بسم الله الرحمن الرحیم

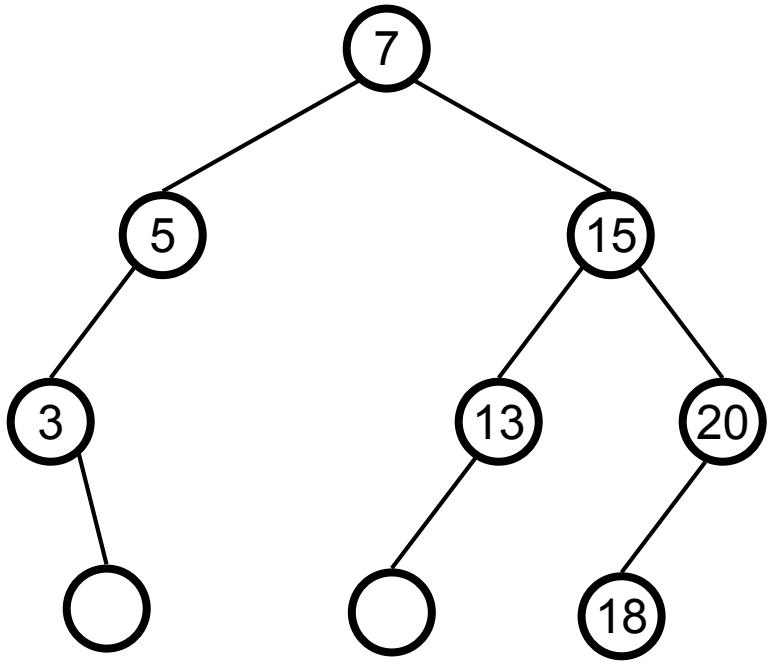
ساختمان‌های داده

جلسه ۱۷

مجتبی خلیلی
دانشکده برق و کامپیوتر
دانشگاه صنعتی اصفهان

مثال

- BST؟

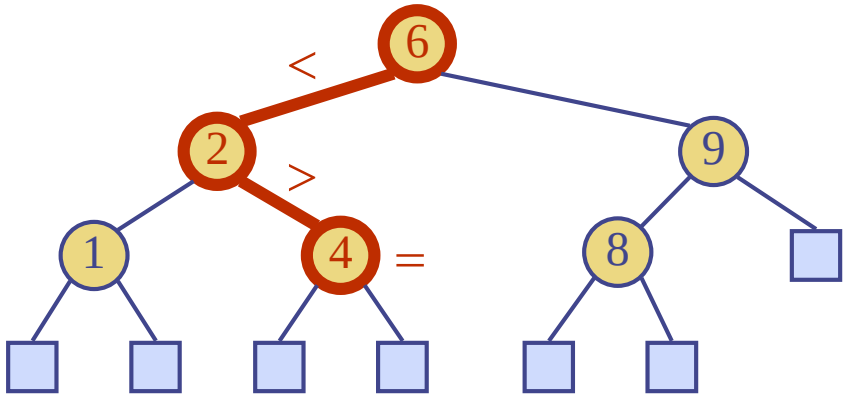


Search

- To search for a key k, we trace a downward path starting at the root
- The next node visited depends on the comparison of k with the key of the current node
- If we reach a leaf, the key is not found
- Example: get(4):
  - Call TreeSearch(4,root)
- The algorithms for floorEntry and ceilingEntry are similar

- Recursive

```
Algorithm TreeSearch(k, v)
    if v.isExternal ()
        return v
    if k < v.key()
        return TreeSearch(k, v.left())
    else if k = v.key()
        return v
    else { k > v.key() }
        return TreeSearch(k, v.right())
```

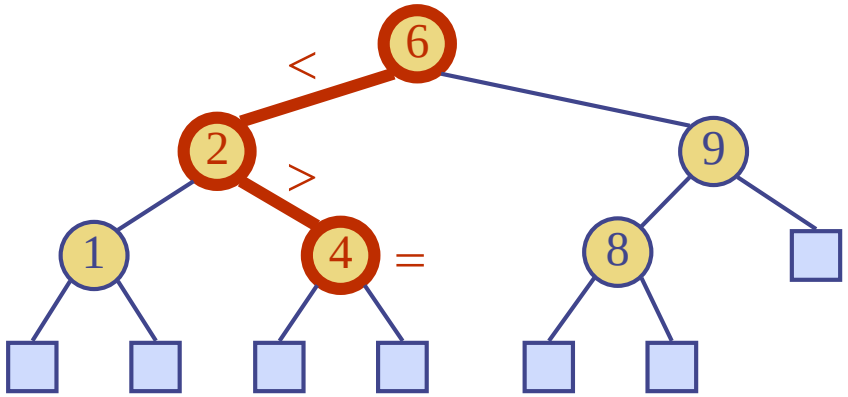


Search

- To search for a key k, we trace a downward path starting at the root
- The next node visited depends on the comparison of k with the key of the current node
- If we reach a leaf, the key is not found
- Example: get(4):
  - Call TreeSearch(4,root)
- The algorithms for floorEntry and ceilingEntry are similar

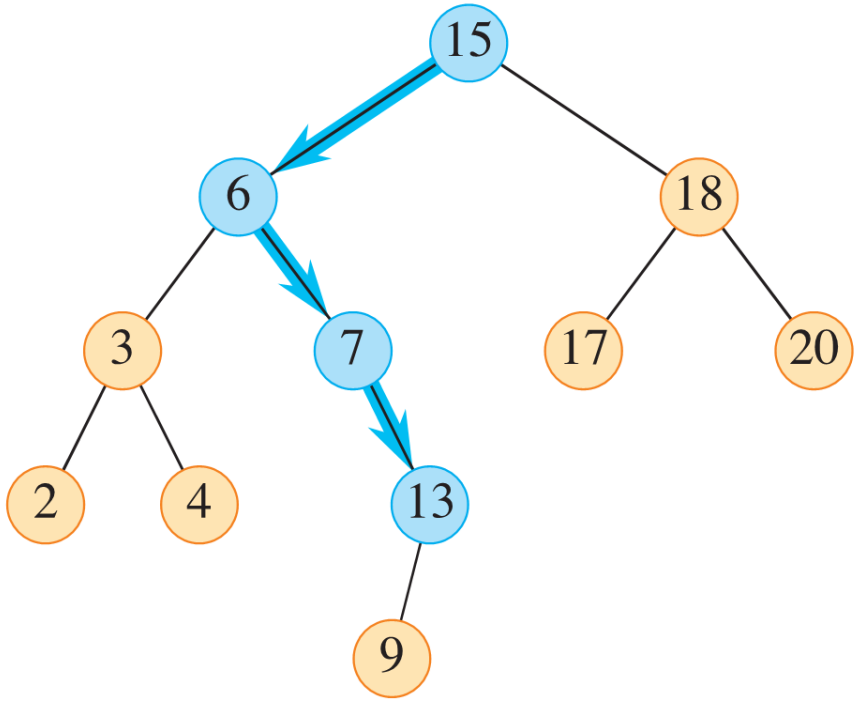
- iterative

```
Algorithm IterativeTreeSearch(k, v)
    while ¬ v.isExternal () and v.key() != k
        if k < v.key()
          v = v.left()
        else  v = v.right()
    return v
```



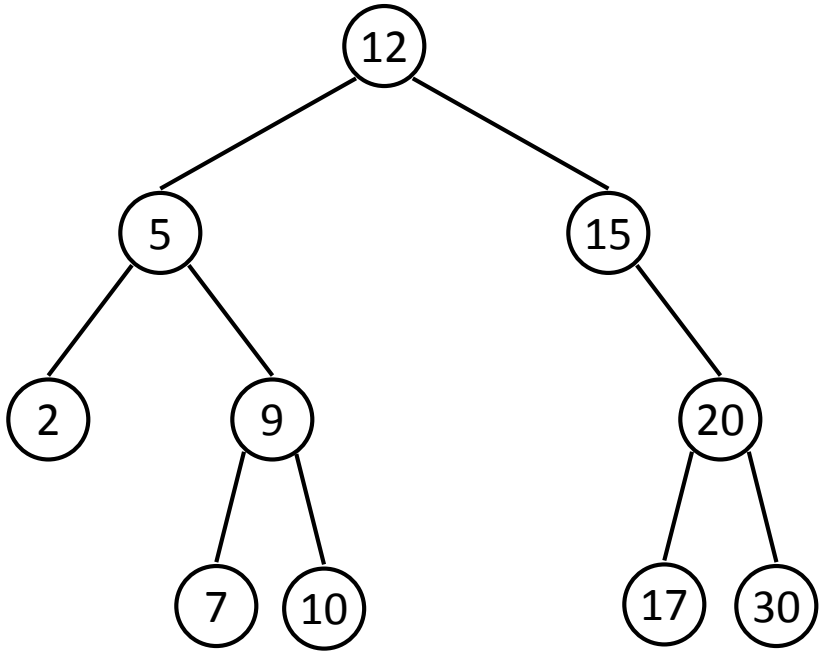
Search

search(13)



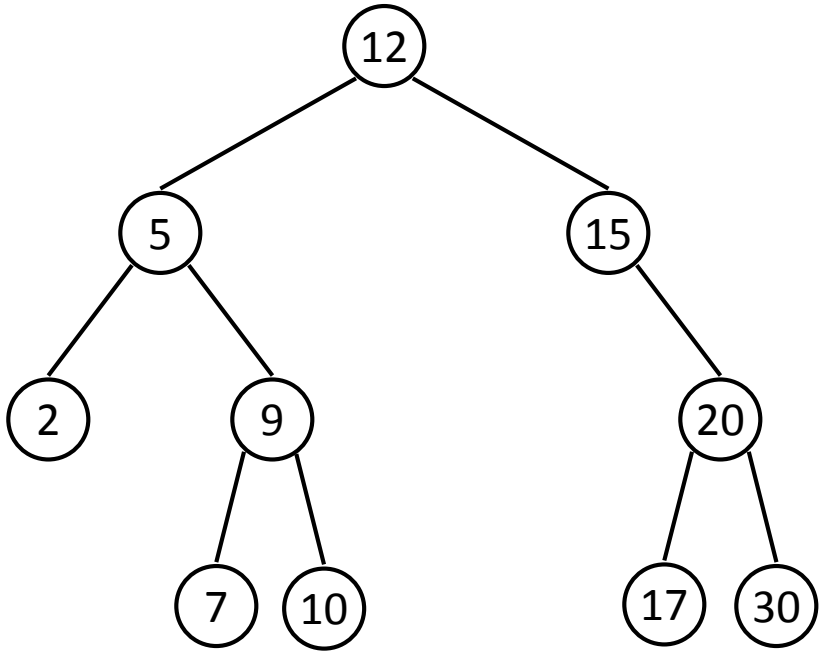
Search

search(17)



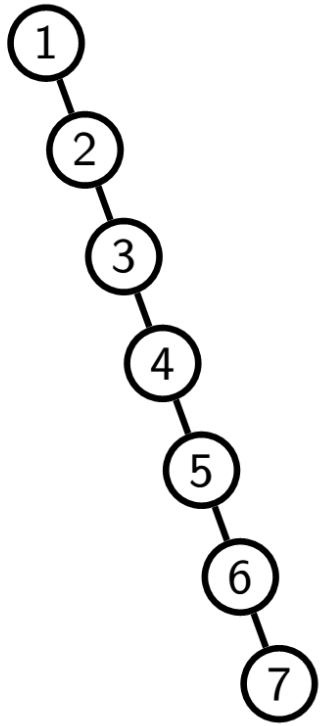
Search

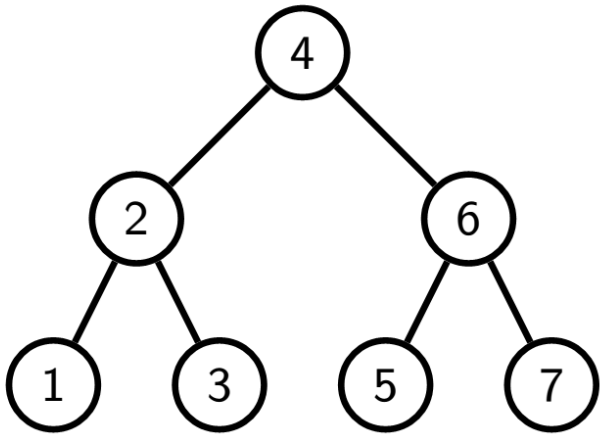
search(18)



Search

- پیچیدگی جستجو؟





Search

- پیچیدگی؟

$$O(h)$$

- حالت میانگین؟
- بهترین حالت؟
- بدترین حالت؟

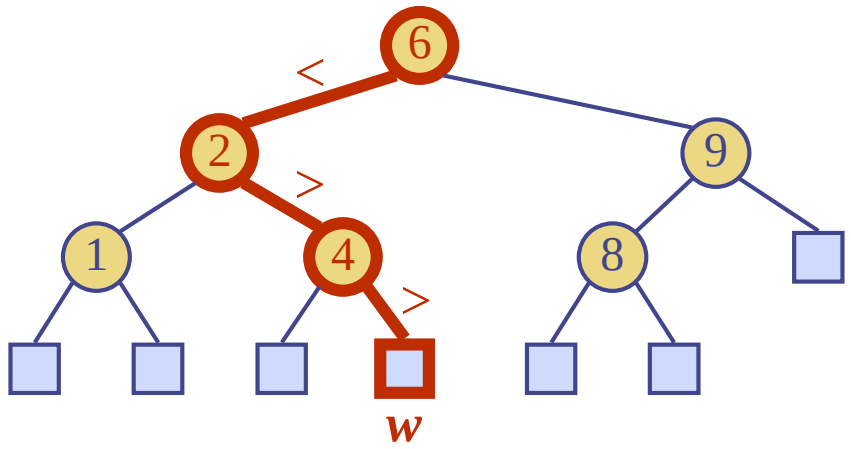
Search

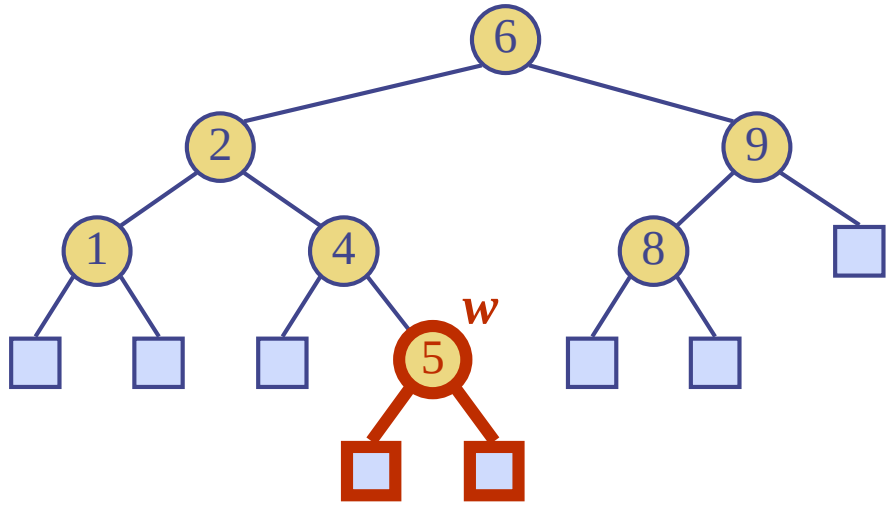
- بسیار سریع
- مثلا جستجوی بین 2^{10000} داده چقدر طول میکشد؟ (فرض توازن درخت و انجام یک عملیات در یک میکروثانیه)

Insertion

- To perform operation put(k, o), we search for key k (using TreeSearch)
- Assume k is not already in the tree, and let w be the leaf reached by the search
- We insert k at node w and expand w into an internal node

Example: insert(5)





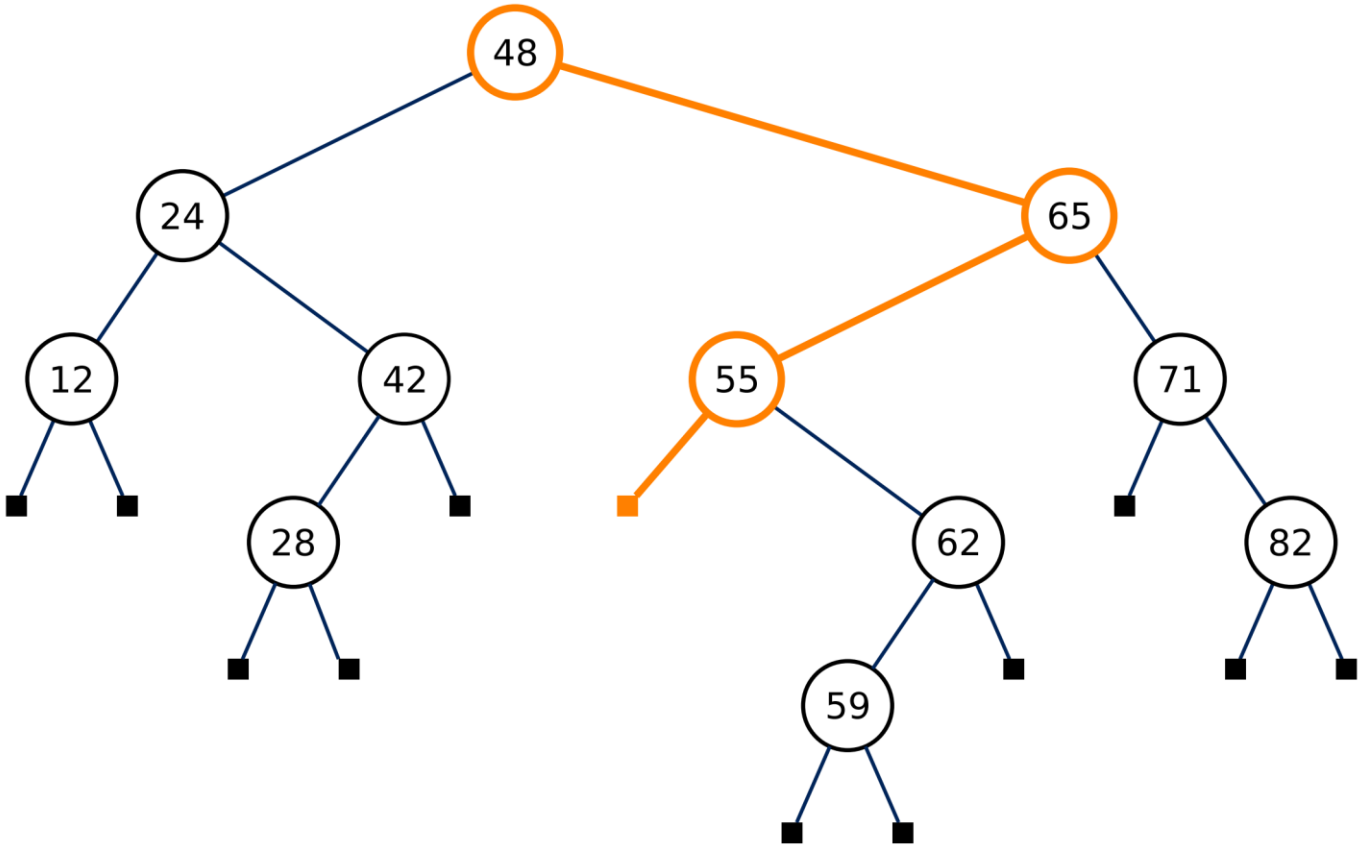
Insertion

- To perform operation put(k, o), we search for key k (using TreeSearch)
- Assume k is not already in the tree, and let w be the leaf reached by the search
- We insert k at node w and expand w into an internal node

```
Algorithm treeInsert(k, o)
    p ←treeSearch(k, root())
    if k = p.key() then
      p.setValue(o)
    else
      expandExternal(p, (k, o))
    return p
```

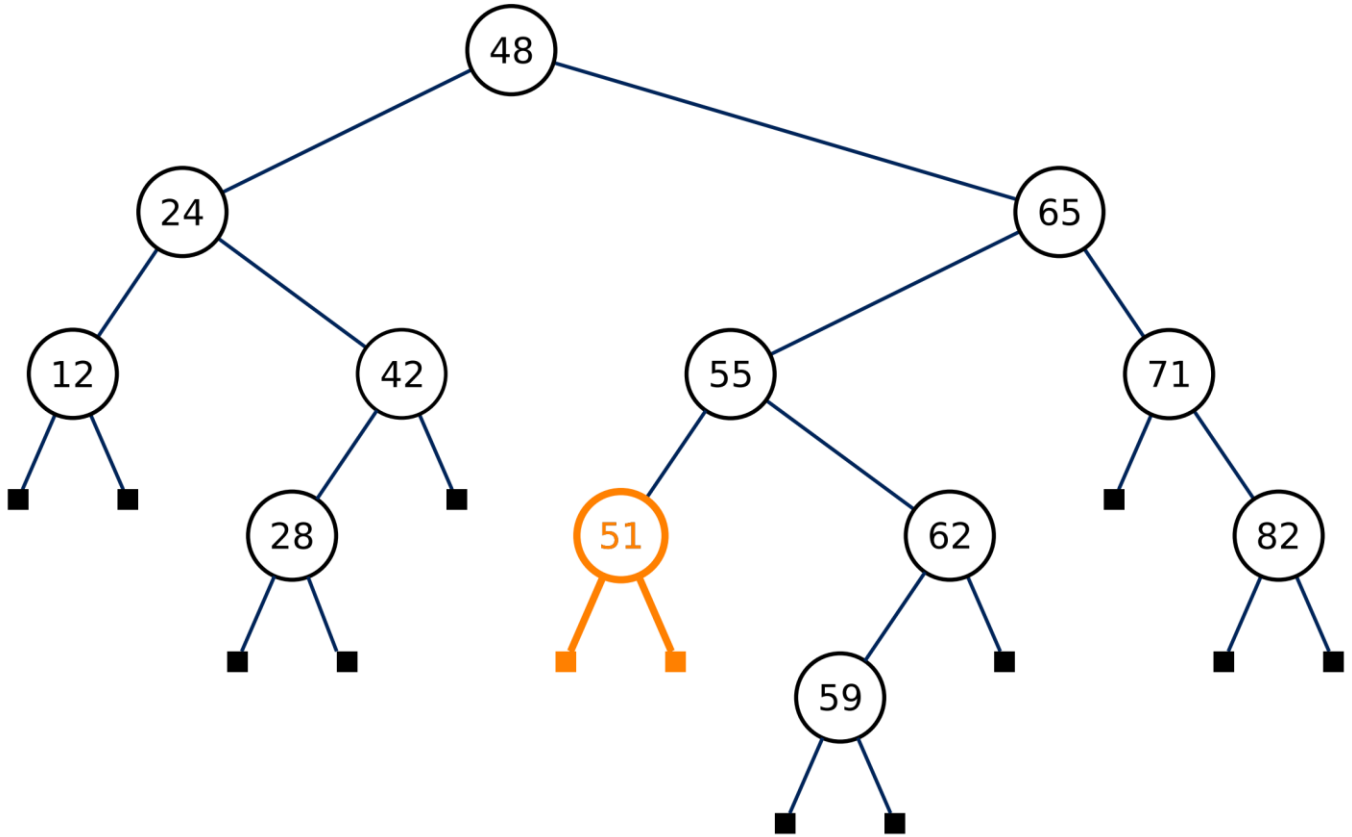
Insertion

insert(51)



Insertion

insert(51)



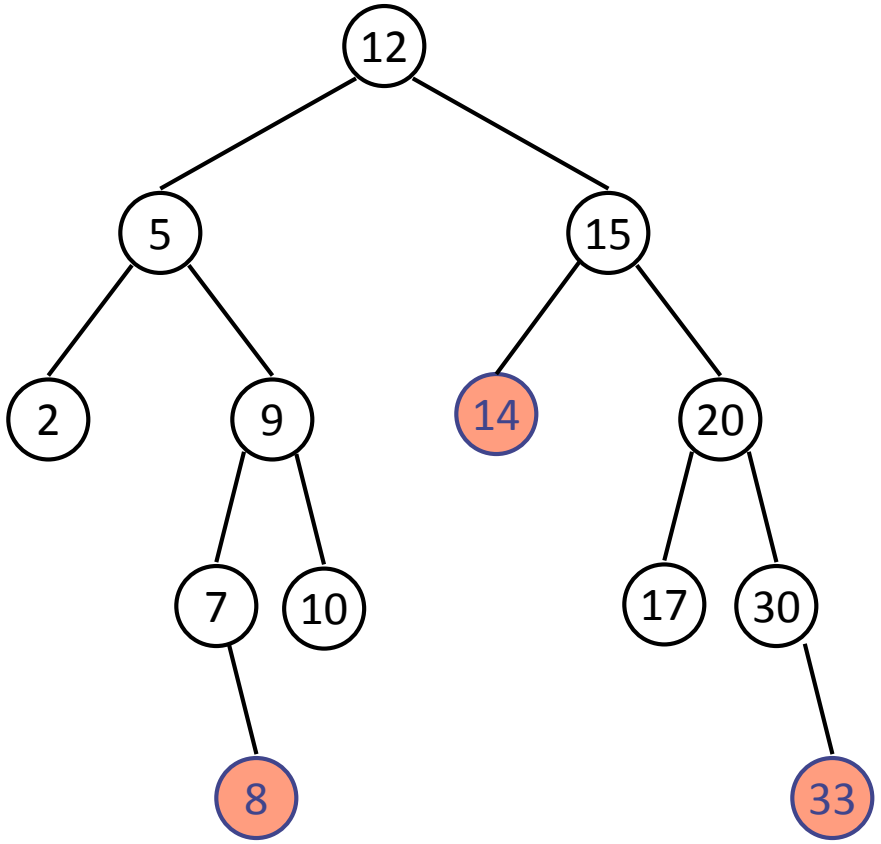
Insertion

- پیچیدگی؟

$$O(h)$$

- حالت میانگین؟
- بدترین حالت؟

Insertion



insert(14)
insert(8)
insert(33)

Ordered Maps

- firstEntry(): smallest key in the map
- lastEntry(): largest key in the map
- floorEntry(k): largest key $\leq$ k
- ceilingEntry(k): smallest key $\geq$ k

Search/Find

- یافتن مقدار کمینه؟ آیا یک گره میانی است؟

firstEntry():

Search/Find

- یافتن مقدار کمینه؟ آیا یک گره میانی است؟ پیچیدگی؟

firstEntry():

```
Algorithm firstEntry(v)
    while v.left != NULL
        v = v.left
    return v
```

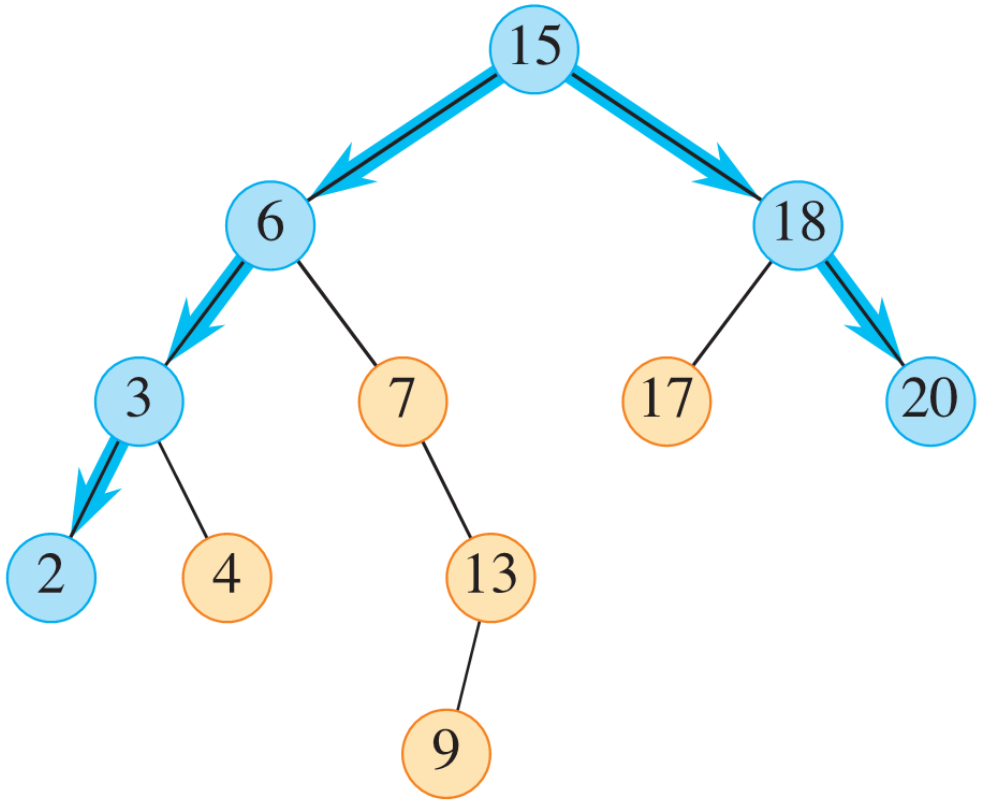
Search/Find

- یافتن مقدار بیشینه؟ آیا یک گره میانی است؟

lastEntry():

```
Algorithm lastEntry(v)
    while v.right!= NULL
        v = v.right
    return v
```

Search/Find

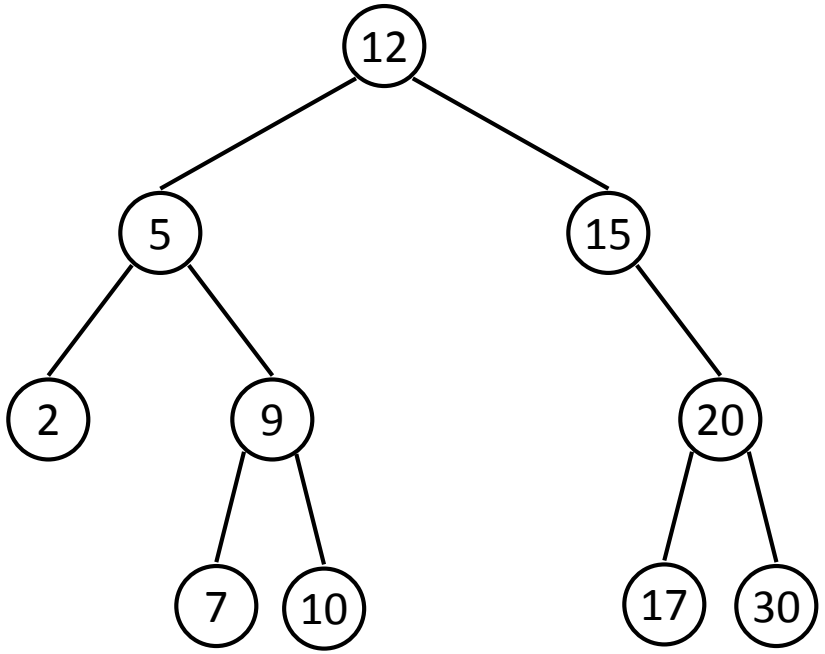


Search/Find

- برای Ceiling یا Successor؟

ceilingEntry(k): smallest key $\geq$ k

Search/Find



ceilingEntry(9):

ceilingEntry(10):

Search/Find

- برای ceilingEntry یا Successor؟
- طبق تعریف میشود گره بعدی که در یک inorder tree walk به آن میرسیم.

Search/Find

- برای ceilingEntry یا Successor؟
- طبق تعریف میشود گره بعدی که در یک inorder tree walk به آن میرسیم.

$x = TreeSearch(k, v)$

```
TREE-SUCCESSOR(x)
1   if x.right ≠ NIL
2         return TREE-MINIMUM(x.right)   // leftmost node in right subtree
3
4
5
6
7
8
```

p: parent
NIL: NULL

TREE-MINIMUM: firstEntry

Search/Find

- برای ceilingEntry یا Successor؟
- طبق تعریف میشود گره بعدی که در یک inorder tree walk به آن میرسیم.

***x* = *TreeSearch*(*k*, *v*)**

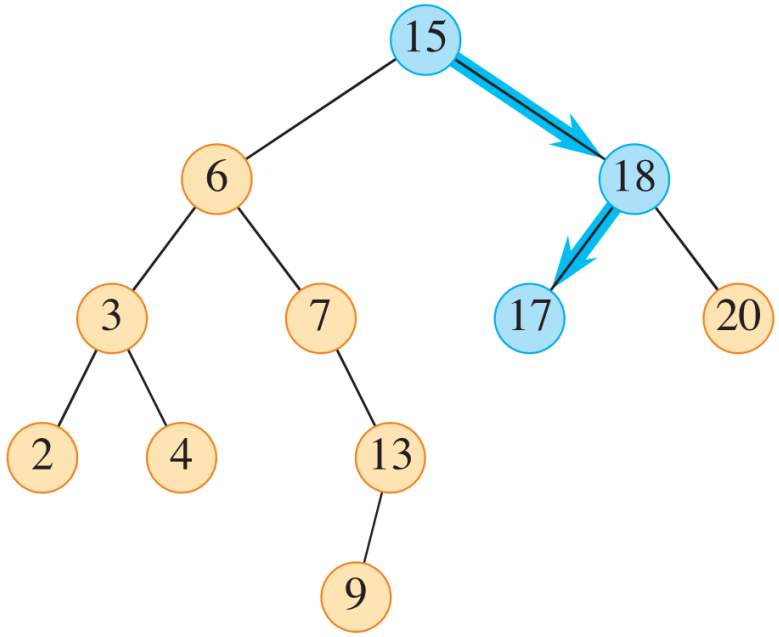
```
TREE-SUCCESSOR(x)
1  if x.right ≠ NIL
2        return TREE-MINIMUM(x.right)   // leftmost node in right subtree
3  else // find the lowest ancestor of x whose left child is an ancestor of x
4        y = x.p
5        while y ≠ NIL and x == y.right
6              x = y
7              y = y.p
8        return y
```

p: parent
NIL: NULL

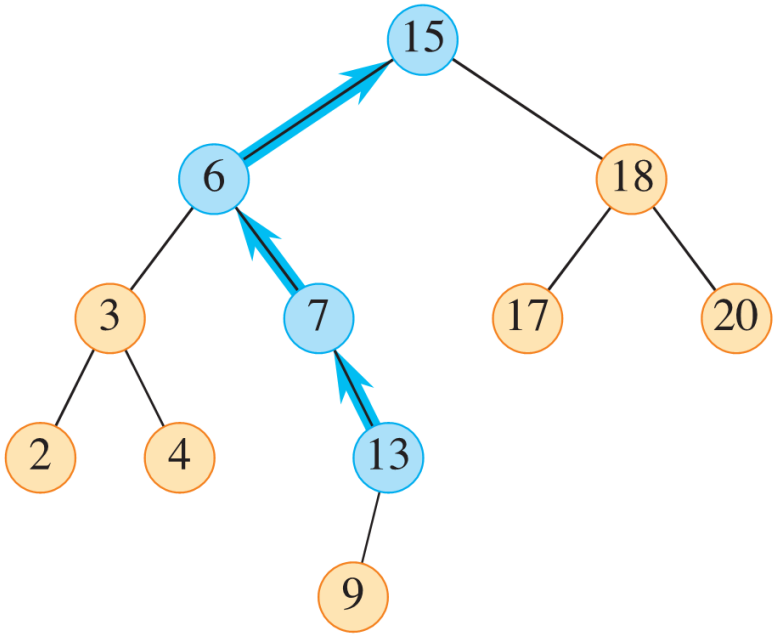
TREE-MINIMUM: firstEntry

Search/Find

- برای ceilingEntry یا Successor؟
- طبق تعریف میشود گره بعدی که در یک inorder tree walk به آن میرسیم.



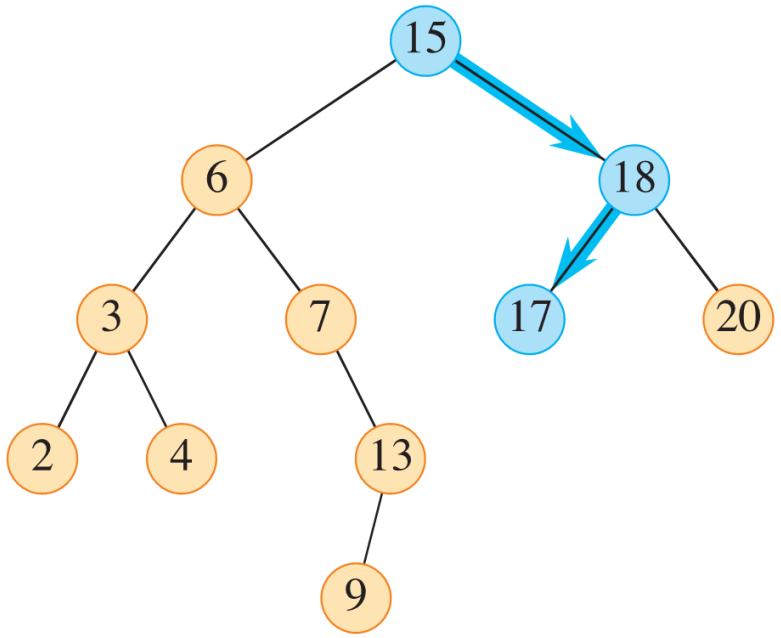
ceilingEntry(15)



ceilingEntry(13)

Search/Find

- برای ceilingEntry یا Successor؟
- طبق تعریف میشود گره بعدی که در یک inorder tree walk به آن میرسیم.



ceilingEntry(5)

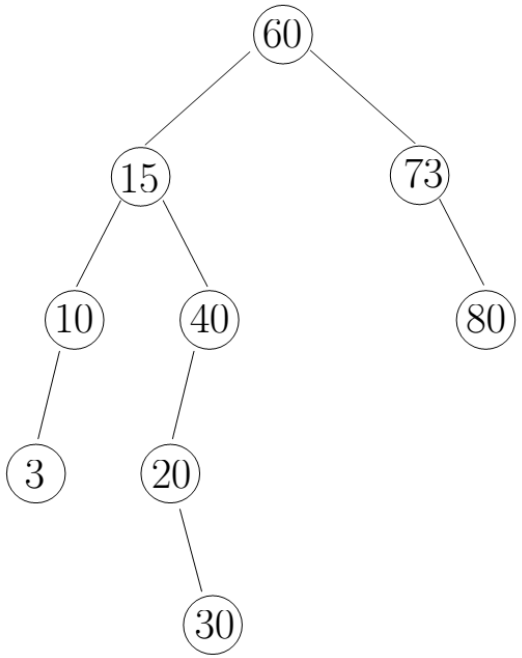
Search/Find

- برای floorEntry یا predecessor؟

مثال

- رسم BST برای مجموعه زیر:

$$S = \{60, 15, 73, 10, 80, 40, 10, 30, 3\}$$



مثال

- رسم BST برای مجموعه زیر:

$$S = \{3, 10, 15, 20, 30, 40, 60, 73, 80\}.$$

- روش بهتر: مرتب کردن و هر بار نصف کردن مجموعه و قرار دادن در سمت چپ و راست

مثال

- بعد از اضافه کردن n گره به صورت بختی به یک BST ، بدترین مقدار برای ارتفاع آن چقدر است؟

مثال

- بعد از اضافه کردن n گره به صورت بختی به یک BST ، مقدار قابل انتظار (امید) برای ارتفاع آن چقدر است؟

Deletion

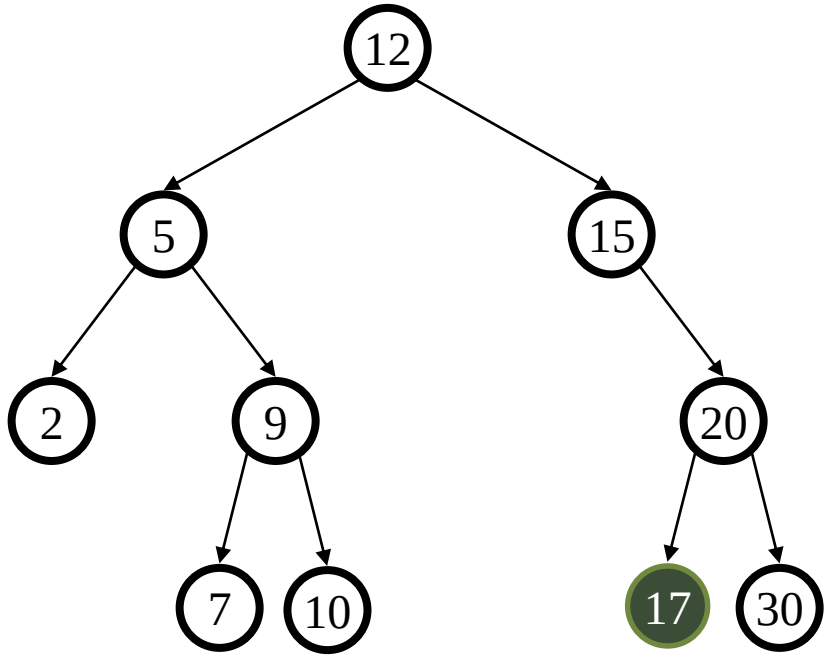
- To perform operation erase(k), we search for key k

- Assume key k is in the tree, and let v be the node storing k

Deletion

- اگر گره مورد نظر یک برگ باشد:

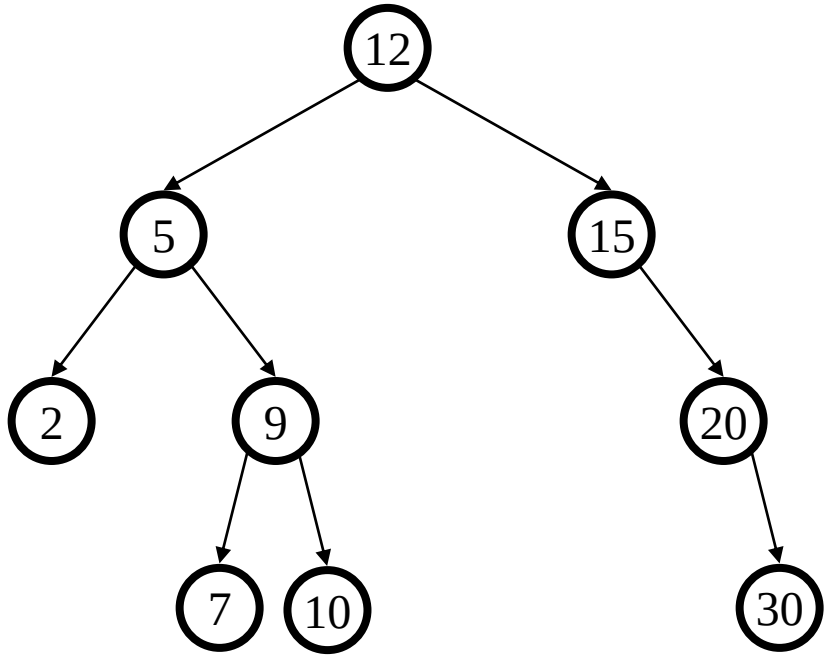
```
delete(17)
```



Deletion

- اگر گره مورد نظر یک برگ باشد:

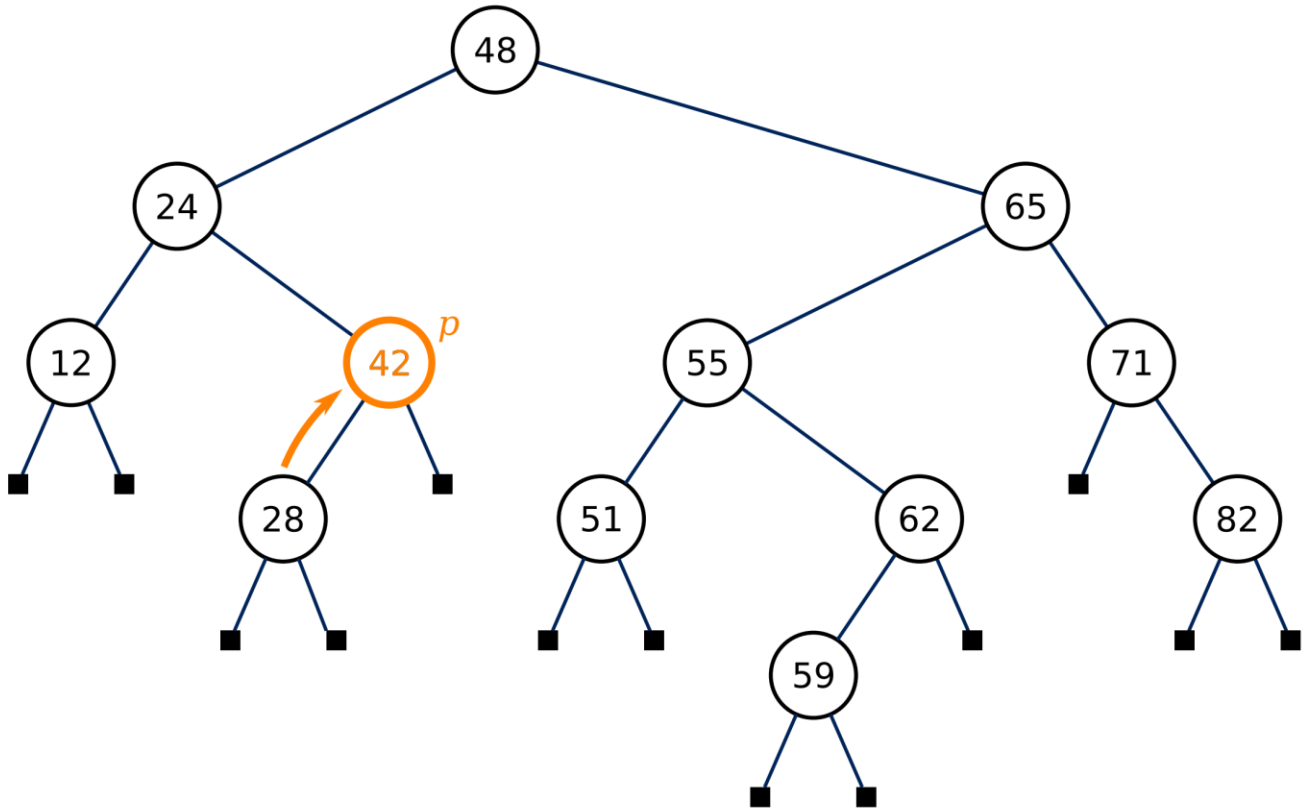
```
delete(17)
```



Deletion

- اگر گره مورد نظر فقط یک فرزند داشته باشد:

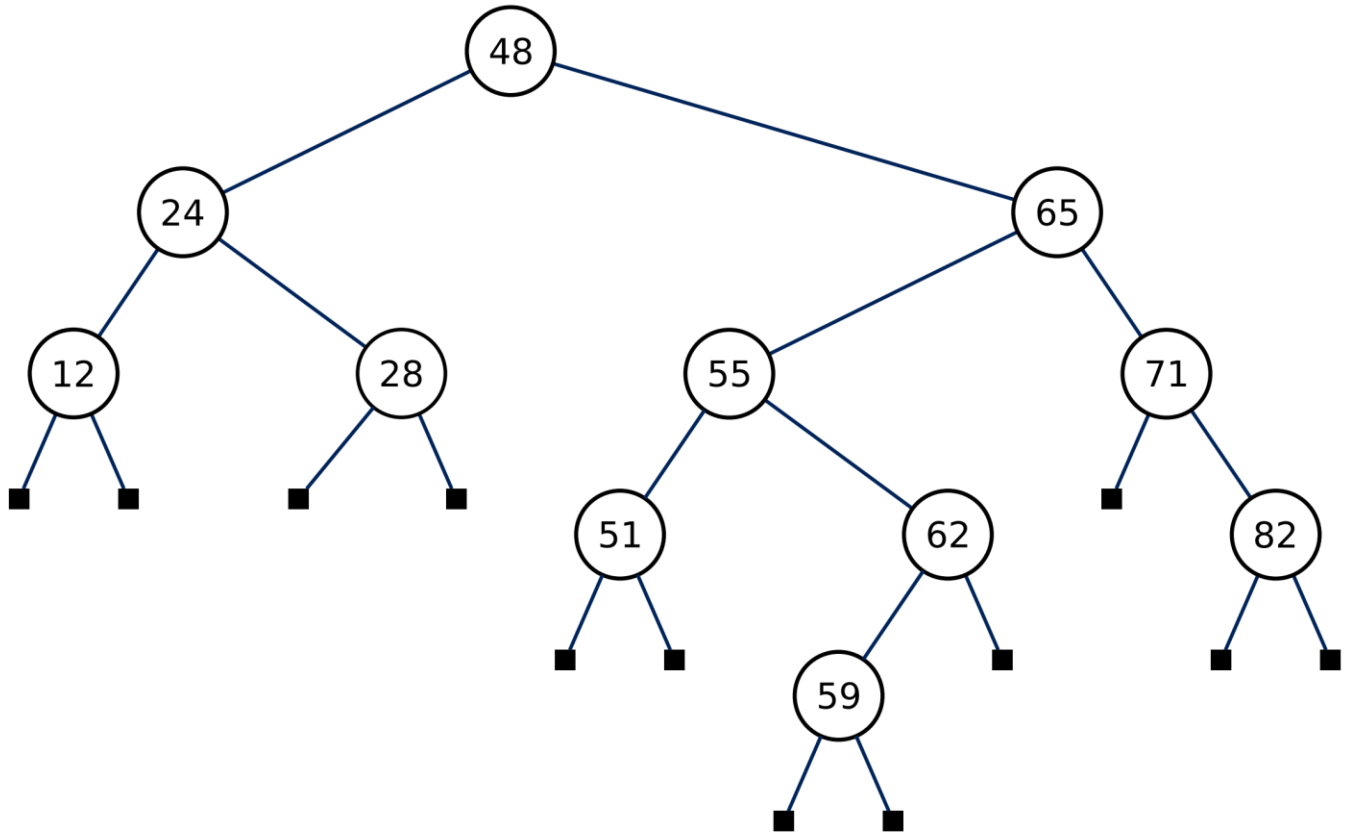
```
delete(42)
```



Deletion

- اگر گره مورد نظر فقط یک فرزند داشته باشد:

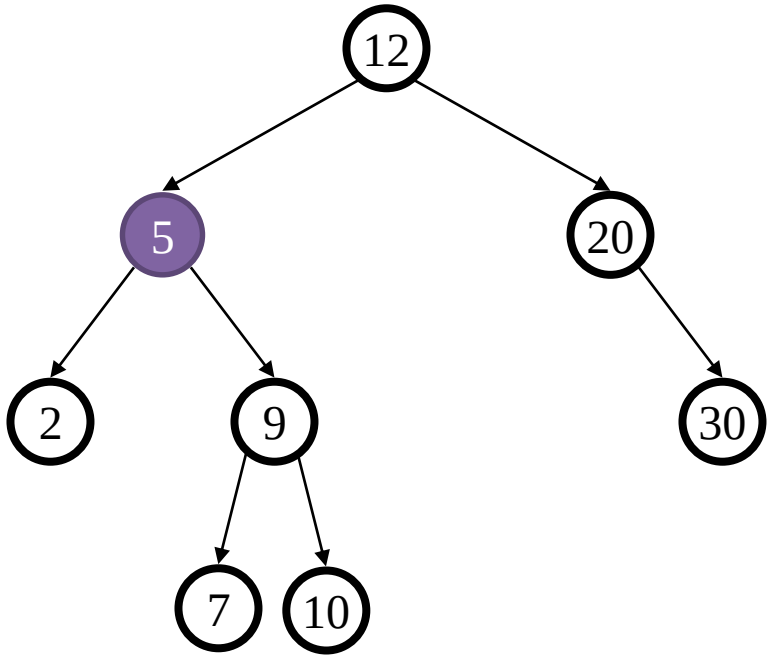
```
delete(42)
```



Deletion

- اگر گره مورد نظر دو فرزند داشته باشد:

```
delete(5)
```



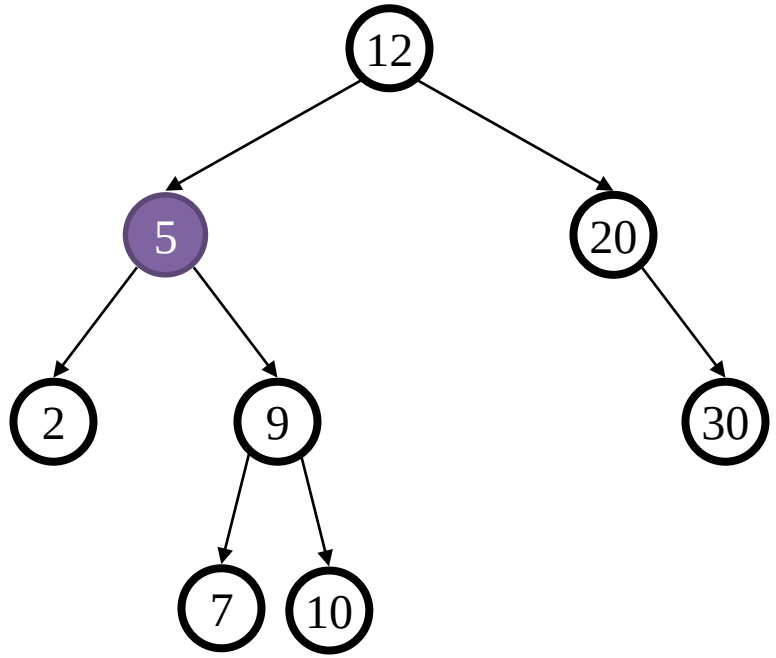
Deletion

- اگر گره مورد نظر دو فرزند داشته باشد:
- در این حالت دو گزینه داریم:
- جایگزینی گره با successor آن (دست چپی ترین گره در زیر درخت سمت راست)
- جایگزنی گره با predecessor آن (دست راستی ترین گره در زیر درخت سمت چپ)

Deletion

- اگر گره مورد نظر دو فرزند داشته باشد (successor):

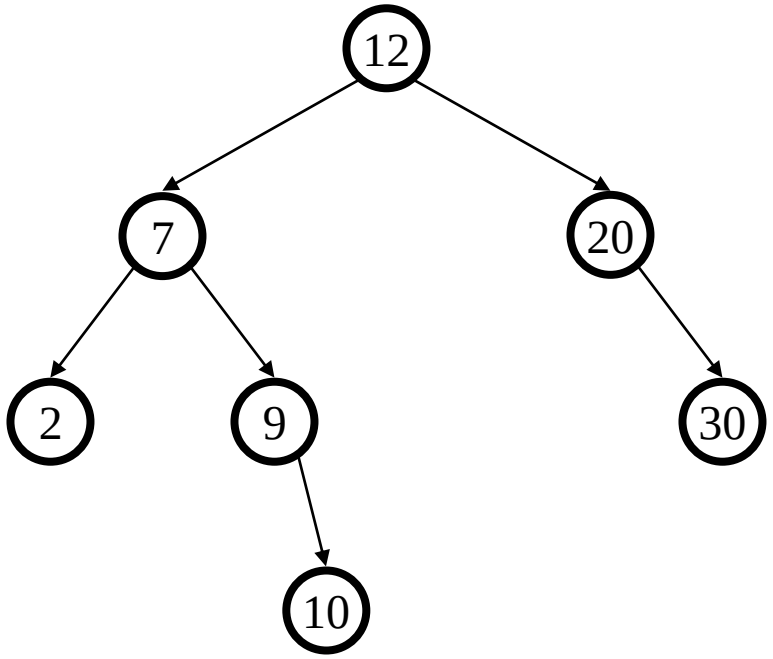
```
delete(5)
```



Deletion

- اگر گره مورد نظر دو فرزند داشته باشد:

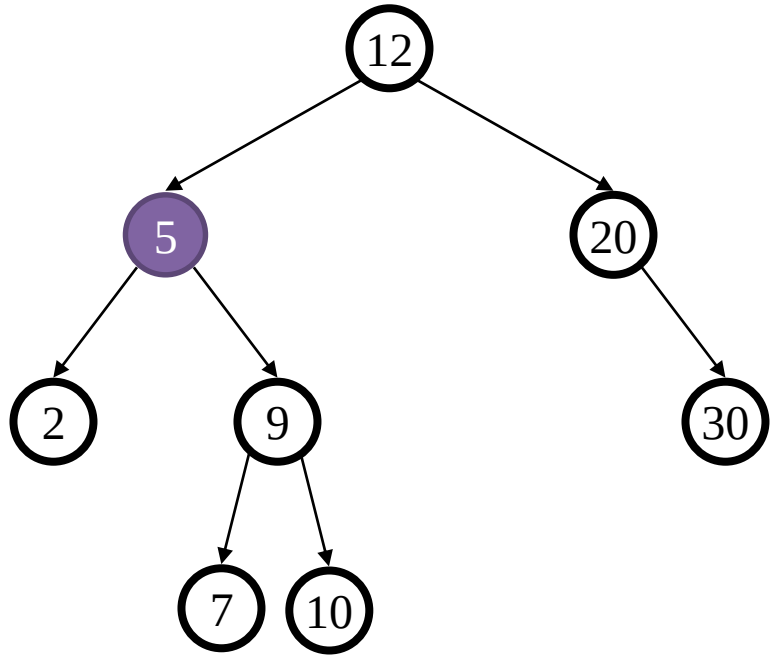
```
delete(5)
```



Deletion

- اگر گره مورد نظر دو فرزند داشته باشد (predecessor):

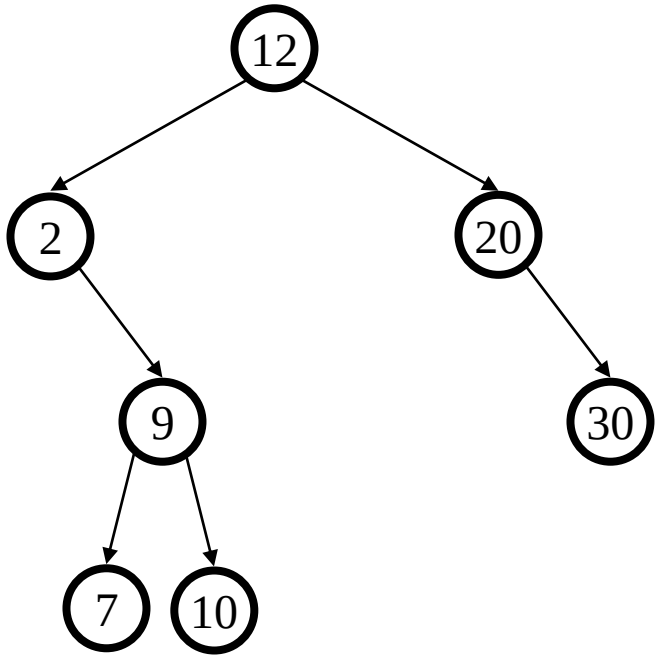
```
delete(5)
```



Deletion

- اگر گره مورد نظر دو فرزند داشته باشد:

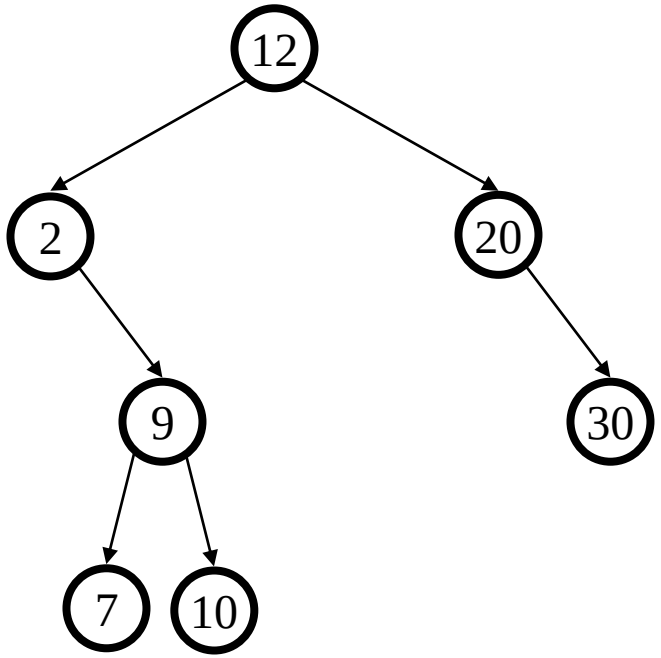
```
delete(5)
```



Deletion

- اگر گره مورد نظر دو فرزند داشته باشد:

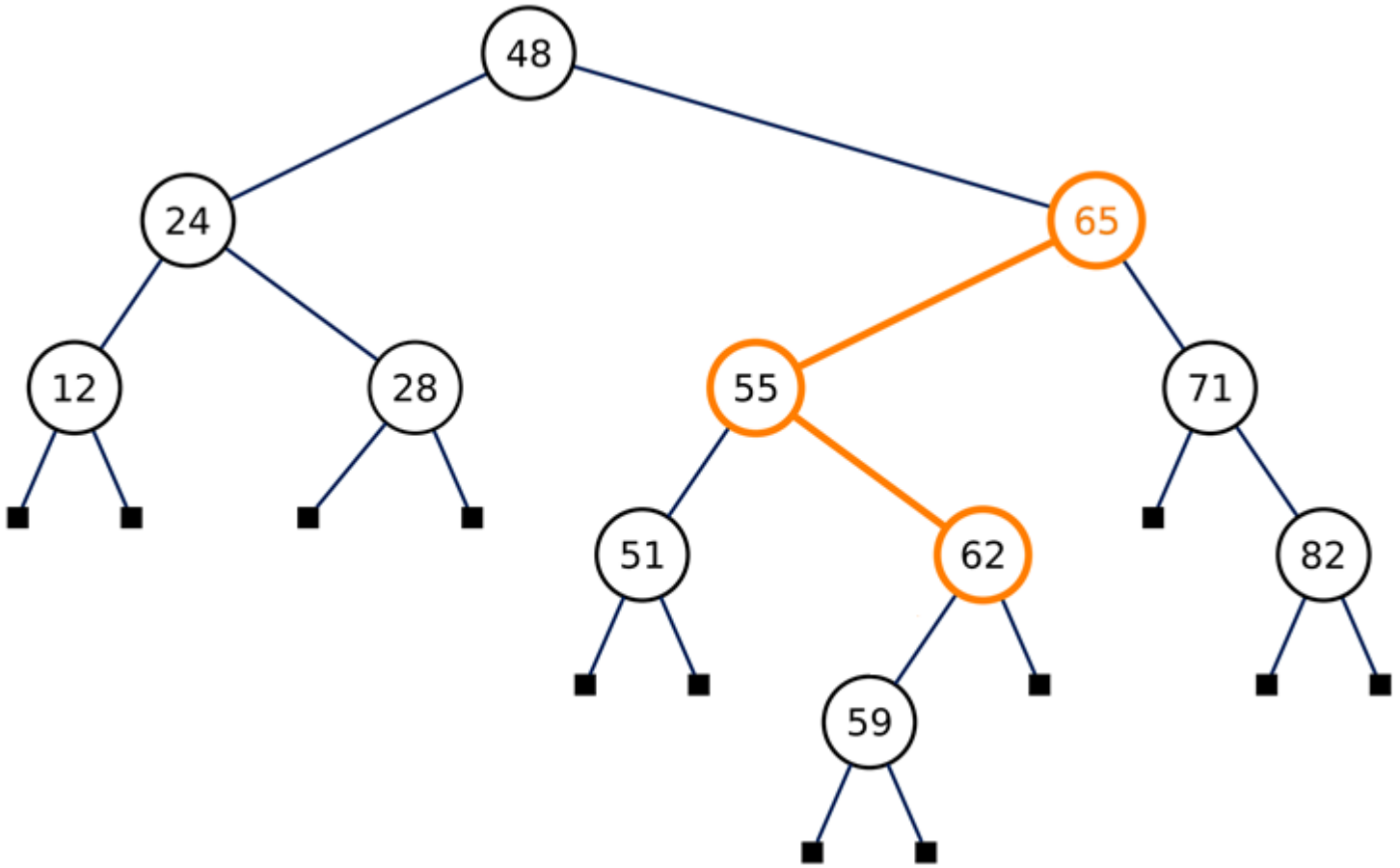
```
delete(5)
```



- Copy کنیم یا move؟

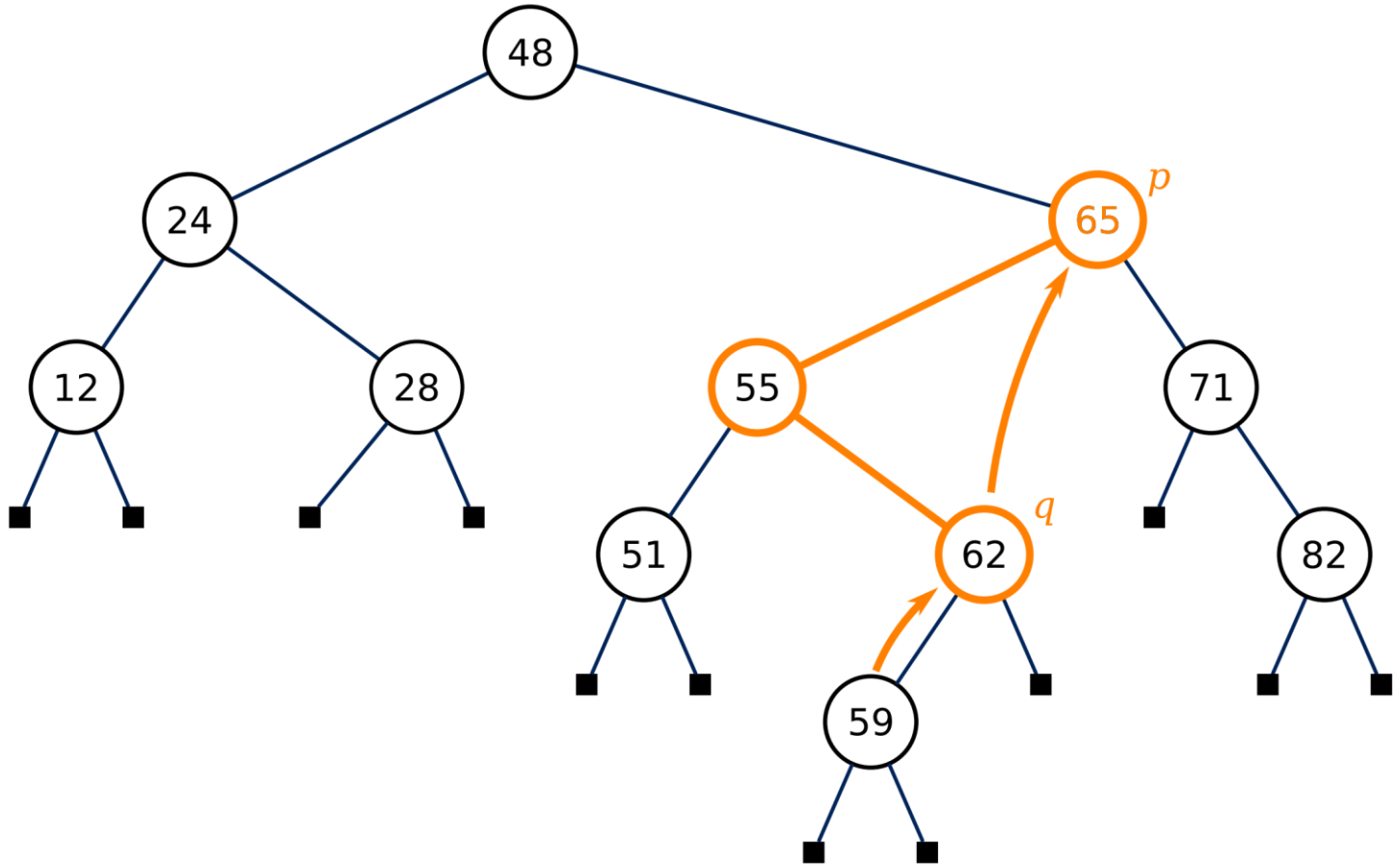
Deletion (Example)

```
delete(65)
```



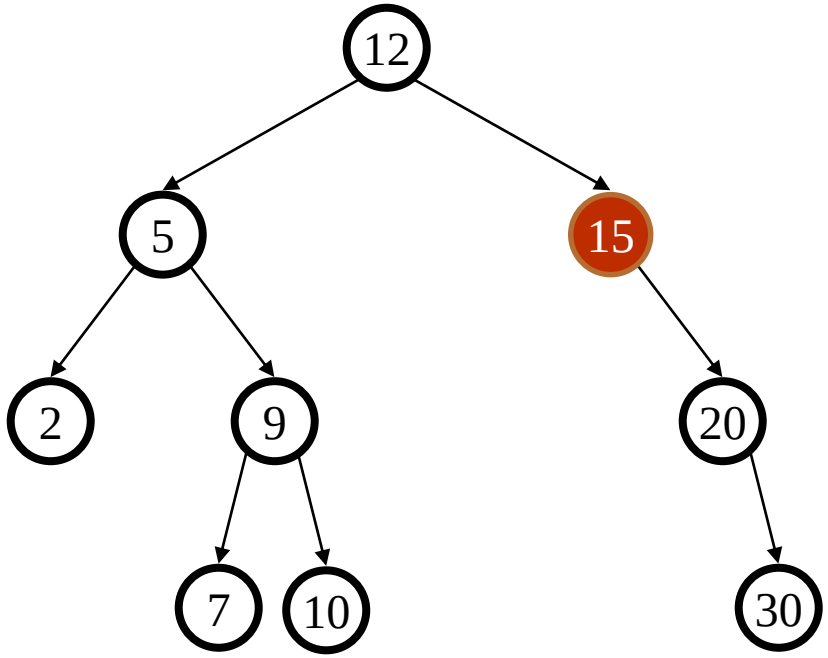
Deletion (Example)

```
delete(65)
```



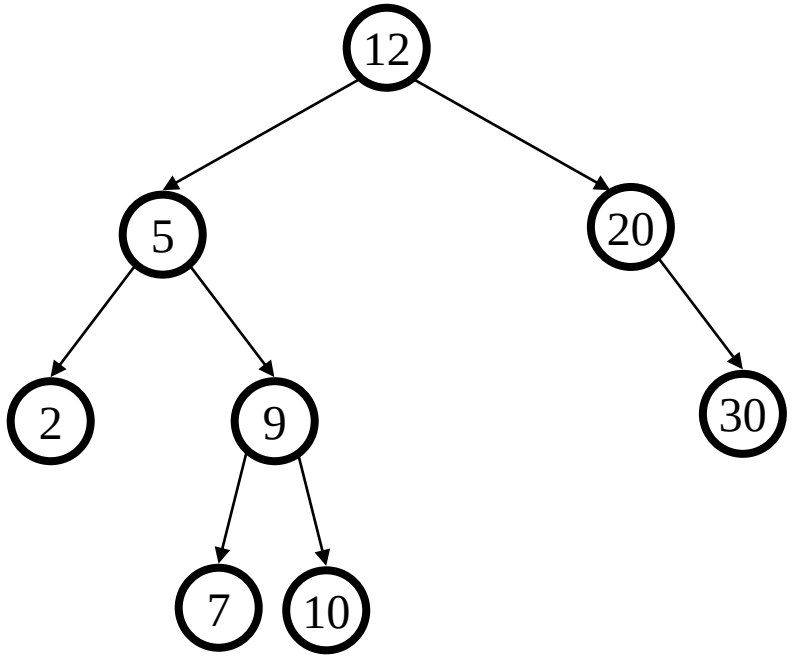
Deletion (Example)

```
delete(15)
```



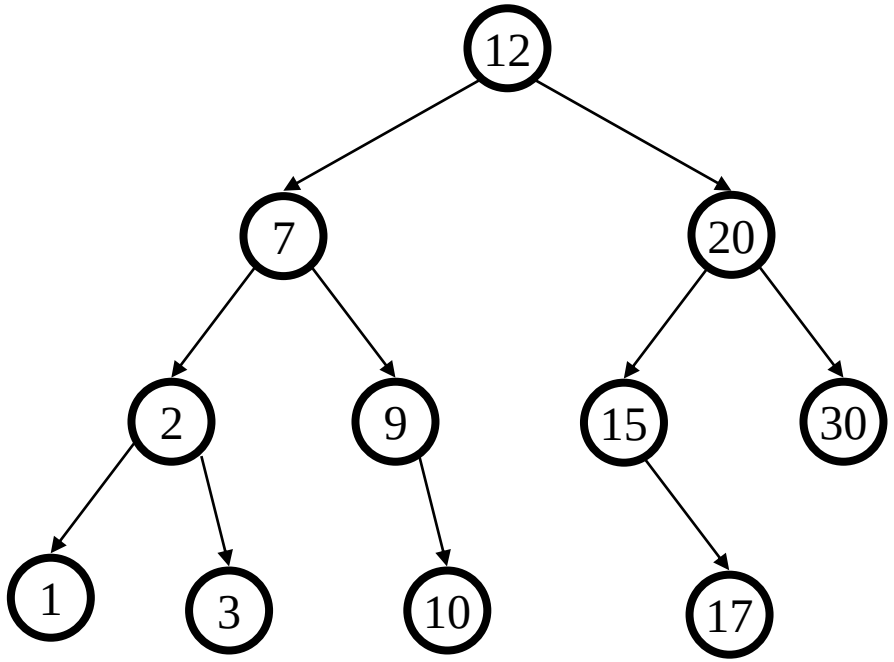
Deletion (Example)

```
delete(15)
```



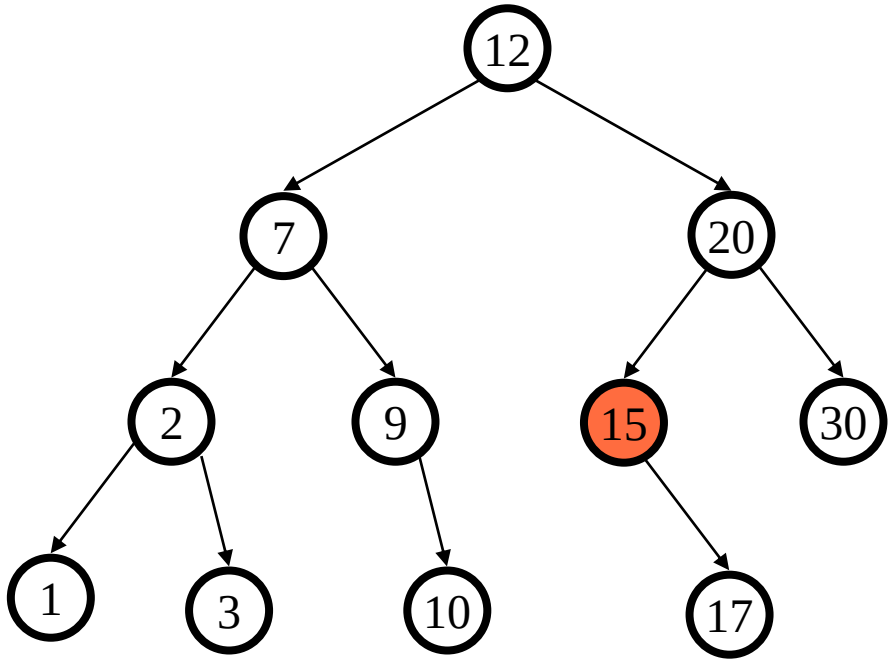
Deletion (Example)

```
delete(12)
```



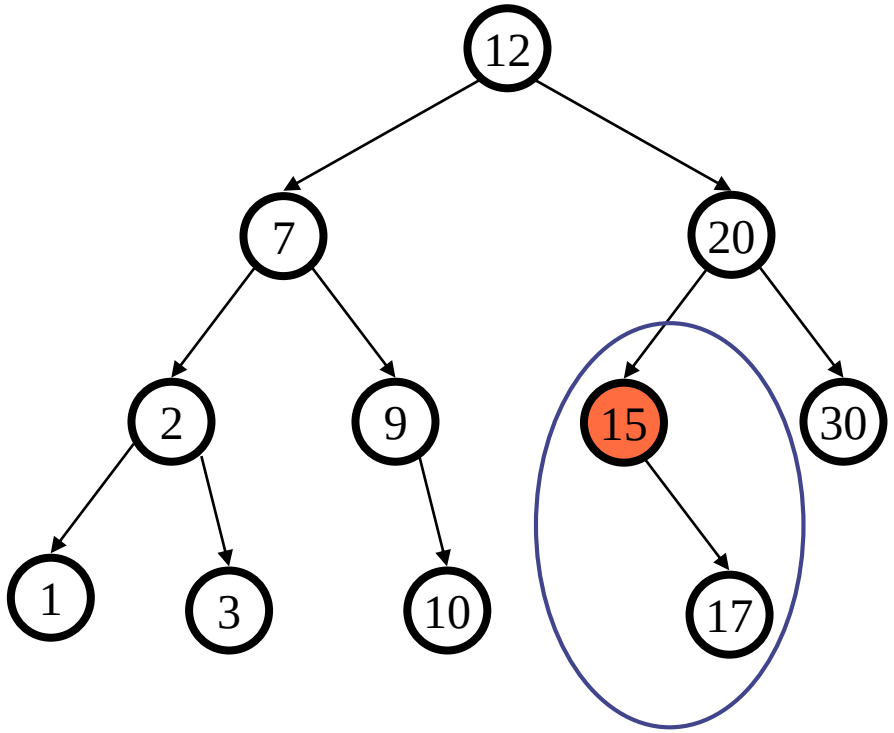
Deletion (Example)

```
delete(12)
```



Deletion (Example)

```
delete(12)
```



Deletion (Example)

```
delete(12)
```