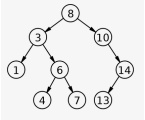

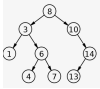
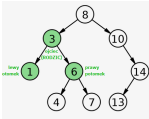
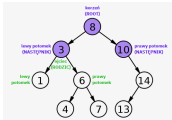
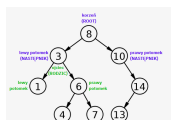
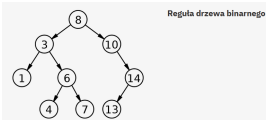



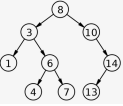
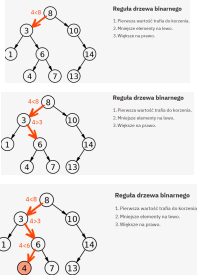
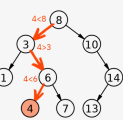
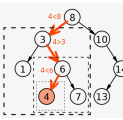


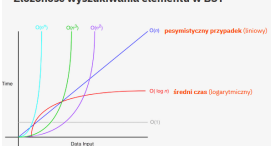



# Binarne drzewo poszukiwań

## 1. Czym jest BST?

	Binarne drzewo poszukiwań (ang. <i>binary search tree</i> - BST) to szczególny rodzaj drzewa binarnego, czyli jednej z fundamentalnych struktur danych.
	Elementy drzewa nazywa się <b>węzłami</b> lub wierzchołkami - i to w nich przechowywane są dane.
	
	Węzeł nadrzędny w drzewie to tzw. <b>rodzic</b> (ojciec). Węzły potomne danego węzła to tzw. <b>potomki</b> (bądź następniki, ang. <i>child node</i> ).
	Każde drzewo binarne ma dokładnie jeden <b>korzeń</b> ( <i>root node</i> ), czyli węzeł nieposiadający rodzica.
	Wszystkie węzły to różne komórki w pamięci połączone wskaźnikami (tak jak pokazują strzałki na rysunku). Strzałki te oznaczają, że węzeł wskazywany (potomek) jest pamiętany przez węzeł, z którego wychodzi strzałka (ojciec).
	Wszystkie węzły drzewa spełniają tzw. regułę drzewa binarnego:
	1. przy każdorazowym dodawaniu do drzewa kolejnego elementu pierwsza wartość trafia do <b>korzenia</b> ,
	2. Później mniejsze elementy trafiają do lewego poddrzewa,
	3. Większe elementy do prawego.

 <p><b>Reguła drzewa binarnego</b></p> <ol style="list-style-type: none"> <li>1. Pierwsza wartość trafia do korzenia.</li> <li>2. Mniejsze elementy na lewo.</li> <li>3. Większe na prawo.</li> </ol>	<p>Reguła ta obowiązuje wszystkie poddrzewa. Śledząc dla przykładu jak wstawiana jest wartość 4:</p>
 <p><b>Reguła drzewa binarnego</b></p> <ol style="list-style-type: none"> <li>1. Pierwsza wartość trafia do korzenia.</li> <li>2. Mniejsze elementy na lewo.</li> <li>3. Większe na prawo.</li> </ol>	<ol style="list-style-type: none"> <li>1. najpierw sprawdzamy czy jest mniejsza czy większa od korzenia? Jest mniejsza więc trafia zgodnie z regułą drzewa binarnego do lewego poddrzewa.</li> <li>2. Następnie porównujemy ją z 3. <math>4 &gt; 3</math> a więc trafia tym razem do prawego poddrzewa.</li> <li>3. Po porównaniu z 6 wychodzi, że <math>4 &lt; 6</math> czyli łąduje w lewym poddrzewie.</li> </ol> <p>W analogiczny sposób wstawiana jest każdy z elementów. <b>Po co nam takie ułożenie danych?</b></p>
 <p><b>Reguła drzewa binarnego</b></p> <p>“Dziel i zwyciężaj”</p>	<p>Dzięki przestrzeganiu reguły drzewa binarnego, można użyć metody “dziel i zwyciężaj”,</p>
 <p><b>Reguła drzewa binarnego</b></p> <p>“Dziel i zwyciężaj”</p>	<p>w której przy poszukiwaniu danego elementu, w ciągu 1 iteracji odrzucamy połowę możliwych węzłów- właśnie dzięki takiemu segregowaniu elementów przy wstawianiu.</p>
	<p>Aby lepiej zobrazować regułę drzewa binarnego przygotowałam krótki program konsolowy.</p> <p>Mamy tu 2 funkcje: dodawania i znajdowania elementu.</p> <ol style="list-style-type: none"> <li>1. <b>Wypełniam</b> program wartościami: 123, 55, 77, 77 - założenie w kodzie jest takie, że większy bądź równy element trafia do prawej gałęzi dlatego 77 trafiła do prawego poddrzewa.</li> <li>2. <b>Wyszukuję</b> pierwsze wystąpienie elementu 77 - znaleziono w węźle numer 5 - i to się zgadza licząc wg założeń że korzeń ma indeks 1.</li> <li>3. Następnie uzupełniam liczbami większymi od korzenia (130, 140 150, 160). Gdy <b>zabraknie</b> węzłów, otrzymamy stosowny komunikat, bo zaprojektowany tablica w programie nie jest w stanie pomieścić więcej elementów.</li> </ol>
 <p>Dodatkowo mam procedurę, która umożliwia ustawienie kursora w konsoli w odpowiednim</p>	<p>Są 2 rodzaje implementacji: wskaźnikowa i tablicowa. Ja zastosowałam tablicową, gdzie korzeniowi przypisałam nr indeksu 1, przez to index lewego potomka danego k-tego węzła wynosi <math>2k</math>, a prawego potomka <math>2k+1</math>.</p> <p>Elementy przechowuję w tablicy intów. Dodatkowa tablica typu bool zwraca true jeśli dana komórka jest jeszcze pusta - oraz false jeśli jest już wypełniona.</p> <p>W funkcji add() szukanie elementu odbywa się za pomocą metody “dziel i zwyciężaj” zgodnie z regułą drzewa. Wewnątrz pętli while poruszam się cały czas w głąb lewego lub prawego poddrzewa</p>

<p>wierszu i kolumnie. Przydaje mi się to w rysowaniu drzewa w późniejszej metodzie ShowTree() wyświetlającej drzewo na konsoli.</p>	<p>zależnie od porównania wartości aktualnie wstawianej ze sprawdzanym właśnie węzłem JUŻ istniejącym w drzewie.</p> <p>Po znalezieniu wolnego miejsca ustawiam flagę bool na wartość true (bo znalazłam miejsce) co kończy pętlę while szukającą dla tej liczby miejsca. Pętla skończy się również gdy wyjdziemy poza 15 węzeł - bo to oznacza, że ta liczba nie zmieści się w zadeklarowanej przeze mnie tablicy.</p> <p>Analogicznie postępuję w funkcji szukającej liczby, w każdym kroku odrzucając mniej więcej połowę pozostałych elementów.</p>
<p>Złożoność wyszukiwania elementu w BST</p> 	<p>Binarne drzewo wyszukiwań ma logarytmiczną złożoność obliczeniową wyszukiwania elementu, która w przypadku pesymistycznym degradowe się do czasu liniowego. Zależność ta w dużej mierze jest uzależniona od średniej wysokości drzewa (czyli zależy od danych wejściowych).</p>
	<p>Binarnych drzew poszukiwań używa się m.in. w bazach danych, a konkretniej specjalnego ich rodzaju, są to tzw. B-drzewa.</p> <p>Baza danych także jest strukturą drzewiastą, <u>dlatego właśnie</u> jest ona w stanie zwrócić jeden konkretny rekord z milionów znajdujących się w niej w ciągu zaledwie ułamków sekund - wyszukuje rekordy przy pomocy metody "dziel i zwyciężaj" analogicznie jak pokazałam to na początku.</p> <p><b>Reasumując:</b></p> <p>Płacąc niewielką mocą obliczeniową i ilością czasu przy wstawianiu elementów do drzewa, zyskujemy później możliwość użycia metody "dziel i zwyciężaj" do przeszukiwania jego zawartości w bardzo korzystnym czasie.</p>
<p><b>Źródła</b></p> <p>1. Bhargava Aditya Y.: "Algoritmy, Ilustrowany przewodnik"; Wydawnictwo Helion, 2017, str. 203-206, ISBN 978-83-283-3445-9</p> <p>2. Zelen M.: "Struktury danych: stos, kolejka, lista, drzewo binarne" [online] <a href="https://mirosławzelen.pl/kurs-c++/struktury-danych-stos-kolejka-lista-drzewo-binarne/">https://mirosławzelen.pl/kurs-c++/struktury-danych-stos-kolejka-lista-drzewo-binarne/</a> [dostęp: 24 stycznia 2022]</p>	<p>Dziękuję.</p>