W tym rozdziale zajmiemy się drzewami: specjalnym przypadkiem grafów. Są one szczególnie przydatne do przechowywania informacji, umożliwiającego szybki dostęp do nich.

Definicja 1. Las *to graf prosty, acykliczny.*

Definicja 2. Drzewo *to graf prosty, spójny, acykliczny (czyli spójny las). Wierzchołki drzewa nazywamy* węzłami. *Podgraf spójny drzewa nazywamy poddrzewem.*

Twierdzenie 1. *Dla grafu $T = (V, E)$ następujące warunki są równoważne:*
1. T jest drzewem.
2. T nie zawiera cykli i ma $|V| - 1$ krawędzi.
3. T jest spójny i ma $|V| - 1$ krawędzi.
4. T jest spójny i każda jego krawędź jest mostem.
5. Dowolne dwa wierzchołki grafu są połączone dokładnie jedną drogą prostą.
6. T nie zawiera cykli, lecz dodanie dowolnej nowej krawędzi tworzy dokładnie jeden cykl.

Definicja 3. Drzewem spinającym (rozpinającym) *grafu G nazywamy podgraf G zawierający wszystkie jego wierzchołki i będący drzewem.*

Łatwo zauważyć, że drzewo spinające jest minimalnym (ze względu na inkluzję krawędzi) podgrafem spójnym łączącym wszystkie wierzchołki grafu G. Dlatego wyznaczanie drzew spinających jest przydatne, gdy wierzchołki grafu symbolizują cele, jakie chcemy osiągnąć.
Przykłady

Twierdzenie 2. *Każdy skończony graf ma drzewo spinające wtedy i tylko wtedy, gdy jest spójny.*

Algorytm 1. *SPINANIE(v):*
***Dane**: v - wierzchołek skończonego grafu spójnego G.*
***Zmienne**: E - zbiór krawędzi, V - zbiór wierzchołków.*
I. Niech $V := \{v\}$, $E := \emptyset$.
II. Dopóki istnieją w grafie G krawędzie łączące wierzchołki ze zbioru V z wierzchołkami, które nie należą do V: wybierz krawędź uw łączącą wierzchołek $u \in V$ z wierzchołkiem $w \notin V$, dołącz w do V i dołącz uw do E.
***Rezultat**: E to zbiór krawędzi drzewa spinającego grafu G.*

Powyższego algorytmu można użyć również jako testu spójności grafu G. Graf jest spójny wtedy i tylko wtedy, gdy po zakończeniu działania algorytmu SPINANIE(v), $V = V(G)$. Czas działania tego algorytmu to $O(|V(G)| + |E(G)|)$.
Przykład (dla sieci) Minimalna konieczna infrastruktura, broadcasting.
Często potrzebne jest znajdowanie w grafie z wagami tzw. minimalnego drzewa spinającego tj. takiego, że jego waga (suma wag jego krawędzi) jest nie większa niż waga jakiegokolwiek innego drzewa spinającego.
Na szczęście, działają tutaj dwa dość proste algorytmy zachłanne: algorytm Kruskala i algorytm Prima.

Algorytm 2. *KRUSKAL:*
***Dane**: G - skończony graf spójny z wagami, którego krawędzie są uporządkowane wg wzrastających wag $e_1, \ldots, e_n$, $n = |E(G)|$.*
***Zmienne**: E - zbiór krawędzi.*
I. Podstaw $E := \emptyset$.
II. Dla $j = 1$ do n wykonuj: jeśli graf $E \cup \{e_j\}$ jest acykliczny, dołącz e_j do E.
***Rezultat**: E to zbiór krawędzi minimalnego drzewa spinającego G.*

Wadą tego algorytmu jest to, że przed jego zastosowaniem musimy jakoś uporządkować krawędzie według wzrastających wag. Jednakże, nawet razem z tym uporządkowaniem, algorytm Kruskala zajmuje tylko $O(|E(G)| \cdot \log |E(G)|)$ czasu. Problemem może być jedynie sytuacja, gdy na początku tego uporządkowania nie da się uzyskać (np. dane o wagach uzyskujemy w miarę tworzenia się naszego drzewa).
Warto zauważyć, że działa on poprawnie, nawet jeśli graf G ma pętle i krawędzie wielokrotne. Nie trzeba nawet zakładać spójności - jednak wtedy znajdziemy nie drzewo, a las spinający.
Drugi z algorytmów nie wymaga uprzedniego posortowania krawędzi według wag i można go użyć w sytuacjach, gdy algorytm Kruskala nie działa:

Algorytm 3. *PRIM:*
***Dane**: G - skończony graf spójny z wagami.*
***Zmienne**: E - zbiór krawędzi, V - zbiór wierzchołków.*
I. Podstaw $E := \emptyset$.
II. Wybierz w - dowolny wierzchołek ze zbioru $V(G)$ i niech $V := \{w\}$.
III. Dopóki $V \neq V(G)$ wykonuj: wybierz w zbiorze $E(G)$ krawędź uv o najmniejszej możliwej wadze, taką, że $u \in V$ i $v \in V(G) \setminus V$, a następnie dołącz krawędź uv do zbioru E i wierzchołek v do zbioru V.
***Rezultat**: E to zbiór krawędzi minimalnego drzewa spinającego G.*

Algorytm Prima da się zaimplementować w czasie $O(n^2)$, gdy graf ma n wierzchołków. Zatem, dla grafów o małej liczbie krawędzi szybszy może być algorytm Kruskala, ale jeśli krawędzi jest wiele, szybszy będzie algorytm Prima (dlaczego?). Pętle i krawędzie wielokrotne nie psują algorytmu Prima, jednak jeśli graf jest niespójny i chcemy uzyskać spinający las, to trzeba ten algorytm nieco zmodyfikować (ćwiczenie).

Definicja 4. Drzewo z wyróżnionym korzeniem *jest to drzewo z wyróżnionym jednym wierzchołkiem, nazywanym* korzeniem.

Z niewiadomych przyczyn, drzewa z wyróżnionym korzeniem rysuje się tak, by korzeń był na samej górze. Czasem interpretuje się je jako graf skierowany, wtedy strzałki idą w dół.

Definicja 5. Liść *drzewa to wierzchołek stopnia* 1*, który nie jest wyróżniony jako korzeń.*

Definicja 6. *Jeśli para (v, w) jest krawędzią drzewa z wyróżnionym korzeniem, to v jest* rodzicem *w, a w jest* dzieckiem *v. Ogólniej, w jest* potomkiem *v, jeśli $w \neq v$ i v jest wierzchołkiem jedynej drogi prostej z korzenia do wierzchołka w. v nazywamy wtedy* przodkiem *w.*

Definicja 7. Poziomem *wierzchołka v nazywamy długość jedynej drogi prostej od korzenia do v.* Wysokość drzewa *z wyróżnionym korzeniem to największy możliwy poziom wierzchołka tego drzewa.*

Definicja 8. Poddrzewo o korzeniu *v jest to drzewo T_v, składające się z korzenia v, wszystkich jego potomków i wszystkich krawędzi (potencjalnie skierowanych) łączących ich.*

Drzewa (zwłaszcza z korzeniem) są szczególnie często stosowanymi w informatyce grafami, tworząc np. wygodne struktury baz danych.
Przykłady Struktura uczelni, stuktura katalogów, drzewo genealogiczne, „drzewkowy" zapis algorytmu postępowania.

Definicja 9. Drzewo binarne *jest to drzewo z wyróżnionym korzeniem, w którym każdy węzeł ma co najwyżej dwoje dzieci, wśród których wyróżniamy dzieci prawe i lewe.*

Przykład Łatwa do przeszukiwania alfabetyczna baza danych.

Definicja 10. Drzewo poszukiwań binarnych *(przykład drzewa z wyróżnionym korzeniem) jest to drzewo binarne składające się z węzłów z etykietami (węzłów indeksowanych zbiorem uporządkowanym), w którym lewe poddrzewo każdego węzła zawiera wyłącznie węzły o etykietach nie większych niż etykieta węzła a prawe poddrzewo zawiera wyłącznie węzły o etykietach nie mniejszych niż etykieta węzła. W ten sposób tworzy się bazy danych, których etykiety są np. liczbami lub słowami (uporządkowanymi alfabetycznie), i które bardzo szybko można przeszukać i znaleźć dany węzeł.*

Algorytm przeszukiwania drzewa poszukiwań binarnych jest przykładem algorytmu rekurencyjnego:

Algorytm 4. *PRZESZUKIWANIE(v,x):*
***Dane**: Drzewo poszukiwań binarnych z korzeniem v, x - poszukiwana wartość etykiety.*
***Zmienne**: w - węzeł.*
I. Podstawiamy $w = v$.
II. Porównujemy etykietę w i x. Jeśli etykiety są równe: STOP.
III. Jeśli etykieta v jest większa od x, to w := lewe dziecko v i przechodzimy do kroku V.
IV. Jeśli etykieta v jest mniejsza od x, to w := prawe dziecko v i przechodzimy do kroku V.
V. Wykonujemy PRZESZUKIWANIE(w,x).
***Rezultat**: w to węzeł o etykiecie x.*

W zależności od tego, jak dobrze jest to drzewo skonstruowane (wyważone) czas działania tego algorytmu wynosi od $O(\log n)$ do $O(n)$.
Ćwiczenie Wypisać algorytmy:
1) Znajdowania największego/najmniejszego elementu w drzewie poszukiwań binarnych.
2) Znajdowania poprzednika/następnika danego elementu w drzewie poszukiwań binarnych.
3) Wstawiania nowego elementu/usuwania jednego z elementów, by nie zaburzyć struktury tego drzewa.
Uogólnienie drzewa poszukiwań binarnych:

Definicja 11. *Jeśli dla każdego węzła drzewa z wyróżnionym korzeniem jego dzieci są uporządkowane tj. wiemy które z nich występuje wcześniej, a które później w jakimś ustalonym porządku, to mówimy, że drzewo jest* uporządkowane*. Na rysunku, tak uporządkowane dzieci rysujemy od strony lewej do prawej. Kolejność dzieci u i v zapisujemy $u < v$, nawet jeśli nie są one liczbami.*

Przykład Porządek alfabetyczny.
Przykład Problem: zapamiętywanie drzewa przez komputer.
Algorytm przechodzenia drzewa to taki, który ustawia w ciąg wszystkie wierzchołki skończonego uporządkowanego drzewa z wyróżnionym korzeniem. Trzy najbardziej popularne takie algorytmy to PREORDER, POSTORDER i INORDER (ten trzeci działa tylko dla drzew binarnych).

Definicja 12. Porządek prefiksowy *drzewa z wyróżnionym korzeniem to takie uporządkowanie wierzchołków, w którym korzeń drzewa umieszczamy na początku listy, a dalej znajdują się poddrzewa uporządkowane w kolejności swoich korzeni (rysunkowo: kolejność od lewej do prawej). Innymi słowy, dzieci w tym porządku pojawiają się po rodzicach. Tworzy się go algorytmem PREORDER.*

Przykład

Algorytm 5. *PREORDER(v):*
***Dane**: skończone, uporządkowane drzewo z wyróżnionym korzeniem v.*
***Zmienne**: L(v) - lista wierzchołków drzewa, w-węzeł.*
I. Umieść v na liście L(v).

II. Dla każdego w - dziecka v (idąc od lewej do prawej) wykonaj: PREORDER(w) {otrzymujemy w ten sposób listę $L(w)$ złożoną z wierzchołka w i jego potomków}, a następnie dołącz otrzymaną listę $L(w)$ na końcu listy $L(v)$.
***Rezultat**: $L(v)$ - lista wierzchołków drzewa, na której rodzice znajdują się zawsze przed swoimi dziećmi (czyli w porządku prefiksowym).*

Definicja 13. Porządek postfiksowy *drzewa z wyróżnionym korzeniem to takie uporządkowanie wierzchołków, w którym poddrzewa są umieszczone najpierw (rysunkowo: kolejność od lewej do prawej), a potem dopiero korzeń. Innymi słowy, dzieci w tym porządku pojawiają się przed rodzicami. Tworzy się go algorytmem POSTORDER.*

Przykład

Algorytm 6. *POSTORDER(v):*
***Dane**: skończone, uporządkowane drzewo z wyróżnionym korzeniem v.*
***Zmienne**: $L(v)$ - lista wierzchołków drzewa, w-węzeł.*
I. Niech $L(v) = \emptyset$ (pusta lista).
II. Dla każdego w - dziecka v (idąc od lewej do prawej) wykonaj: POSTORDER(w) {otrzymujemy w ten sposób listę $L(w)$ złożoną z wierzchołka w i jego potomków}, a następnie dołącz otrzymaną listę $L(w)$ na końcu listy $L(v)$.
III. Dołącz wierzchołek v na końcu listy $L(v)$.
***Rezultat**: $L(v)$ - lista wierzchołków drzewa, na której rodzice znajdują się zawsze po swoich dzieciach (czyli w porządku postfiksowym).*

Trzeci sposób przechodzenia drzewa dotyczy tylko uporządkowanych drzew binarnych z wyróżnionym korzeniem.

Definicja 14. Porządek infiksowy *drzewa binarnego z wyróżnionym korzeniem to takie uporządkowanie wierzchołków, w którym dany korzeń jest umieszczony pomiędzy wierzchołkami poddrzewa, którego korzeniem jest jego lewe dziecko, a wierzchołkami poddrzewa, którego korzeniem jest jego prawe dziecko. Innymi słowy, lewe dzieci w tym porządku pojawiają się przed rodzicami, a prawe po nich. Tworzy się go algorytmem INORDER.*

Algorytm 7. *INORDER(v):*
***Dane**: skończone, uporządkowane drzewo binarne z wyróżnionym korzeniem v.*
***Zmienne**: $L(v)$ - lista wierzchołków drzewa, u, w-węzły.*
I. Niech $L(v) = \emptyset$ (pusta lista).
II. Jeśli wierzchołek v ma lewe dziecko w, wykonaj: INORDER(w) {otrzymujemy w ten sposób listę $L(w)$ złożoną z wierzchołka w i jego potomków}, a następnie dołącz otrzymaną listę $L(w)$ na końcu listy $L(v)$.
III. Dołącz v na koniec listy $L(v)$.
IV. Jeśli wierzchołek v ma prawe dziecko u, wykonaj: INORDER(u) {otrzymujemy w ten sposób listę $L(u)$ złożoną z wierzchołka u i jego potomków}, a następnie dołącz otrzymaną listę $L(u)$ na końcu listy $L(v)$
***Rezultat**: $L(v)$ - lista wierzchołków drzewa, na której lewe dzieci znajdują się przed rodzicami, a prawe - za nimi (czyli w porządku infiksowym).*

Warto zwrócić uwagę, że algorytm działa również jeśli niektórzy rodzice mają tylko jedno dziecko.
Czas działania wszystkich trzech algorytmów to $O(|V|)$, gdzie V jest zbiorem wierzchołków drzewa.
Wiemy już, jak można zakodować drzewo z wyróżnionym korzeniem. Jednak, czy z odkodowaniem pójdzie tak łatwo? To znaczy, czy możemy odtworzyć drzewo mając daną tylko kolejność wierzchołków? Na ogół odpowiedź jest przecząca.
Przykłady.
Jeśli jednak tak jest, jaki sens ma kodowanie drzewa? Otóż, w pewnych istotnych przypadkach, da się drzewo odtworzyć.

Twierdzenie 3. *Niech T będzie skończonym, uporządkowanym drzewem z wyróżnionym korzeniem, którego wierzchołki są wypisane w porządku prefiksowym lub postfiksowym. Przypuśćmy, że znana jest liczba dzieci każdego wierzchołka. Wtedy to drzewo jest wyznaczone jednoznacznie i może być odtworzone z ciągu wierzchołków.*

Niestety, nie da się uzyskać takiego wyniku dla porządku infiksowego (przykład), dlatego ten jest mniej używany w komputerowych strukturach danych.
Poniższe algorytmy odkodowują drzewa zadane w porządku prefiksowym i postfiksowym:

Algorytm 8. *DEPRE($v_1, \dots v_n$, $c_1, \dots c_n$):*
***Dane**: $v_1, \dots v_n$ - ciąg wierzchołków drzewa T w porządku prefiksowym, $c_1, \dots c_n$ - ciąg liczb dzieci tych wierzchołków.*
***Zmienne**: i, j - liczniki pętli wierzchołków. $S(v_i)$ - zbiory wierzchołków , $U(v_i)$ - liczby całkowite.*
I. Dla każdego i podstawiamy $U(v_i) := c_i$, $S(v_i) = \emptyset$. Następnie $i := 1$.
II. Dopóki $i \leq n$ wykonujemy:
Znajdujemy maksymalne możliwe $j < i$, takie, że $U(v_j) \neq 0$. Wtedy $U(v_j) := U(v_j) - 1$, $S(v_j) := S(v_j) \cup \{v_i\}$, $i := i + 1$.
***Rezultat**: $S(v_i)$ - ciąg zbiorów, zawierających wierzchołki będące dziećmi kolejnych węzłów v_i.*

Algorytm 9. *DEPOST($v_1, \dots v_n$, $c_1, \dots c_n$):*
***Dane**: $v_1, \dots v_n$ - ciąg wierzchołków drzewa T w porządku postfiksowym, $c_1, \dots c_n$ - ciąg liczb dzieci tych wierzchołków.*
***Zmienne**: i - licznik pętli wierzchołków. $S(v_i)$ - zbiory wierzchołków , $U(v_i)$- ciągi wierzchołków.*
Po kolei, dla każdego $i = 1, 2, \dots, n$ definiujemy:
a) Jeśli $c_i = 0$, to $S(v_i) = \emptyset$, a $U(v_i)$ jest dowolne.
b) Ib. Jeśli $c_i \neq 0$ to $U(v_i) = \{v_k : k < i\} \setminus \bigcup_{k<i} S(v_i)$ (uporządkowany w kolejności danych wejściowych). Wtedy definiujemy $S(v_i)$ jako c_i ostatnich elementów ciągu $U(v_i)$.
***Rezultat**: $S(v_i)$ - ciąg zbiorów, zawierających wierzchołki będące dziećmi kolejnych węzłów v_i. Z tego już łatwo można odtworzyć drzewo*

Przykłady