3 $+$ $\cdot$ b 4 $+$ $/$" dla $a = 2$ i $b = 1$.

			3				4		
		2	2	5		1	1	5	
stos	2	2	2	2	10	10	10	10	**2**
element	2	$a\ (=2)$	3	$+$	$\cdot$	$b\ (=1)$	4	$+$	$/$

W oparciu o ten algorytm można też zbudować drzewo wyr. arytm.