Drzewa czerwono-czarne

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){}
};
```

1. 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 == NULL || tree->x == x)
    {
        return tree;
    }
    if (x < tree->x)
        return find(tree->left, x);
    else
        return find(tree->right, x);
}
```

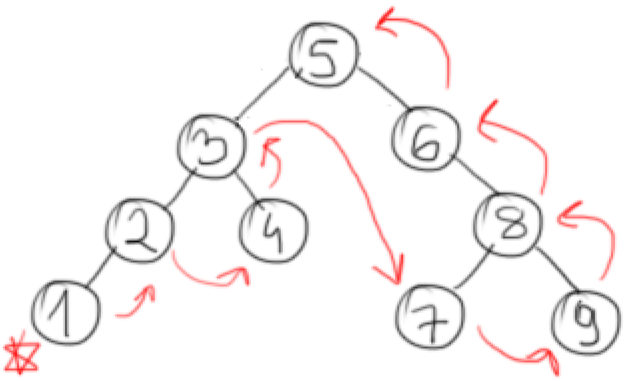
2. Napisz rekurencyjną wersję procedury void insert (node *& tree, int x)

```
void insert(node*& tree, int x)
{
    if (!tree)
        tree = new node(x, NULL, NULL);
    else
        if (x < tree->x)
            insert(tree->left, x);
        else
            insert(tree->right, x);
}
```

***Czy potrafisz zmienić ją tak, by zwracała poziom na którym został dodany węzeł (0- nowy węzeł to korzeń, bo drzewo było puste, 1 - bezpośrednio pod korzeniem itd.)?**

```
int insert(node*& tree, int x, int i)
{
   if (!tree)
   {
       tree = new node(x, NULL, NULL);
       return i;
   }
   else
   {
       i++;
       if (x < tree->x)
           return insert(tree->left, x, i);
       return insert(tree->right, x, i);
   }
}
```

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



Strzałki pokazują przechodzenie postorder (lewy, prawy, klucz): 1 2 4 3 7 9 8 6 5

4. Napisz procedurę void destroy(node* tree), która odwiedza węzły drzewa tree w porządku postorder lecz zamiast wypisywać klucz odwiedzanego węzła wywołuje delete tego węzła.

```
void destroy(node *tree)
{
    if(tree)
    {
        destroy(tree->left);
        destroy(tree->right);
        delete(tree)
    }
}
```

5. Podaj definicję drzewa czerwono-czarnego. Narysuj poprawne drzewo, które na lewo od korzenia ma 1 węzeł, a na prawo 7 węzłów.

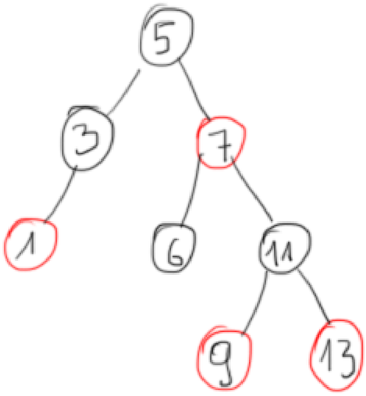
Drzewo RB - samobalansujące się binarne drzewo wyszukiwań, które:

- posiada węzły o kolorze czarnym lub czerwonym
- na każdej ścieżce między liściem a korzeniem znajduje się taka sama ilość czarnych węzłów
- każdy liść i korzeń są czarne
- lewe dzieci mają mniejsze wartości, prawe - większe lub równe
- każdy węzeł może mieć co najwyżej dwóch synów

6. Zadeklaruj strukturę RBnode tak, aby dziedziczyła z node.
Napisz funkcję int HB(RBnode * tree), która oblicza czarną wysokość drzewa tree.

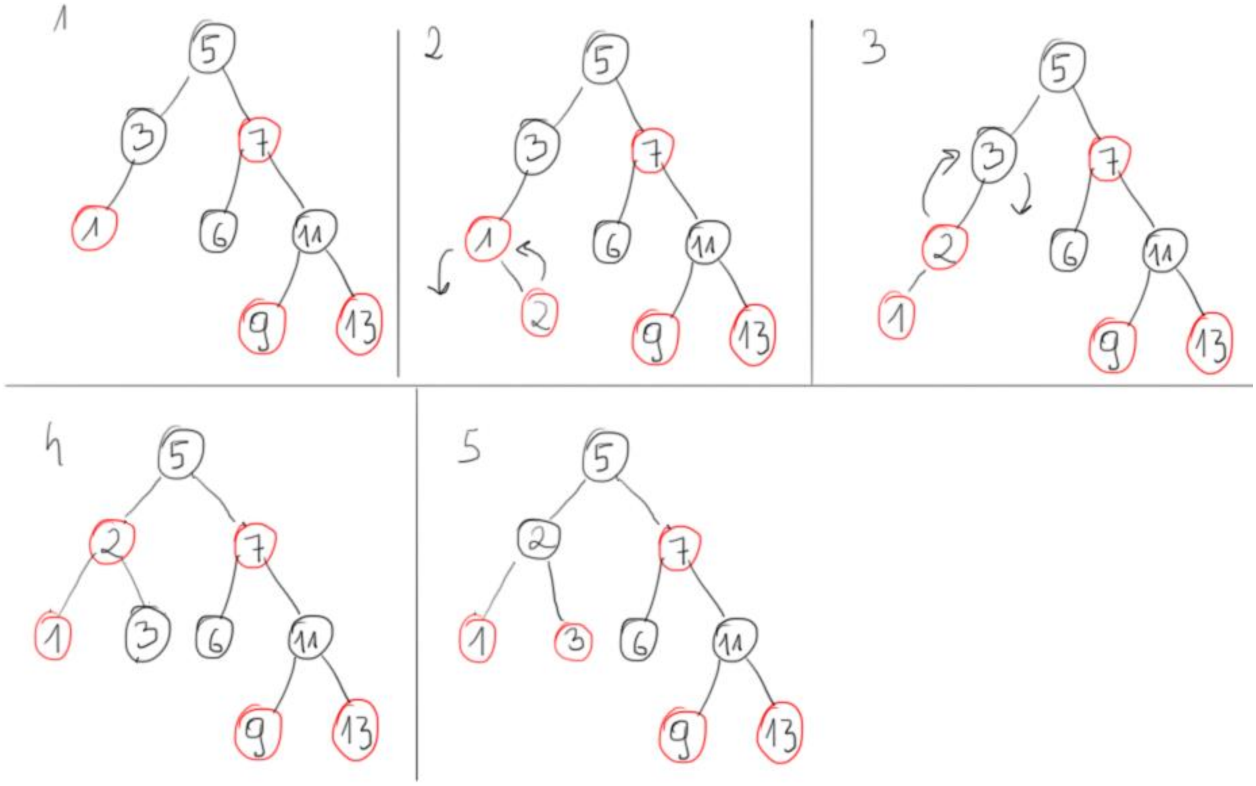
```
struct RBnode : public node
{
   node* parent;
   bool isBlack;
   RBnode(int k, node* l, node* r, node* p, bool B=false):node(k,l,r),parent(p),
isBlack(B){};
}
```

7. W poniższym drzewie czerwono-czarnym:



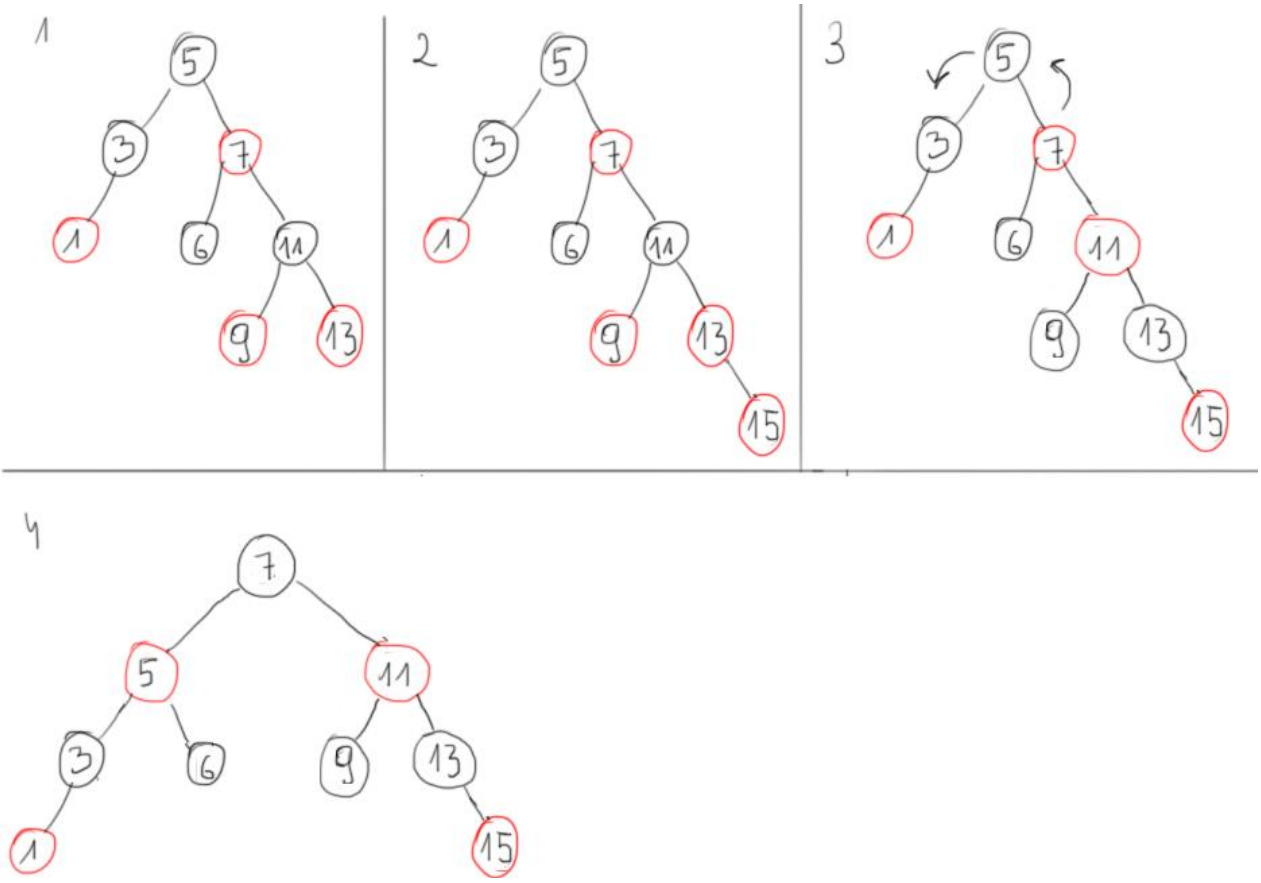
Za każdym razem narysuj wynikowe drzewo i napisz ile rotacji i przekolorowań wykonałeś. Nie rysuj etapów pośrednich.

a) wstaw 2



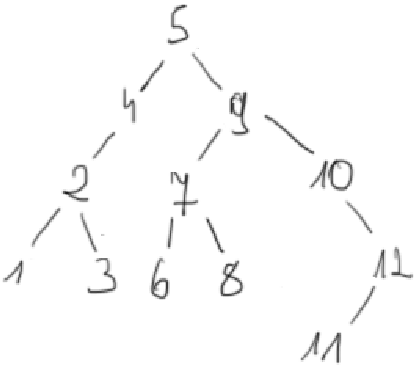
- rotacja w lewo 1 i 2 (ojciec dwójki jest czerwony, wujek, czyli liść, jest czarny, ojciec jest lewym dzieckiem a dwójka jest prawym dzieckiem więc zwykła rotacja w lewo)
- rotacja w prawo 2 i 3 (ojciec jedynki jest czerwony, wujek czarny, więc rotacja. Jedynka i dwójka są lewymi dziećmi więc rotacja ojciec-dziadek w prawo)
- zamiana kolorów 2 i 3 (bo po rotacji ojciec-dziadek)

b) wstaw 15



- zniesienie czarnego z 11 na 9 i 13 (bo 9 i 13 jako ojciec i wujek piętnastki są czerwone)
- rotacja w lewo 5 i 7 (bo 11 i 7 są prawymi dziećmi, więc rotacja ojciec-dziadek jedenastki w lewo)
- zamiana kolorów 5 i 7 (bo po rotacji ojciec-dziadek)

8*. Dane są klucze drzewa BST wypisane w porządku preorder: 5 4 2 1 3 9 7 6 8 10 12 11. Narysuj to drzewo.



B-drzewa

9. a) Jakie informacje przechowujemy w wewn. węzłach B-drzewa, a jakie w liściach?

To zadanie mam na 0.8 punkta, nie wiem kompletnie jak je zrobić poprawnie, bo nigdy podczas tego kursu nie było podziału na węzeł wewnętrzny i liść, jedynie był po prostu węzeł, który ma zmienną booleanową, która określa, czy dany węzeł jest liściem czy nie

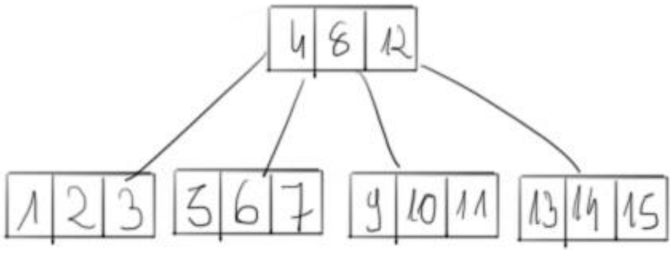
- Lista dzieci (tylko wierzchołek)
- Lista kluczy
- Ilość kluczy
- Czy jest liściem (boolean)

b) Jakie warunki musi spełniać drzewo złożone z takich węzłów, by nazwać je B-drzewem?

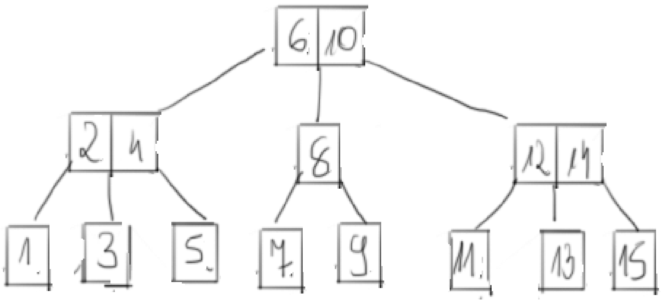
- Wszystkie lewe dzieci są mniejsze
- Wszystkie prawe dzieci są większe lub równe
- Każdy węzeł (poza korzeniem) może mieć od t-1 do 2t-1 kluczy
- Każdy węzeł zawsze ma o 1 więcej dzieci od kluczy

10. Narysuj poprawne B-drzewo o t=2 zawierające dokładnie 15 kluczy.

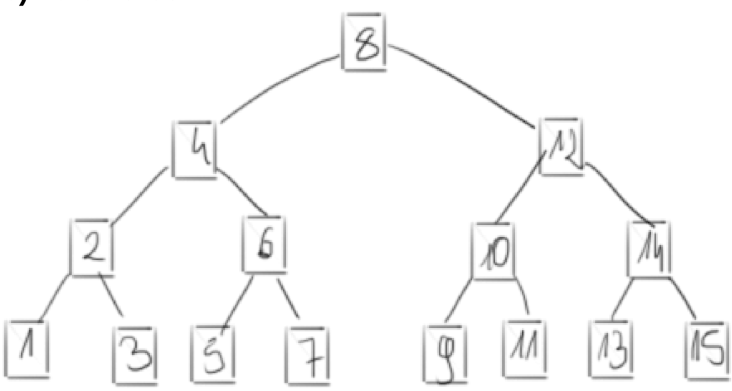
a) na dwóch poziomach (korzeń i dzieci)



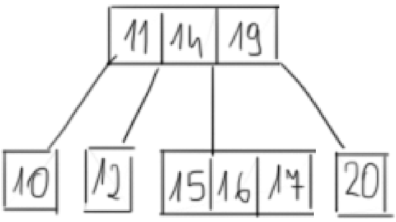
b) na trzech



c) na czterech



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



- dodaj do niego 9

Tutaj mamy dwie opcje, albo po prostu dołączamy dziewiątkę do dziesiątki (i to rozwiązanie jest poprawne, ale kij wi, czy CJowi ono odpowiada) i dopiero później naprawiamy ewentualne błędy rekurencyjnie idąc z dołu w górę **(down-top)**

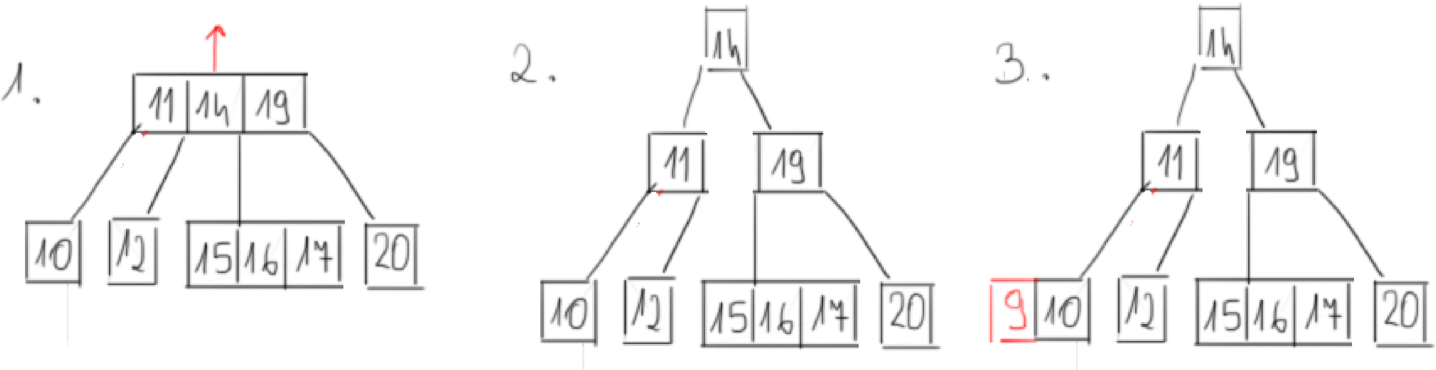
Druga opcja to “przygotowywanie” drzewa przed wstawieniem elementu idąc w dół. **(top-down)**

Polega to na tym że usuwać i dodawać można tylko do liścia i

W przypadku usuwania: jeżeli liczba kluczy jest minimalna (t-1) to nie wchodź, zrzucamy jednego rodzica w dół i łączymy z bratem

W przypadku dodawania: jeżeli liczba kluczy jest max (2t-1) to robimy splita

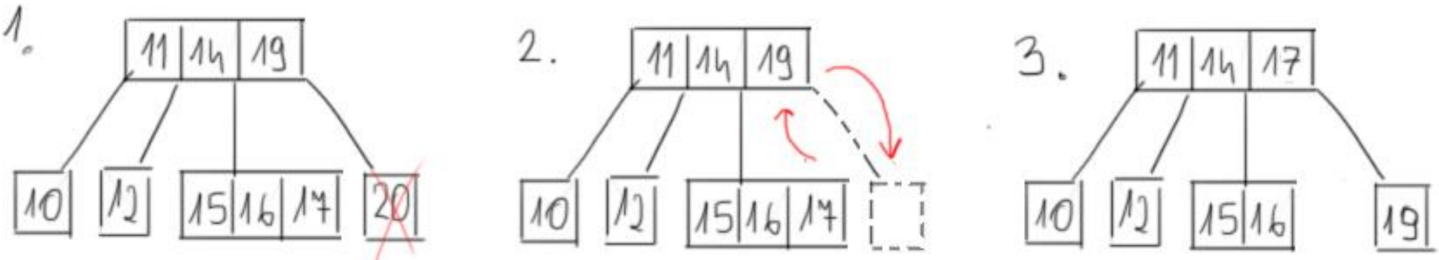
top-down:



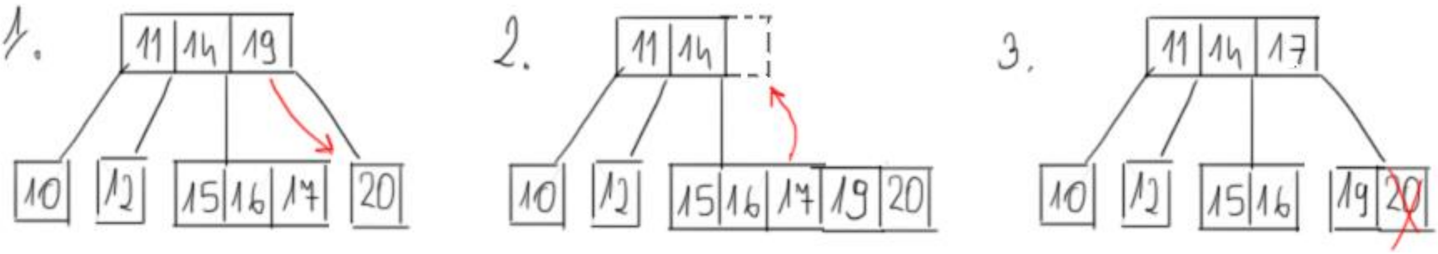
(korzeń już jest pełny więc od razu go rozbijamy)

- usuń z niego 20

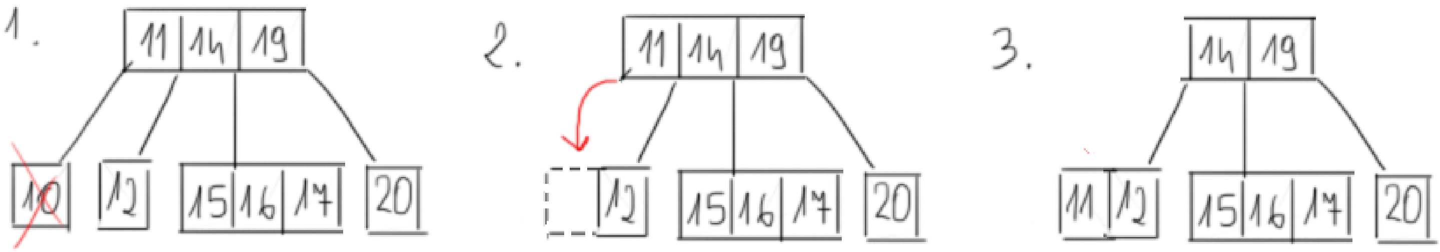
down-top:



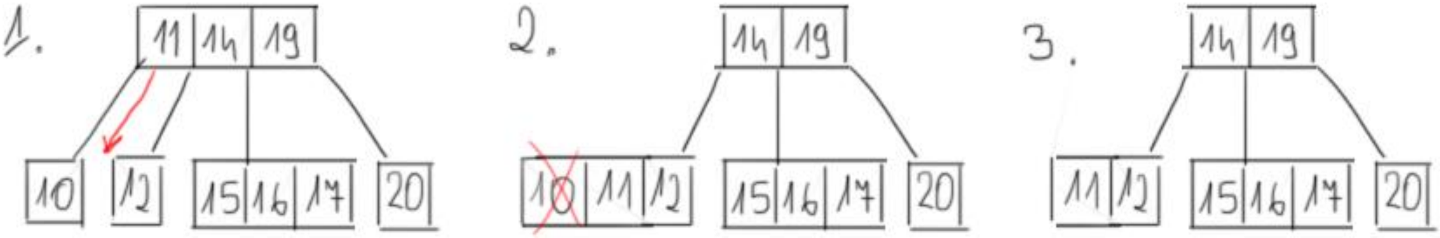
top-down:



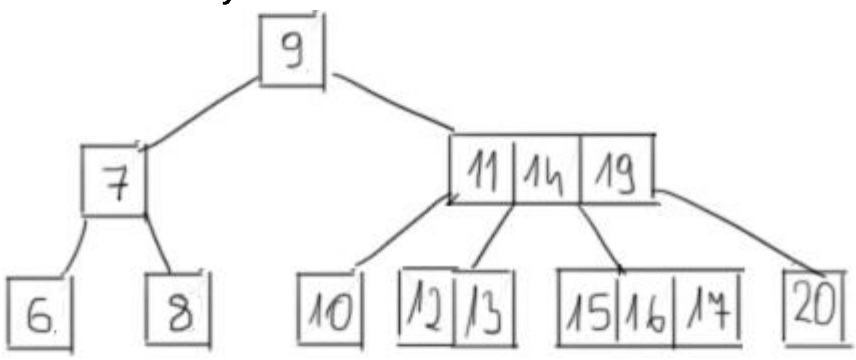
- usuń z niego 10
down-top:



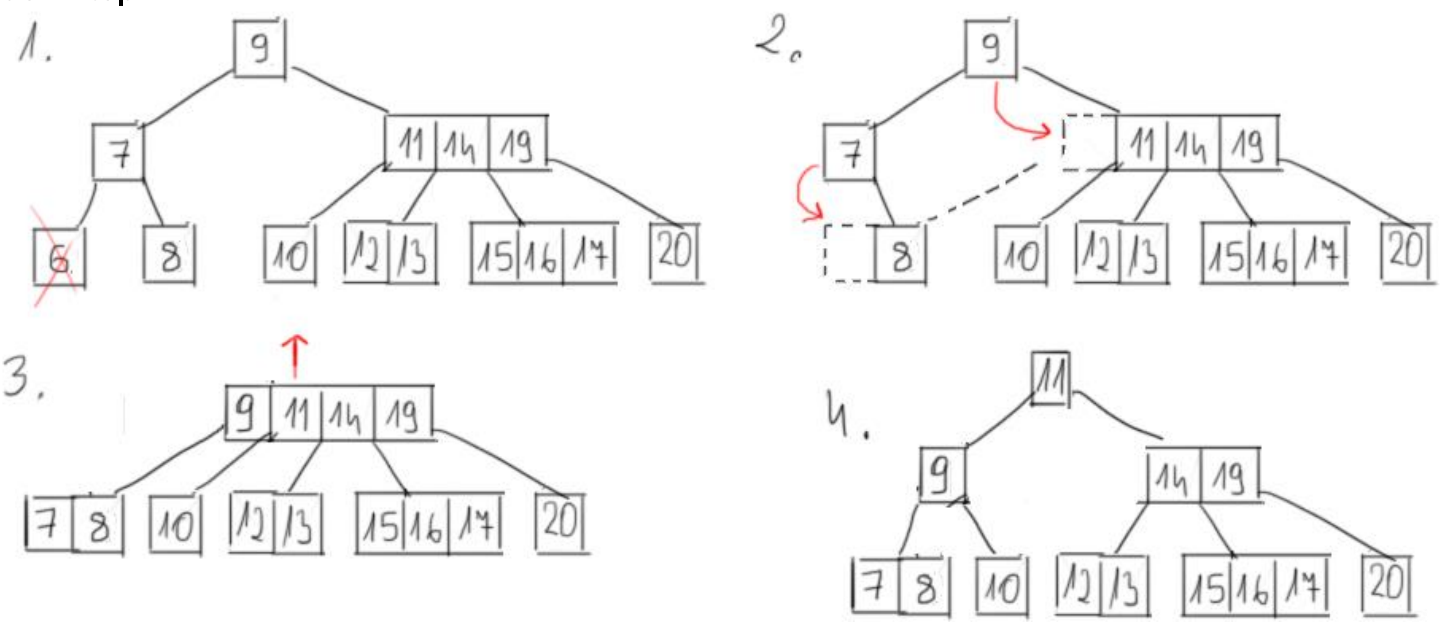
top-down:



12. Podano na rysunku B-drzewo o t=2:



- usuń z niego 6
down-top:



- dodaj do niego 21
tutaj rysunek

13. W B-drzewie o ustalonym t=5 oblicz:
a) ile maksymalnie węzłów może występować na k-tym poziomie (przyjmując, że korzeń to poziom 0)
$(2t)^k$
b) sumując ten wzór, oblicz ile maksymalnie węzłów może występować w całym drzewie o głębokości k

$$\sum_{i=0}^{k} (2t)^i$$

c) ile zatem maksymalnie kluczy może występować w drzewie o głębokości k?

$$\sum_{i=0}^{k} (2t)^i(2t-1)$$

d) o ile wzrośnie całkowita liczba kluczy, jeśli dla liści t zwiększymy do wartości t=10 (a dla pozostałych węzłów pozostawimy t=5)?

PRZYPOMNIENIE

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

1. Zapisz warunki jakie muszą spełniać klucze drzewa BST.
   - **Jeżeli węzeł posiada lewe poddrzewo, to wszystkie węzły w tym poddrzewie mają wartość nie większą od wartości danego węzła.**
   - **Jeżeli węzeł posiada prawe poddrzewo, to wszystkie węzły w tym poddrzewie mają wartość nie mniejszą od wartości danego węzła.**

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 == NULL || tree->x == x) return tree;
    if(x < tree->x) return find(tree->left, x);
    else return find(tree->right, x);
}
```

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

```
void instert(node*& tree, int x) {
    if(!tree) tree = new node(x, NULL, NULL);
    else
        if(x < tree->x) insert(tree->left, x);
        else insert(tree->right, x);
}
```

4. Drzewo BST o różnych kluczach można odtworzyć z listy par `klucz_wezła:klucz_ojca`.
(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.

```
                    (5)
          (4)                  (9)
     (2)                  (7)
(1)       (3)        (6)       (8)
```

Inorder - 1, 2, 3, 4, 5, 6, 7, 8, 9
Preorder - 5, 4, 2, 1, 3, 9, 7, 6, 8
Postorder - 1, 3, 2, 4, 6, 8, 7, 9, 5

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 inorder(node *t)  // wypisanie kluczy w porzadku "in order"
{
    if(t)
    {
        inorder(t->left);
        std::cout << t->x <<" ";
        inorder(t->right);
    }
}

void prerder(node *t) // wypisanie kluczy w porzadku "pre order"
{
    if(t)
    {
        std::cout<<t->x<<" ";
        prerder(t->left);
        prerder(t->right);
    }
}

void postorder(node *t) // wypisanie kluczy w porzadku "post order"
{
    if(t)
    {
        postorder(t->left);
        postorder(t->right);
        std::cout<<t->x<<" ";
    }
}
```

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?

Każdy węzeł drzewa czerwono-czarnego zabiera atrybuty:

- **Kolor: czerwony albo czarny**
- **Klucz**
- **Wskaźnik na prawego lub lewego syna**
- **Wskaźnik na ojca**

Drzewem czerwono-czarnym nazywamy drzewo przeszukiwań binarnych dla którego każdy węzeł posiada dodatkowy bit koloru – czarny lub czerwony – o następujących właściwościach

- **Każdy węzeł posiada kolor**
- **Korzeń jest koloru czarnego**
- **Każdy liść (NULL) jest czarny**
- **Dla czerwonego węzła dzieci są zawsze czarne**
- **Każda ścieżka z węzła do liścia ma zawsze taką samą liczbę czarnych węzłów**

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.

Rotacja:

[A] [B]
(B) (C) -> [X] (A)
[X] (C)

(B)
[X] [A]
(C)

Przekolorowanie:

---

8. W poniższym drzewie czerwono-czarnym (czarne węzły oznaczono nawiasem kwadratowym):

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

- wstaw do niego 10.
- usuń z wyjściowego drzewa 1.

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

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

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

- Tablica kluczy
- Stopień
- Tablica wskaźników na dzieci
- Liczba aktualnych kluczy
- Czy jest liściem

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.

(3 6 9 12)
(1 2) (4 5) (7 8) (10 11) (13 14 15 16 17)

(6)
(3) (9 12)
(1 2) (4 5) (7 8) (10 11) (13 14 15 16 17)

(5)
(3) (9 12)
(1 2) (4) (7 8) (10 11) (13 14 15 16 17)

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

(9)
(7) (11 14 19)
(6) (8) (10) (12 13) (15 16 17) (20)

- usuń z tego drzewa 7.
- dodaj do niego 18.

(9)
(6 8) (11 14 19)
(10) (12 13) (15 16 17 18) (20)

(9)
(6 8) (11 14 16 19)
(10) (12 13) (15) (17 18) (20)

(9 14)
(6 8) (11) (16 19)
(10) (12 13) (15) (17 18) (20)

12. W B-drzewie o $t = 10$:
- ile kluczy może zawierać korzeń (podaj przedział),
- ile dzieci może mieć korzeń (podaj przedział),
- ile kluczy może mieć potomek korzenia (podaj przedział),
- ile dzieci może mieć potomek korzenia (podaj przedział),
- ile maksymalnie węzłów może być na k-tym poziomie (przyjmując, że korzeń to poziom 0).
- ile łącznie kluczy może być na k-tym poziomie (podaj przedział).

t-1 2t-1

- 1 - 19
- 0 – 20 \ {1} - bez 1
- 9 – 19 - t-1 2t-1
- 0 - 20 \ {1}