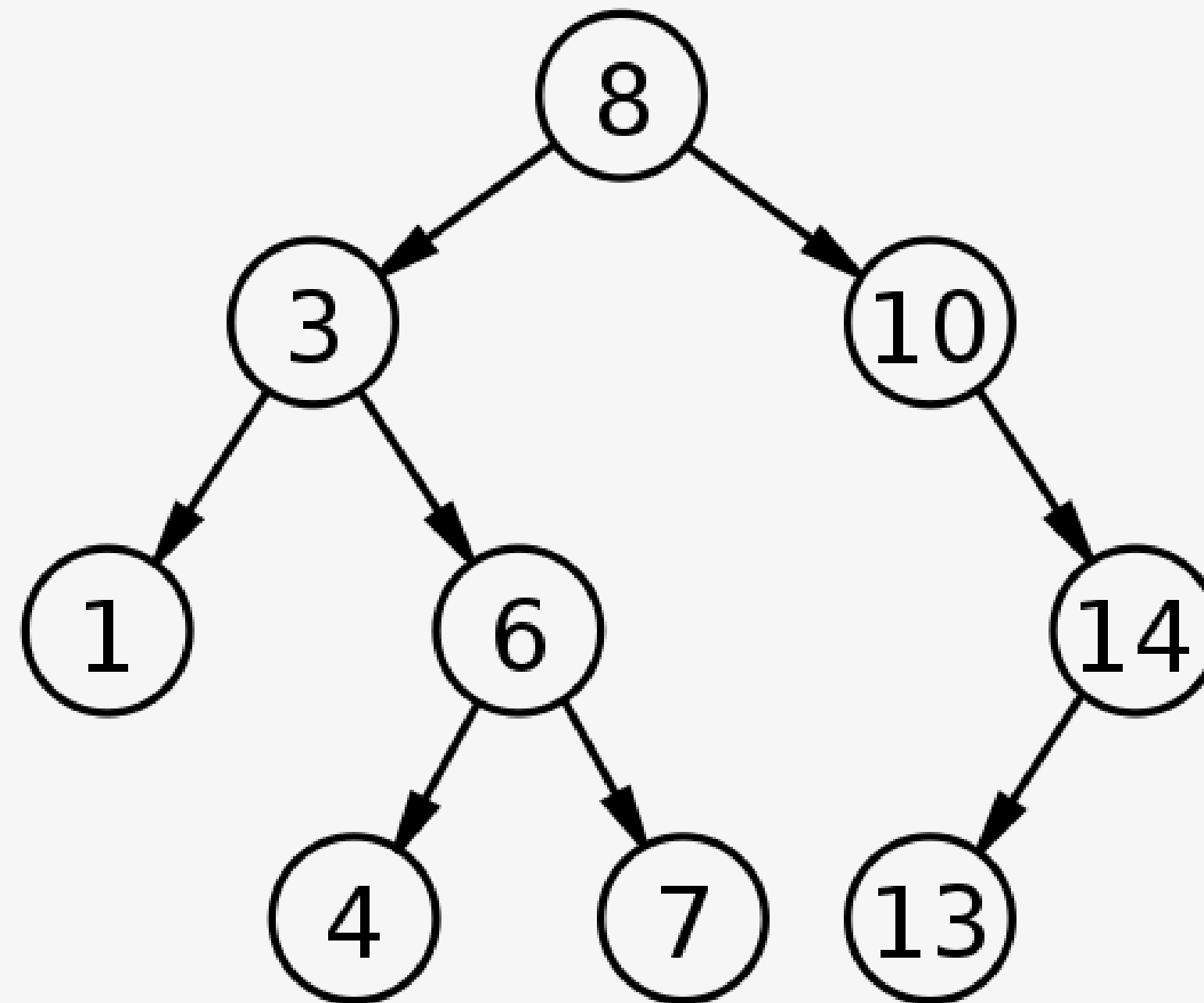
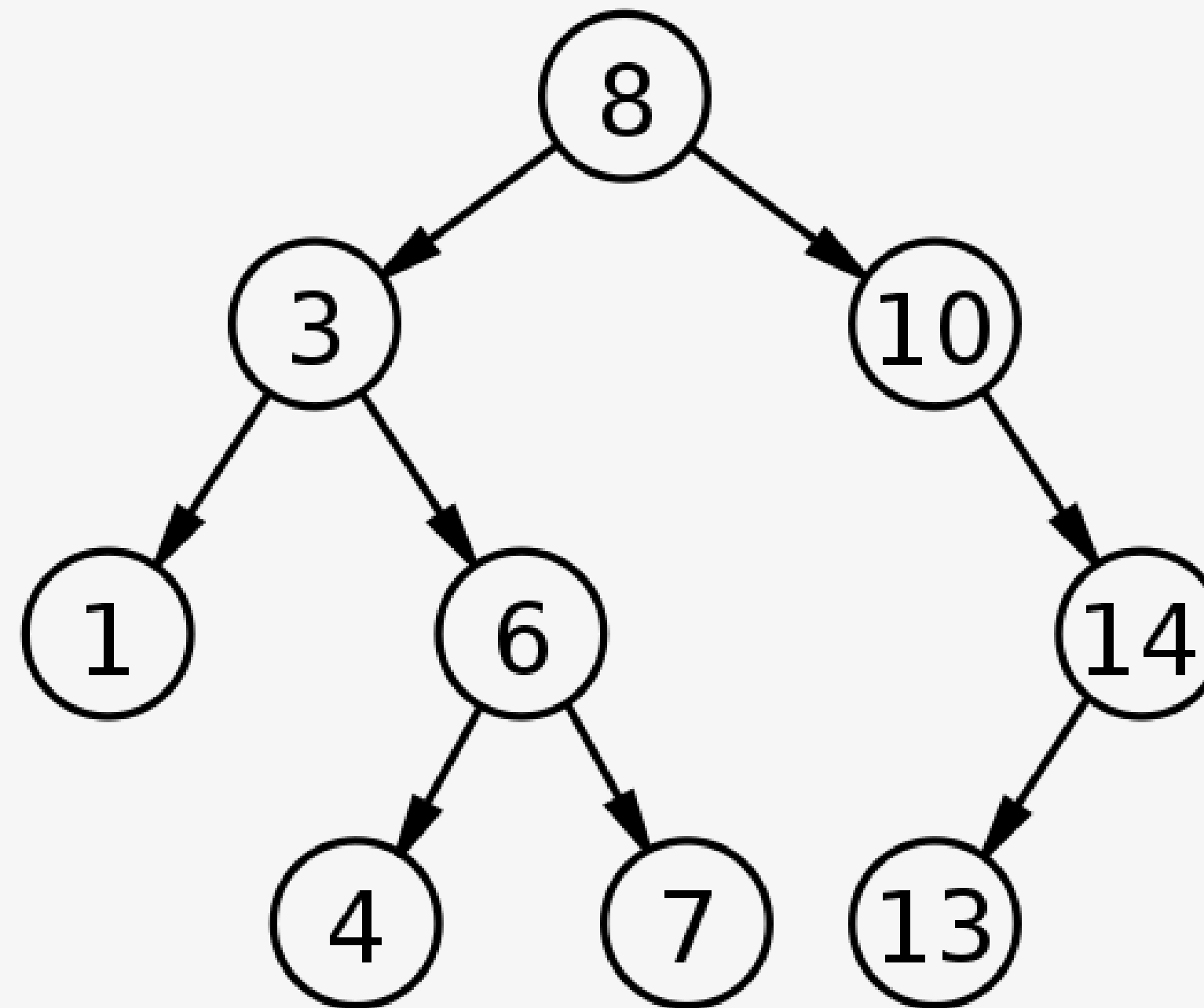


# BINARNE DRZEWO POSZUKIWAŃ

Magda Szafrńska







węzły/ wierzchołki  
(ang. NODES)

# Binary Search Tree

Czym jest  
BST

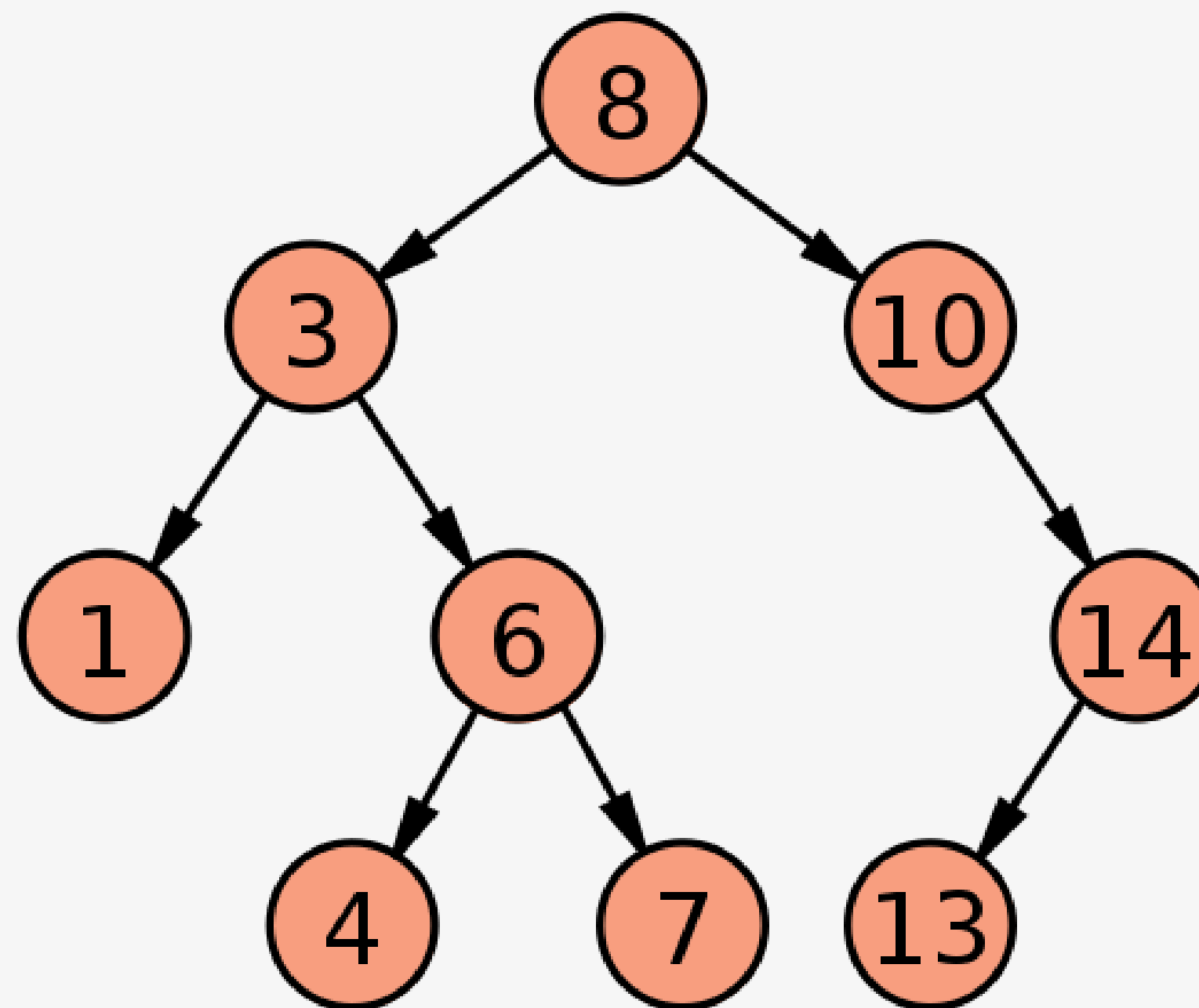
Zasada  
działania

Program  
w C++

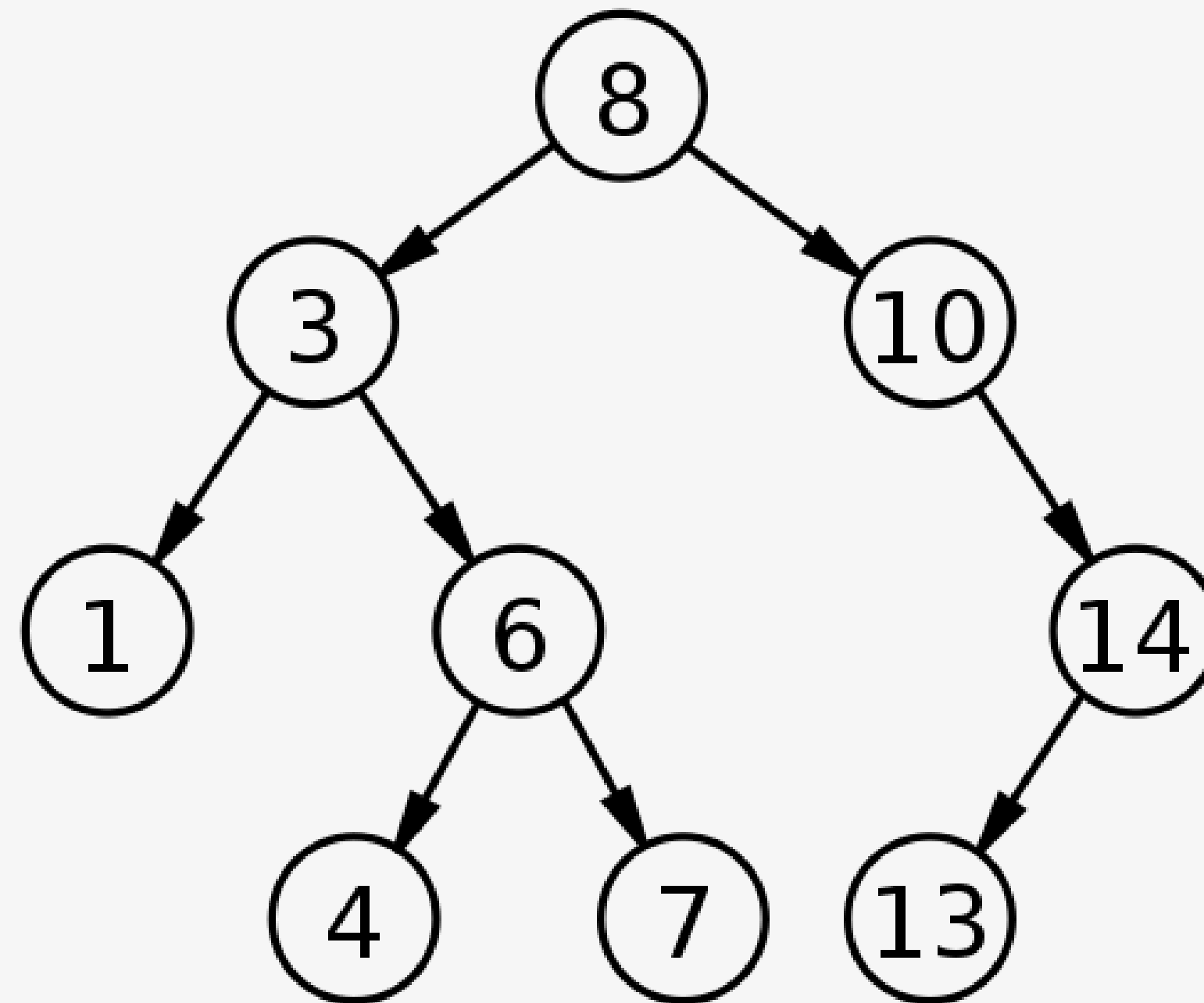
Implementacja

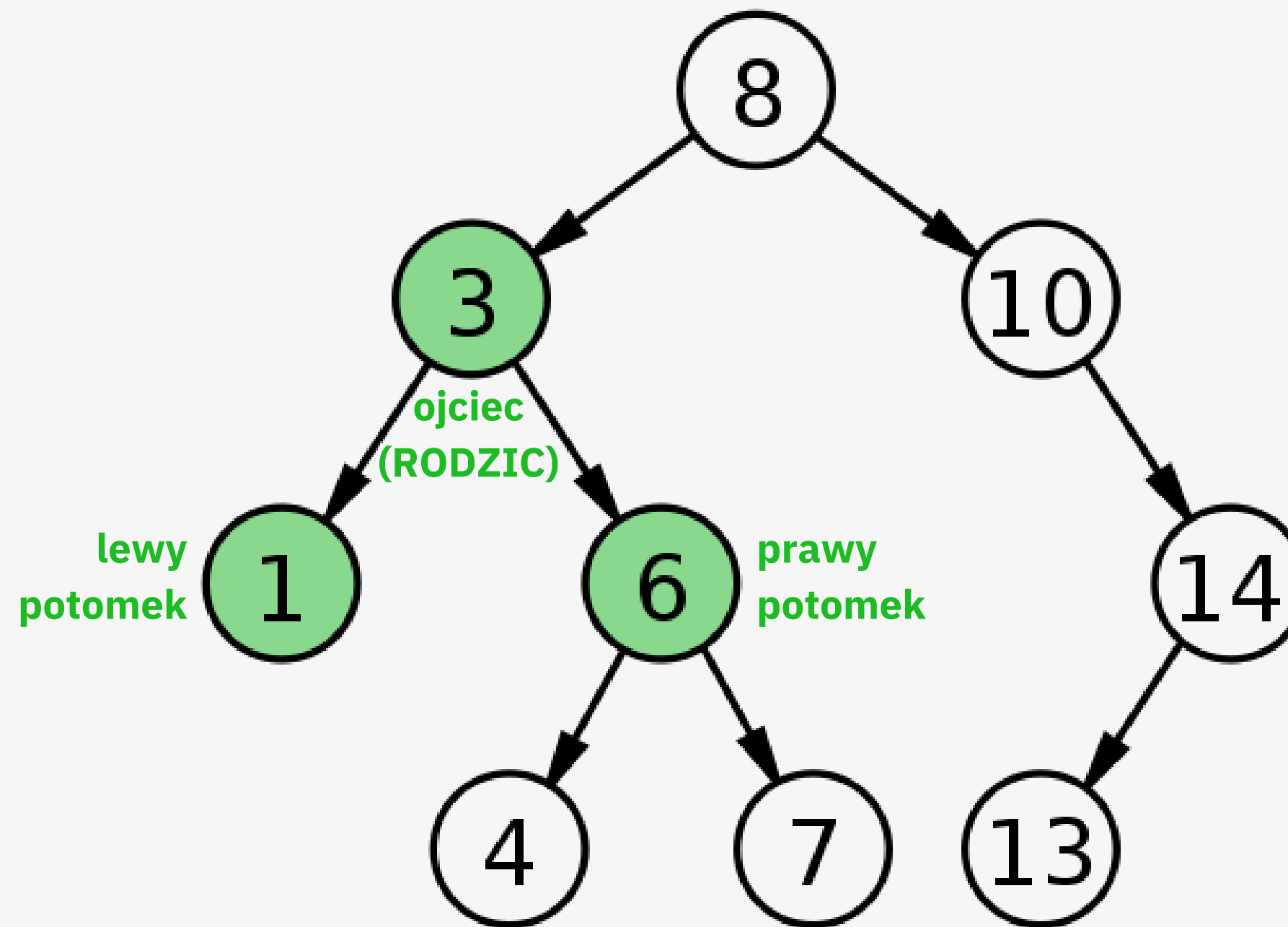
Złożoność  
obliczeniowa

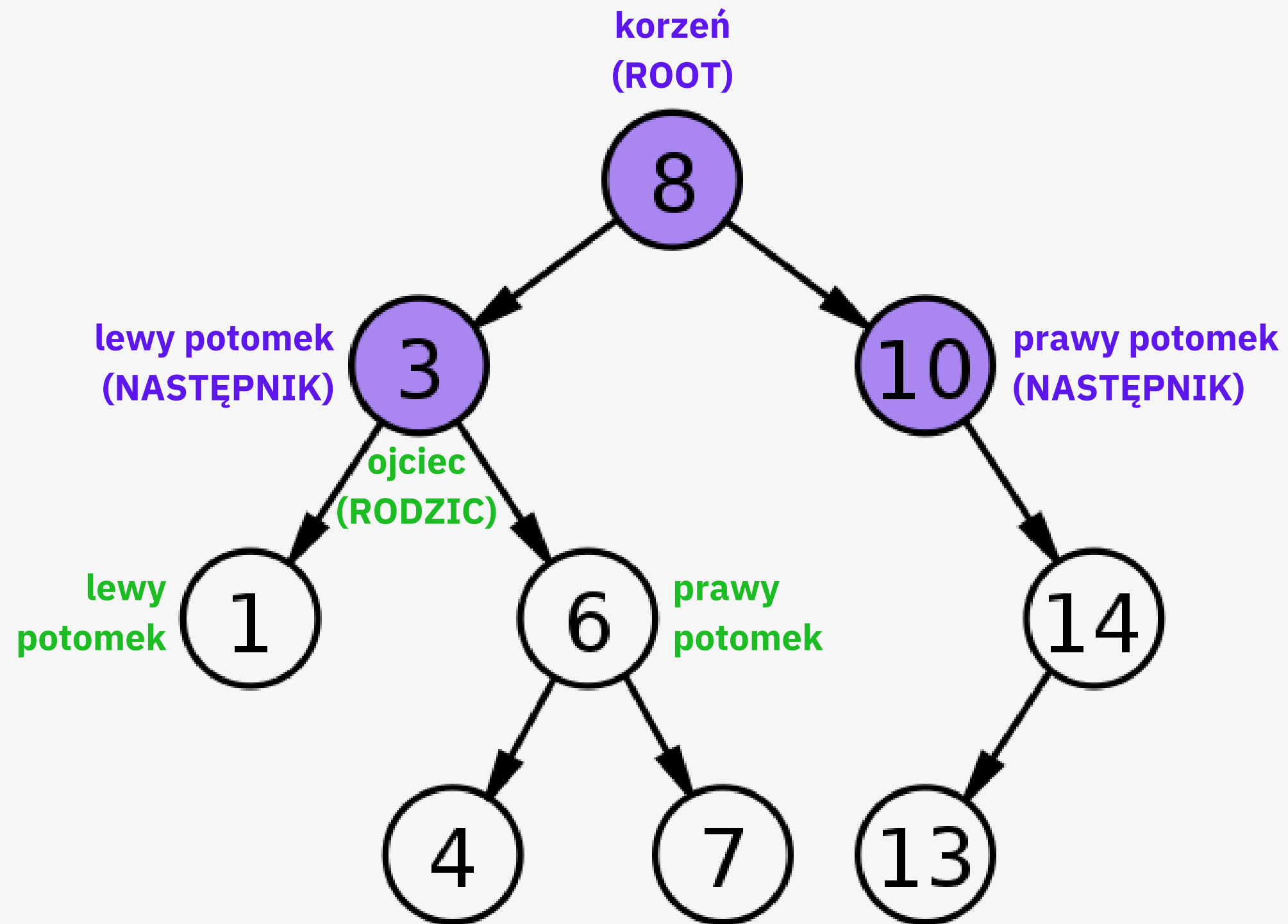
Wykorzystanie

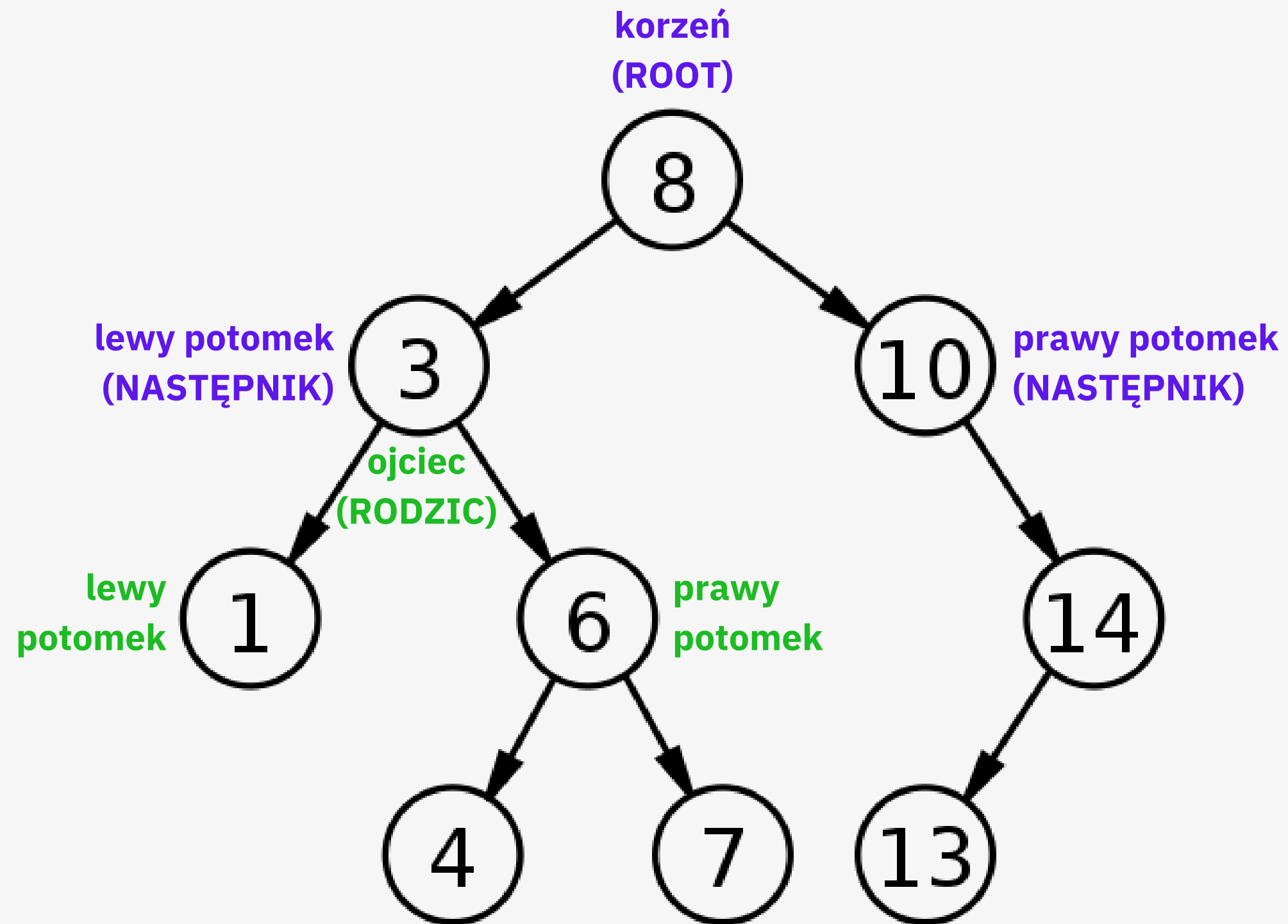


węzły/ wierzchołki  
(ang. NODES)

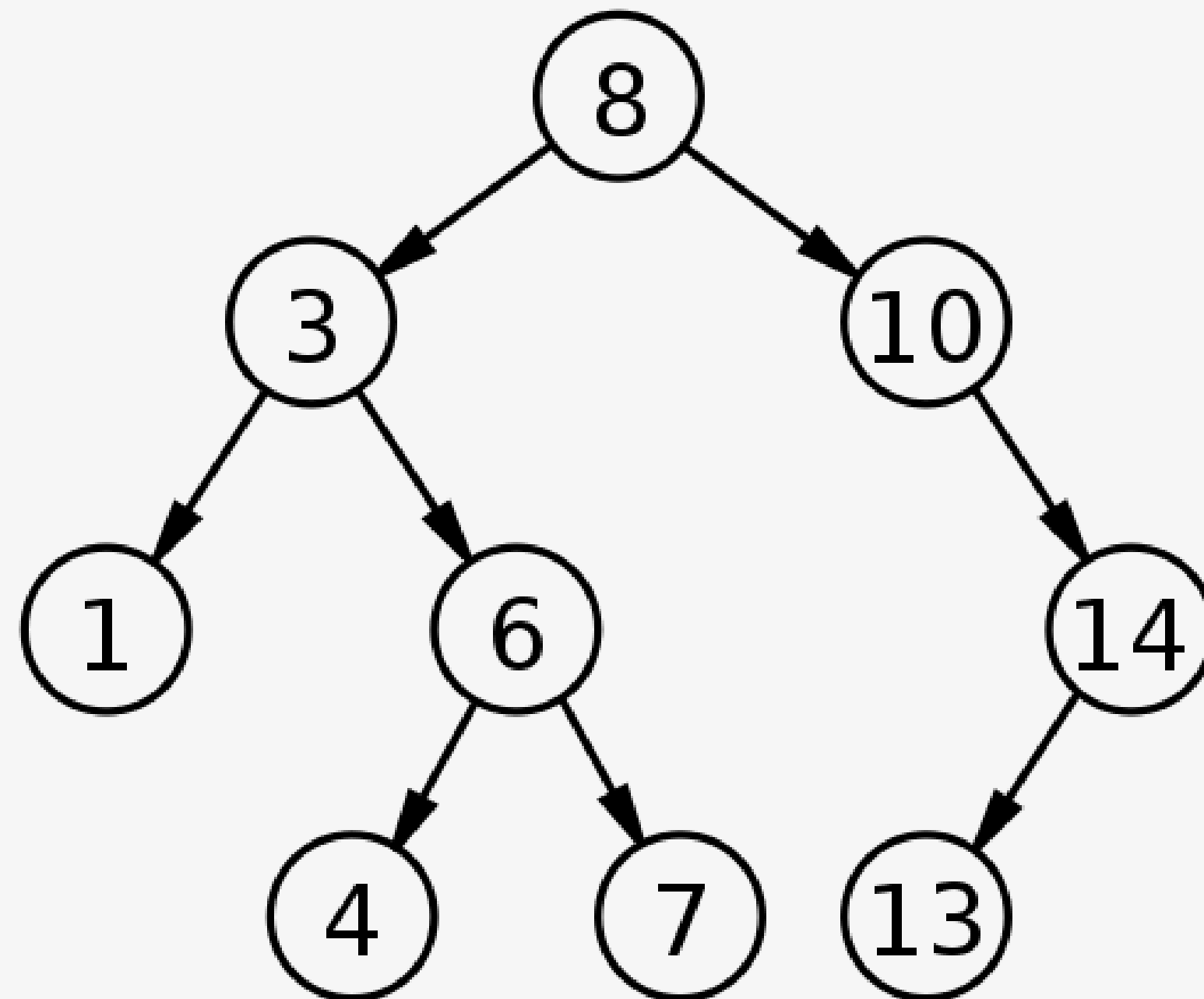




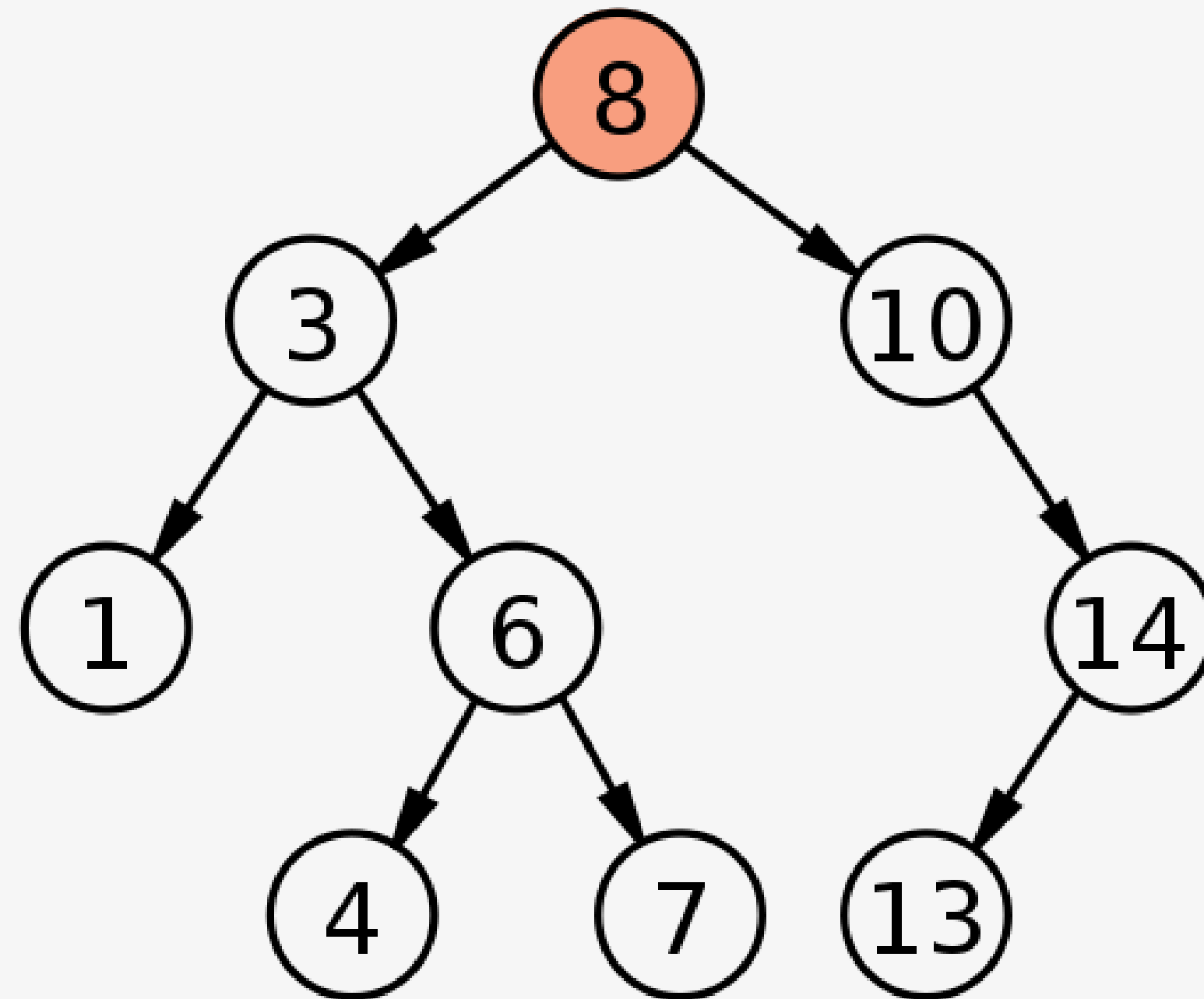






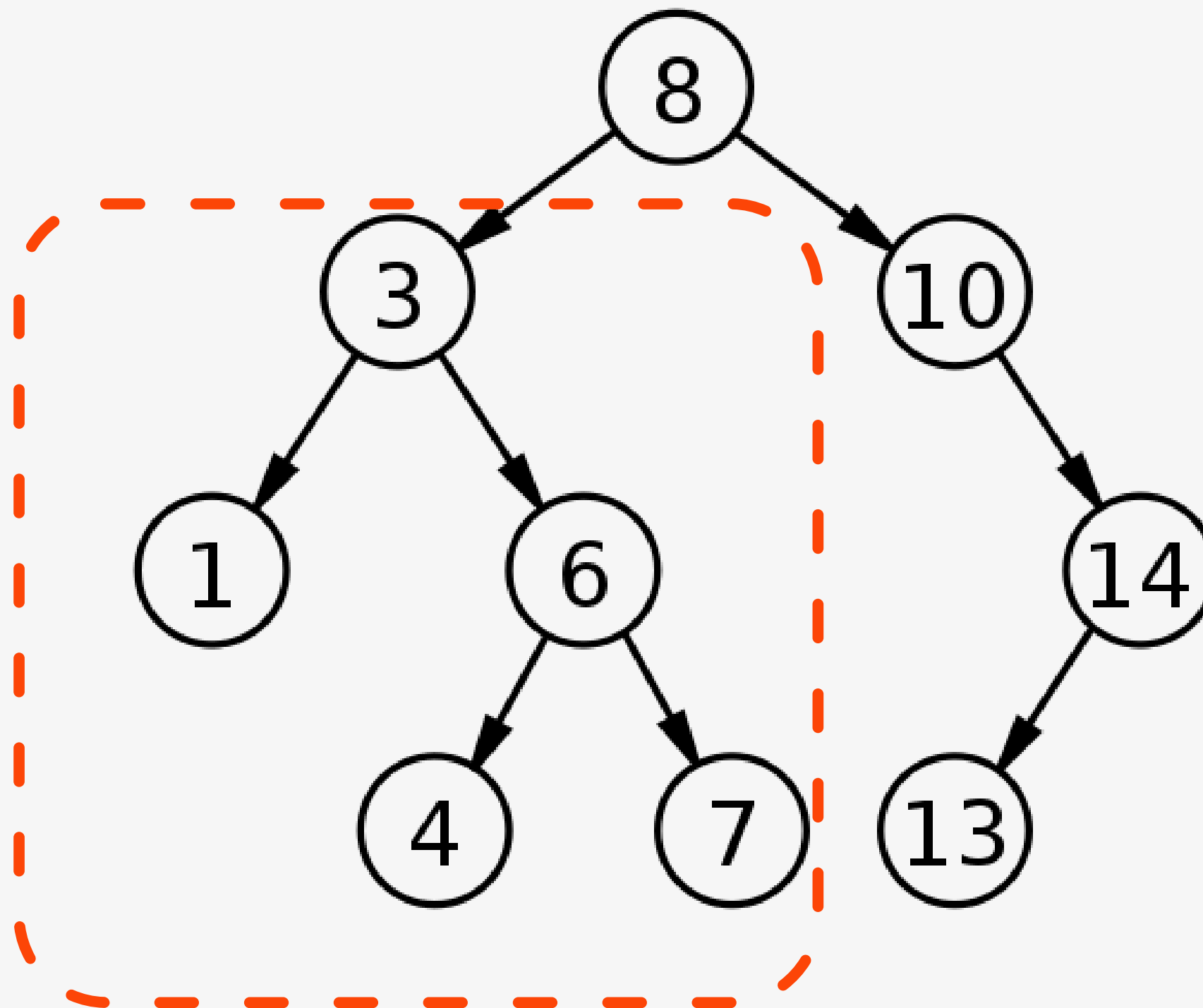


**Reguła drzewa binarnego**



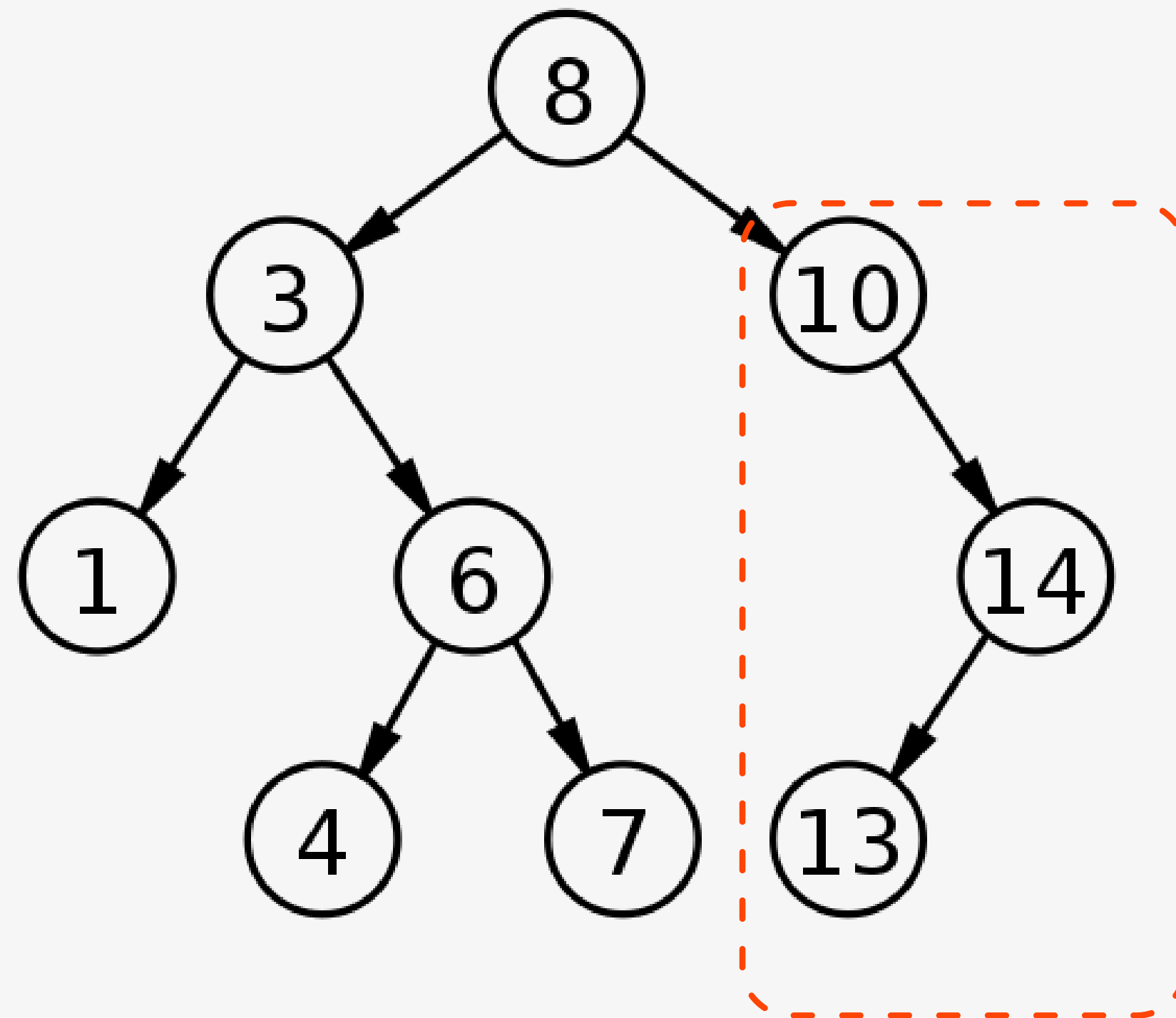
## Reguła drzewa binarnego

1. Pierwsza wartość trafia do **korzenia**.



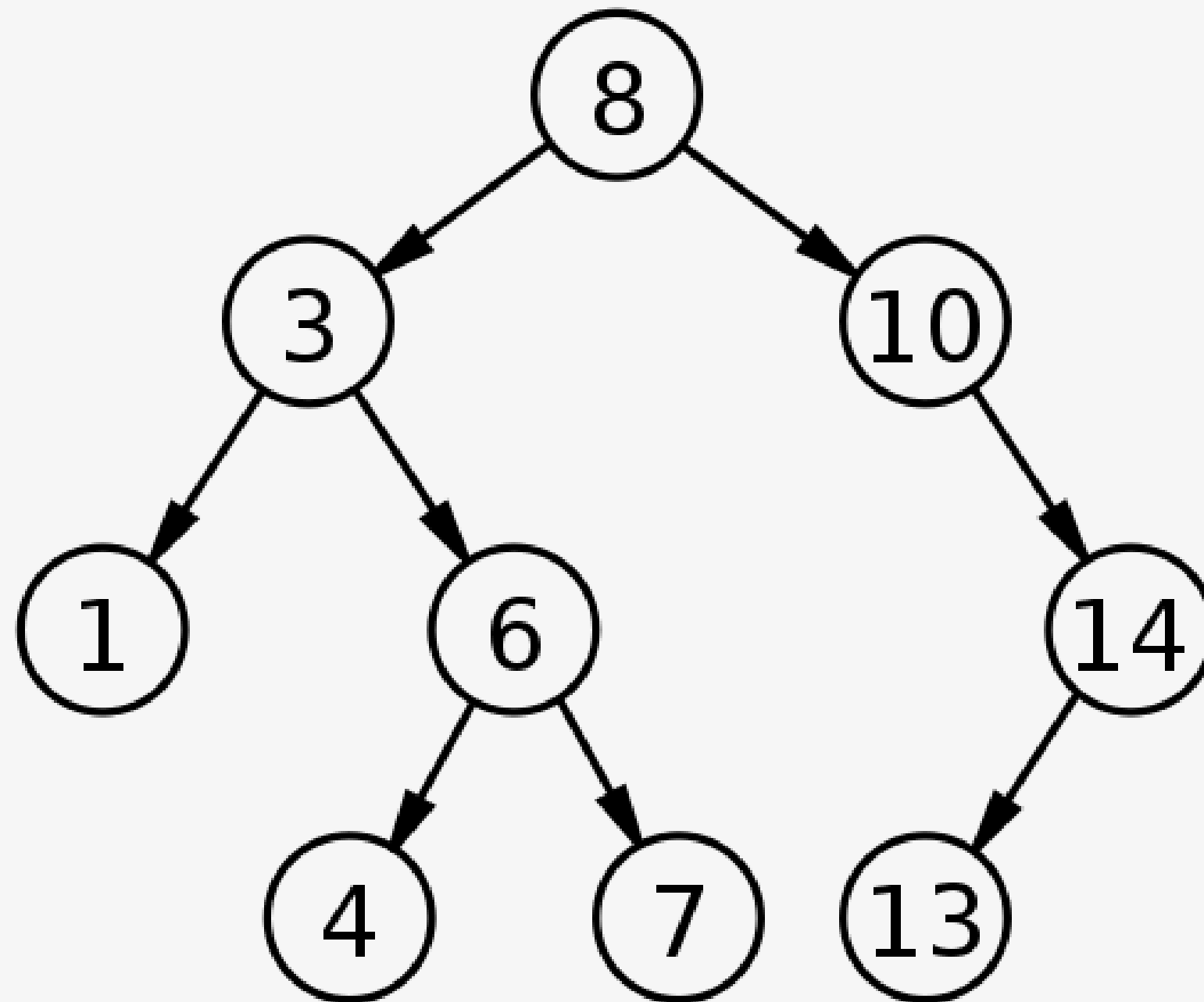
## Reguła drzewa binarnego

1. Pierwsza wartość trafia do korzenia.
2. Mniejsze elementy **na lewo**.



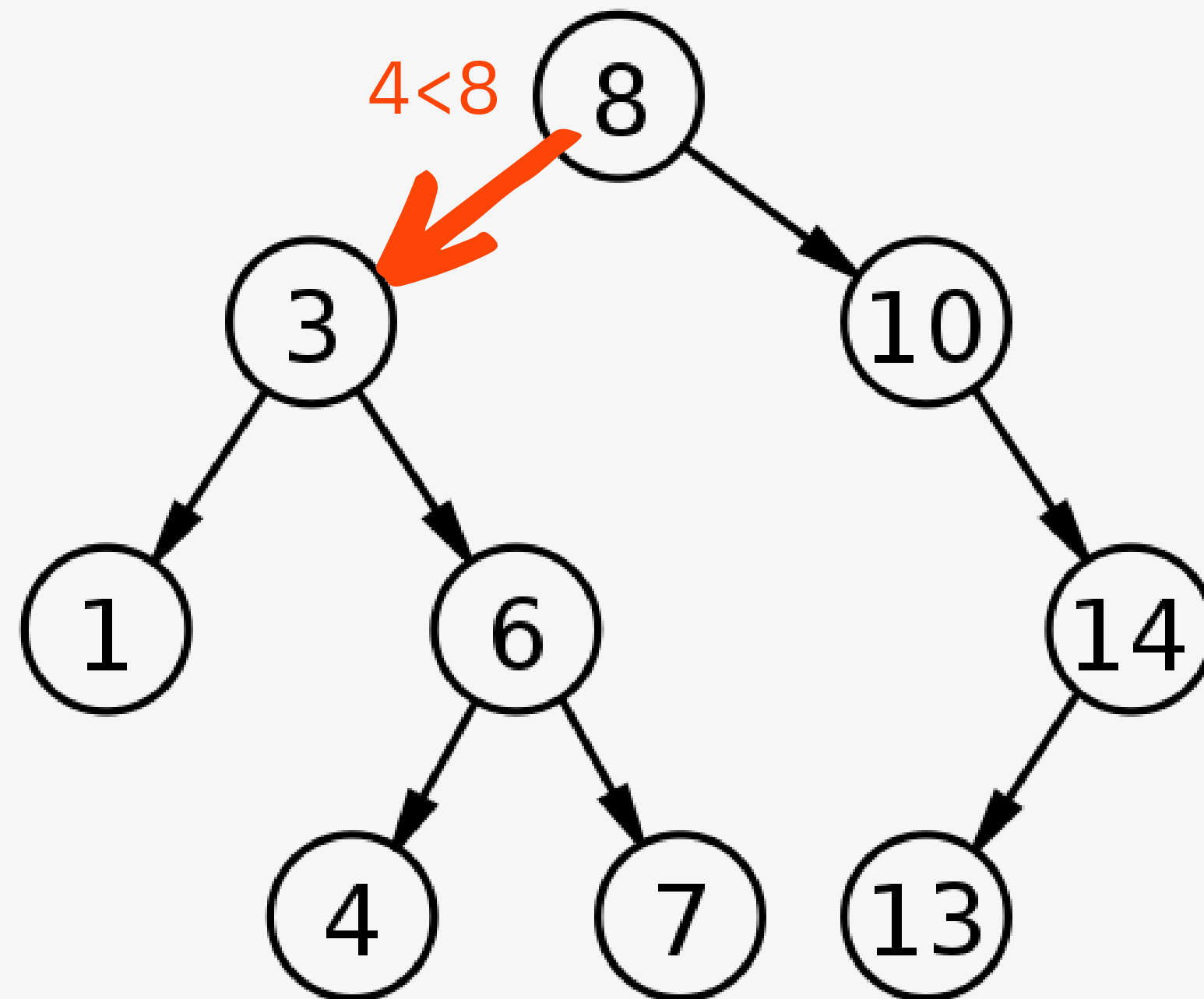
## Reguła drzewa binarnego

1. Pierwsza wartość trafia do korzenia.
2. Mniejsze elementy na lewo.
3. Większe **na prawo**.



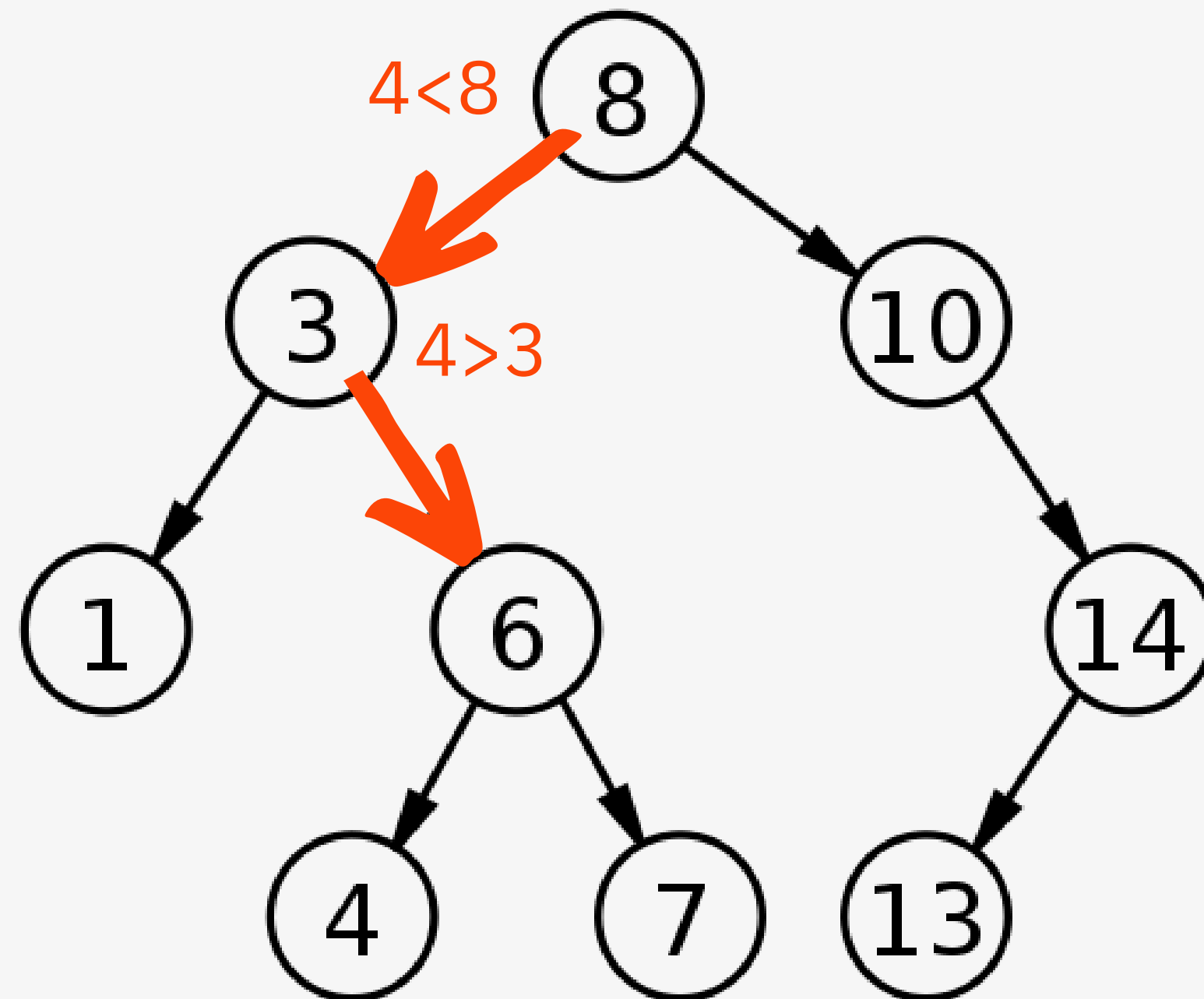
## Reguła drzewa binarnego

1. Pierwsza wartość trafia do korzenia.
2. Mniejsze elementy na lewo.
3. Większe na prawo.



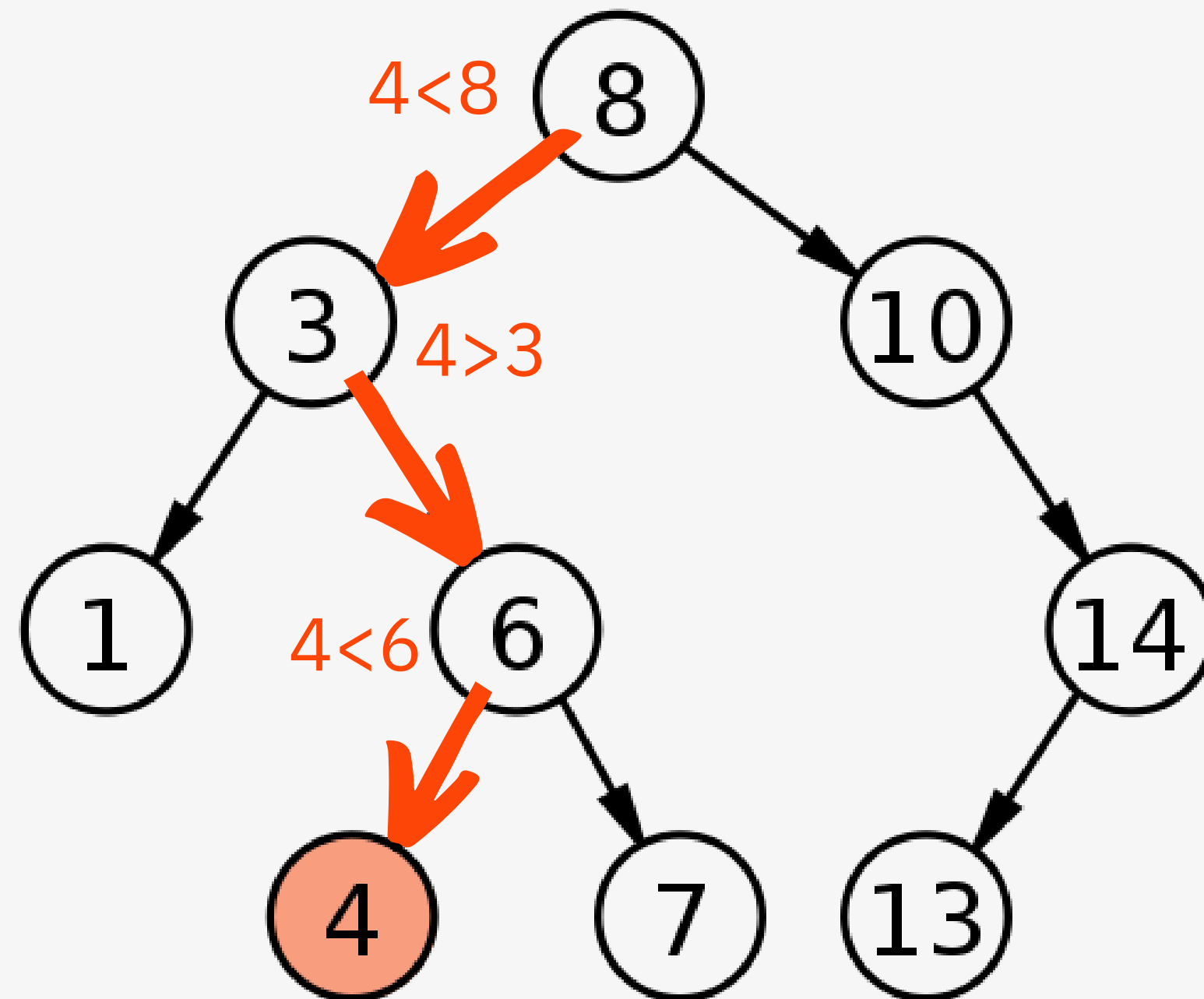
## Reguła drzewa binarnego

1. Pierwsza wartość trafia do korzenia.
2. Mniejsze elementy na lewo.
3. Większe na prawo.



## Reguła drzewa binarnego

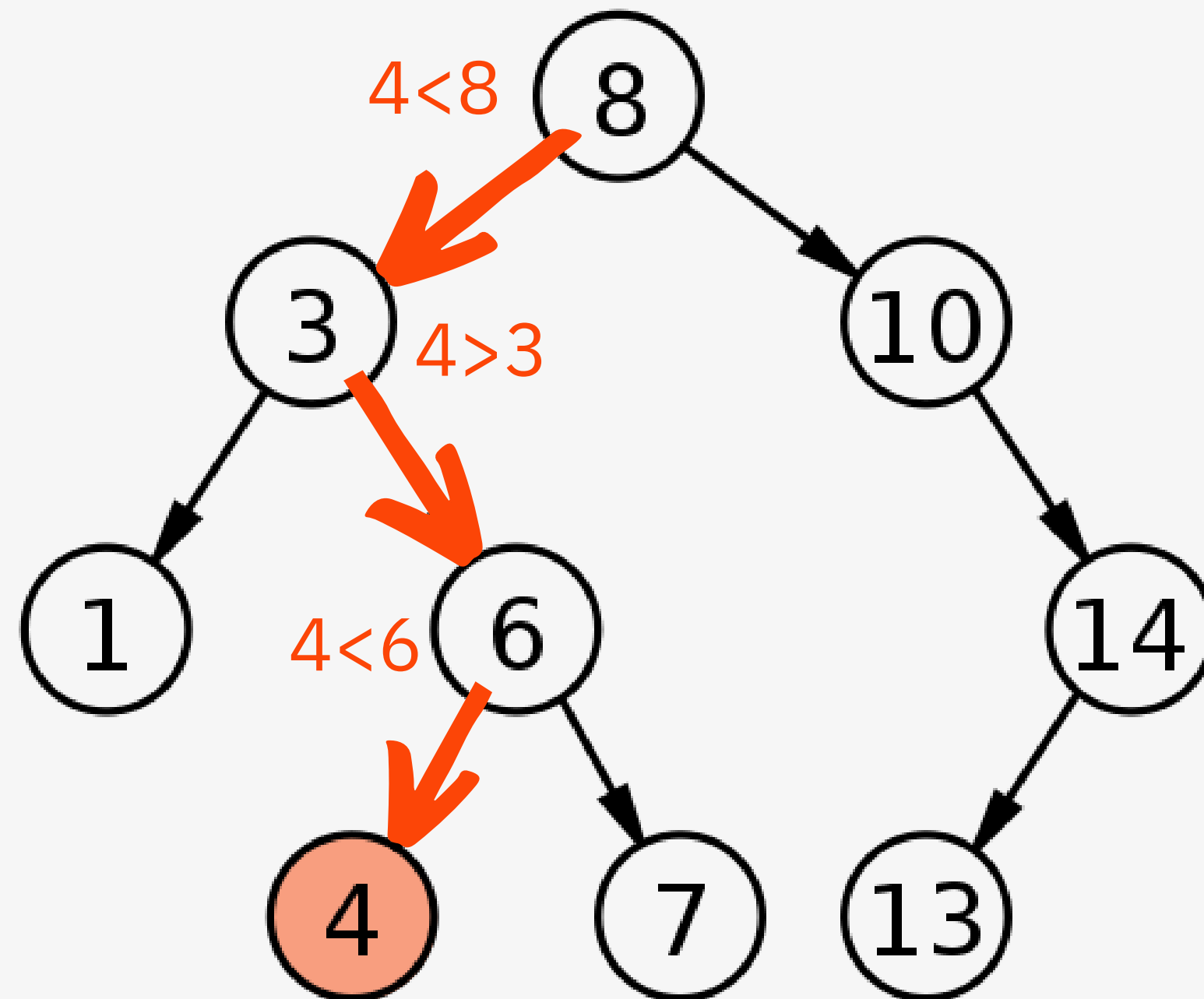
1. Pierwsza wartość trafia do korzenia.
2. Mniejsze elementy na lewo.
3. Większe na prawo.



## Reguła drzewa binarnego

1. Pierwsza wartość trafia do korzenia.
2. Mniejsze elementy na lewo.
3. Większe na prawo.

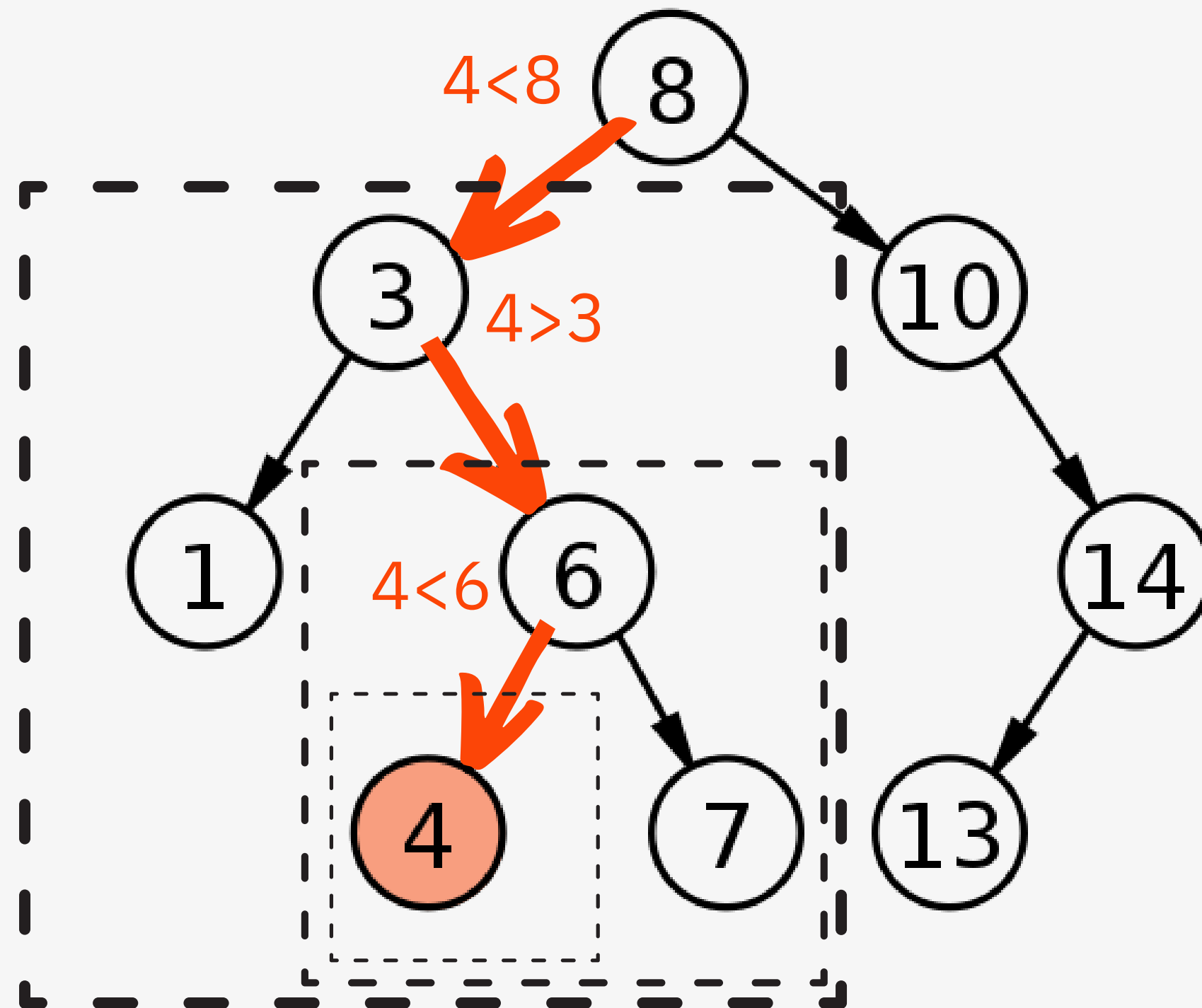




Reguła drzewa binarnego



“Dziel i zwyciężaj”



## Reguła drzewa binarnego



“Dziel i zwyciężaj”

# Binary Search Tree

Czym jest  
BST

Zasada  
działania

Program  
w C++

Implementacja

Złożoność  
obliczeniowa

Wykorzystanie

```
C:\Users\Modify\Downloads\AHNS\Algorytmy\PRESENTATION\BinarySearchTree\Debug\BinarySearchTree.exe

      _
     _
    _
   _
  _
 _
_

-----
EMULATOR DRZEWA BINARNEGO
-----
1. ADD (dodaje element do drzewa)
2. FIND (szuka elementu w drzewie)
3. EXIT (zakończy ten program)
-----
WYBOR:
```

## Dodawanie elementu

```

47 void add()
48 {
49     int choosen_number;
50     cout << "Jaka liczbe dodac do drzewa: ";
51     cin >> choosen_number;
52
53     // ----- empty tree -----
54     if (empty_array[1] == true)
55     {
56         root = &data_array[1];
57         *root = choosen_number;
58         empty_array[1] = false;
59     }
60     else
61     // ----- not empty tree -----
62     {
63         bool found_place = false;
64         int node_nr = 1;
65
66         while (found_place == false)
67         {
68             if (empty_array[node_nr] == true)
69             {
70                 found_place = true;
71                 data_array[node_nr] = choosen_number;
72                 empty_array[node_nr] = false;
73             }
74             else if (choosen_number < data_array[node_nr])
75             {
76                 // ----- put to the LEFT -----
77                 node_nr = 2 * node_nr;
78             }
79             else
80             {
81                 // ----- put to the RIGHT -----
82                 node_nr = 2 * node_nr + 1;
83             }
84             if (node_nr > 15)
85             {
86                 cout << "Potrzebne bylyby wieksze drzewo!";
87                 Sleep(3000);
88                 found_place = true;
89             }
90         }
91     }
92 }
93

```

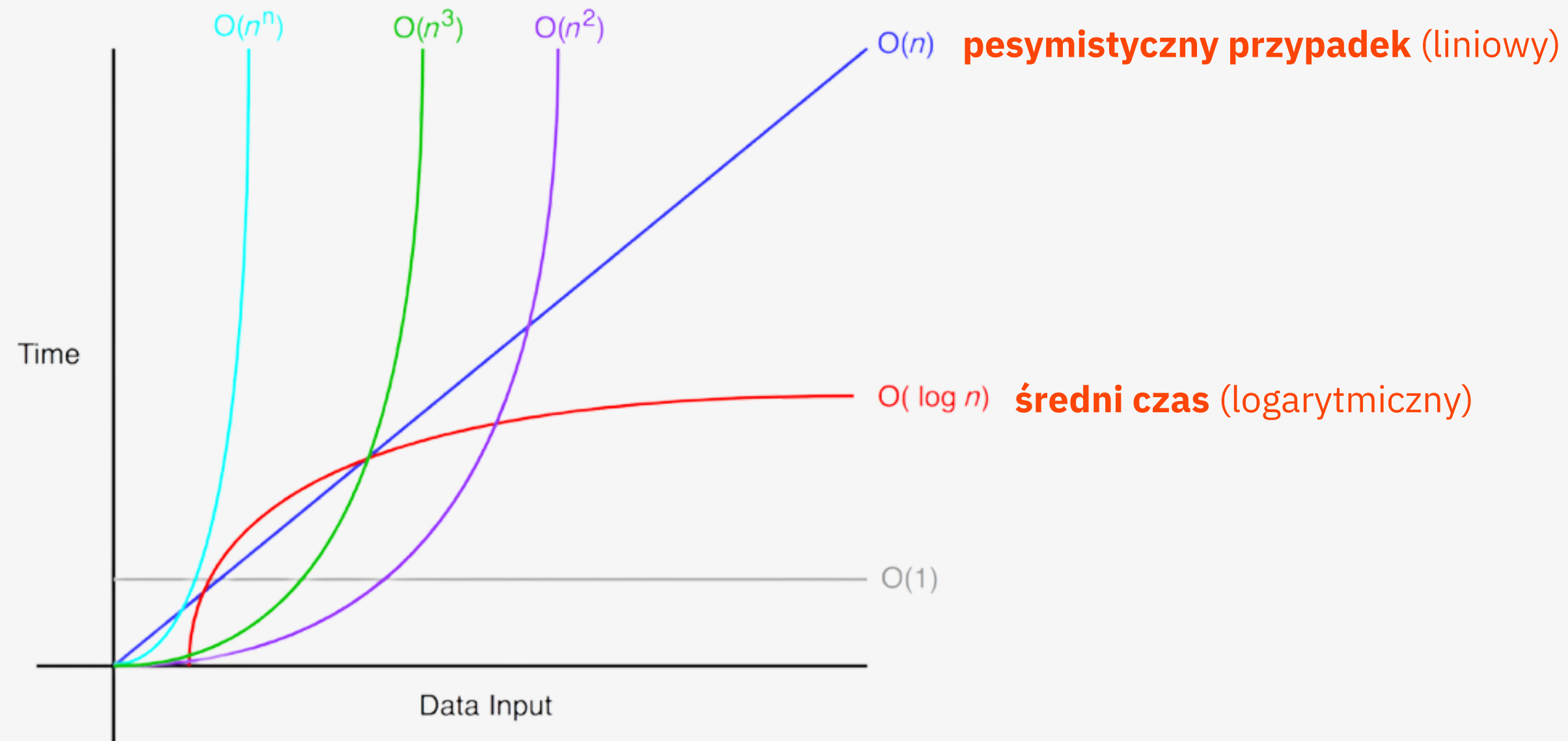
## Szukanie elementu

```

95 void find()
96 {
97     int number;
98     cout << "Jaka liczbe znalezc w drzewie: ";
99     cin >> number;
100
101     bool found = false;
102     int node_of_the_number = 1;
103
104     while (found == false)
105     {
106         if (number == data_array[node_of_the_number])
107         {
108             cout << "Znaleziono liczbe w wezle nr: " << node_of_the_number;
109             Sleep(3000);
110             found = true;
111         }
112         else if (number < data_array[node_of_the_number])
113         {
114             // ----- go to the LEFT -----
115             node_of_the_number = 2 * node_of_the_number;
116         }
117         else
118         {
119             // ----- go to the RIGHT -----
120             node_of_the_number = 2 * node_of_the_number + 1;
121         }
122         if (node_of_the_number > 15)
123         {
124             cout << "Nie znaleziono!";
125             Sleep(3000);
126             found = true;
127         }
128     }
129 }

```

## Złożoność wyszukiwania elementu w BST









# Źródła

1. Bhargava Aditya Y.: “Algorytmy, Ilustrowany przewodnik”: Wydawnictwo Helion, 2017, str. 203-206. ISBN 978-83-283-3445-8
2. Zelent M.: “Struktury danych: stos, kolejka, lista, drzewo binarne” [online] <https://miroslawzelent.pl/kurs-c++/struktury-danych-stos-kolejka-lista-drzewo-binarne/> [dostęp: 24 stycznia 2022]