Algorytmy i struktury danych (Lista 7)

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

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

Zadanie 1 Warunki klucza drzewa BST

1. Lewa strona zawiera wartości mniejsze od rodzica
2. Prawa większe lub równe
3. Llucz z wartością

Zadanie 2 Napisz procedurę `node* find(node* tree, int x)`, która zwraca wskaźnik na węzeł zawierający x, lub NULL, jeśli nie ma takiego węzła.

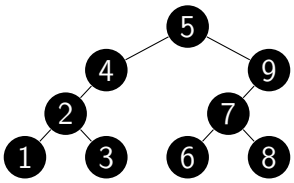
```
node* find(node* tree, int x) {
    while (tree != null && tree->key != x)
        if (x < tree->key) tree = tree->left
        else tree = tree->right
}
```

Zadanie 3 Napisz procedurę `void insert(node*& tree, int x)` (dodaje do drzewa tree klucz x).

```
node* insert(node*& tree, int x) {
    node **node = &tree;
    while (*tree != null)
        if (x < (**node).key) node = &(**node).left
        else node = &(**node).right

    *node = new node(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

1. INORDER: 1 2 3 4 5 6 7 8 9
2. PREORDER: 5 4 2 1 3 9 7 6 8
3. POSTORDER:1 3 2 4 6 8 7 9 5

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?

RBT: find tak samo (5), wypisz/print tak samo (5), insert nie (3)

Oprócz lewego/prawego dziecka kolor węzła (czerwony lub czarny). Logika/definicja: drzewo czerwono-czarne jest drzewem BST, w którym każdy węzeł ma dodatkowo kolor (czerwony lub czarny). Węzły czerwone mają 2 czarnych dzieci. Korzeń jest czarny. Każda ścieżka od korzenia do liścia ma tę samą liczbę czarnych węzłów. Każdy liść jest czarny (wartownik) a czerwony węzeł nie może mieć czerwonego dziecka.

```
struct BSTnode:
    int key;
    BSTnode *left;
    BSTnode *right;
end

struct RBTnode : BSTnode:
    bool color;
    RBT* parent;
end
```

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

Zadanie 8 W poniższym drzewie czerwono-czarnym (czarne węzły oznaczono nawiasem kwadratowym), usuń 1, dodaj do wyściowego 10:

```
        [5]
   (3)      [11]
[1]  [4]   (9)
```

```
      [5]
(3)       (10)
 [4]    [9]  [11]
```

```
          [5]
     (3)      [11]
[1]    [4]   (9)
              (10)
```

```
      [5]
[3]       (10)
 (4)    [9]  [11]
```

```
          [5]
     (3)       [11]
[1]    [4]    (10)
             (9)
```

```
          [5]
     (3)       [10]
[1]    [4]    (9) (11)
```

```
          [5]
     (3)       (10)
[1]    [4]    [9] [11]
```

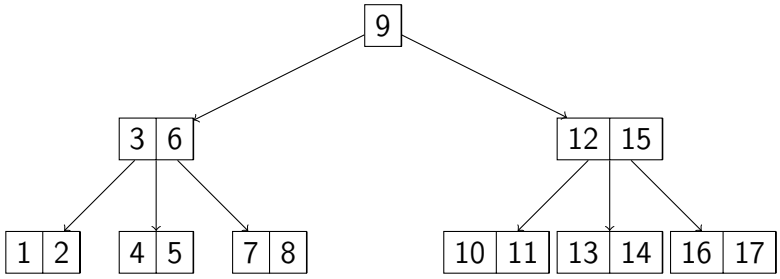
Zadanie 9 Jakie informacje przechowujemy w węźle B-drzewa? Podaj definicję B-drzewa

```
struct BTnode:
    int t;              // minimum degree
    int n;              // current number of keys
    int* keys;          // array of keys in non-decreasing order
    bool leaf;          // is it a leaf?
    BTnode** children;
end
```

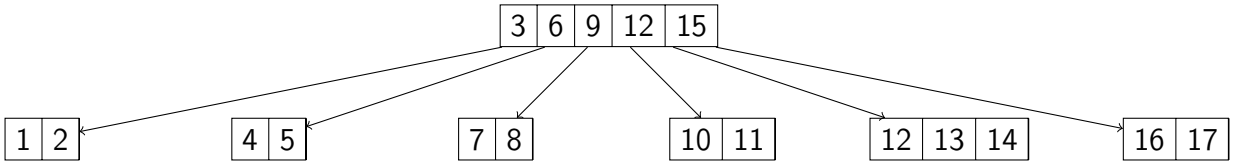
Inne założenia (oprócz zawartych w komentarzach):

1. Węzeł wewnętrzny zawiera $n+1$ wskaźników do synów.
2. Klucze rozdzielają dzieci na przedziały (n+1).
3. Każdy węzeł różny od korzenia musi mieć co najmniej $t-1$ kluczy i co najwyżej $2t-1$ kluczy. (korzeń może mieć od 1 do $2t-1$ kluczy)
4. Wszystkie liście leżą na tej samej wysokości równym h.

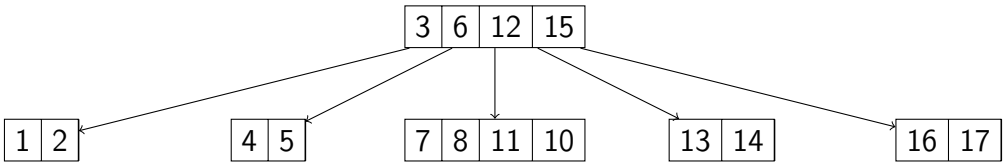
Zadanie 10 Narysuj B-drzewo o t = 3 zawierające dokładnie 17 kluczy na trzech poziomach: korzeń jego dzieci i wnuki. Następnie usuń z tego drzewa korzeń.



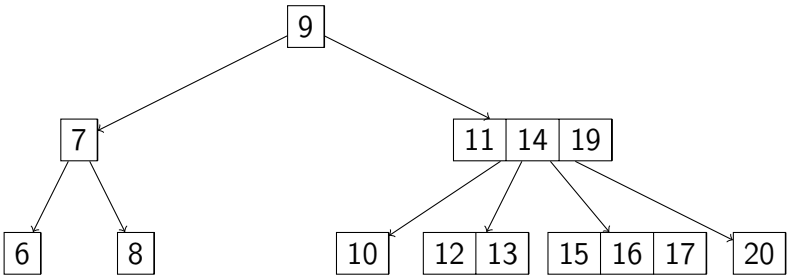
Dzieci korzenia mają minimalną liczbę kluczy, więc robimy po prostu unsplit.



Teraz możemy usunąć 9. Jednak nie możemy zastąpić jej najmniejszym kluczem prawego dziecka, dlatego robimy unsplit dzieci.



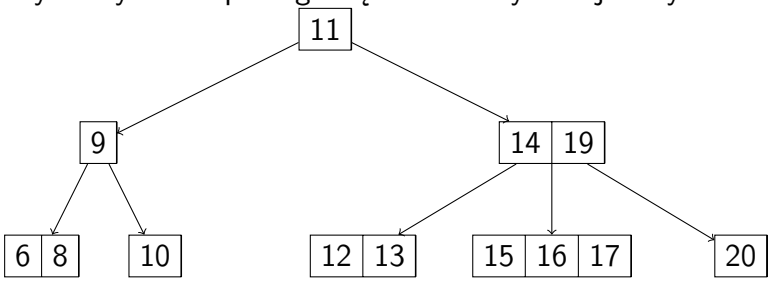
Zadanie 11 Podano na rysunku B-drzewo o $t = 2$:



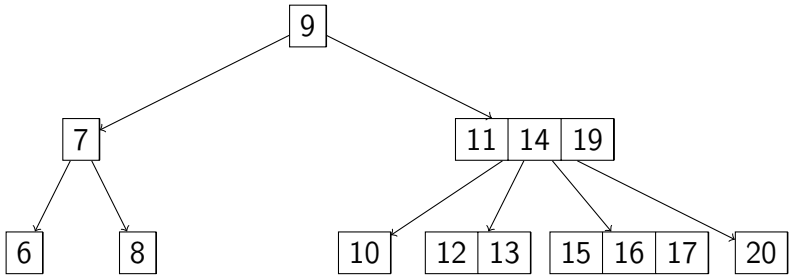
- usuń z tego drzewa 7.

- do drzewa widocznego powyżej dodaj 18.

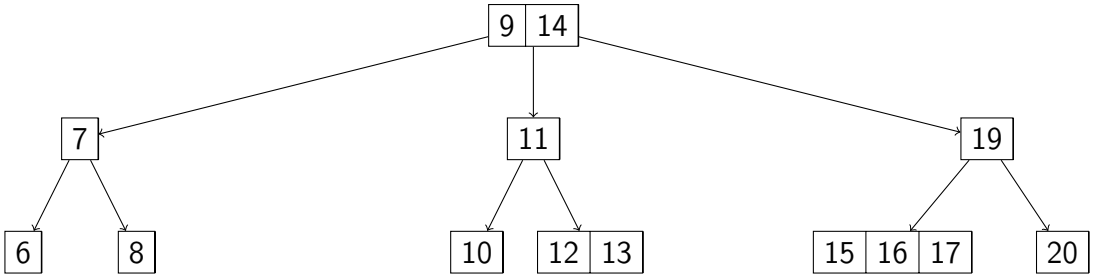
Nie wchodzimy do dziecka o minimalnej liczbie ani maksymalnej liczbie węzłów. Dlatego od razu pożyczamy 11 od pełnego węzła i robimy z niej nowy korzeń.



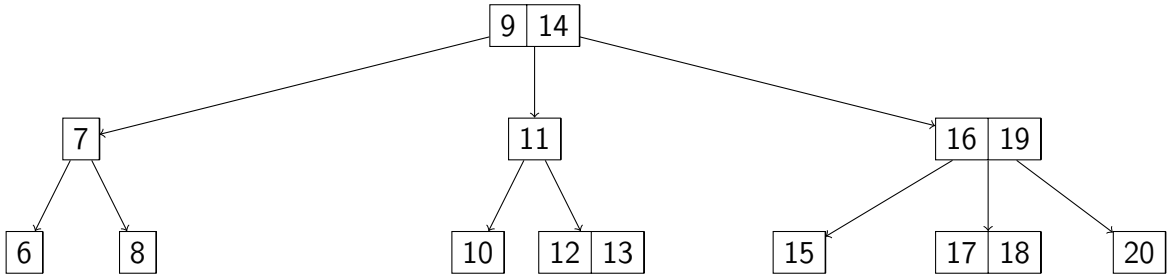
Dodanie 18:



Analogicznie jak wcześniej nie wchodzimy do pełnego węzła, tylko od razu robimy split i 14 idzie do góry.



Widząc znowu pełny węzeł robimy split i 16 idzie do rodzica.



Zadanie 12 W B-drzewie o t = 10:

(a) ile kluczy może zawierać korzeń (podaj przedział),
Korzeń zawiera od 1 do 19 kluczy. (max $2t - 1$)

(b) ile dzieci może mieć korzeń (podaj przedział),
Korzeń może mieć od 2 do 20 dzieci. (min t max $2t$)

(c) ile kluczy może mieć potomek korzenia (podaj przedział),
Potomek korzenia może mieć od 9 do 19 kluczy. (min $t - 1$ max $2t - 1$)

(d) ile dzieci może mieć potomek korzenia (podaj przedział),
Potomek korzenia może mieć od 10 do 20 dzieci. (min t max $2t$)

(e) 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.

(f) 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 $(2min)t^{k-1}$ max $(max)t^k$