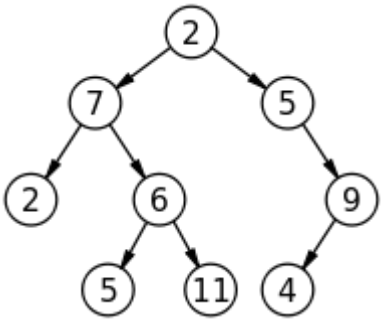
Algorytmy i złożoność obliczeniowa – studia niestacjonarne

Przygotowanie do Laboratorium 4: Drzewa binarne. Drzewa BST.

Drzewa binarne – wprowadzenie.

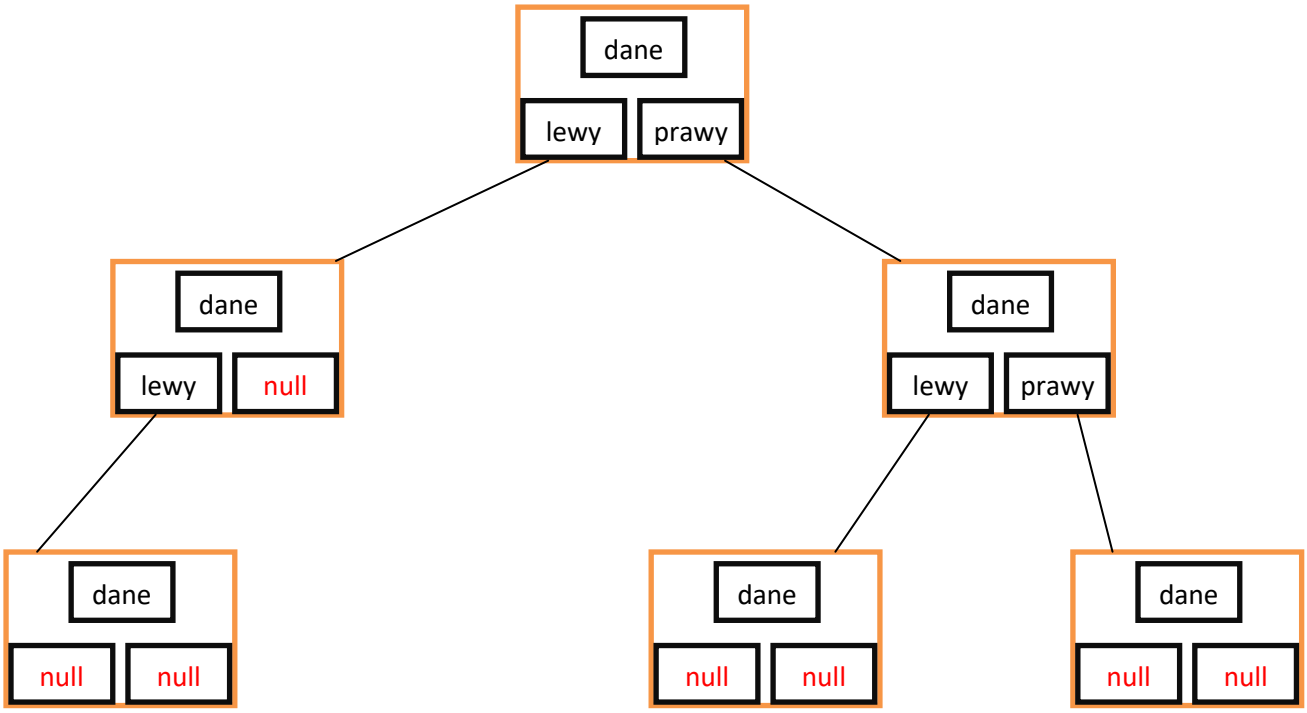


W informatyce drzewo binarne jest strukturą danych, w której każdy węzeł ma co najwyżej dwa węzły podrzędne, zwykle rozpoznawane jako "lewy" i "prawy".

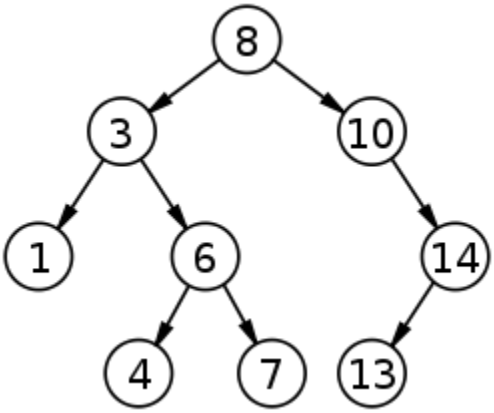
Każdy węzeł w strukturze danych jest osiągalny z korzenia – wystarczy podążać za referencjami (lewy lub prawy) aby przechodzić coraz niżej w strukturze drzewa. Drzewa binarne są podstawą implementacji drzew BST (Binary Search Tree) oraz kopców binarnych.

1. **Implementacja drzewa binarnego.**

   Węzły drzewa są obiektami składającymi się z trzech komponentów: danych przechowywanych w węźle oraz dwóch referencji do węzłów potomnych: *prawy* i *lewy*. Na zewnątrz całe drzewo jest reprezentowane jako referencja do korzenia drzewa.



2. **Porządek w drzewach BST.**



   W drzewach BST węzły są uporządkowane.

   Wartości znajdujące się w lewym poddrzewie są nie większe niż wartość w węźle nadrzędnym, a z kolei wartości w prawym poddrzewie są większe, niż wartość w węźle nadrzędnym.

3. Iteracyjne wstawianie i wyszukiwanie węzła w drzewie

Drzewa BST posiadają tę własność, iż jego węzły są uporządkowane. Wartości znajdujące się w lewym poddrzewie są nie większe niż wartość w węźle nadrzędnym, a z kolei wartości w prawym poddrzewie są większe, niż wartość w węźle nadrzędnym. W celu budowania lub przeszukiwania drzewa BST wykorzystywane są często algorytmy rekurencyjne. Można jednak algorytmy te zapisać iteracyjnie przez „rozwinięcie“ rekursji w pętlę `while`. Często ta druga wersja jest nawet efektywniejsza.

Pseudokod: Iteracyjne wyszukiwanie węzła w drzewie. Funkcja zwraca element szukany i jego poprzednika(rodzica).

```
function search(v : T; r: node, var y: node) : node;
 {T jest dowolnym typem liniowo uporządkowanym; r jest korzeniem drzewa
BST; y jest poprzednikiem wierzchołka wyszukiwanego przez search}
var x : node;
begin
    x := r; y:=nil;
    while (x < > nil) and (key(x) < > v) do begin
        y:=x;
        if v < key(x) then x := left(x) else x := right (x);
    end;
    search := x
end search;
```

Procedura `insert` najpierw sprawdza czy element nie istnieje w drzewie. Następnie tworzy nowy wierzchołek i wstawia tam element v. Następnie przeszukuje drzewo w celu znalezienia „odpowiedniego” miejsca do dowiązania wierzchołka x.

Pseudokod: Iteracyjne wstawianie węzła do drzewa BST.

```
procedure insert (v : T; var r : node);
var x, y: node;
begin
     if search(v, r, y) = nil then
     begin
          new(x) ;
          left(x) := nil; key(x) := v; right(x) := nil;
          if y= nil then r := x else
            if v< key(y) then left(y) := x
                         else right(y) := x
     end
end insert;
```

5. Zadania.

1. Zaimplementować klasę reprezentującą drzewo binarne (węzeł drzewa). Zaimplementować metodę pozwalającą na wypisanie drzewa.
2. Zaimplementować przeszukiwanie drzewa BST w celu znalezienia określonej wartości na dwa sposoby: rekurencyjnie oraz iteracyjnie.
3. Zaimplementować dodawanie wartości do drzewa BST na dwa sposoby: rekurencyjnie oraz iteracyjnie.
4. Zaimplementować usuwanie wartości z drzewa BST.