ASD, Wyk. 10, 2022.04.28

Drzewa BST, kontynuacja

Def

T jest drzewem BST jeśli T jest drzewem binarnym z liniowym porządkiem na swoich kluczach oraz $\forall v \in T$

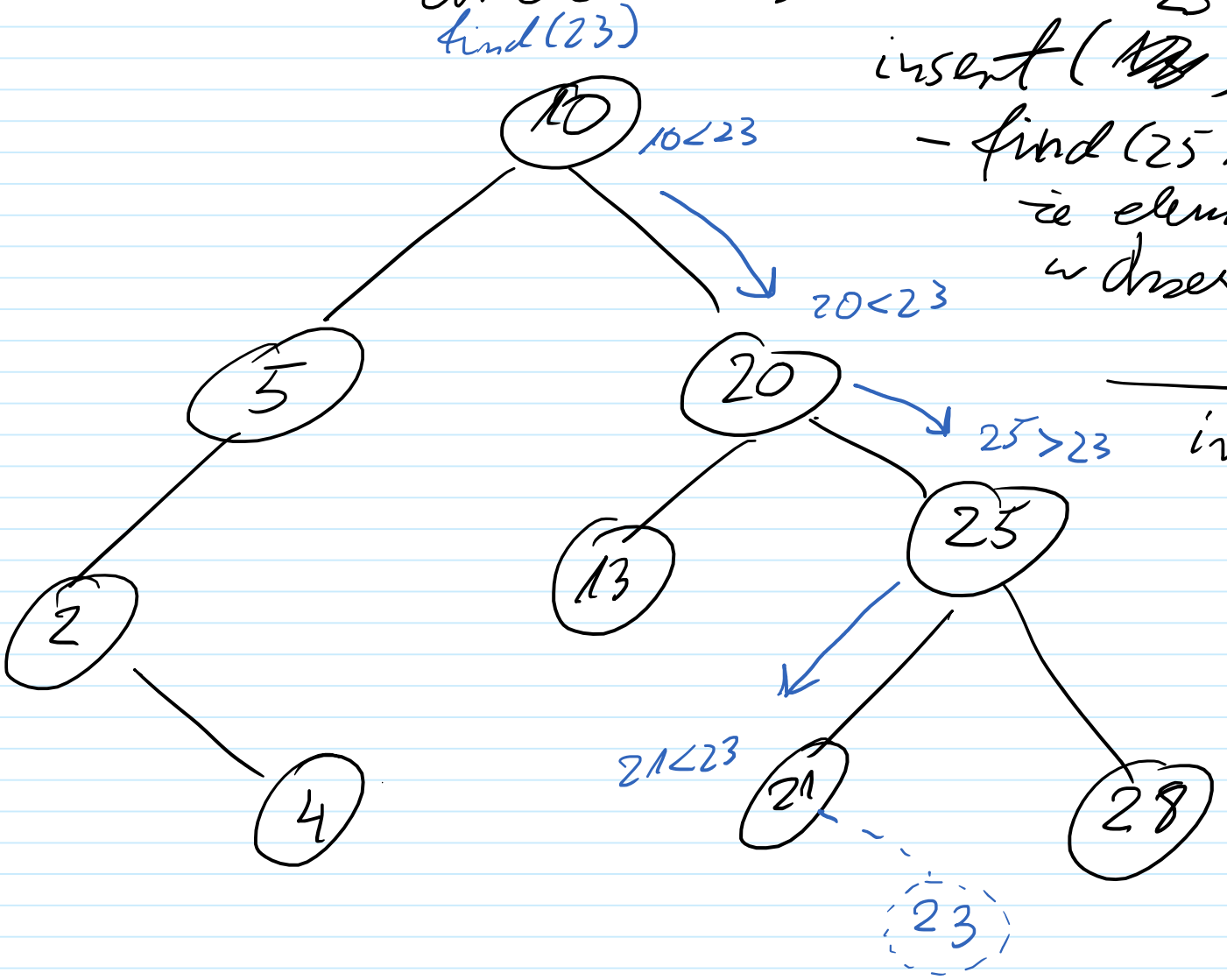
$\forall w \in Tree(v \to left) \quad w \to key < v \to key$

$\forall w \in Tree(v \to right) \quad v \to key < w \to key.$

Def

Wysokość drzewa T to długość najdłuższej ścieżki w T od korzenia do liścia.

Wstawianie do drzewa BST



insert(25)
- find(25) zwraca, że element jest już w drzewie → nic nie robimy

insert(23)
- find(23)

Wstawianie

- nowe elementy dodajemy jako liście w drzewie BST T istnieje dokładnie jedno miejsce, na które możemy dotrzeć nowy element x.

Kod

```
insert(int k){ //załóżmy, że trzymamy w drzewie tylko klucze, bez
dodatkowych danych
   Node *ptr=root;
   Node*prev=NULL);
   while(ptr!=NULL){
        prev=ptr;
        if(ptr->key==k) //element k już jest w drzewie
            return;
        if(ptr->key> k) ptr=ptr->left;
        if(ptr->key<k) ptr=ptr->right;
   }
   //wskaźnik prev wskazuje gdzie umieścić nowy rekord.
   Node *ptr2=new Node;
   ptr2->key=k;
   ptr2->lef=ptr2->right=NULL;
   if(prev==NULL) root=ptr2;
   else{
         if(prev->key<k) prev->right=ptr2;
        else prev->left=ptr2;
   }
}
```

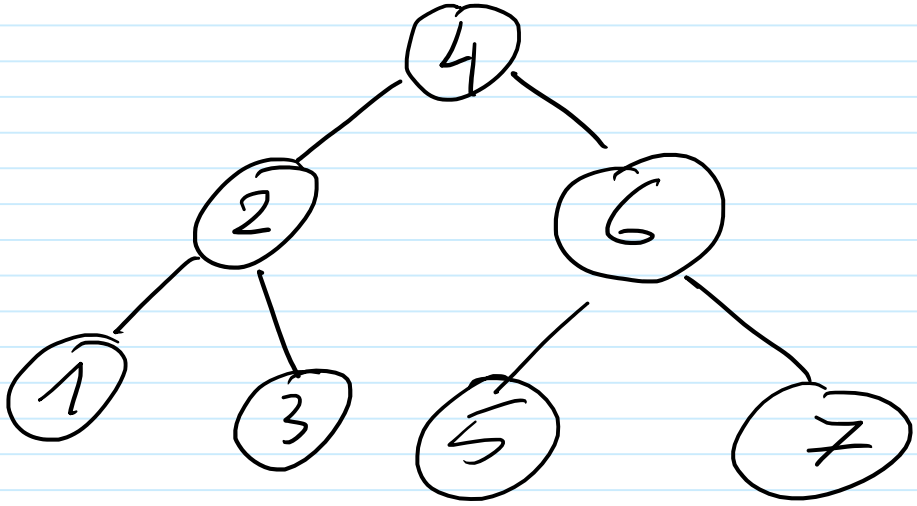
nic nie rób

wyszukanie miejsca na nowy element.

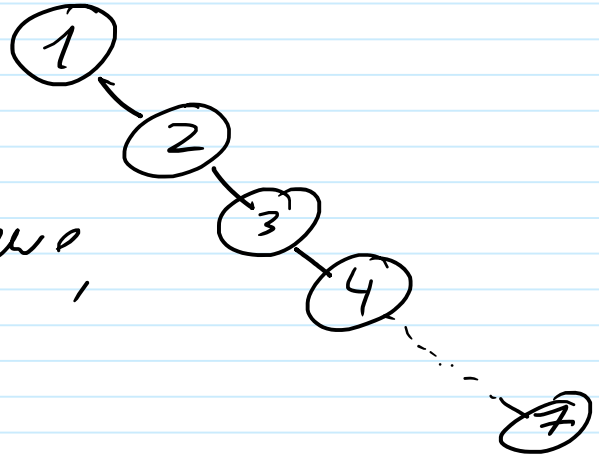
tworzymy nowy element i umieszczamy go w drzewie.

Przykład

Wstawiamy 4, 2, 1, 6, 5, 7, 3 →



Wstawiamy 1 2 3 4 5 6 7 →



wysokość drzewa to 6.

Mamy zdegenerowane drzewo, lista.

Algorytmy działające na drzewie BST

insert find delete

mają pesymistyczną złożoność

taką jak wysokość drzewa.

Czyli jeśli drzewo BST jest zdegenerowane,

to otrzymujemy złożoność liniową.

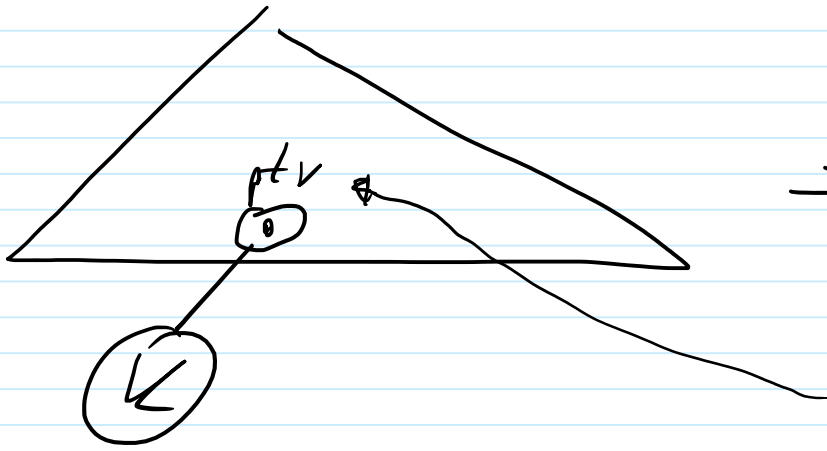
Na szczęście szanse na
taki scenariusz są małe
(wykładniczo małe wzgl. n).

W średnim przypadku:
jeśli wstawiamy losowe elementy
do drzewa BST to oczekiwana
wysokość drzewa to
$\simeq 1.39 \log_2(n)$.

Usuwanie z drzewa BST

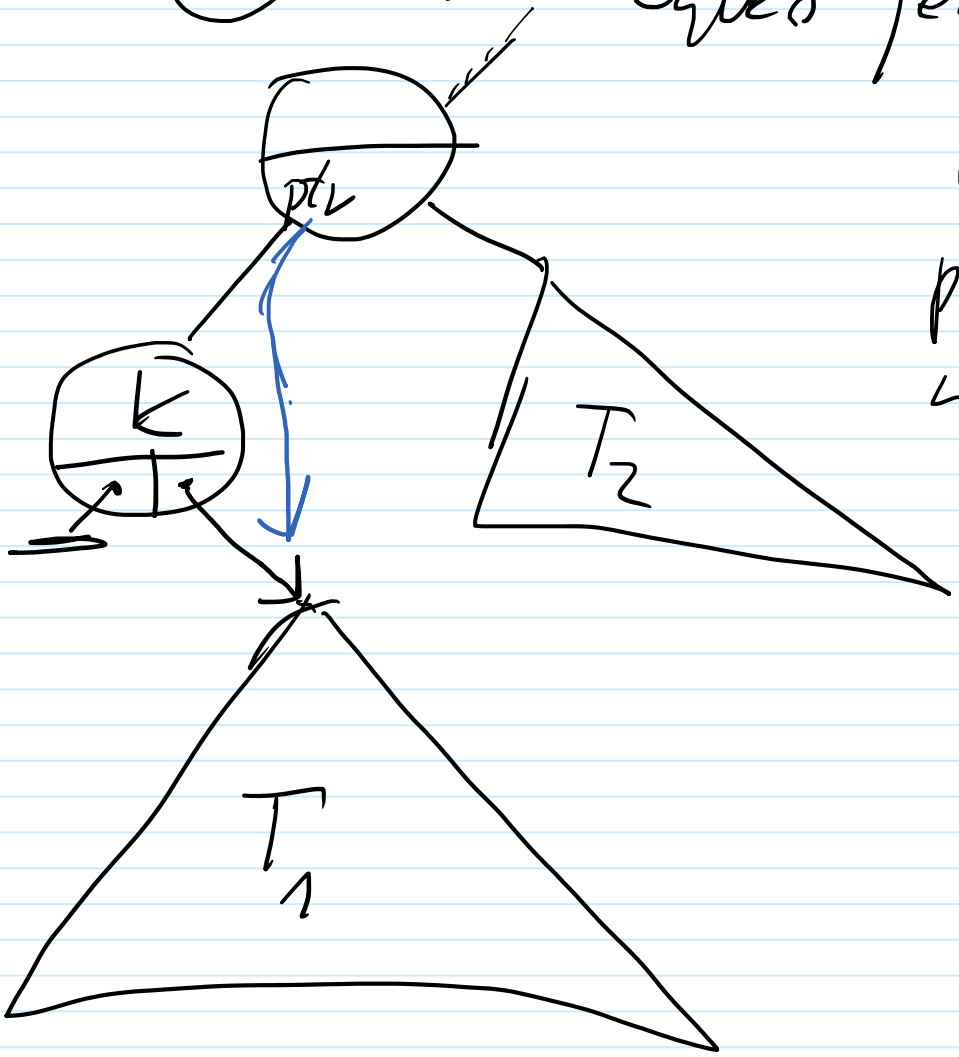
Mamy trzy przypadki, które musimy rozważyć podczas wykonywania delete(k).

① k jest w liściu.

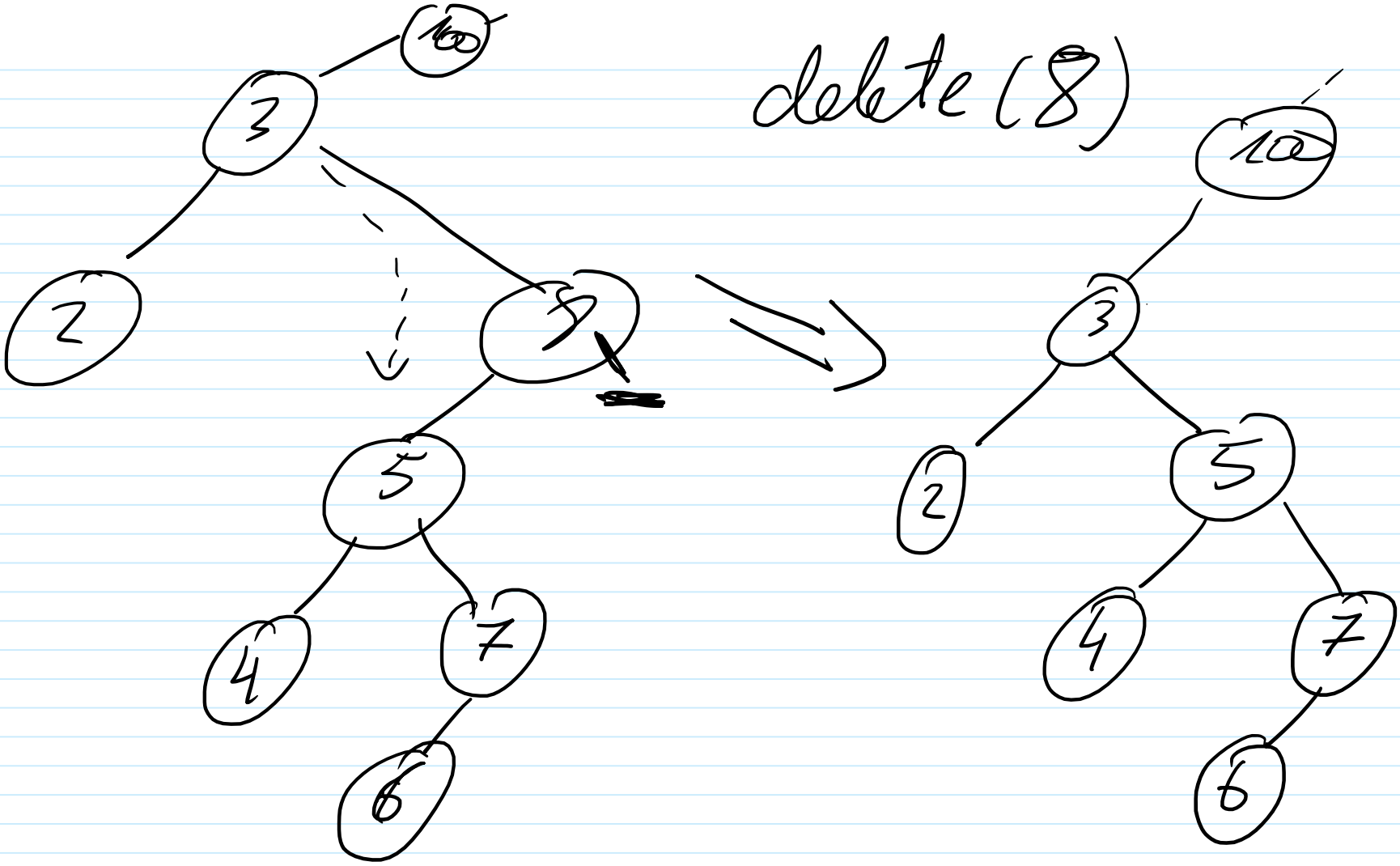


→ usuwamy k, zwalniamy pamięć wskaźnika na k, ustawiamy na NULL

(2) węzeł (k) ma tylko jedno dziecko.



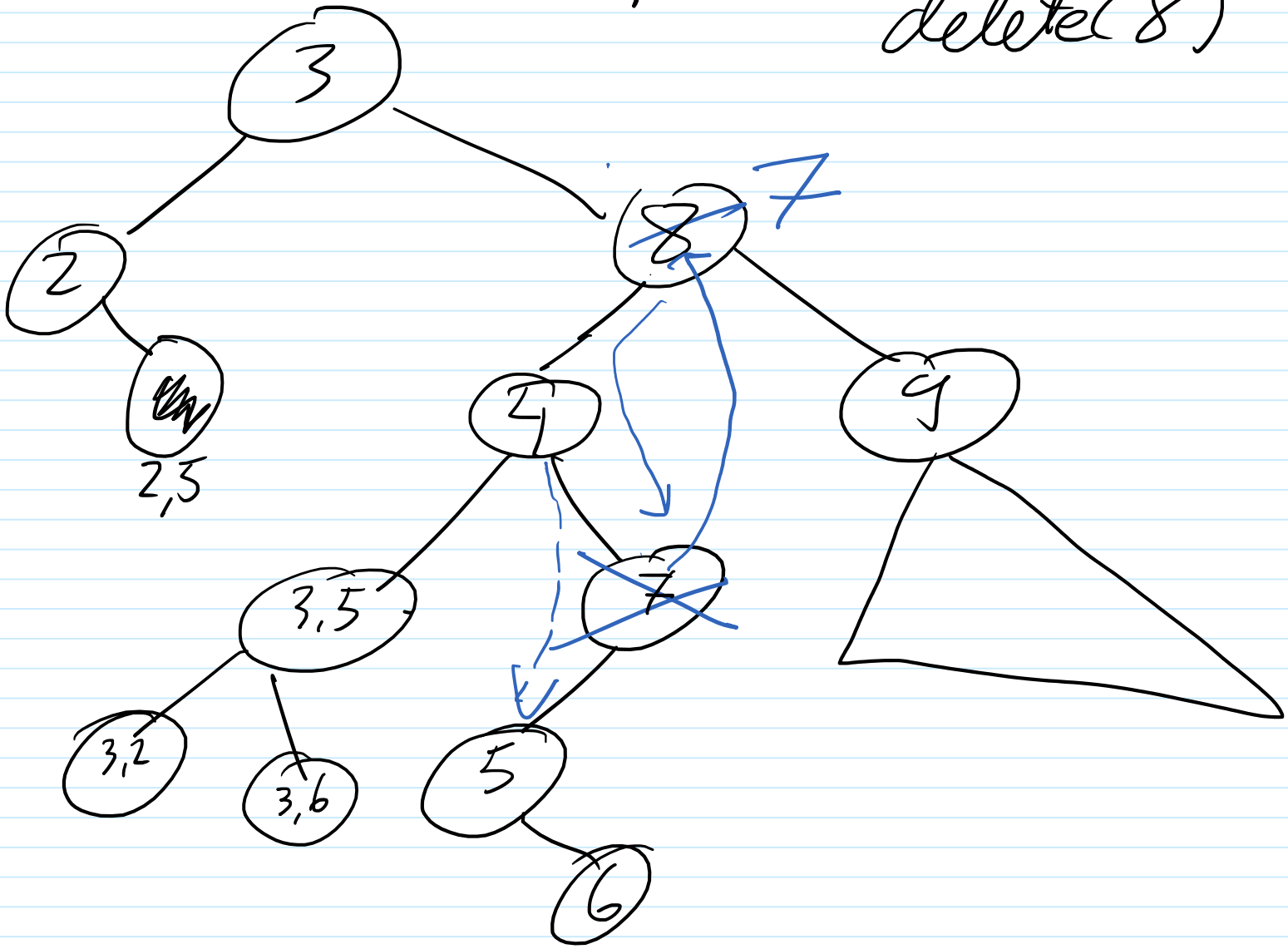
Ustawiamy ptr tak że wskazuje na jedyne dziecko (k) i usuwamy (k)



W Tasmanii żyją drzewem BST jest zachłanna.

3) Węzeł (8) ma dwoje dzieci

delete(8)



W tedy

① Znajdujemy największy element w lewym poddrzewie (L); niech to będzie (W)

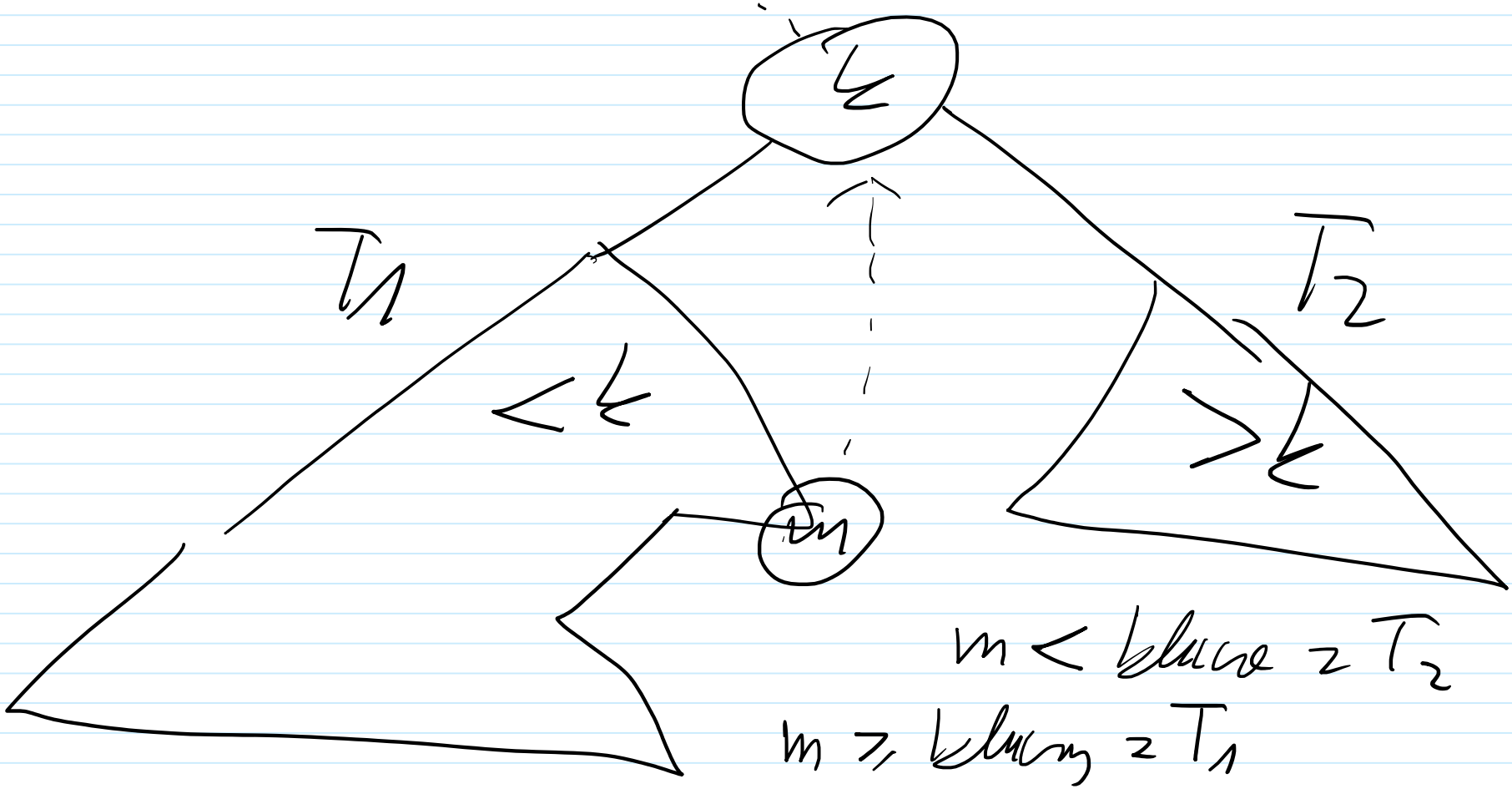
Uwaga (W) ma najwyżej jedno dziecko, bo nie może mieć prawego dziecka

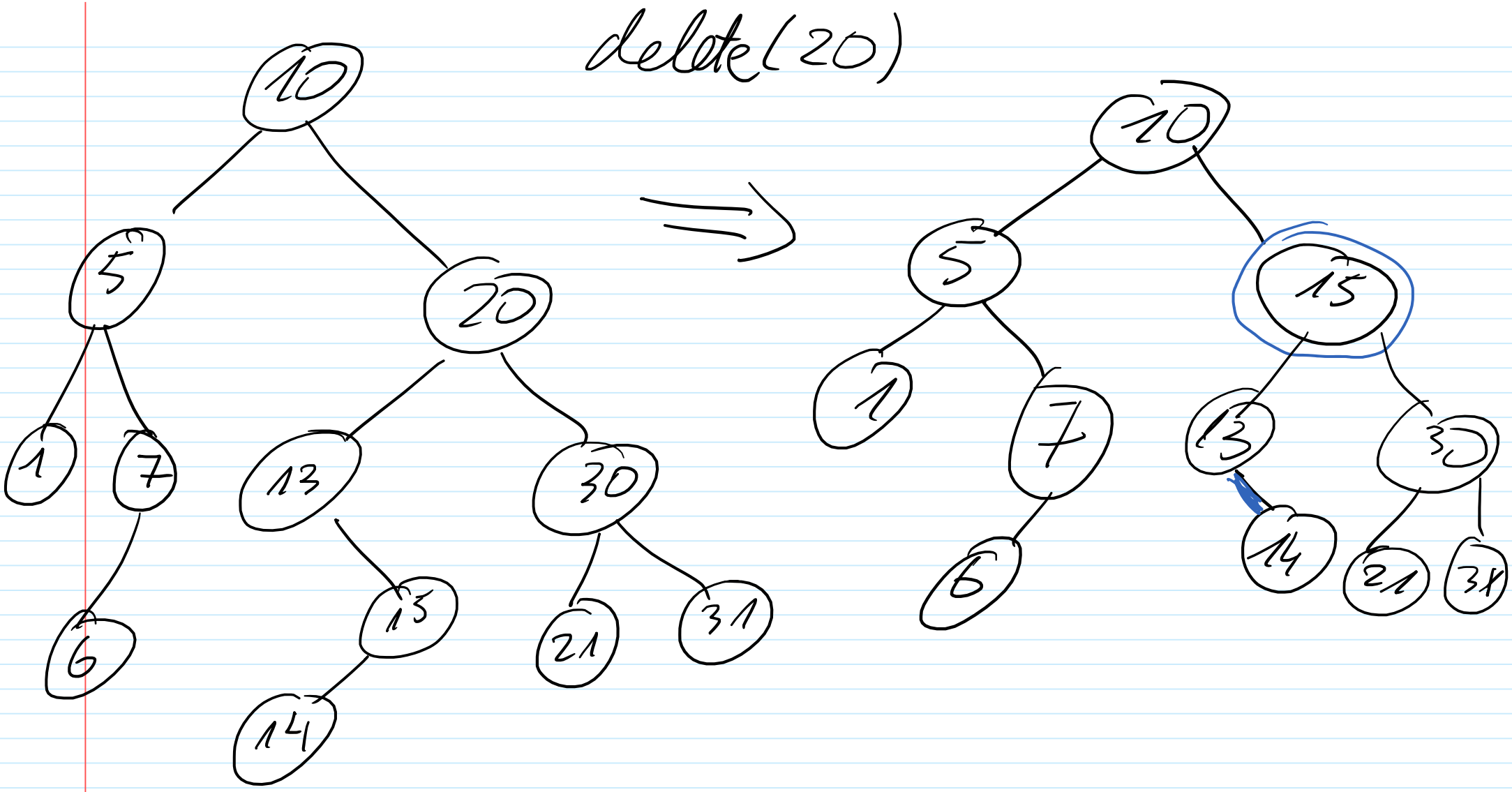
② Przepisujemy zawartość rekordu (W) do (L)

③ Usuwamy fizycznie rekord (W), bo to jest jeden z poprzednich przypadków

Usuwanie

Nowe drzewo dalej jest BST.




delete(20)
10
5
20
1
7
13
30
6
15
14
21
31

Kod

```
void _delete_key(Node ** r, int k){
	if((*r)==NULL)
		return;
	if((*r)->key<k)
		_delete_key(&((*r)->right),k);
	if((*r)->key>k)
		_delete_key(&((*r)->left),k);
	//teraz: *r!=NULL && (*r)->key==k
	if((*r)->left==NULL){
		Node *tmp=(*r)->right;
		delete(*r);
		*r=tmp;
		return;
	}
	if((*r)->right==NULL){
		Node *tmp=(*r)->left;
		delete(*r);
		*r=tmp;
		return;
	}
	Node **tmp=&((*r)->left);
	while((*tmp)->right!=NULL)
		tmp=&((*tmp)->right);
	(*r)->key=(*tmp)->key;
	_delete_key(tmp,(*tmp)->key);
}
```

```
void delete_key(int k){
  _delete_key(&root,k);
}
```

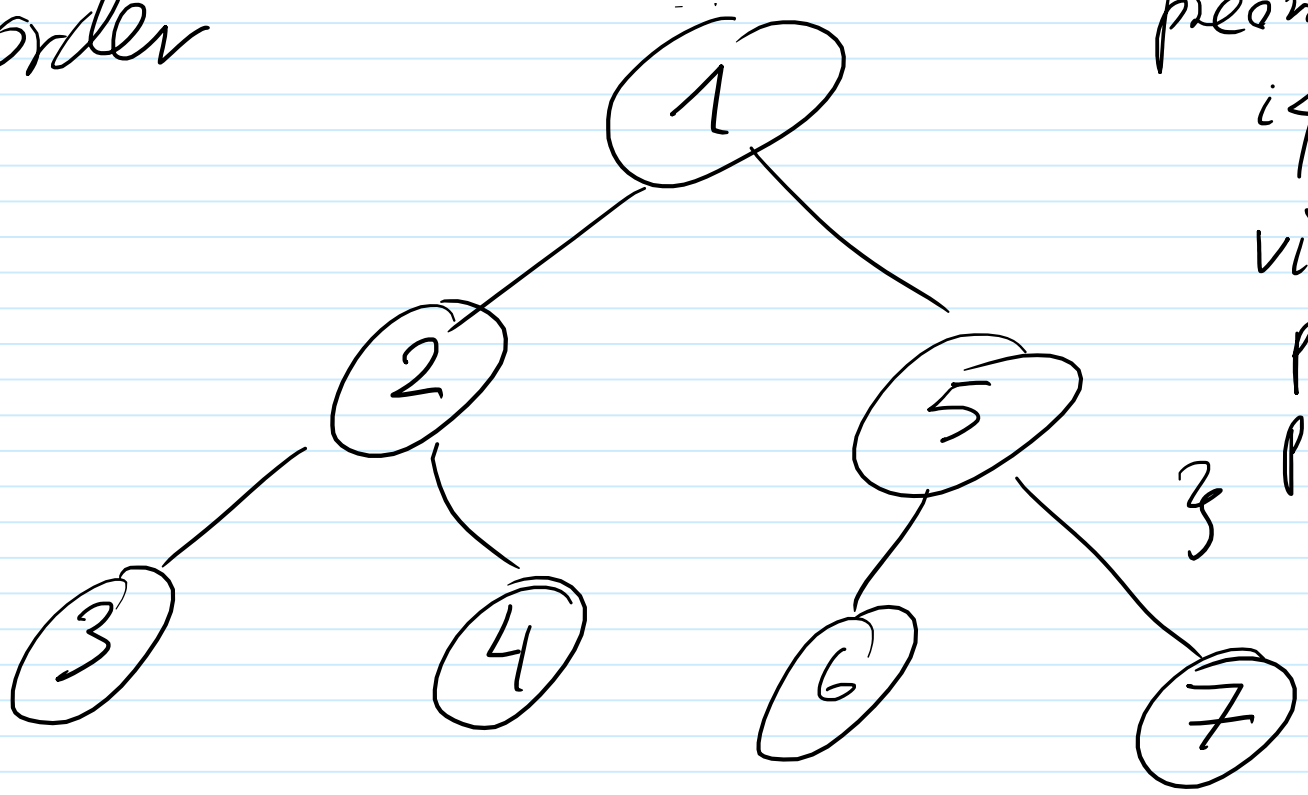
Warto ten zaimplementowany kod przeanalizować, ale najważniejsze jest zrozumienie logiki operacji delete().

Przeglądanie drzewa

Mamy cztery naturalne sposoby przeglądania drzewa

- pre order
- in order
- post order
- level order.

Pre order



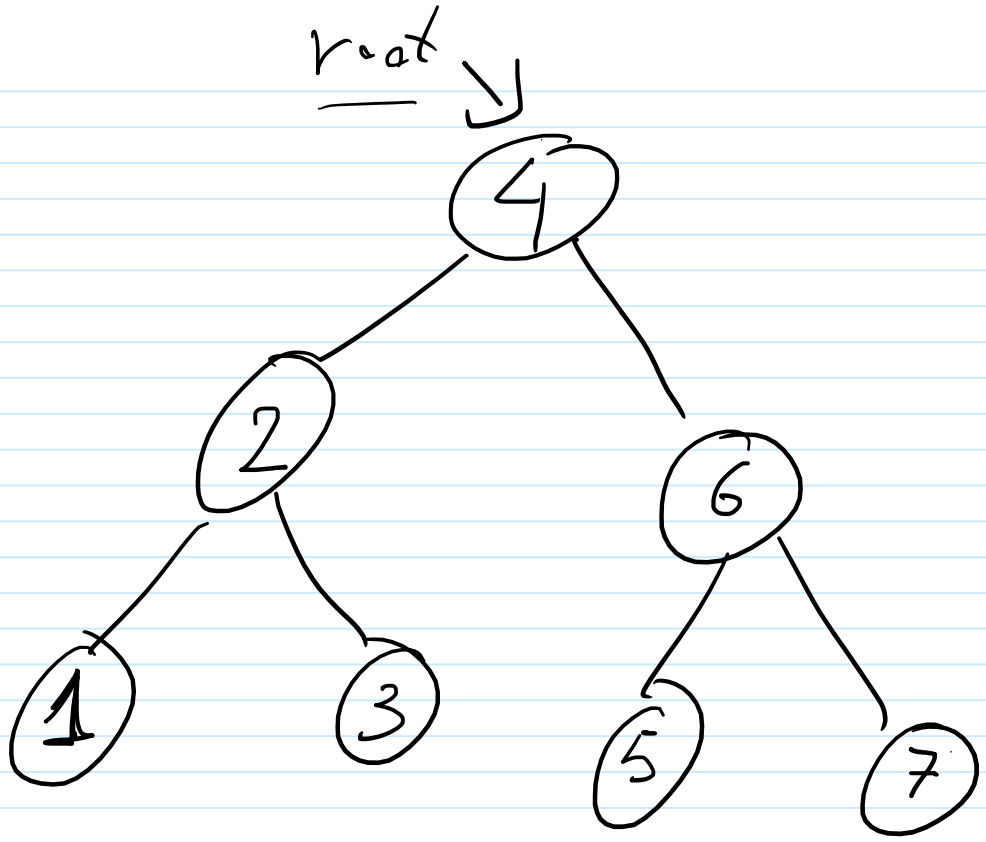

```
preorder(Node *r){
  if(r==NULL)
      return;
  visit(r);
  preorder(r->left);
  preorder(r->right);
};
```

gdzie visit()
to np.
```
visit(Node *r){
  print(r->key);
}
```

Najpierw odwiedź korzeń a potem, rekurencyjnie, lewe i prawe podrzewo

In order

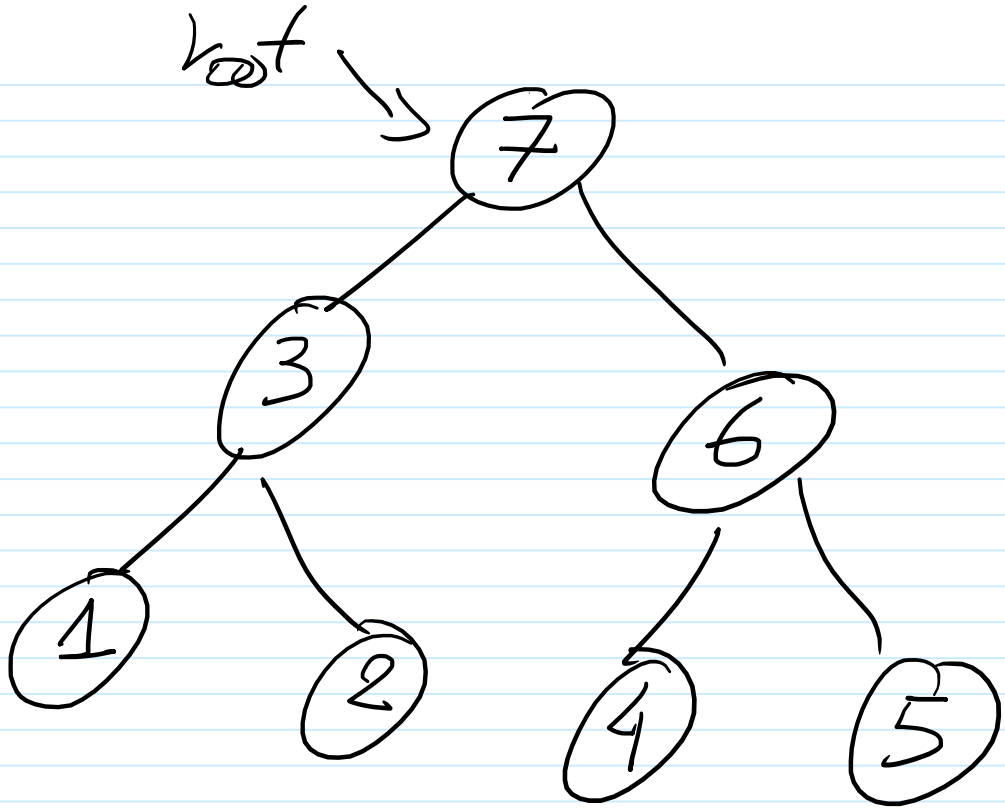
```
inorder(Node *r){
    if(r==NULL)
        return;
    inorder(r->left);
    visit(r);
    inorder(r->right);
}
```



Kolejność odwiedzania wierzchołków jest zgodna z wielkością elementów w drzewie BST o takiej strukturze.

Postorder

```
postorder(Node *r){
  if(r == NULL)
      return;
  postorder(r->left);
  postorder(r->right);
  visit(r);
}
my_visit(r){
  delete(r);
}
```



Postorder można wykorzystać do usuwania drzewa.