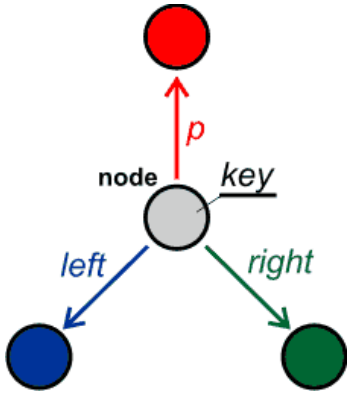
# ALGORYTMY I STRUKTURY DANYCH

## STRUKTURA DANYCH – DRZEWO POSZUKIWAŃ BINARNYCH

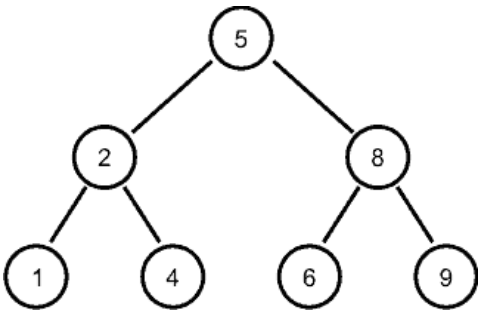
**Drzewo poszukiwań binarnych BST** (ang. *Binary Search Tree*) jest dynamiczną strukturą danych zbudowaną z **węzłów** (ang. *node*). Każdy węzeł może posiadać dwóch **potomków** (*left* – lewy, *right* - prawy) oraz jednego **przodka** (p - *parent*). Z każdym węzłem dodatkowo związany jest **klucz** (*key*).

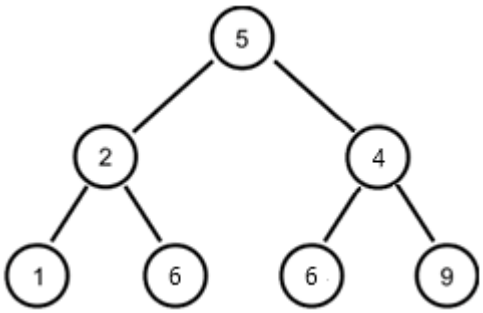



Dla każdego węzła w drzewie BST zachodzą następujące własności:

- Wartości kluczy węzłów leżących *w lewym poddrzewie* węzła są *mniejsze lub równe* wartości klucza danego węzła.
- Wartości kluczy węzłów leżących *w prawym poddrzewie* węzła są *większe lub równe* wartości klucza danego węzła.

**Zadanie**: które z poniższych drzew spełnia powyżej wymienione własności drzewa BST?




Strukturę węzłów drzewa BST deklarujemy następująco:

```
class BSTNode {
   public:
       int data;
       BSTNode* left, * right;
};
```

**Zadanie 1**. Zbuduj drzewo BST dla następującego ciągu wartości: 8, 3, 6, 1, 7, 10, 0, 2, 11, 4, 9. (*Pliki do wykorzystania*: *zadania_BST.xlsx*, arkusz *zadanie_1*).

**Zadanie 2**. Przeanalizuj kod budowania drzewa BST (*Pliki do wykorzystania*: *projekt BST*).

Złożoność czasowa funkcji *insert*() jest rzędu **O(*h*)**, gdzie h jest **wysokością** drzewa. Wysokość drzewa, to jest odległość korzenia (w sensie liczby krawędzi) od najbardziej oddalonego liścia drzewa.

## Wyszukiwanie węzła o zadanym kluczu

**Zadanie 3**. Napisz funkcje *search*(), która wyszukuje w drzewie BST węzeł o zadanym kluczu. Jeśli taki węzeł istnieje, to zwraca wskaźnik na ten węzeł. Jeśli nie – zwraca NULL. Oszacuj złożoność obliczeniową operacji wyszukiwania.

## Wyszukiwanie najmniejszego i największego klucza

Przy wyszukiwaniu najmniejszego klucza poruszamy się po lewych krawędziach węzłów aż osiągniemy węzeł, którego lewy potomek jest pustym liściem. Klucz tego węzła posiada wartość minimalną, co wynika bezpośrednio z własności drzewa BST.

**Zadanie 4.** Napisz funkcję wyszukującą minimalną wartość w drzewie. Zakładamy, że drzewo nie jest puste.
**Zadanie 5.** Napisz funkcję wyszukującą maksymalną wartość w drzewie.

## Przechodzenie przez drzewo BST

### Algorytm inorder

W algorytmie **inorder** przetwarzamy kolejno:

1. lewą gałąź drzewa BST,
2. węzeł,
3. prawą gałąź drzewa BST.

W efekcie rozpoczynamy od węzła drzewa o minimalnym kluczu i odwiedzamy kolejne węzły w porządku rosnącym ich kluczy, kończąc na węźle maksymalnym.

(*Pliki do wykorzystania*: *zadania_BST.xlsx*, arkusz *zadanie_6_7_8*)
**Zadanie 6**. Na podstawie losowego ciągu kluczy utwórz drzewo BST i przeglądnij go algorytmem **inorder**.

**Zadanie 7.** Napisz funkcje **preorder**, w której przetwarzamy kolejno: węzeł, lewą gałąź drzewa BST, prawą gałąź drzewa BST.

**Zadanie 8.** Napisz funkcje **postorder**, w której przetwarzamy kolejno: lewą gałąź drzewa BST, prawą gałąź drzewa BST, węzeł.