# Algorytmy i struktury danych - Drzewo BST

Marcin Żurowski

16 stycznia 2020

# Drzewo BST

```
STRUCT-TREE T
INIT(T) /*inicjuje zmienne*/
TREE-INSERT(T,15)
TREE-INSERT(T,5)
TREE-INSERT(T,16)
TREE-INSERT(T,3)
TREE-INSERT(T,12)
TREE-INSERT(T,20)
TREE-INSERT(T,10)
TREE-INSERT(T,6)
TREE-INSERT(T,7)
TREE-INSERT(T,13)
TREE-INSERT(T,18)
TREE-INSERT(T,23)
```

# Drzewo BST

```
WRITE(T) //
//    23
//   20
//     18
// 16
//15
//    13
//   12
//    10
//       7
//      6
// 5
//  3
```

# Drzewo BST

```
WRITE(TREE-SEARCH-RECURSIVE(T,13)) //13
WRITE(TREE-SEARCH-ITERATIVE(T,13)) //13
WRITE(TREE-SUCCESSOR(TREE-SEARCH-RECURSIVE(T,5))) //6
WRITE(TREE-SUCCESSOR(TREE-SEARCH-RECURSIVE(T,13)))
//15
```

# Drzewo BST

```
TREE-DELETE(T,13)
WRITE(T) //
//    23
//   20
//     18
// 16
//15
//   12
//     10
//       7
//      6
// 5
//   3
```

# Drzewo BST

```
TREE-DELETE(T,16)
WRITE(T)) //
//   23
// 20
//   18
//15
//   12
//     10
//       7
//      6
// 5
//   3
```

# Drzewo BST

```
TREE-DELETE(T,5)
WRITE(T) //
//   23
// 20
//   18
//15
//   12
//     10
//       7
// 6
//   3
CLEAR(T) /*zwalnia pamięć*/
WRITE(T) //
```

# Drzewo BST - funkcje i procedury

```
STRUCT-TREE
 root = NIL
STRUCT-NODE
 key = NIL
 p = NIL
 left = NIL
 right = NIL
TREE-SEARCH-RECURSIVE(T,k)
TREE-SEARCH-ITERATIVE(T,k)
TREE-SUCCESSOR(x)
TREE-INSERT(T,k)
TREE-DELETE(T,k)
```

# Zadania

Napisz algorytm który:

1. `TREE-MINIMUM`
2. `TREE-MAXIMUM`
3. `TREE-PREDECESSOR`
4. Następnik najmniejszego (0.5 pkt. lub zad.5)
5. Poprzednik największego (0.5 pkt. lub zad.4)