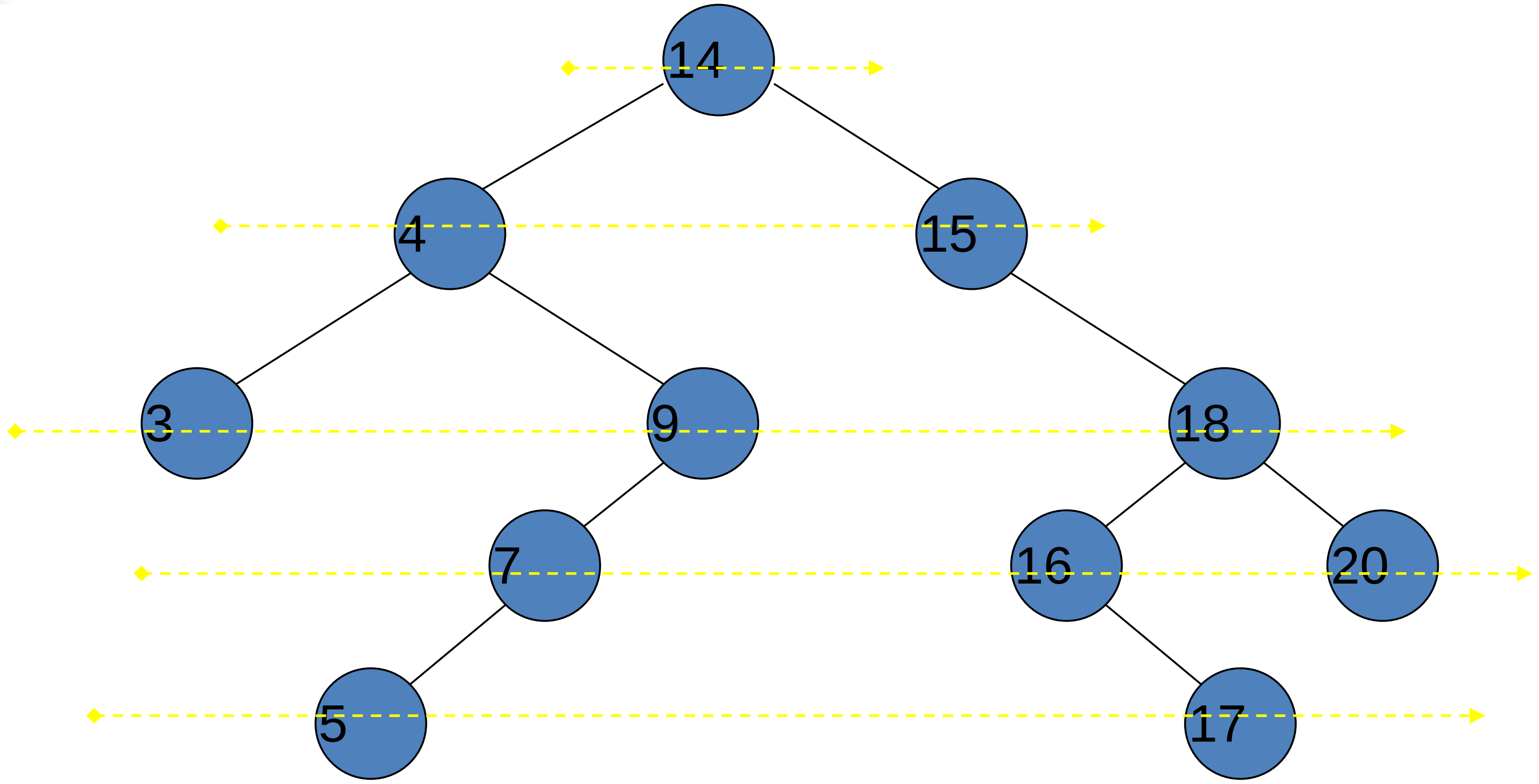
Level Order Binary Tree Traversal

Level-order Traversal

- There is yet another way of traversing a binary tree that is not related to recursive traversal procedures discussed previously.
- In level-order traversal, we visit the nodes at each level before proceeding to the next level.
- At each level, we visit the nodes in a left-to-right order.

Level-order Traversal



Level-order: 14 4 15 3 9 18 7 16 20 5 17

Level-order Traversal

- How do we do level-order traversal?
- Surprisingly, if we use a queue instead of a stack, we can visit the nodes in level-order.
- Here is the code for level-order traversal:

Level-order Traversal

```
void levelorder(TreeNode* treeNode)
{
    Queue q;
    if( treeNode == NULL ) return;
    q.enqueue( treeNode);
    while( !q.empty() )
    {
        treeNode = q.dequeue();
        cout << (treeNode->getInfo()) << " ";
        if(treeNode->getLeft() != NULL )
             q.enqueue( treeNode->getLeft());
        if(treeNode->getRight() != NULL )
             q.enqueue( treeNode->getRight());
    }
    cout << endl;
}
```

Level-order Traversal

```
void levelorder(TreeNode* treeNode)
{
    Queue q;
    if( treeNode == NULL ) return;
    q.enqueue( treeNode);
    while( !q.empty() )
    {
        treeNode = q.dequeue();
        cout << (treeNode->getInfo()) << " ";
        if(treeNode->getLeft() != NULL )
             q.enqueue( treeNode->getLeft());
        if(treeNode->getRight() != NULL )
             q.enqueue( treeNode->getRight());
    }
    cout << endl;
}
```

Level-order Traversal

```
void levelorder(TreeNode* treeNode)
{
    Queue q;
    if( treeNode == NULL ) return;
    q.enqueue( treeNode);
    while( !q.empty() )
    {
        treeNode = q.dequeue();
        cout << (treeNode->getInfo()) << " ";
        if(treeNode->getLeft() != NULL )
             q.enqueue( treeNode->getLeft());
        if(treeNode->getRight() != NULL )
             q.enqueue( treeNode->getRight());
    }
    cout << endl;
}
```

Level-order Traversal

```
void levelorder(TreeNode* treeNode)
{
    Queue q;
    if( treeNode == NULL ) return;
    q.enqueue( treeNode);
    while( !q.empty() )
    {
        treeNode = q.dequeue();
        cout << (treeNode->getInfo()) << " ";
        if(treeNode->getLeft() != NULL )
             q.enqueue( treeNode->getLeft());
        if(treeNode->getRight() != NULL )
             q.enqueue( treeNode->getRight());
    }
    cout << endl;
}
```

Level-order Traversal

```
void levelorder(TreeNode* treeNode)
{
    Queue q;
    if( treeNode == NULL ) return;
    q.enqueue( treeNode);
    while( !q.empty() )
    {
        treeNode = q.dequeue();
        cout << (treeNode->getInfo()) << " ";
        if(treeNode->getLeft() != NULL )
             q.enqueue( treeNode->getLeft());
        if(treeNode->getRight() != NULL )
             q.enqueue( treeNode->getRight());
    }
    cout << endl;
}
```

Level-order Traversal

```
void levelorder(TreeNode* treeNode)
{
    Queue q;
    if( treeNode == NULL ) return;
    q.enqueue( treeNode);
    while( !q.empty() )
    {
        treeNode = q.dequeue();
        cout << (treeNode->getInfo()) << " ";
        if(treeNode->getLeft() != NULL )
             q.enqueue( treeNode->getLeft());
        if(treeNode->getRight() != NULL )
             q.enqueue( treeNode->getRight());
    }
    cout << endl;
}
```

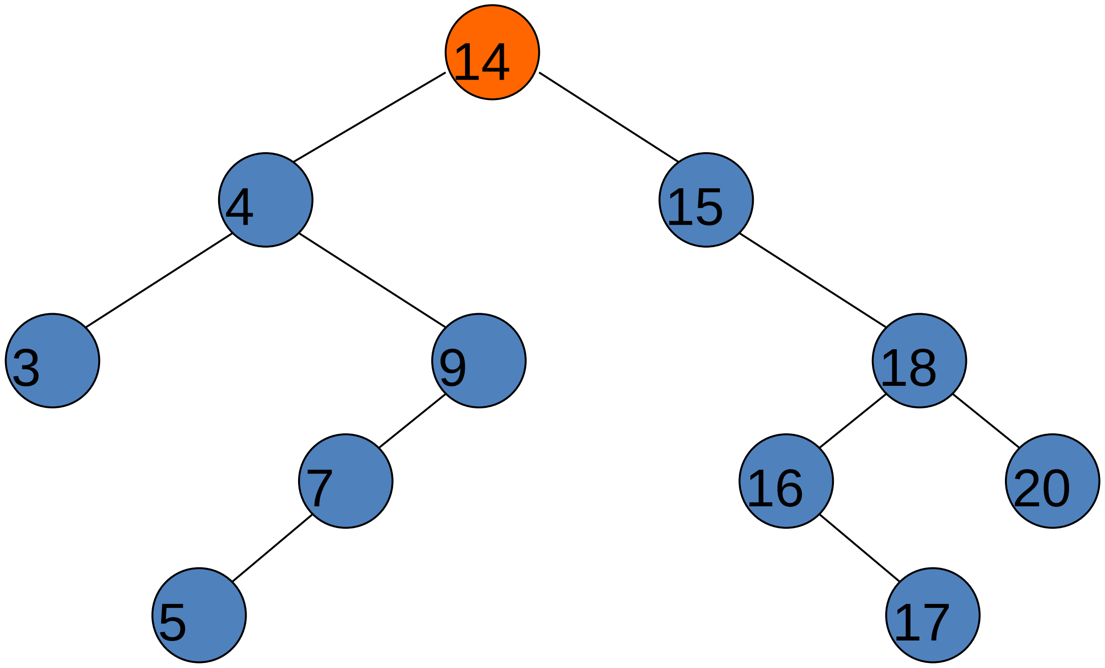
Level-order Traversal

```
void levelorder(TreeNode* treeNode)
{
    Queue q;
    if( treeNode == NULL ) return;
    q.enqueue( treeNode);
    while( !q.empty() )
    {
        treeNode = q.dequeue();
        cout << (treeNode->getInfo()) << " ";
        if(treeNode->getLeft() != NULL )
            q.enqueue( treeNode->getLeft());
        if(treeNode->getRight() != NULL )
            q.enqueue( treeNode->getRight());
    }
    cout << endl;
}
```

Level-order Traversal

```
void levelorder(TreeNode* treeNode)
{
    Queue q;
    if( treeNode == NULL ) return;
    q.enqueue( treeNode);
    while( !q.empty() )
    {
        treeNode = q.dequeue();
        cout << (treeNode->getInfo()) << " ";
        if(treeNode->getLeft() != NULL )
             q.enqueue( treeNode->getLeft());
        if(treeNode->getRight() != NULL )
             q.enqueue( treeNode->getRight());
    }
    cout << endl;
}
```

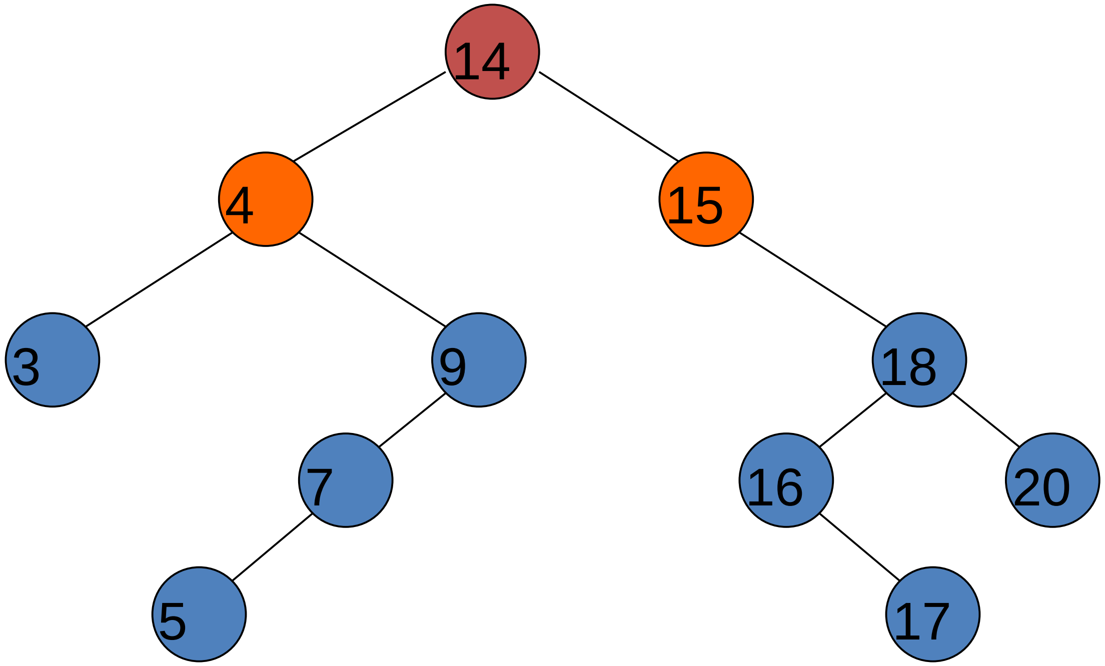
Level-order Traversal



Queue: 14

Output:

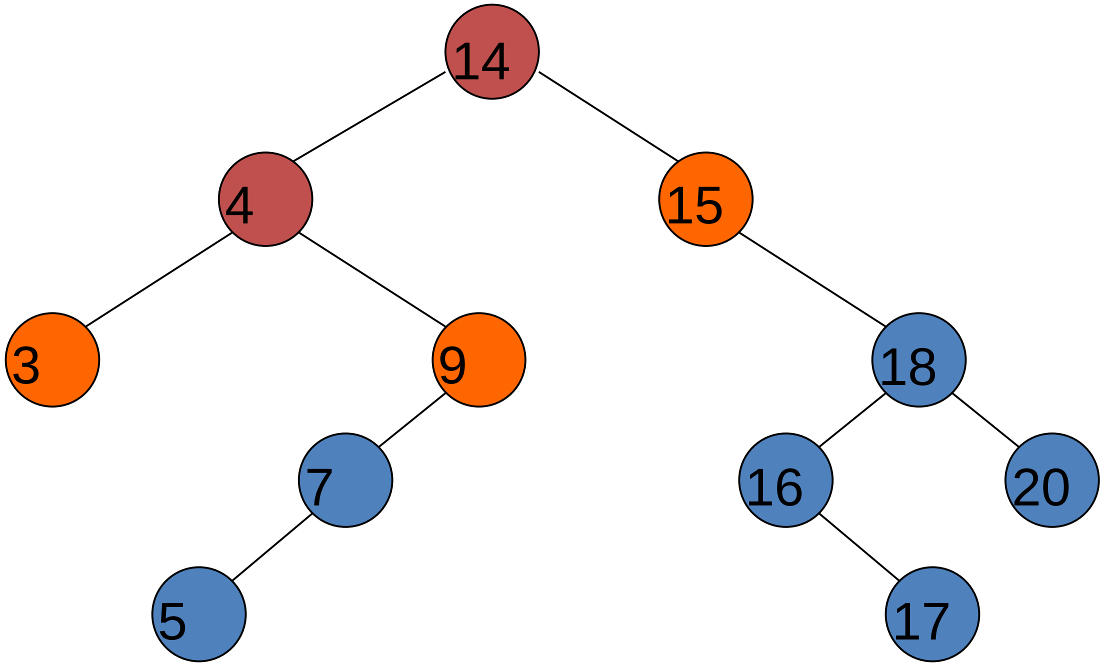
Level-order Traversal



Queue: 4 15

Output: 14

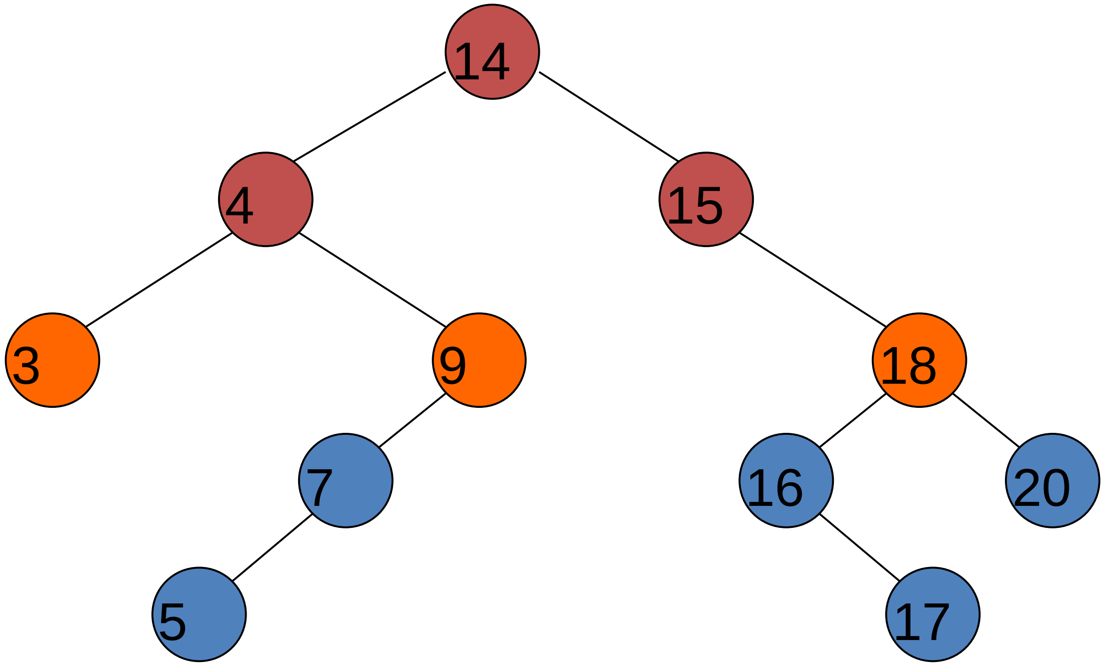
Level-order Traversal



Queue: 15 3 9

Output: 14 4

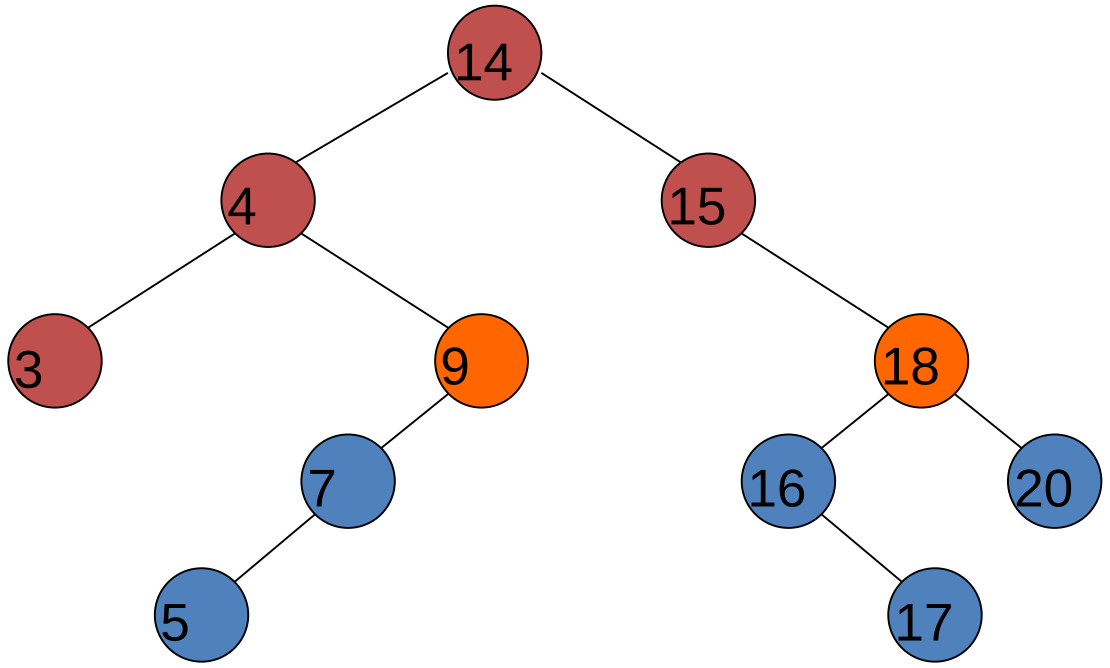
Level-order Traversal



Queue: 3 9 18

Output: 14 4 15

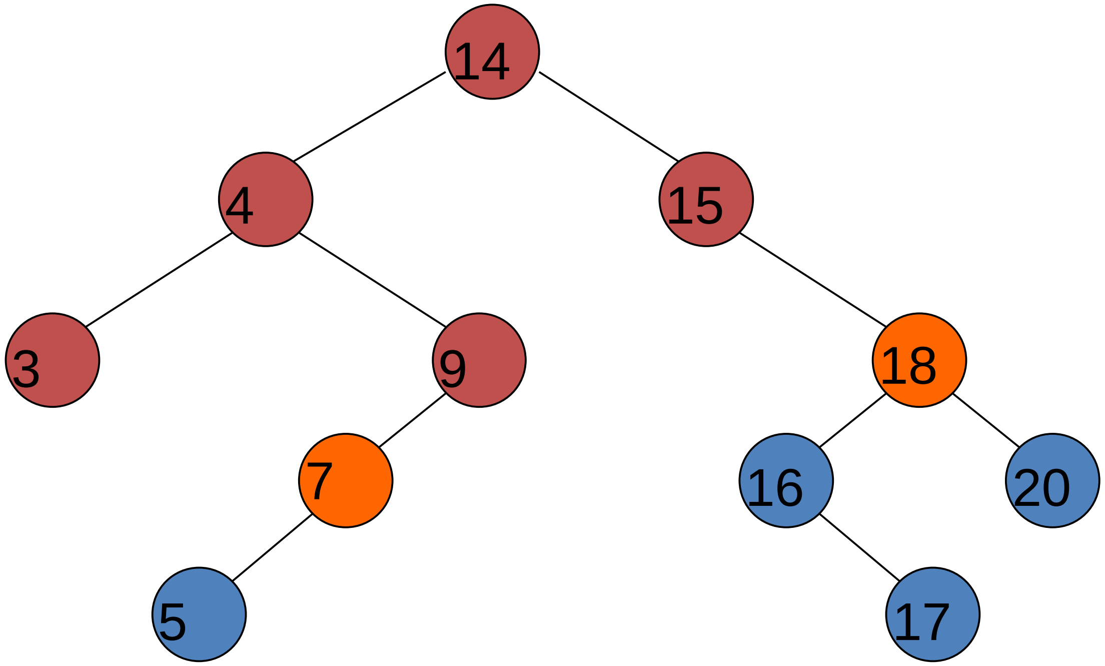
Level-order Traversal



Queue: 9 18

Output: 14 4 15 3

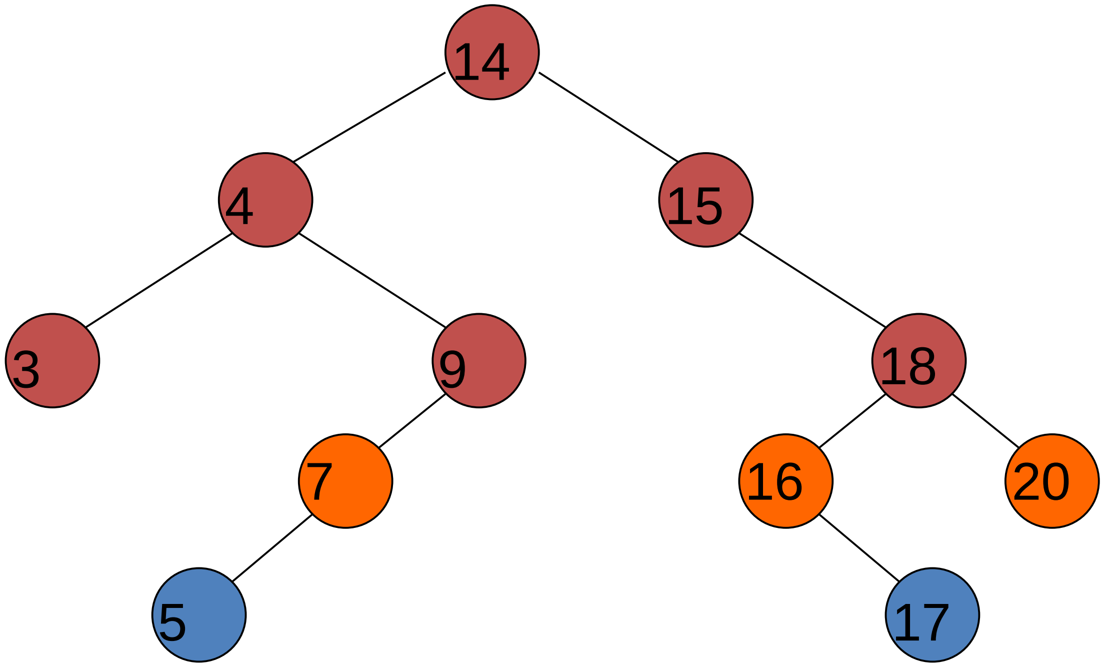
Level-order Traversal



Queue: 18 7

Output: 14 4 15 3   9

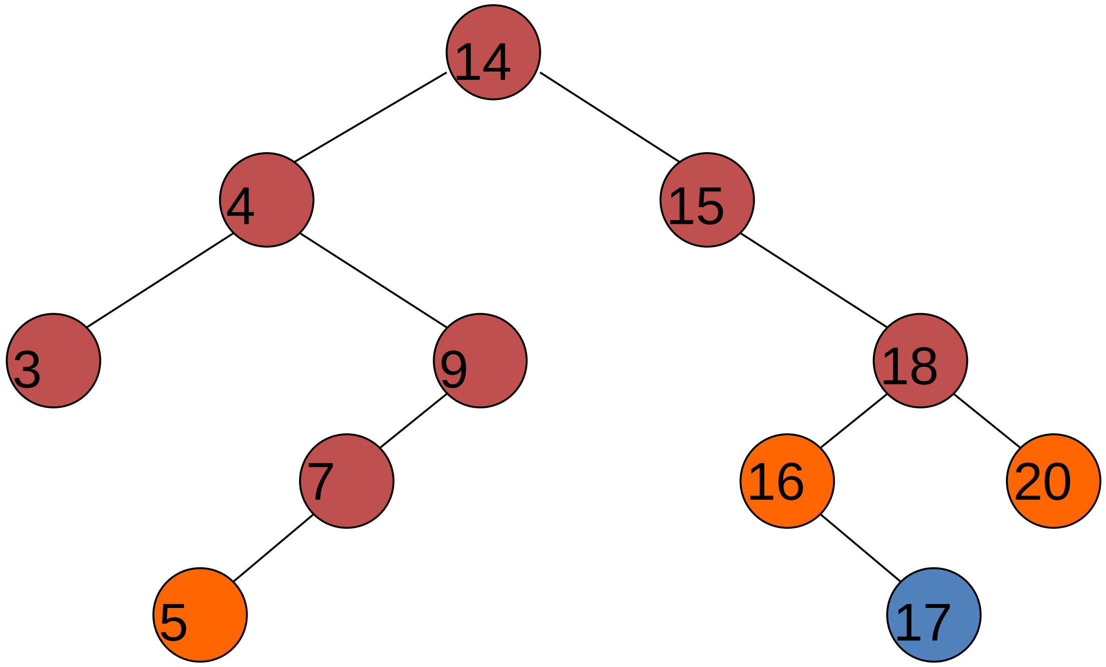
Level-order Traversal



Queue: 7 16 20

Output: 14 4 15 3 9 18

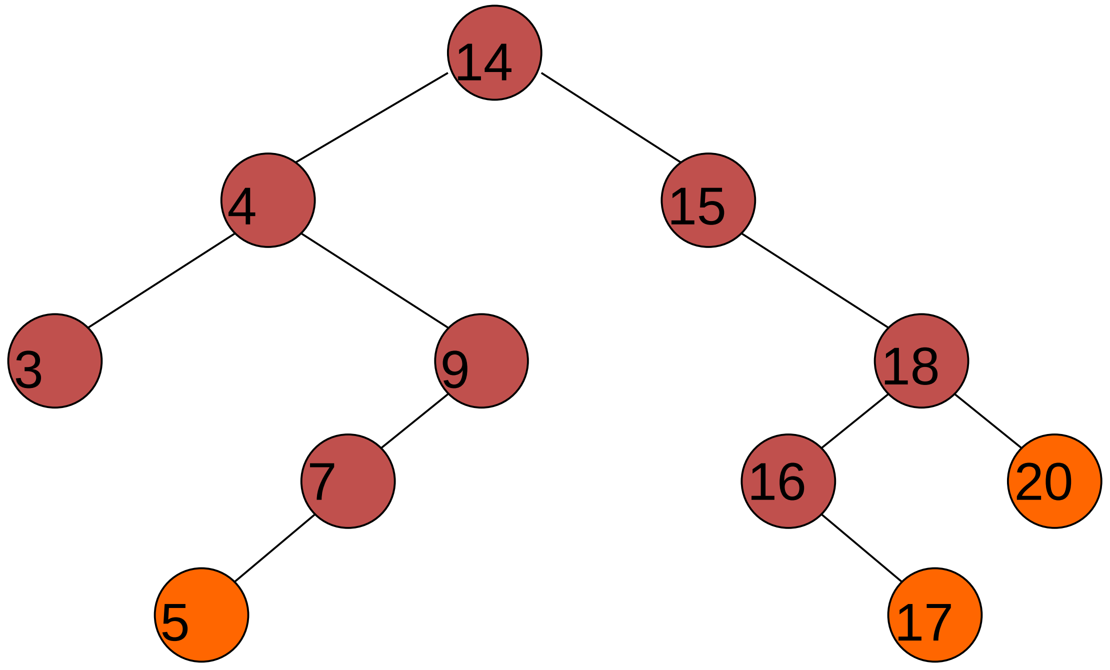
Level-order Traversal



Queue: 16 20 5

Output: 14 4 15 3   9 18   7

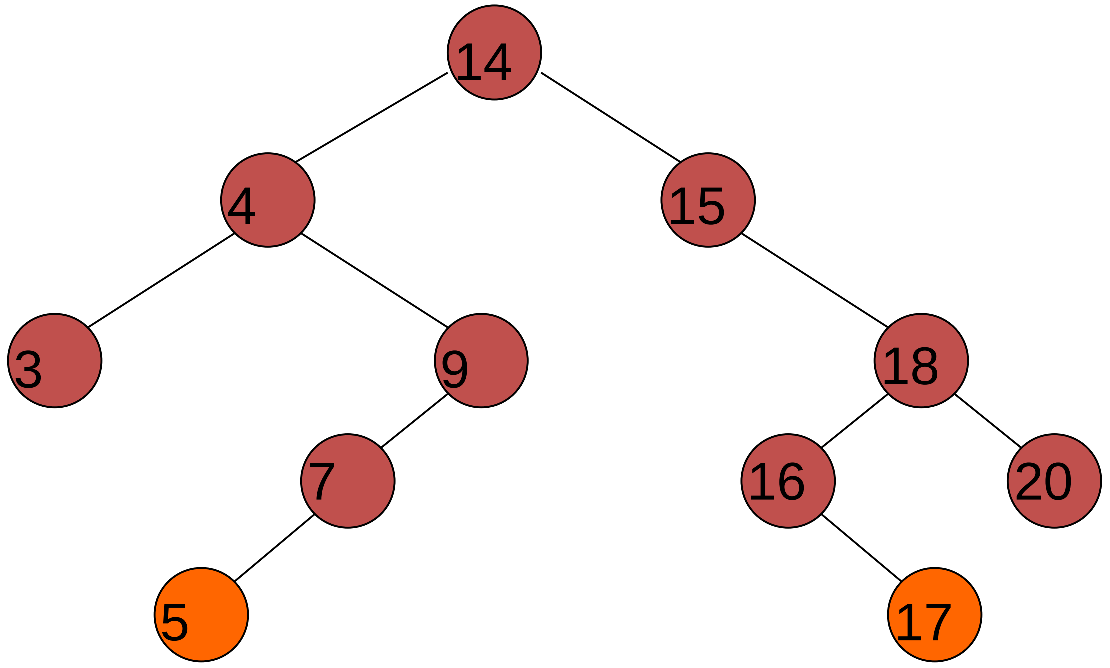
Level-order Traversal



Queue: 20 5 17

Output: 14 4 15 3   9 18   7 16

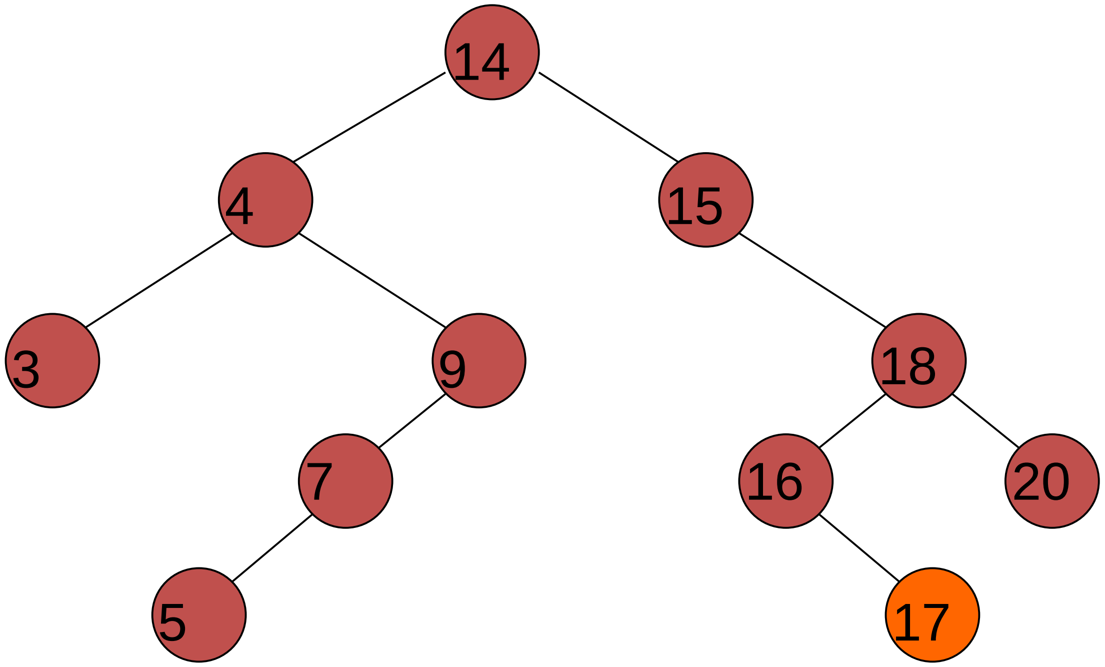
Level-order Traversal



Queue: 5 17

Output: 14 4 15 3 9 18 7 16 20

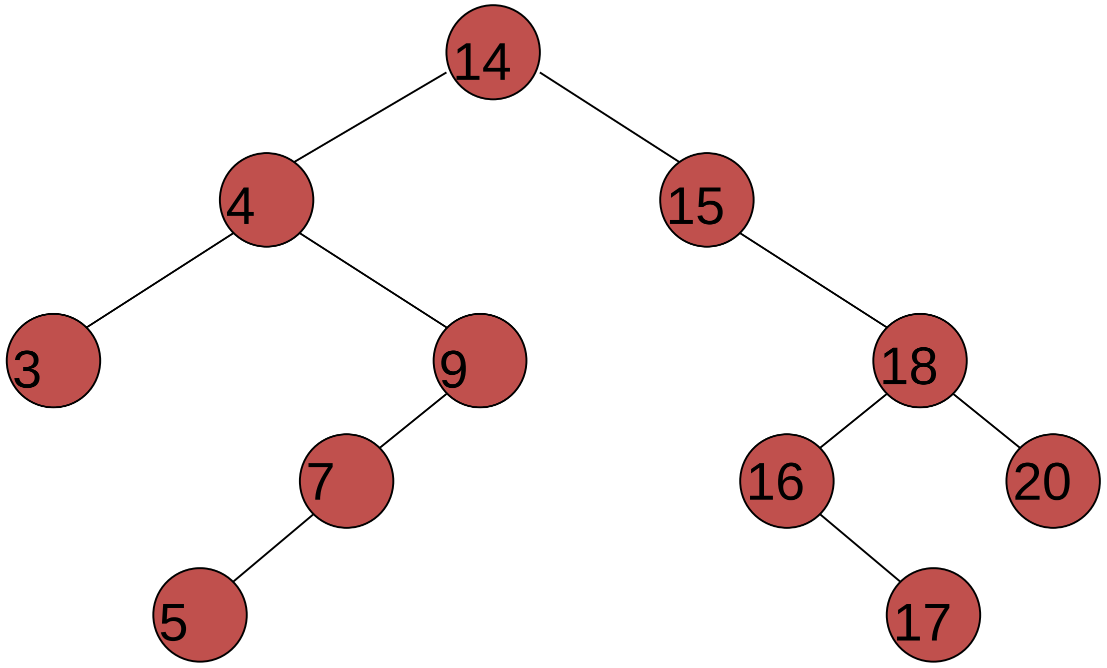
Level-order Traversal



Queue: 17

Output: 14 4 15 3 9 18 7 16 20 5

Level-order Traversal



Queue:

Output: 14 4 15 3 9 18 7 16 20 5 17