

# Algorytmy i struktury danych

Dana jest struktura danych będąca węzłem drzewa BST

```
struct node {
    int key;
    node* left;
    node* right;
    node(int k, node* l, node* r) : key(k), left(l), right(r) {}
};
```

## Zadanie 1

Zapisz warunki jakie muszą spełniać klucze drzewa BST.

Klucze w lewym poddrzewie są mniejsze od klucza węzła, natomiast w prawym poddrzewie są większe lub równe.

## Zadanie 2

Napisz procedurę `node* find(node* tree, int x)`, która zwraca wskaźnik na węzeł zawierający `x`, lub `NULL`, jeśli nie ma takiego węzła.

```
node* find(node* tree, int x) {
    if (tree == nullptr) return nullptr;
    if (tree->key == x) return tree;
    if (tree->key > x) return find(tree->left, x);
    return find(tree->right, x);
}
```

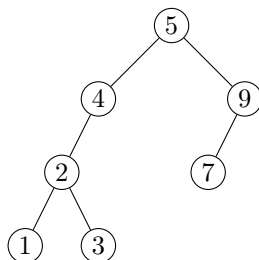
## Zadanie 3

Napisz procedurę `void insert(node*& tree, int x)` (dodaje do drzewa `tree` klucz `x`).

```
void insert(node*& tree, int x) {
    if (tree == nullptr) {
        tree = new node(x, nullptr, nullptr);
        return;
    }
    if (tree->key > x) insert(tree->left, x);
    else insert(tree->right, x);
}
```

## Zadanie 4

Drzewo BST o różnych kluczach można odtworzyć z listy par kluczWezła:kluczOjca. (a) Narysuj drzewo BST reprezentowane przez listę par: 1:2, 2:4, 3:2, 4:5, 6:7, 7:9, 8:7, 9:5. (b) wypisz jego klucze w porządku: INORDER, (c) PREORDER, (d) POSTORDER



(b) 1234567

(c) 5421397

(d) 1324795

## Zadanie 5

Napisz procedurę `void wypisz(node *tree, int order=0)`, która wypisuje klucze drzewa `tree` w porządku inorder gdy `order=0`, preorder gdy `order=1`, postorder gdy `order=2`.

```
void wypisz(node *tree, int order = 0) {
    if (tree == nullptr) return;
    if (order == 1) std::cout << tree->key;
    wypisz(tree->left, order);
    if (order == 0) std::cout << tree->key;
    wypisz(tree->right, order);
    if (order == 2) std::cout << tree->key;
}
```

## Zadanie 6

Jakie informacje przechowujemy w węźle drzewa czerwono-czarnego? Podaj definicję drzewa czerwono czarnego. Zadeklaruj strukturę `RBnode` tak, by dziedziczyła z `node`. Czy można dla niej użyć funkcji napisanych w zadaniach 2, 3 i 5?

```
struct NodeBST
{
    int32_t value;
    NodeBST* left;
    NodeBST* right;
};

struct NodeRBT : public NodeBST
{
    bool isBlack;
    NodeRBT* parent;
};
```

Drzewo czerwono-czarne musi przestrzegać następujące wymagania:

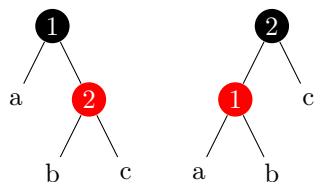
1. Każdy węzeł jest albo czerwony albo czarny.
2. Korzeń jest czarny.
3. Każdy liść jest czarny.

4. Czerwony węzeł ma czarnego ojca.
5. Każda ścieżka od korzenia do liścia ma tę samą liczbę czarnych węzłów.

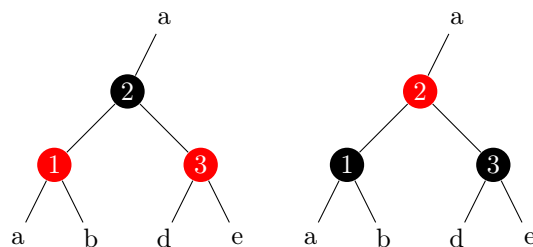
Można użyć dla niej funkcji `find` i `wypisz` z zadań 2 i 5, ale nie `insert` z zadania 3.

## Zadanie 7

Uzasadnij posługując się rysunkiem i opisem, że operacje na drzewie czerwono-czarnym (rotacja i przekolorowanie) nie zmieniają ilości czarnych węzłów, na żadnej ścieżce od korzenia do liścia



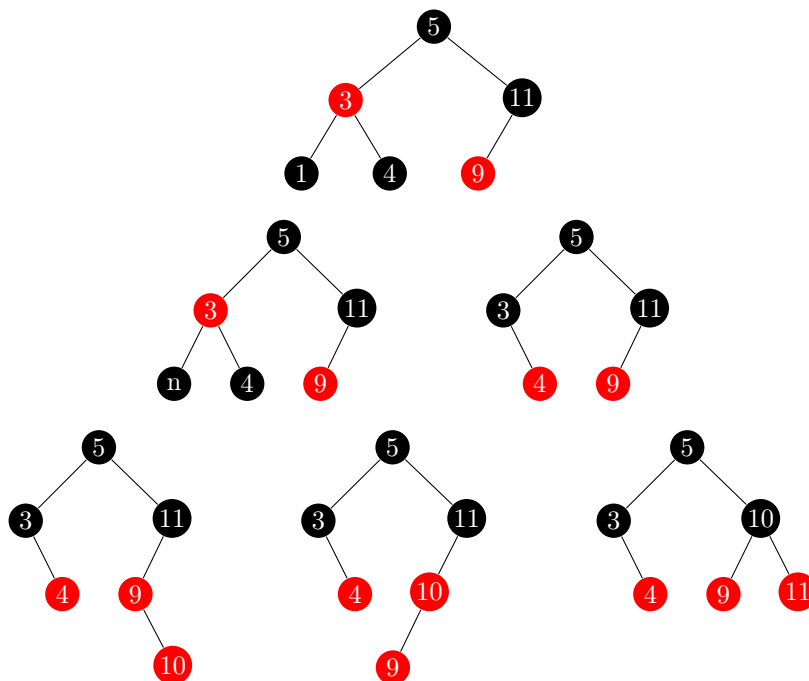
Po wykonaniu rotacji liczba czarnych węzłów na ścieżce nie ulega zmianie, a rotowane węzły wymieniają się piętami i kolorami.



Kolory czarne z ścieżek wychodzących zostają wypchnięte do węzła nadrzędnego.

## Zadanie 8

W poniższym drzewie czerwono-czarnym (czarne węzły oznaczono nawiasem kwadratowym), usuń 1, dodaj do wyjściowego 10:



## Zadanie 9

Jakie informacje przechowujemy w węźle B-drzewa? Podaj definicję B-drzewa

```
struct BTree
{
    uint32_t t;
    bool isLeaf;
    size_t n;
```

```

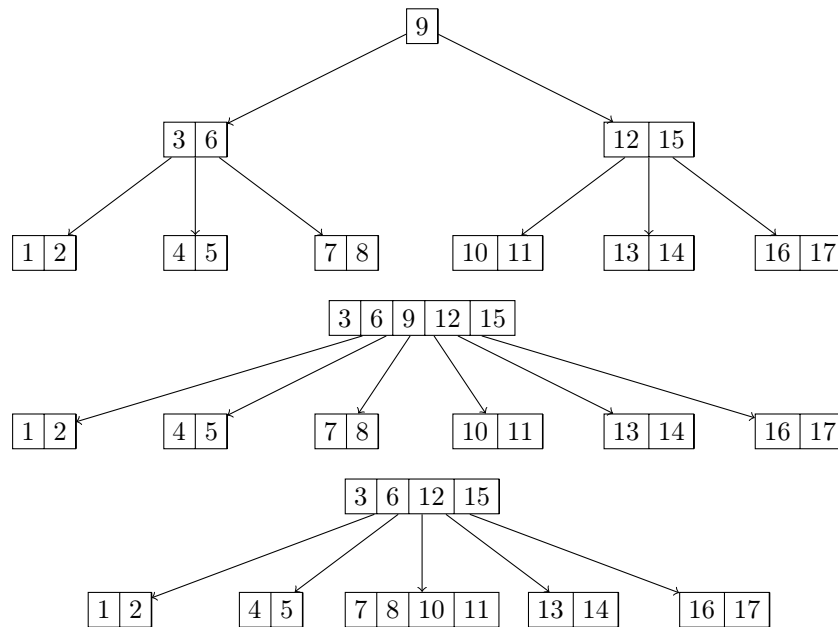
    int32_t *keys;
    size_t *offsets;
};

```

1. Każdy węzeł posiada  $n$  kluczy, przechowywanych w kolejności niemalejącej, a także informację o tym czy jest on liściem.
2. Dodatkowo każdy węzeł posiada  $n + 1$  wskaźników do swoich dzieci.
3. Klucze węzła dzielą zbiór kluczy przechowywanych w jego dzieciach na  $n + 1$  przedziałów.
4. Wszystkie liście znajdują się na tym samym poziomie równym wysokości drzewa  $h$ .
5. Każdy węzeł, z wyjątkiem korzenia, posiada co najmniej  $t - 1$  kluczy.
6. Każdy węzeł może posiadać maksymalnie  $2t - 1$  kluczy.

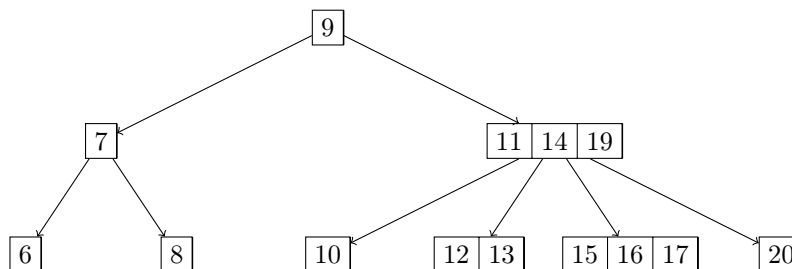
## Zadanie 10

Narysuj B-drzewo o  $t = 3$  zawierające dokładnie 17 kluczy na trzech poziomach: korzeń jego dzieci i wnuki. Następnie usuń z tego drzewa korzeń.

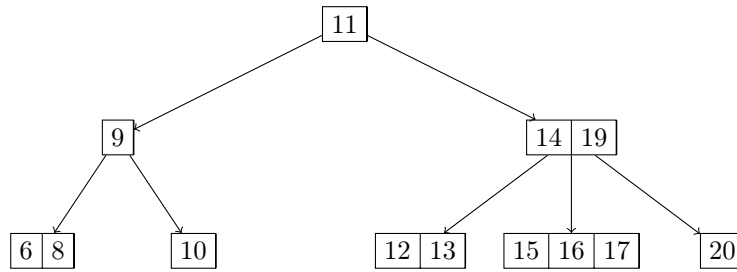


## Zadanie 11

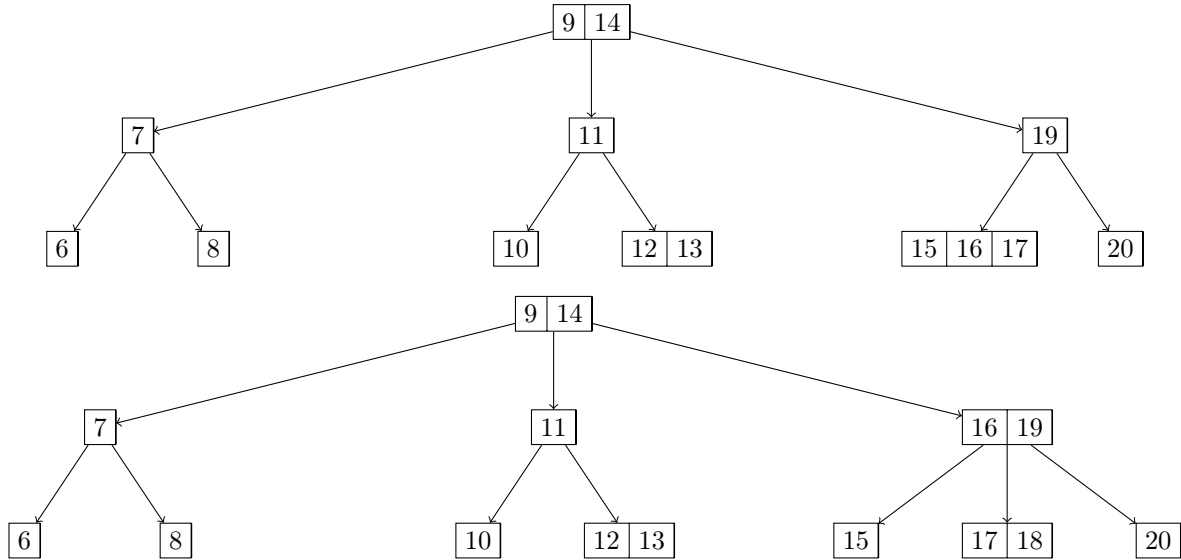
Podano na rysunku B-drzewo o  $t = 2$ :



- usuń z tego drzewa 7.



- do drzewa widocznego powyżej dodaj 18.



## Zadanie 12

W B-drzewie o  $t = 10$ :

- ile kluczy może zawierać korzeń (podaj przedział),  
Korzeń zawiera od 1 do 19 kluczy. ( $\max 2t - 1$ )
- ile dzieci może mieć korzeń (podaj przedział),  
Korzeń może mieć od 2 do 20 dzieci. ( $\min t \max 2t$ )
- ile kluczy może mieć potomek korzenia (podaj przedział),  
Potomek korzenia może mieć od 9 do 19 kluczy. ( $\min t - 1 \max 2t - 1$ )
- ile dzieci może mieć potomek korzenia (podaj przedział),  
Potomek korzenia może mieć od 10 do 20 dzieci. ( $\min t \max 2t$ )
- ile maksymalnie węzłów może być na  $k$ -tym poziomie (przyjmując, że korzeń to poziom 0)  
Na  $k$ -tym poziomie może być maksymalnie  $(2t)^k$  węzłów.
- ile łącznie kluczy może być na  $k$ -tym poziomie (podaj przedział).  
Nie licząc korzenia dla którego minimum to 1 klicz to na  $k$ -tym poziomie może być od  $2(t - 1)t^{k-1}$  do  $(2t - 1)(2t)^k$  kluczy. ( $\min (2\min)t^{k-1} \max (\max)t^k$ )