Data Structures and Object Oriented Programming

Lecture 14

Dr. Naveed Anwar Bhatti

Webpage: naveedanwarbhatti.github.io

Object-Oriented Programming in C++

Binary Search Trees

Traversal

`PreOrder()`, **`InOrder()`** and `PostOrder()`

BST – Traversal

Definition: “traversal” we mean visiting all the nodes in a tree.

Traversal strategies can be specified by the ordering of the three objects to visit: the **current node**, the **left subtree**, and the **right subtree**.

Most common tree traversal orders:

- Pre-order
- **In-order**
- Post-order

BST – Traversal

Inorder:

The ordering is: the left subtree, the current node, the right subtree.

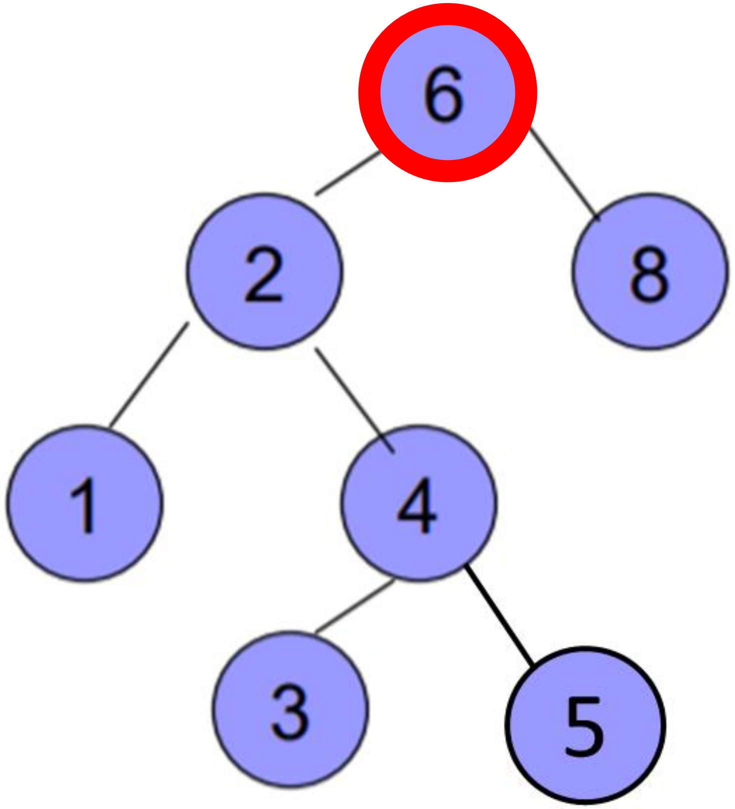
Preorder:

The ordering is: the current node, the left subtree, the right subtree.

Postorder:

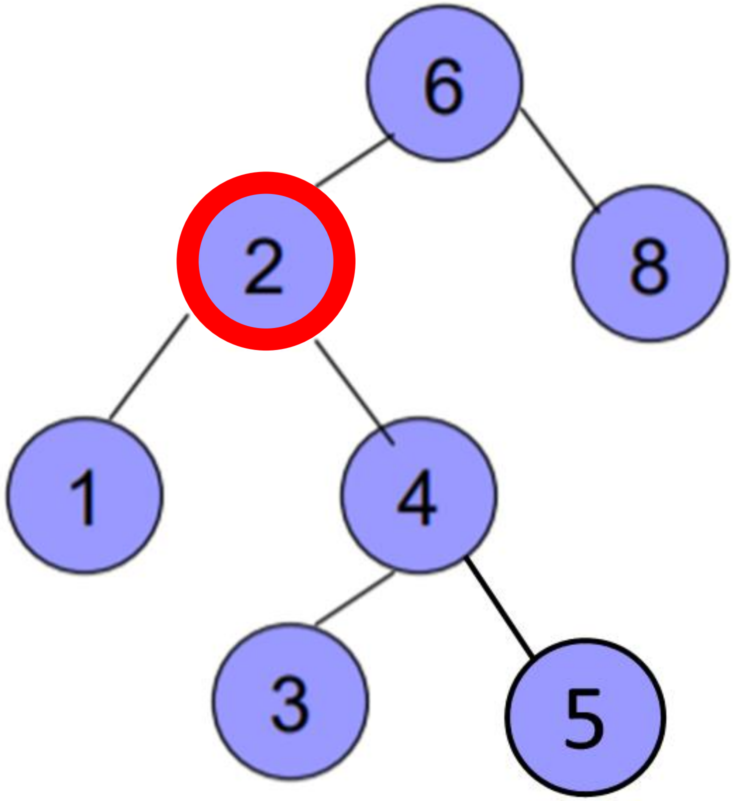
The ordering is: the left subtree, the right subtree, the current node.

Practice: Inorder Traversal



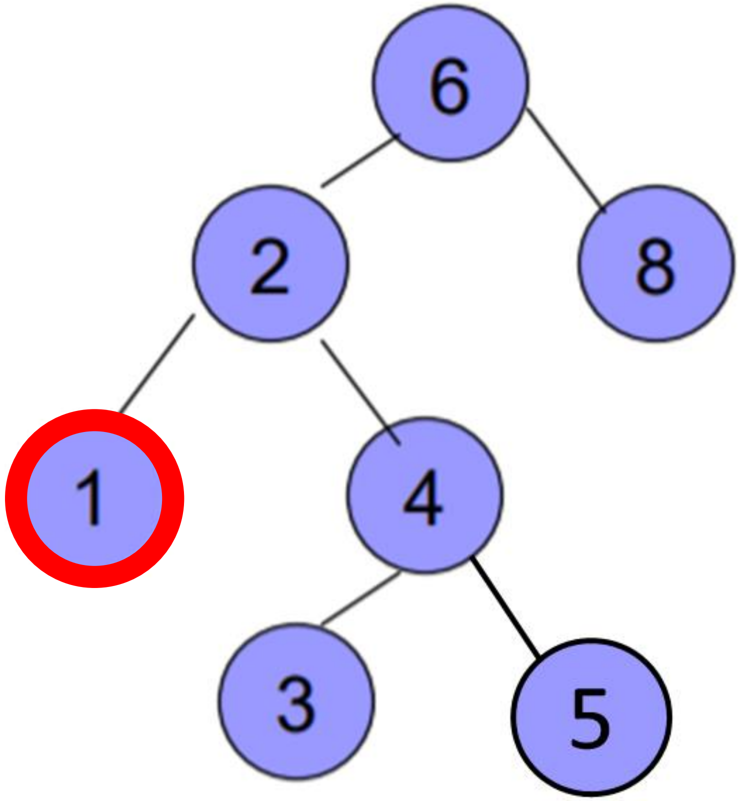
Left – Current – Right

Practice: Inorder Traversal



Left – Current – Right

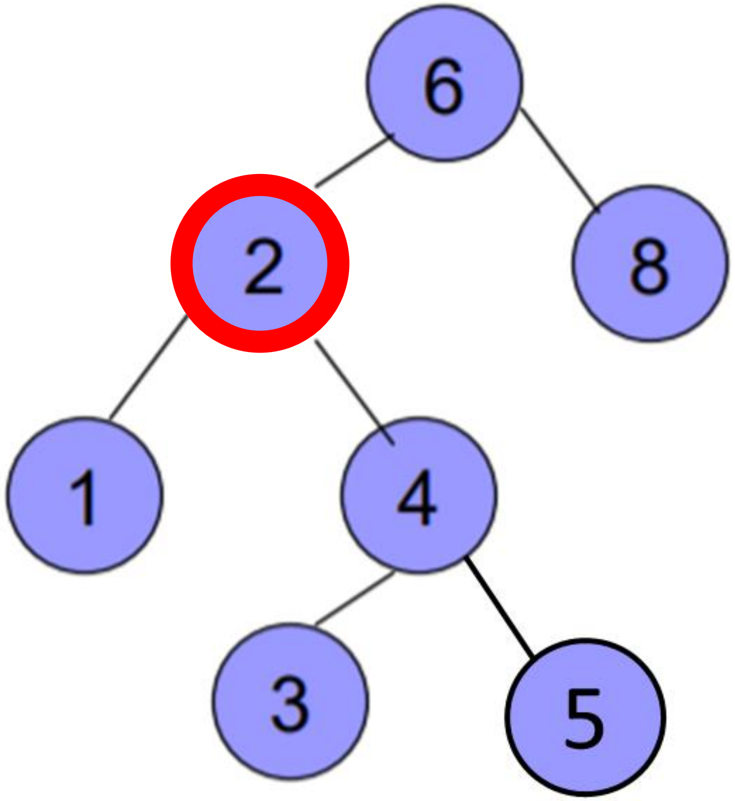
Practice: Inorder Traversal



Left – Current – Right

1

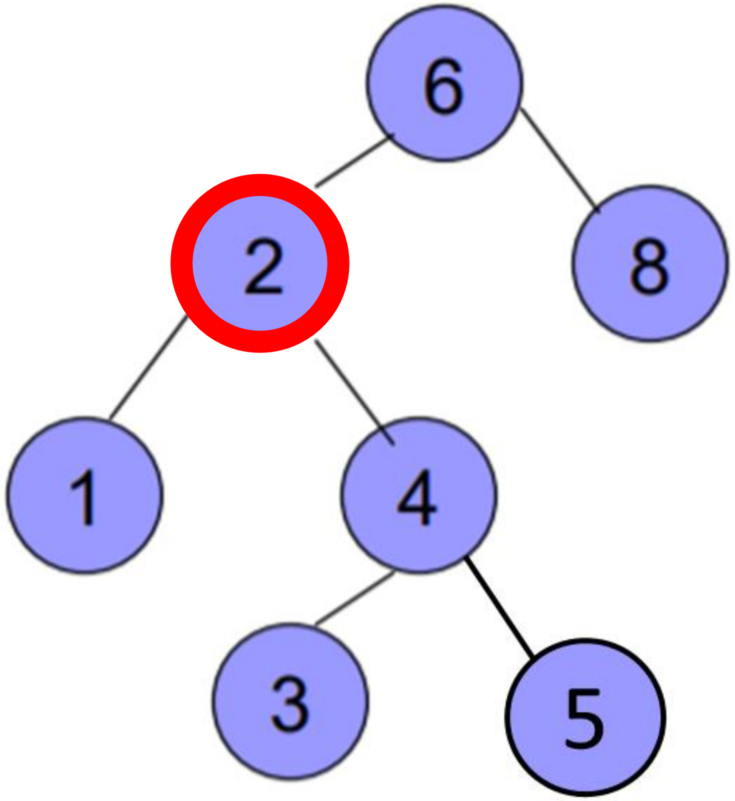
Practice: Inorder Traversal



Left – Current – Right

1 2

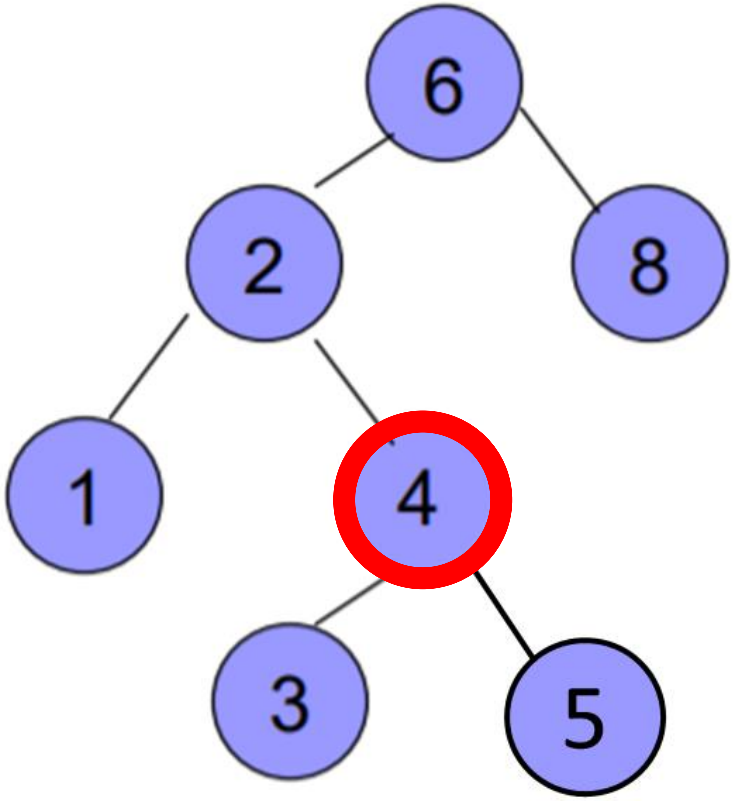
Practice: Inorder Traversal



Left – Current – Right

1 2

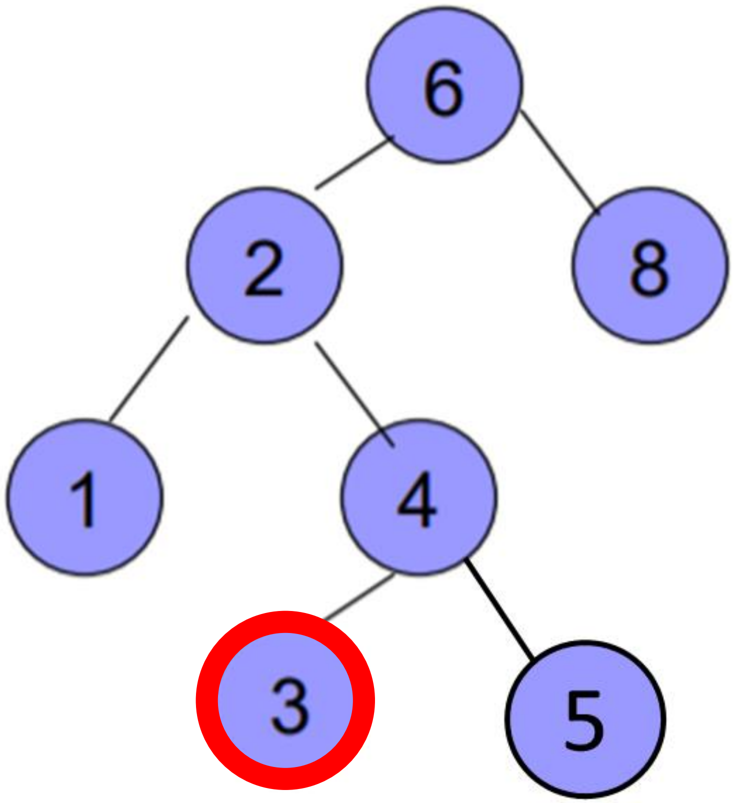
Practice: Inorder Traversal



Left – Current – Right

1 2

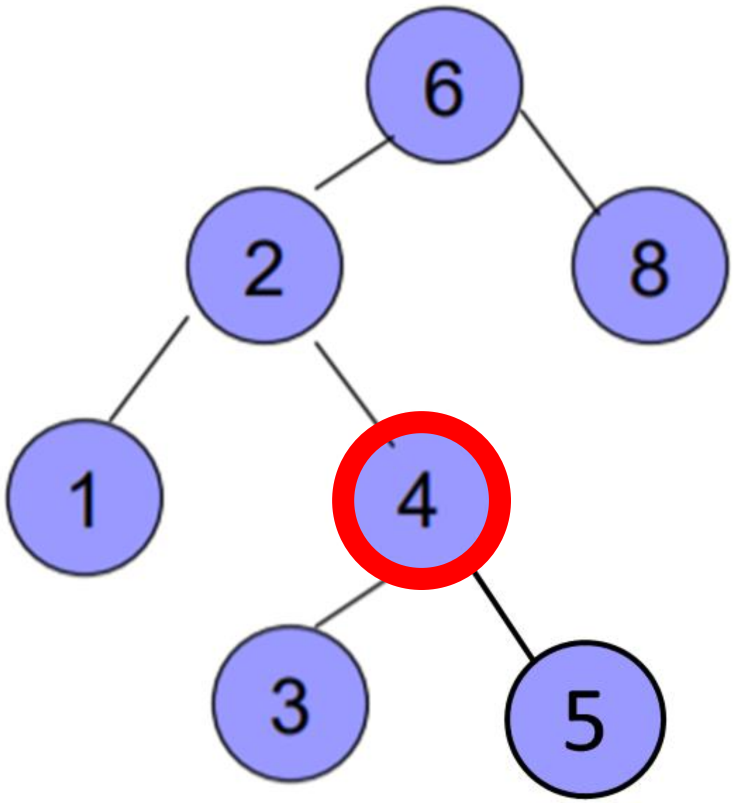
Practice: Inorder Traversal



Left – Current – Right

1 2 3

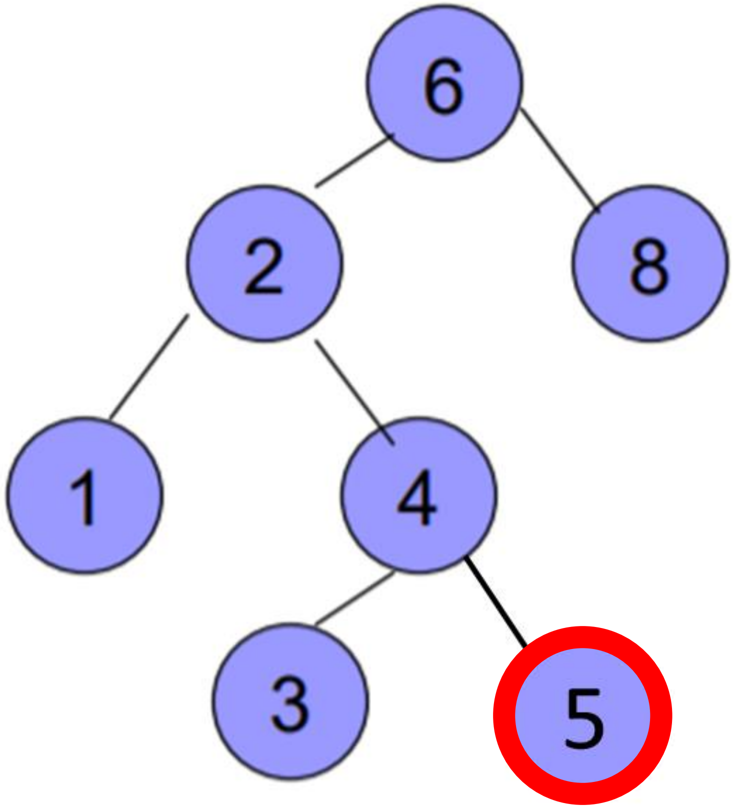
Practice: Inorder Traversal



Left – Current – Right

1 2 3 4

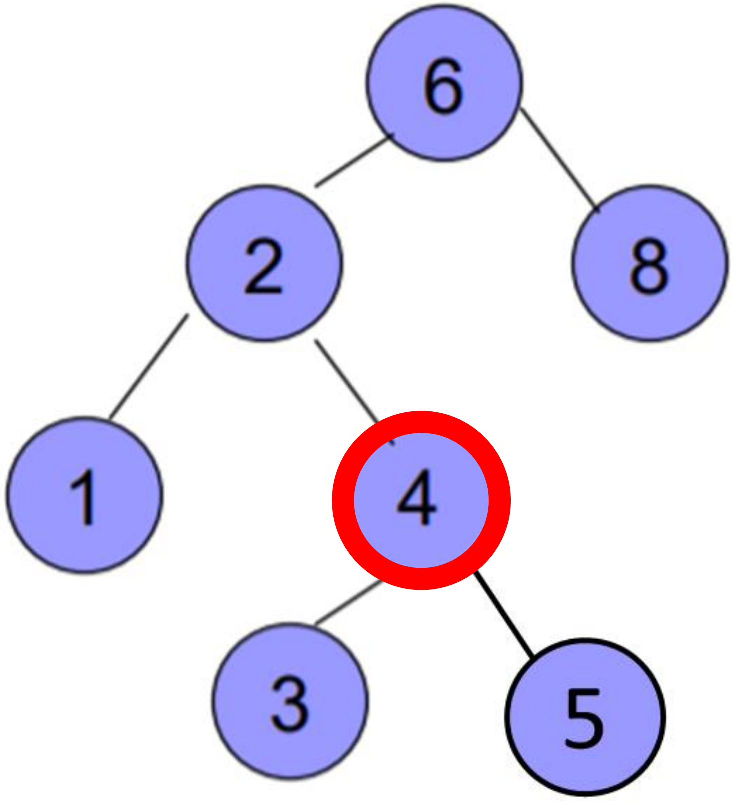
Practice: Inorder Traversal



Left – Current – Right

1 2 3 4 5

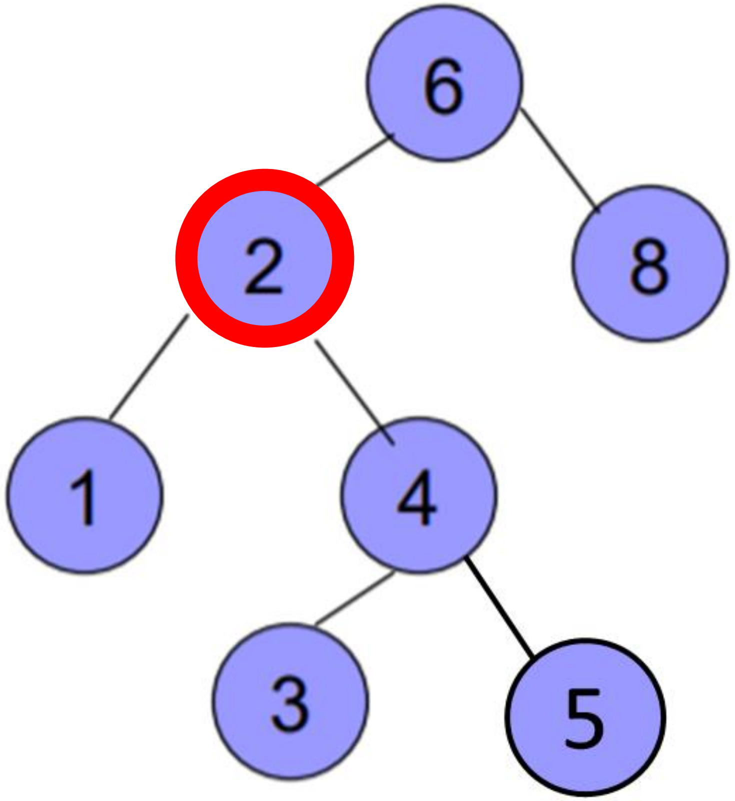
Practice: Inorder Traversal



Left – Current – Right

1 2 3 4 5

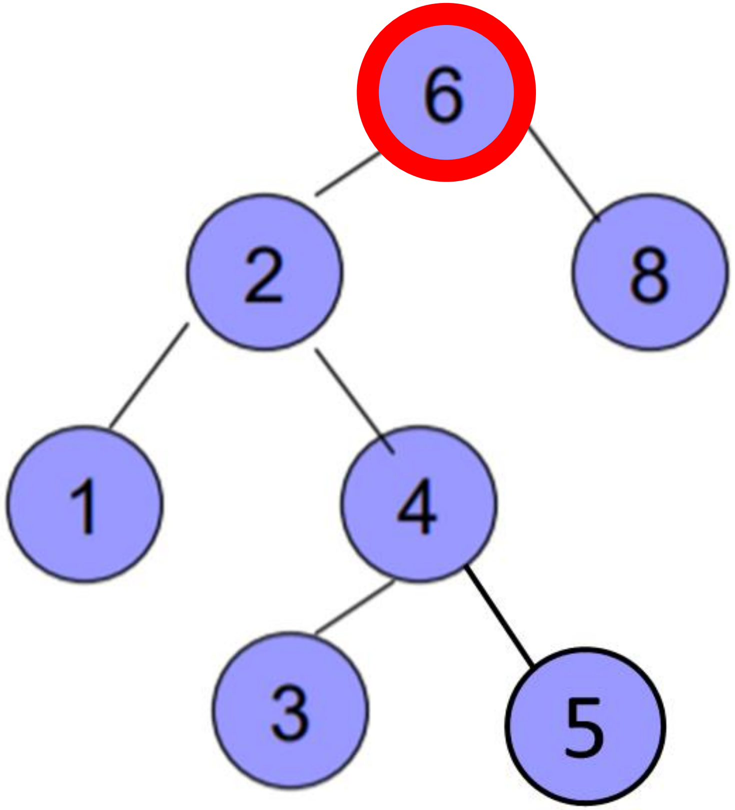
Practice: Inorder Traversal



Left – Current – Right

1 2 3 4 5

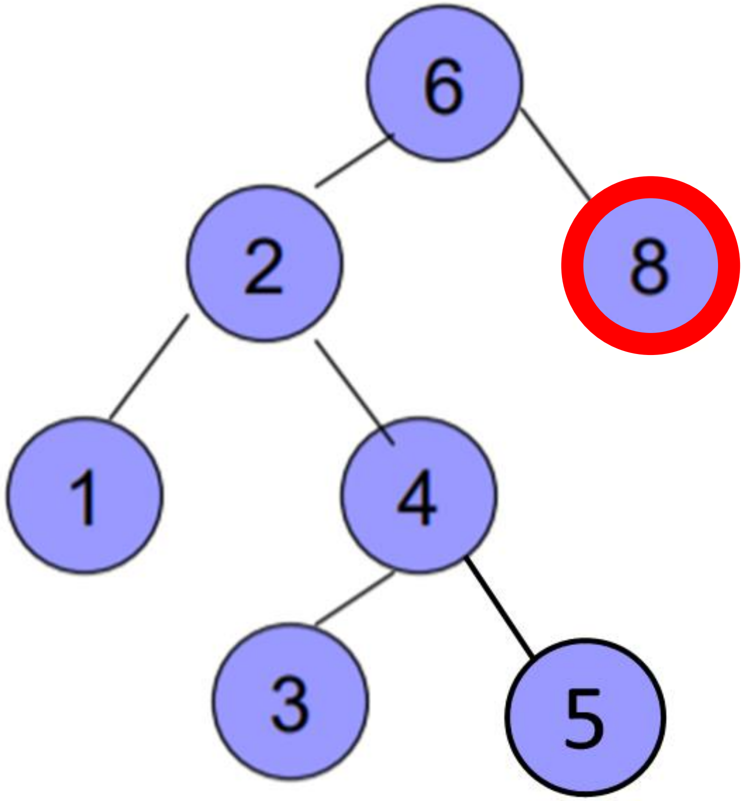
Practice: Inorder Traversal



Left – Current – Right

1 2 3 4 5 6

Practice: Inorder Traversal

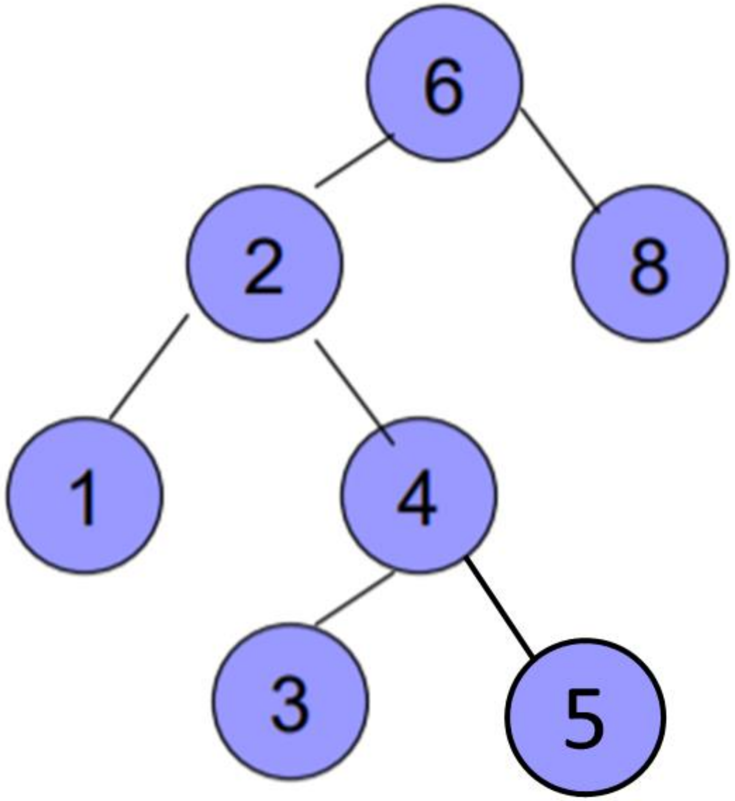


Left – Current – Right

1 2 3 4 5 6 8

Pre-Order

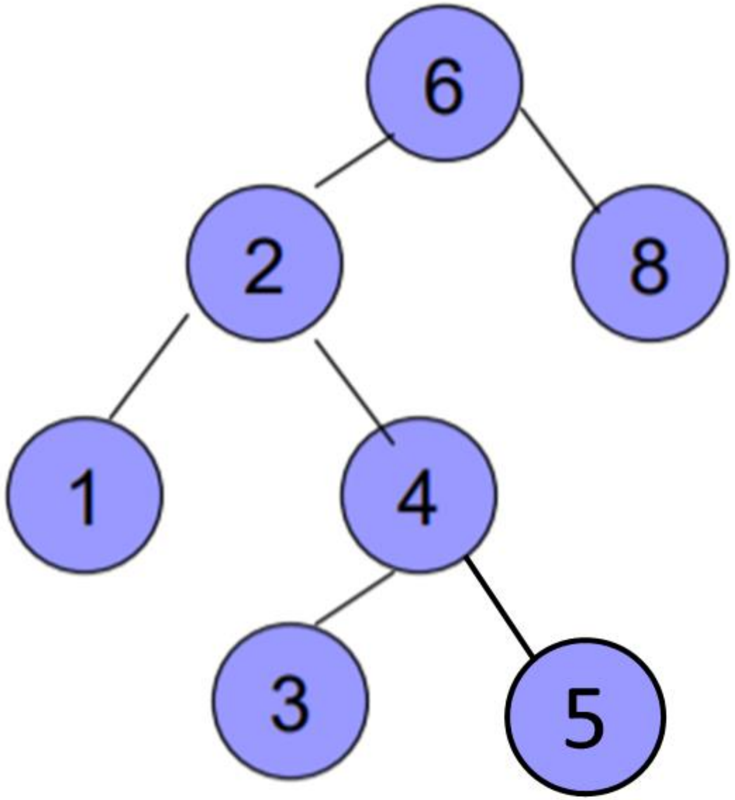
Current – Left – Right



6 2 1 4 3 5 8

Post-Order

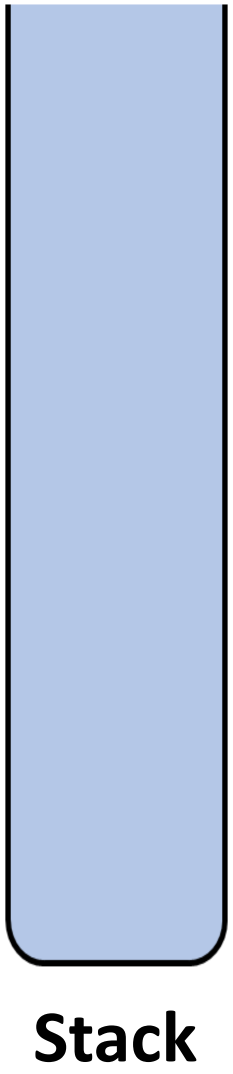
Left – Right – Current

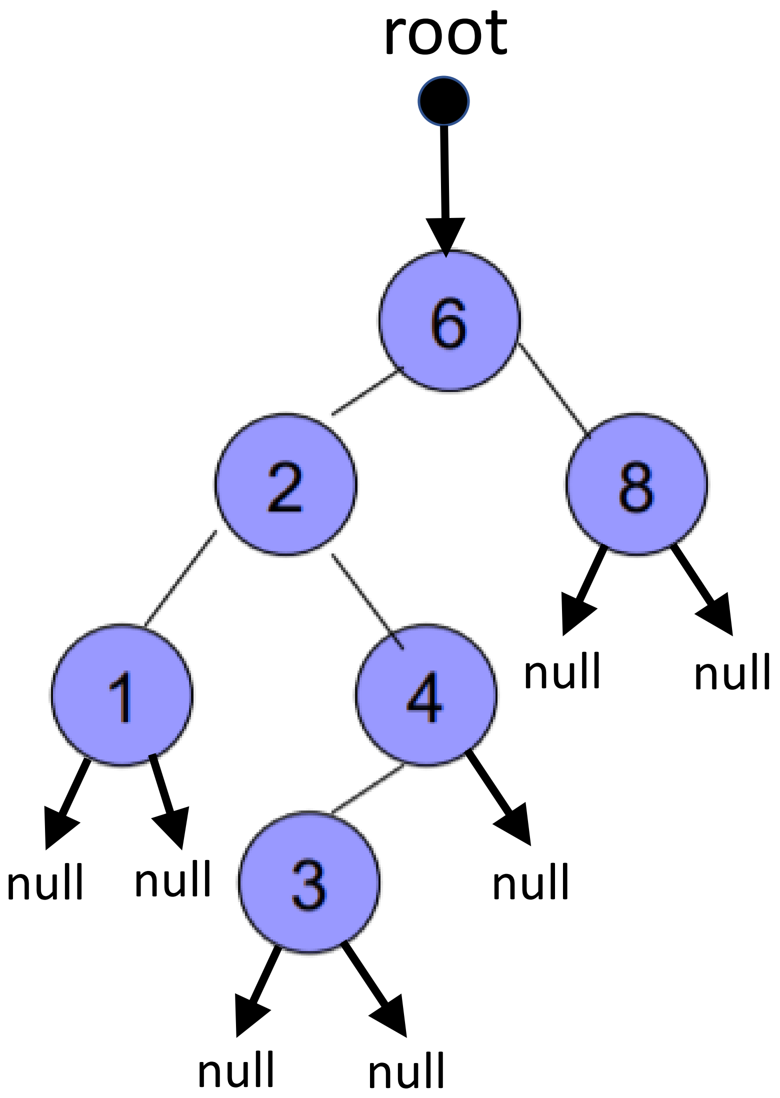


1 3 5 4 2 8 6

BST - Inorder Traversal using Stack

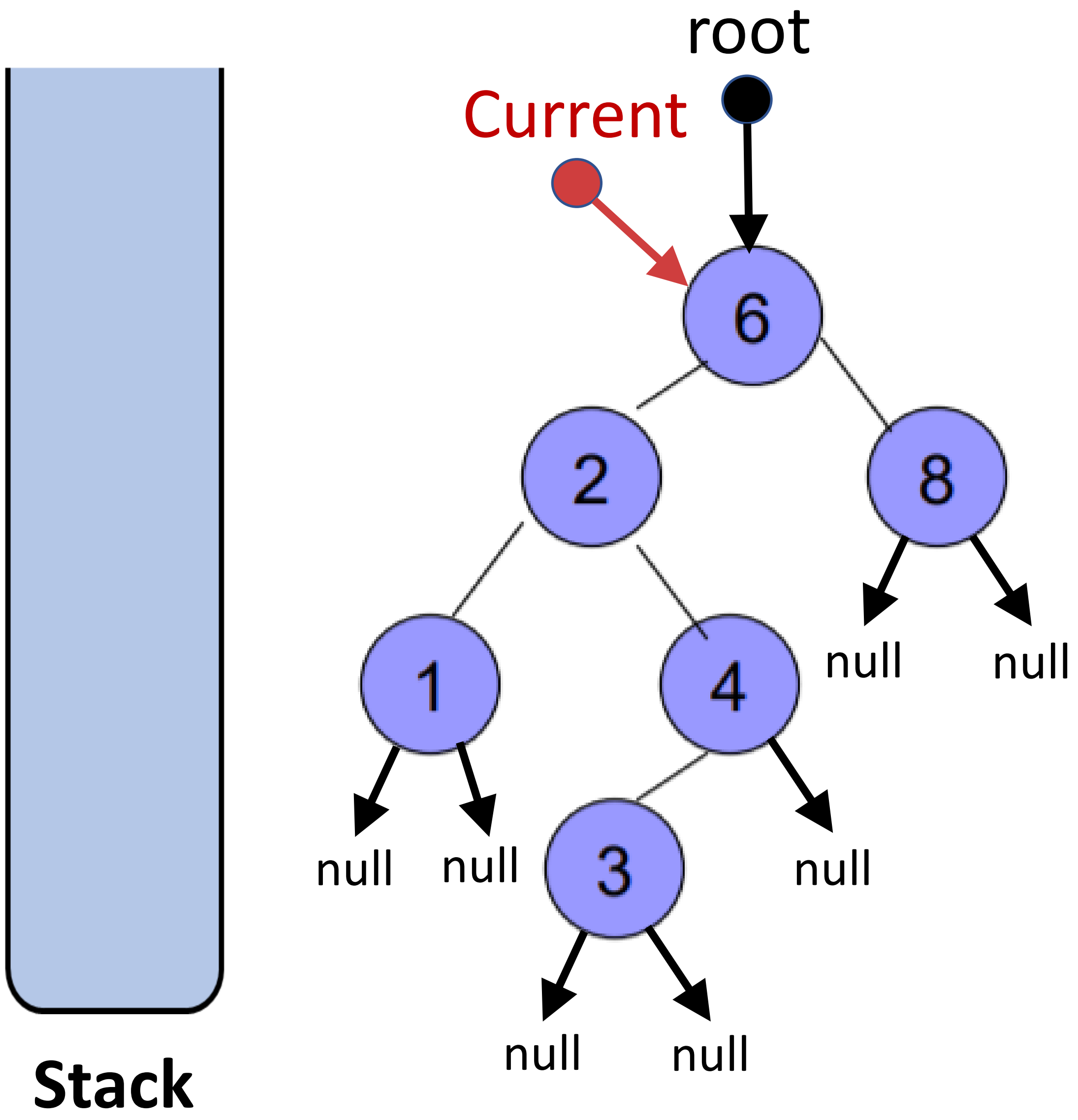
```
void  inorder()
 {



 }
```





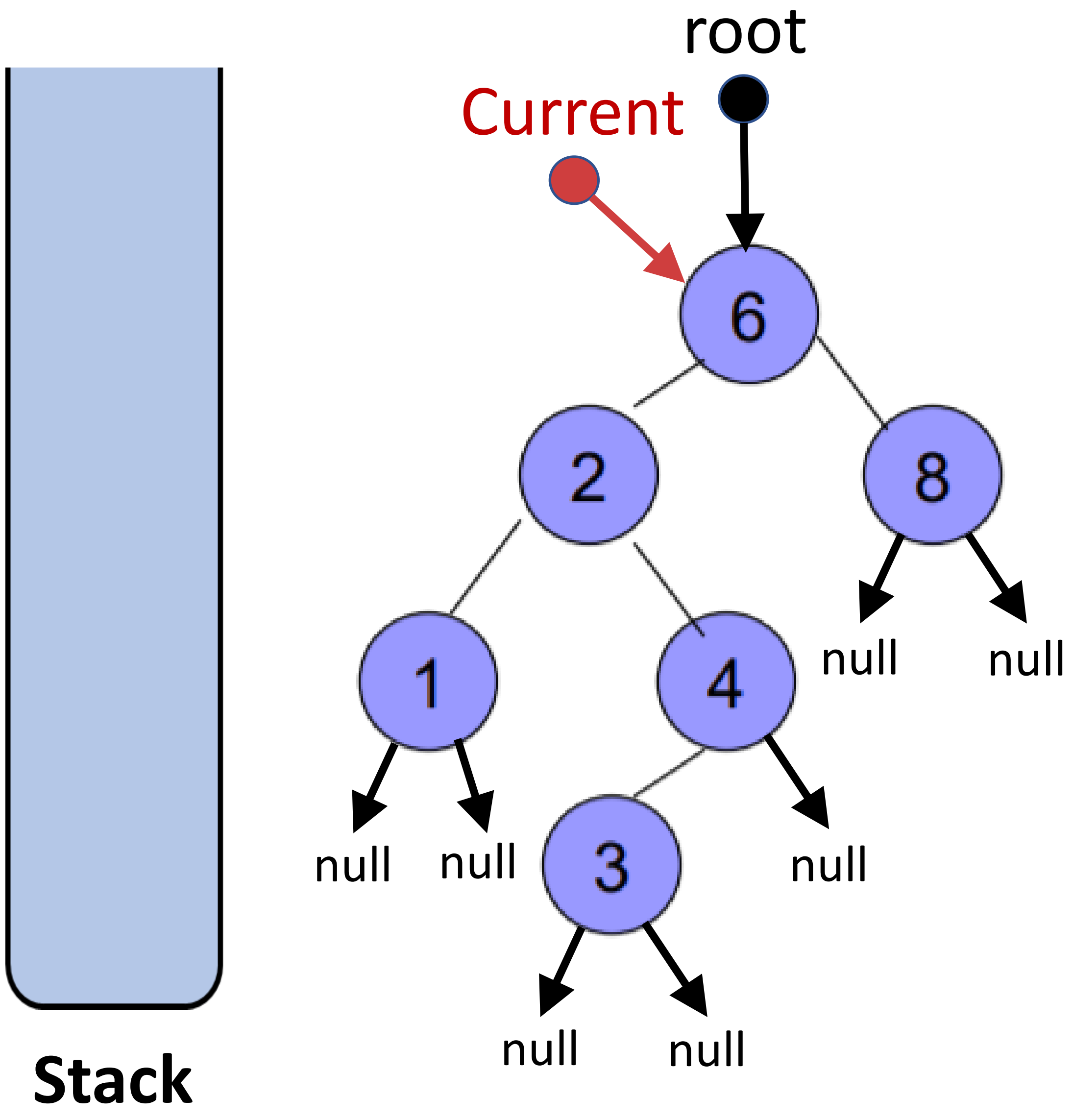
BST - Inorder Traversal using Stack

```
void  inorder()
 {
    node* current=root;



 }
```



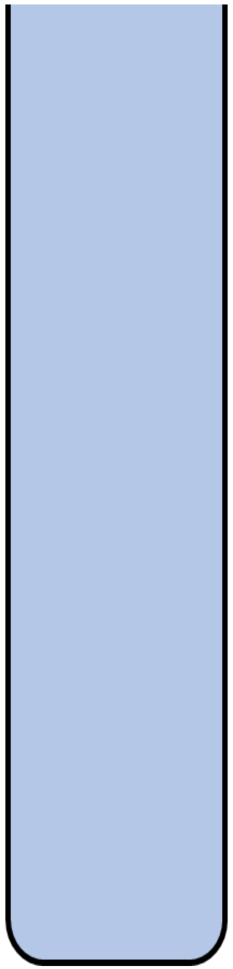
BST - Inorder Traversal using Stack

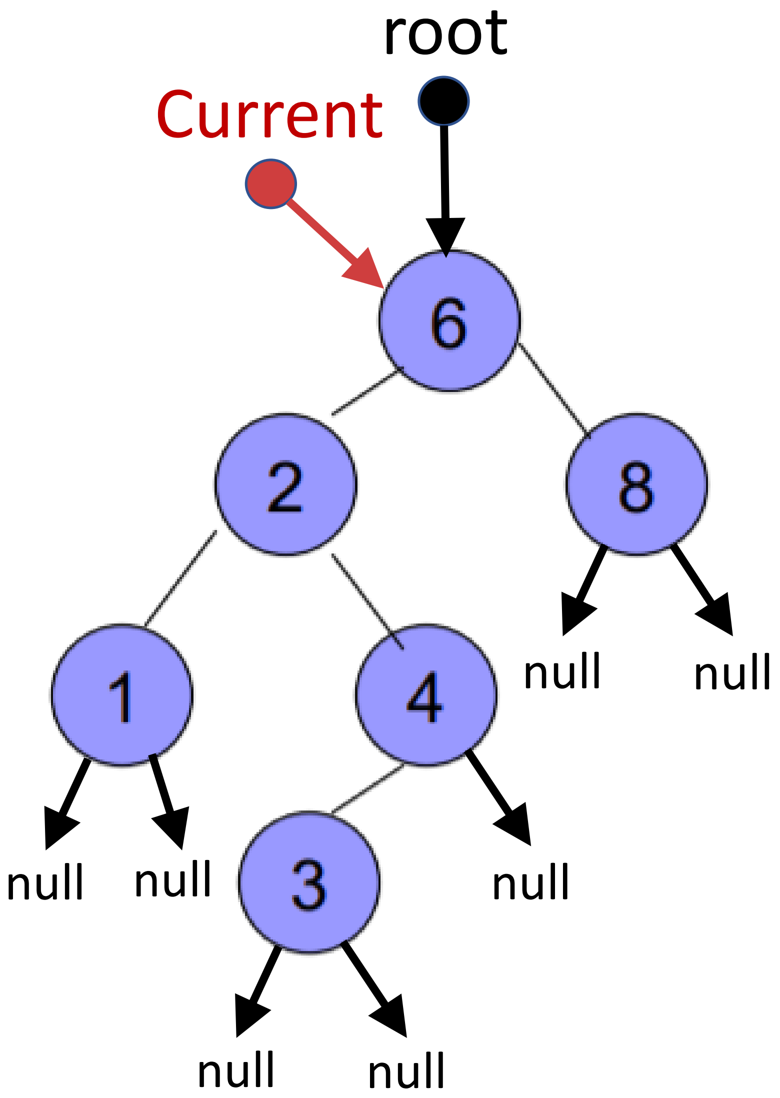
```
void  inorder()
 {
        node* current=root;

        while(1)

            if (current != NULL)
                    stack.push(current)
                    current = current->left
            else if  (!stack.empty() )
                    current = stack.top()
                    stack.pop()
                    cout<< current->data;
                    current = current->right
            else
                    return
  }
```



BST - Inorder Traversal using Stack

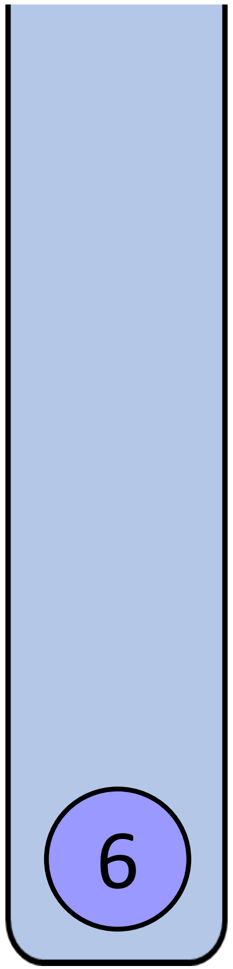
```
void  inorder()
 {
      node* current=root;

      while(1)

         if (current != NULL)
               stack.push(current)
               current = current->left
         else if  (!stack.empty() )
               current = stack.top()
               stack.pop()
               cout<< current->data;
               current = current->right
         else
               return
 }
```



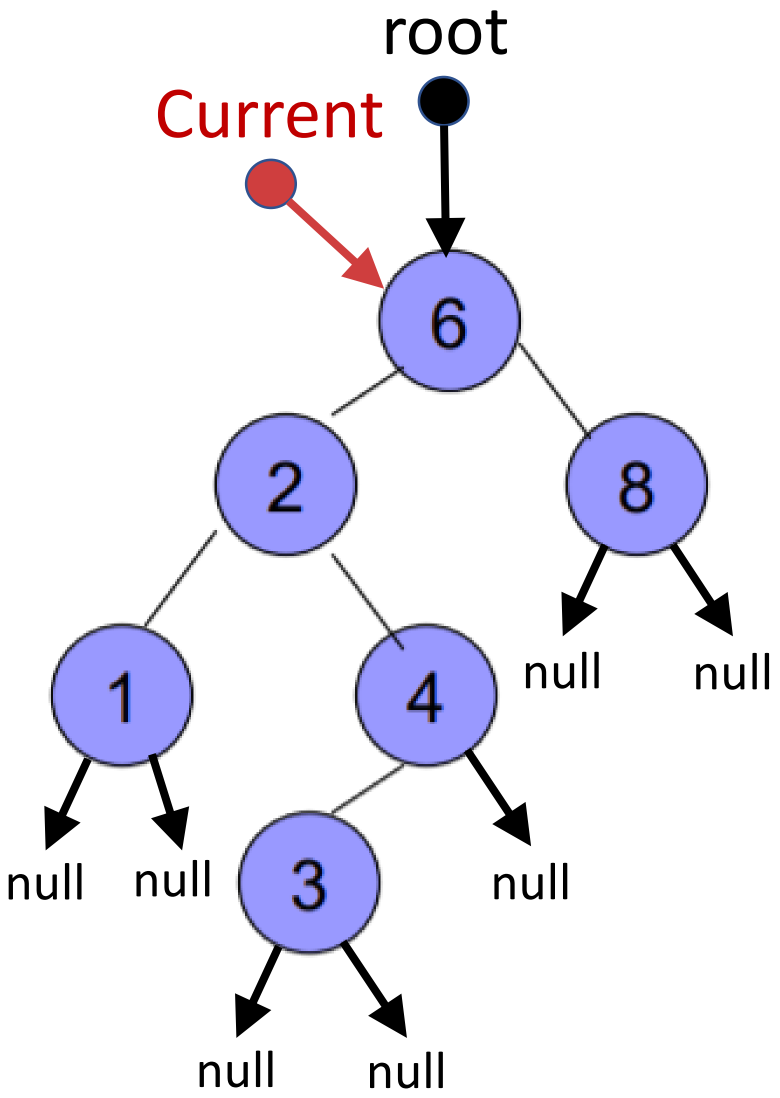


BST - Inorder Traversal using Stack

```
void  inorder()
 {
        node* current=root;

        while(1)

            if (current != NULL)
        ⇨   stack.push(current)
                current = current->left
            else if  (!stack.empty() )
                current = stack.top()
                stack.pop()
                cout<< current->data;
                current = current->right
            else
                return
 }
```

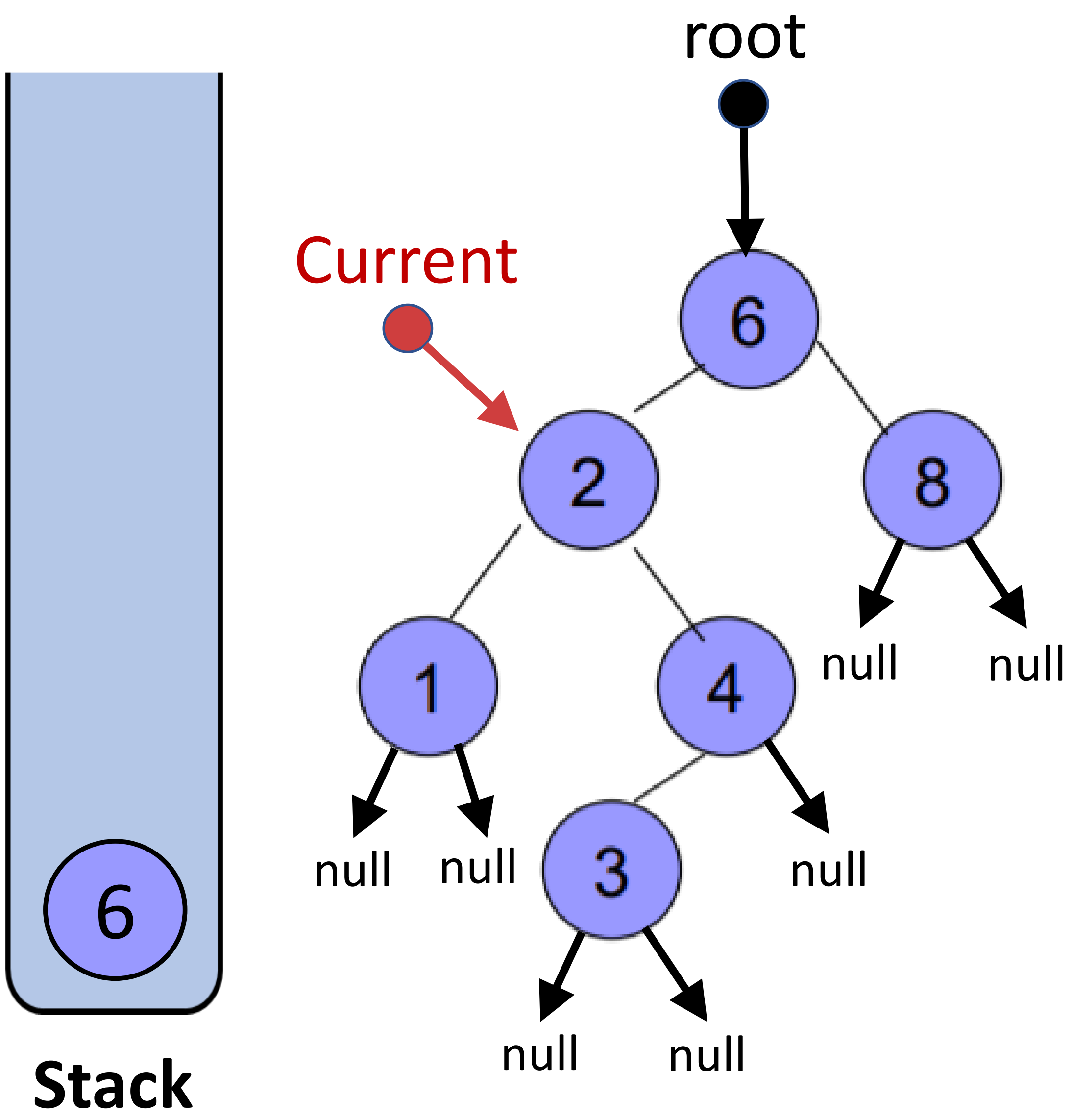


Stack



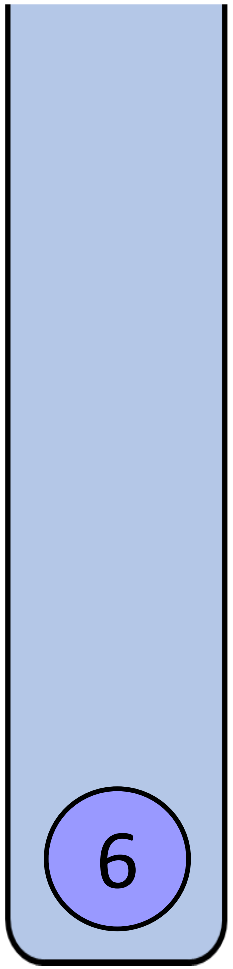
BST - Inorder Traversal using Stack

```
void  inorder()
 {
        node* current=root;

        while(1)

             if (current != NULL)
                    stack.push(current)
                    current = current->left
             else if  (!stack.empty() )
                    current = stack.top()
                    stack.pop()
                    cout<< current->data;
                    current = current->right
             else
                    return
 }
```

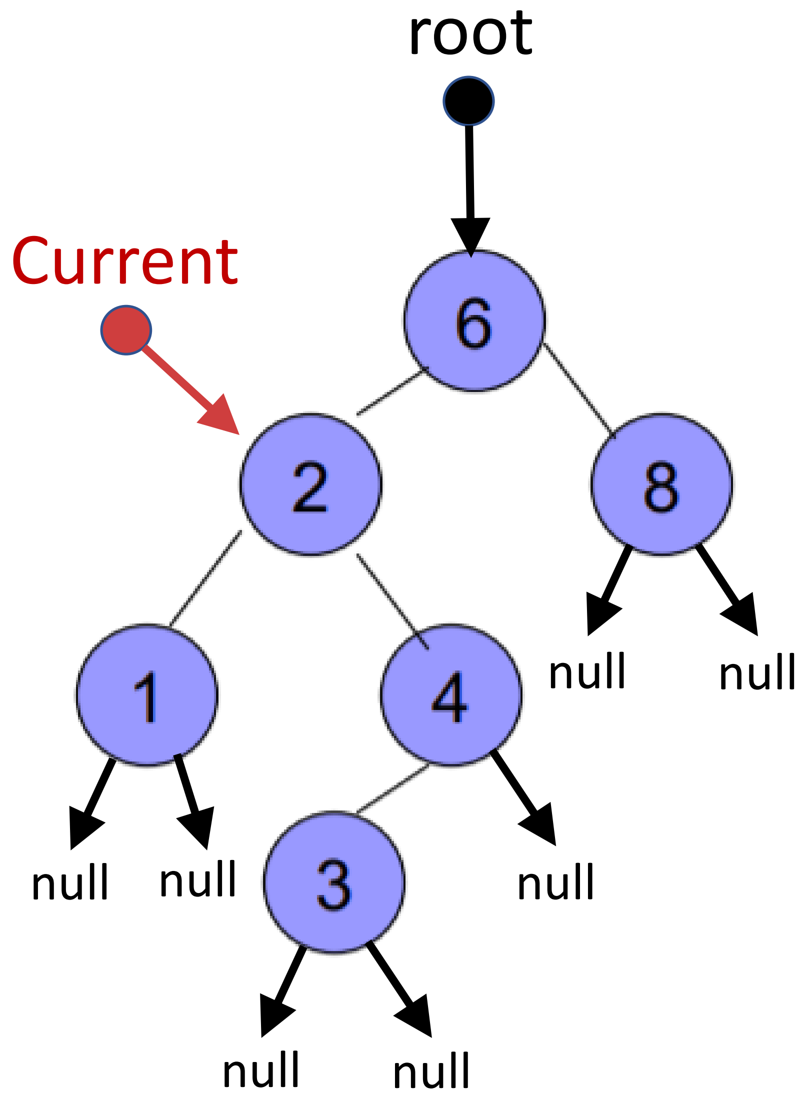


BST - Inorder Traversal using Stack

```
void  inorder()
 {
        node* current=root;

        while(1)

          if (current != NULL)
                 stack.push(current)
                 current = current->left
          else if  (!stack.empty() )
                 current = stack.top()
                 stack.pop()
                 cout<< current->data;
                 current = current->right
          else
                 return
 }
```

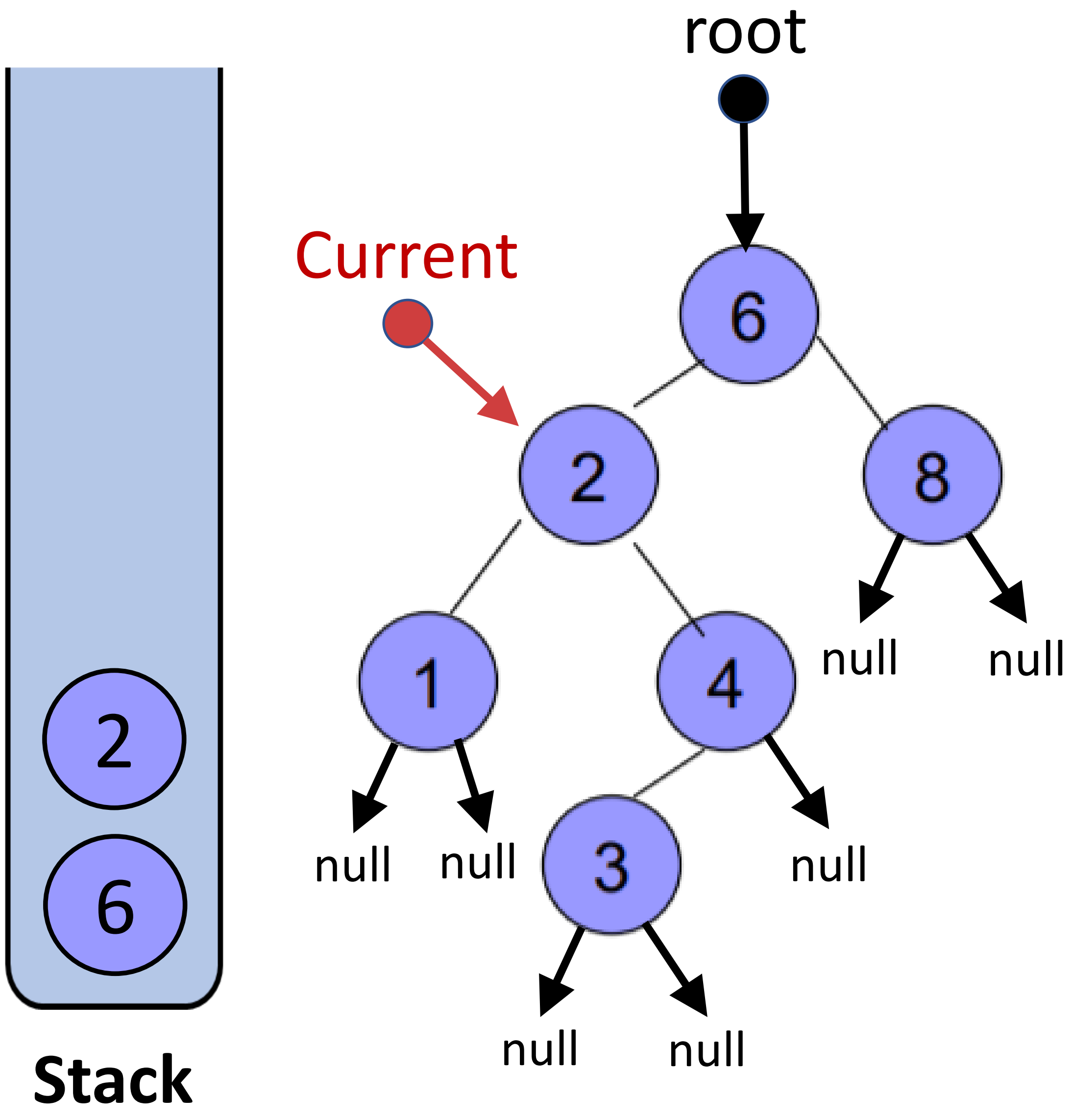


Stack



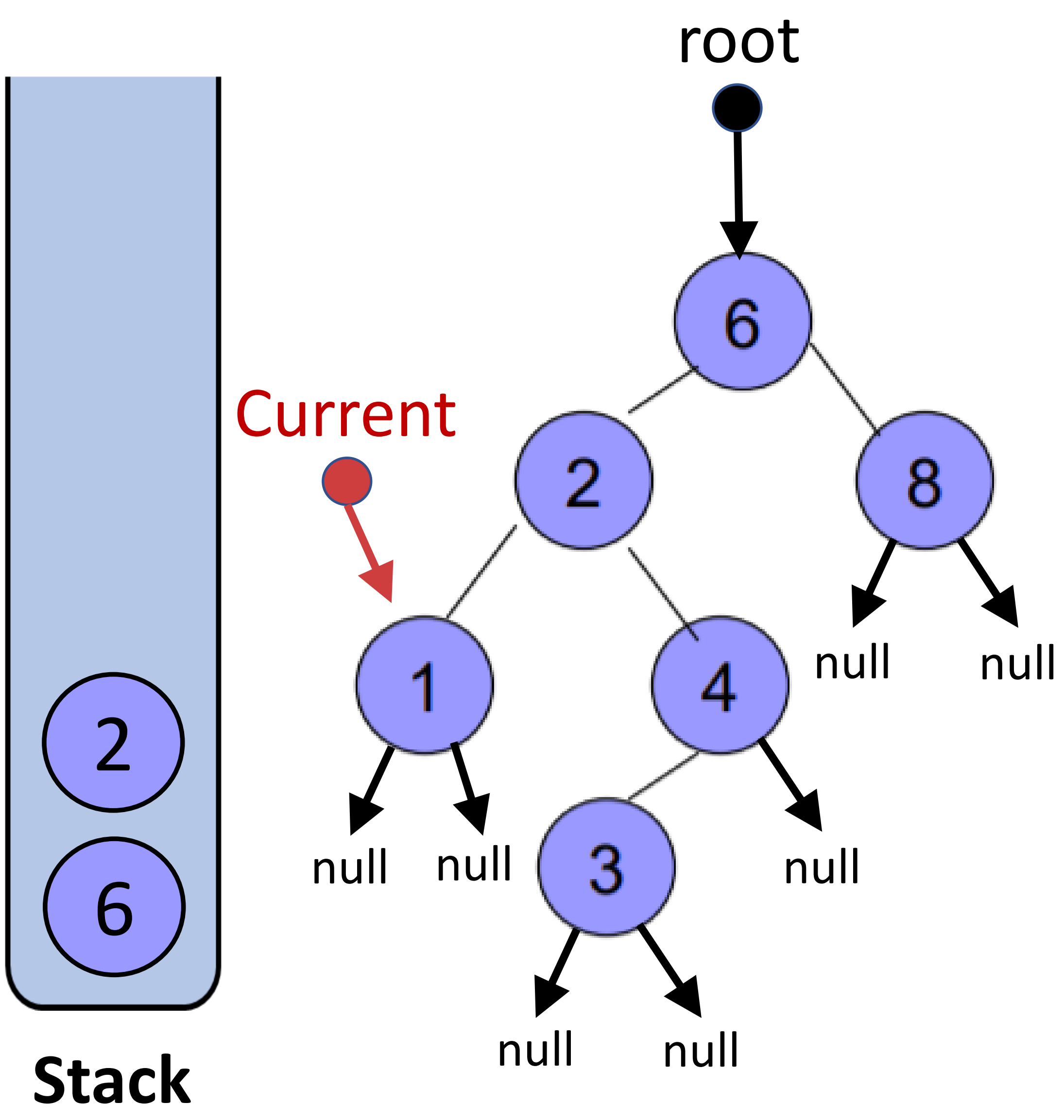
BST - Inorder Traversal using Stack

```
void  inorder()
 {
        node* current=root;

        while(1)

            if (current != NULL)
          ⇨ stack.push(current)
                   current = current->left
            else if  (!stack.empty() )
                   current = stack.top()
                   stack.pop()
                   cout<< current->data;
                   current = current->right
            else
                   return
  }
```



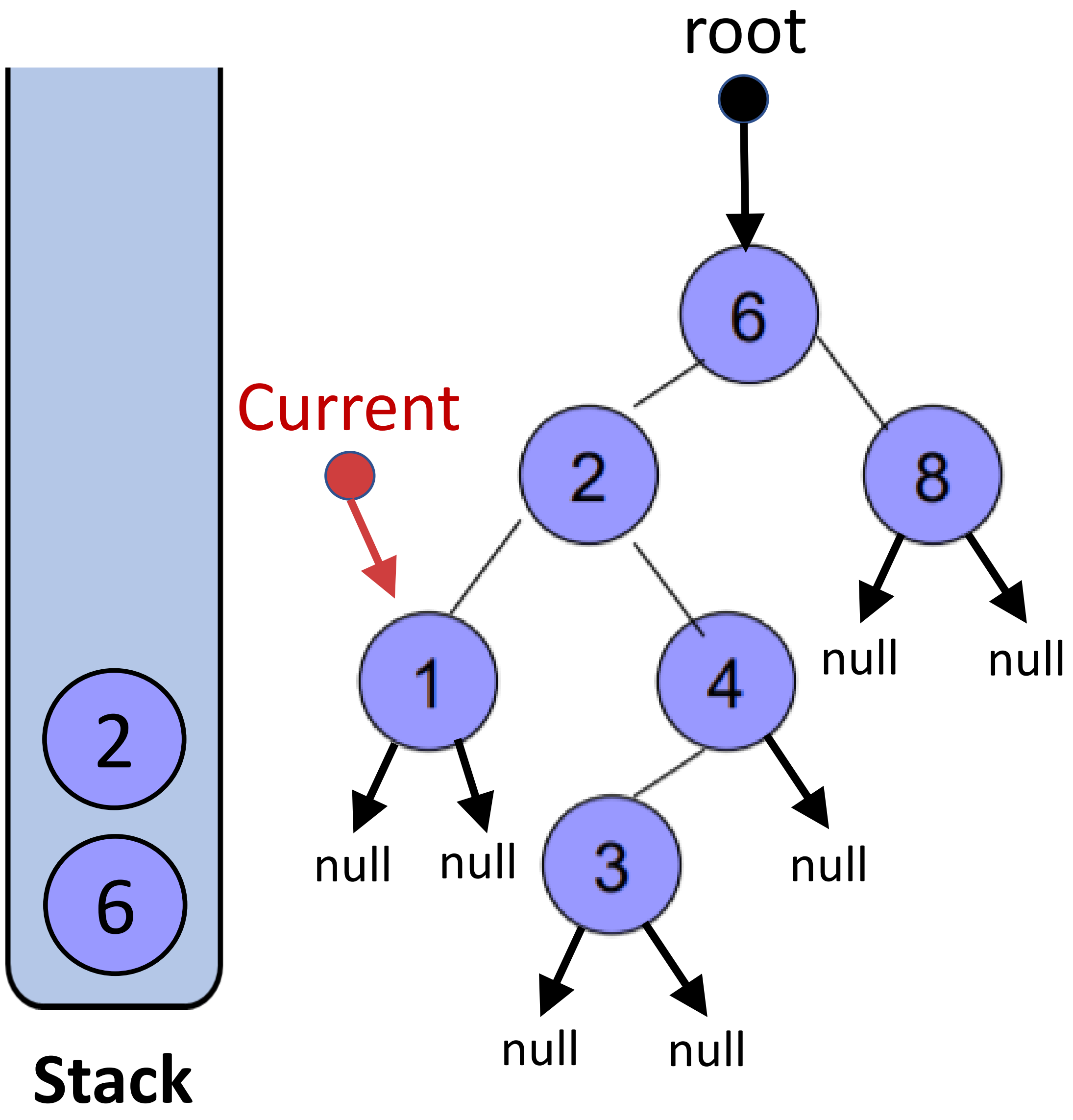
BST - Inorder Traversal using Stack

```
void  inorder()
 {
        node* current=root;

        while(1)

            if (current != NULL)
                    stack.push(current)
         ⇨          current = current->left
            else if  (!stack.empty() )
                    current = stack.top()
                    stack.pop()
                    cout<< current->data;
                    current = current->right
            else
                    return
  }
```



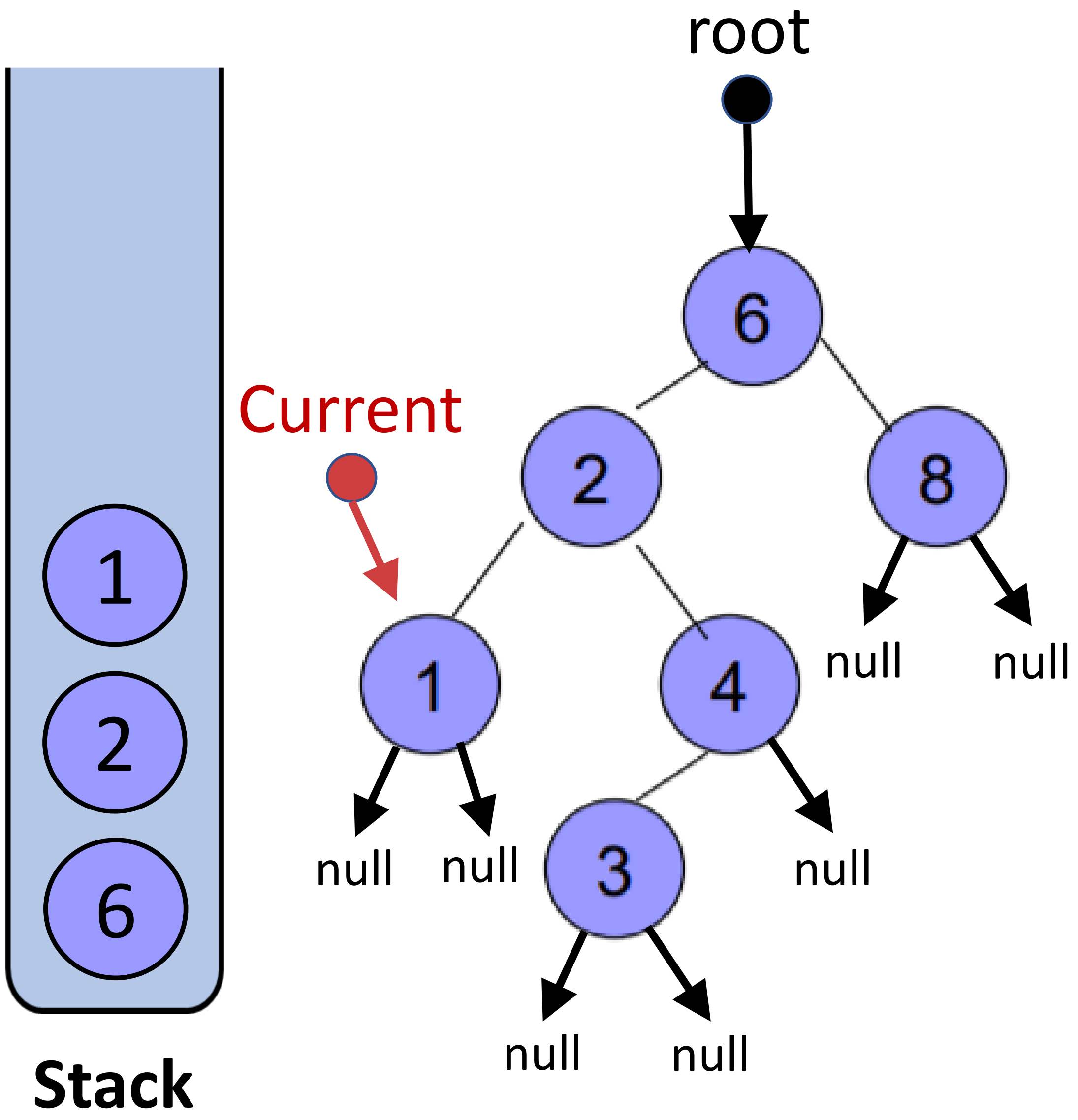
BST - Inorder Traversal using Stack

```
void  inorder()
 {
      node* current=root;

      while(1)

        if (current != NULL)
              stack.push(current)
              current = current->left
        else if  (!stack.empty() )
              current = stack.top()
              stack.pop()
              cout<< current->data;
              current = current->right
        else
              return
 }
```



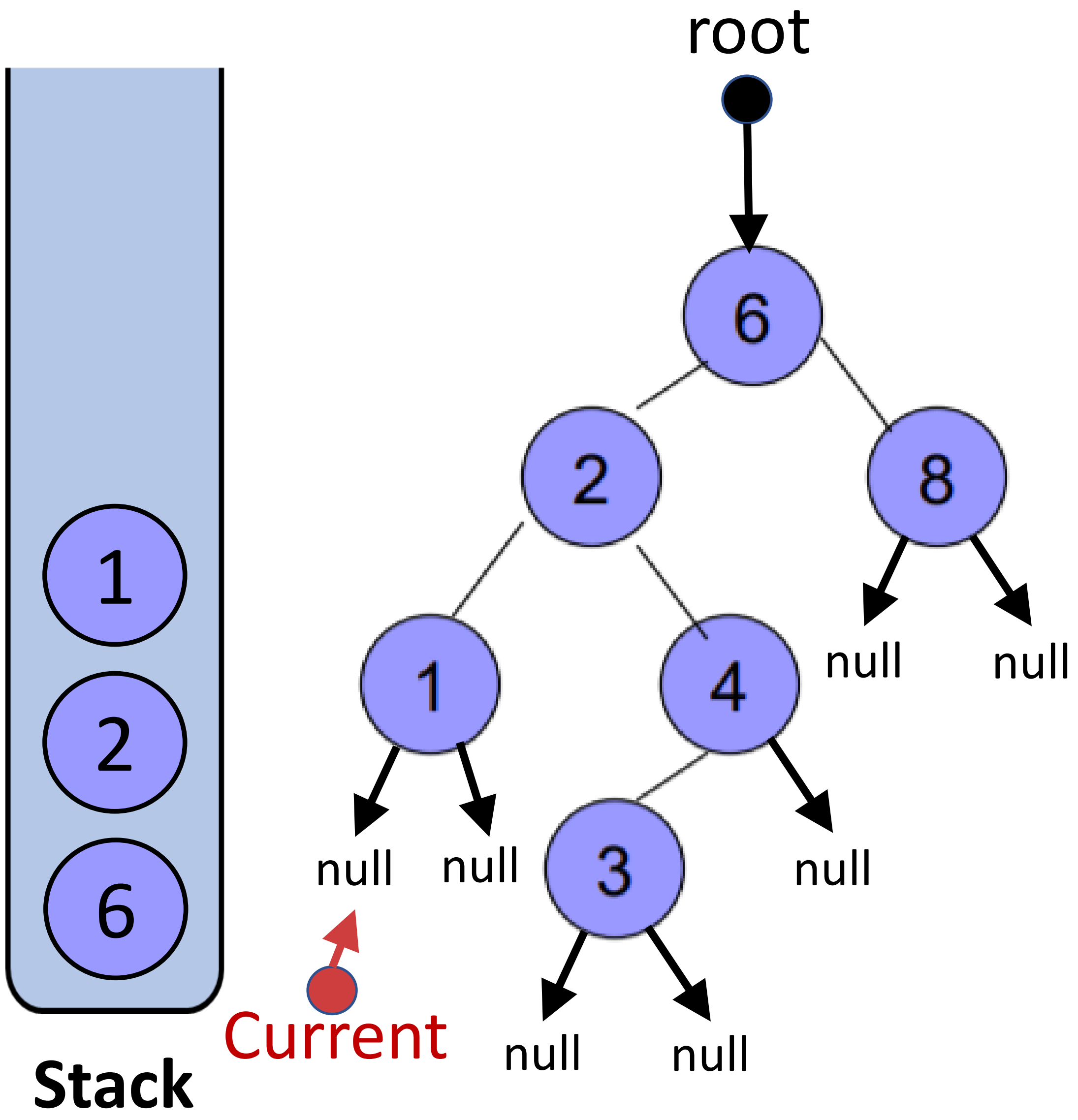
BST - Inorder Traversal using Stack

```
void  inorder()
 {
        node* current=root;

        while(1)

            if (current != NULL)
                   stack.push(current)
                   current = current->left
            else if  (!stack.empty() )
                   current = stack.top()
                   stack.pop()
                   cout<< current->data;
                   current = current->right
            else
                   return
  }
```



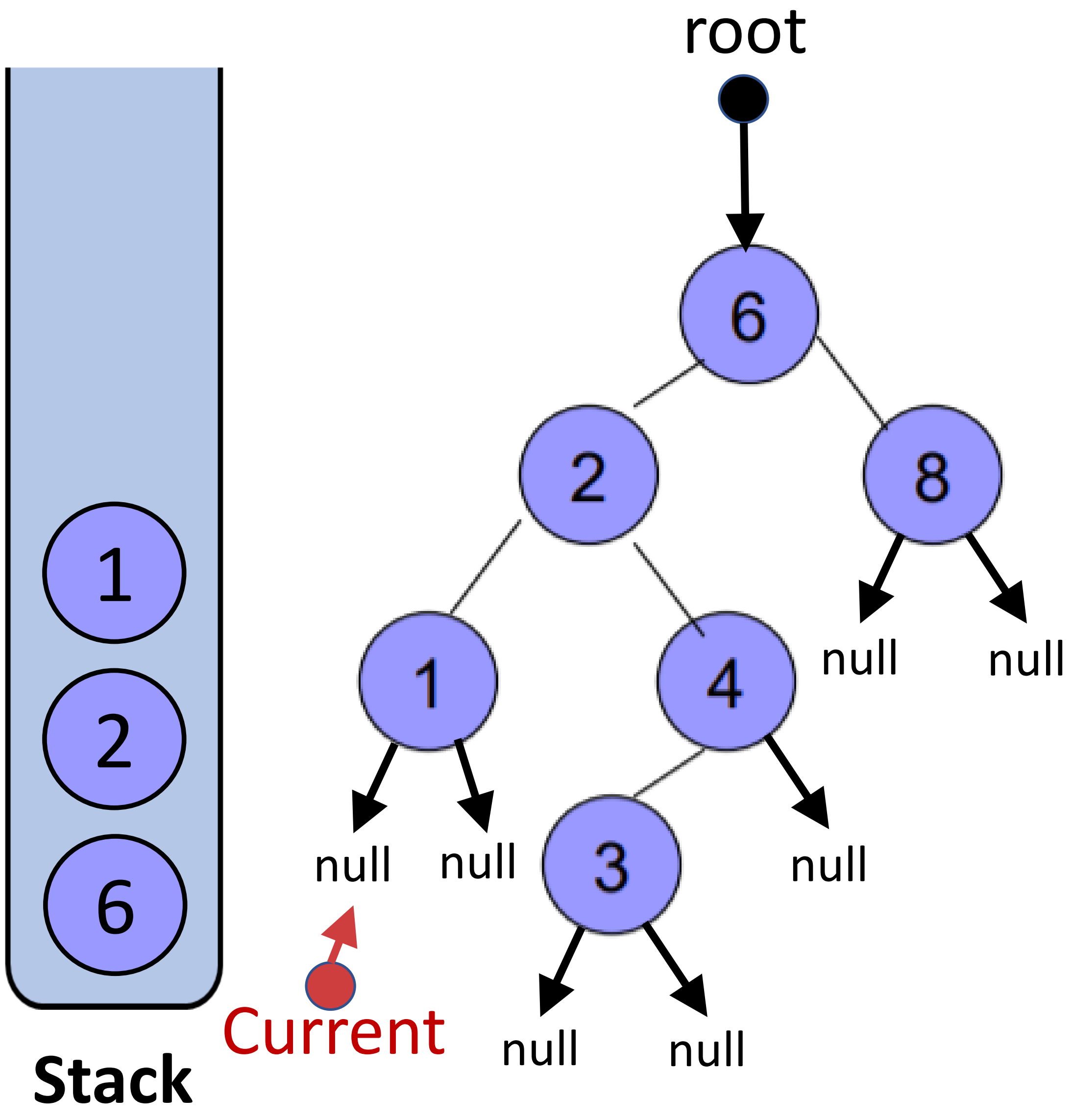
BST - Inorder Traversal using Stack

```
void  inorder()
 {
        node* current=root;

        while(1)

            if (current != NULL)
                    stack.push(current)
                    current = current->left
            else if  (!stack.empty() )
                    current = stack.top()
                    stack.pop()
                    cout<< current->data;
                    current = current->right
            else
                    return
  }
```



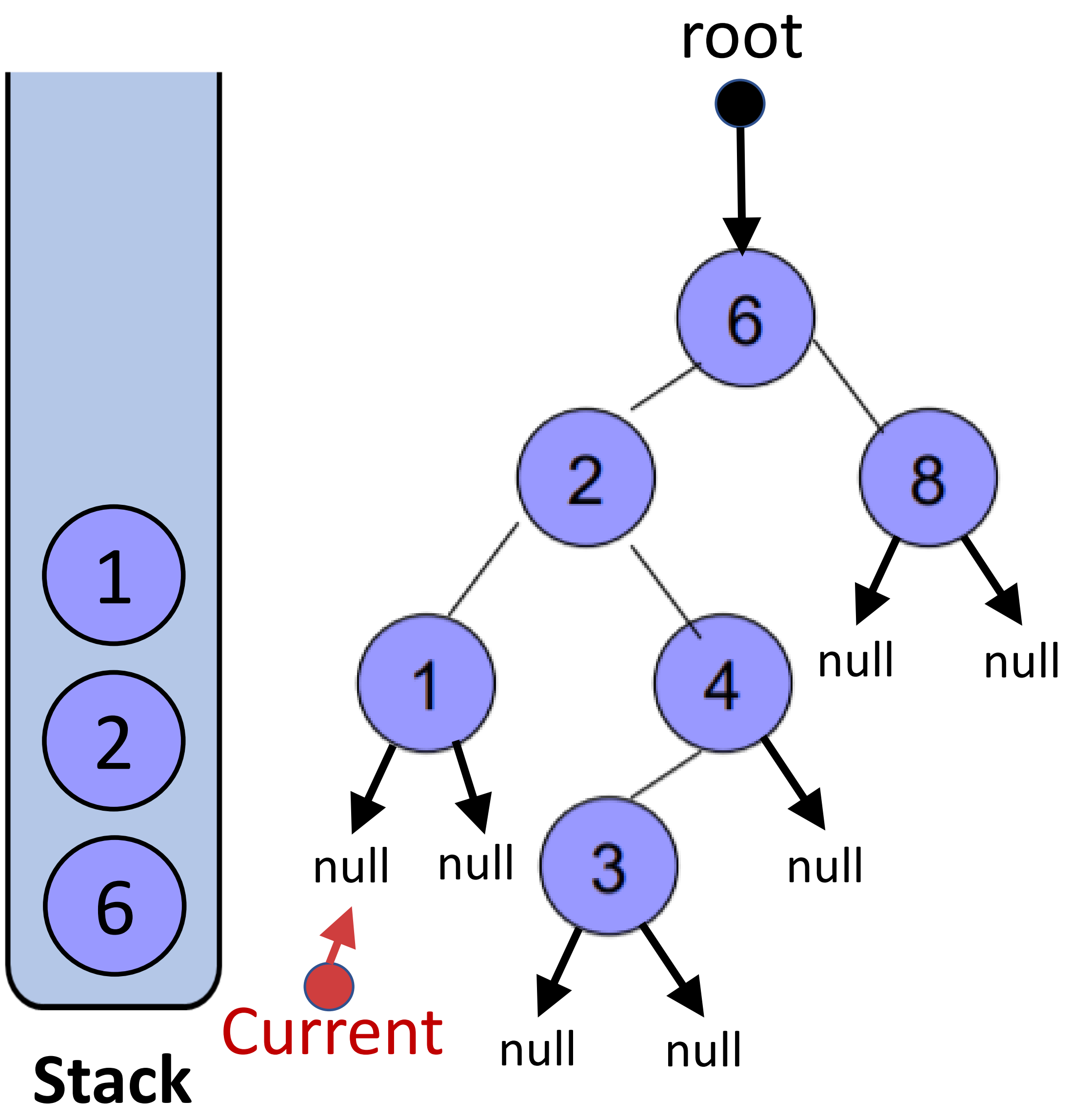
BST - Inorder Traversal using Stack

```
void  inorder()
 {
      node* current=root;
      while(1)

        if (current != NULL)
              stack.push(current)
              current = current->left
        else if  (!stack.empty() )
              current = stack.top()
              stack.pop()
              cout<< current->data;
              current = current->right
        else
              return
 }
```



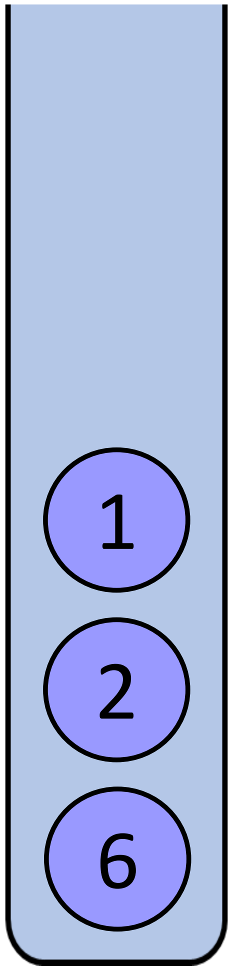
BST - Inorder Traversal using Stack

```
void  inorder()
 {
       node* current=root;

       while(1)

          if (current != NULL)
                stack.push(current)
                current = current->left
          else if  (!stack.empty() )
                current = stack.top()
                stack.pop()
                cout<< current->data;
                current = current->right
          else
                return
 }
```

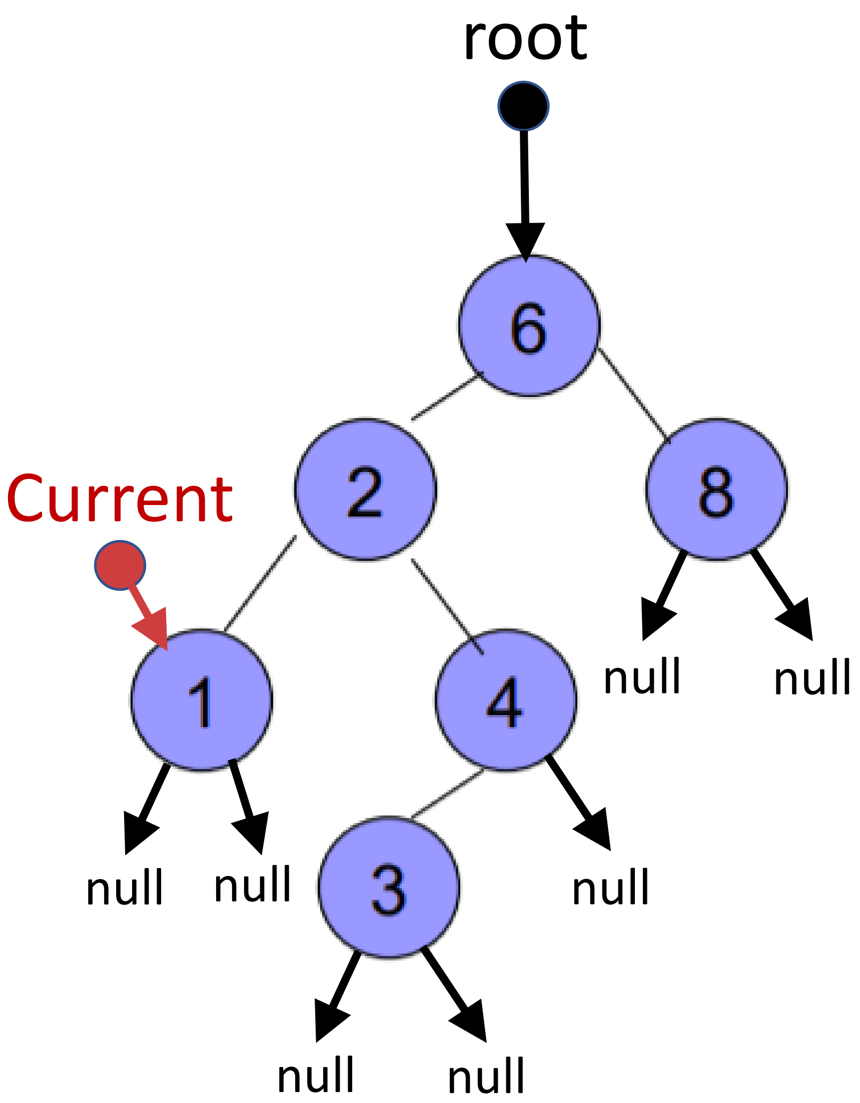


BST - Inorder Traversal using Stack

```
void  inorder()
 {
        node* current=root;

        while(1)

            if (current != NULL)
                    stack.push(current)
                    current = current->left
            else if  (!stack.empty() )
              ➡     current = stack.top()
                    stack.pop()
                    cout<< current->data;
                    current = current->right
            else
                    return
  }
```

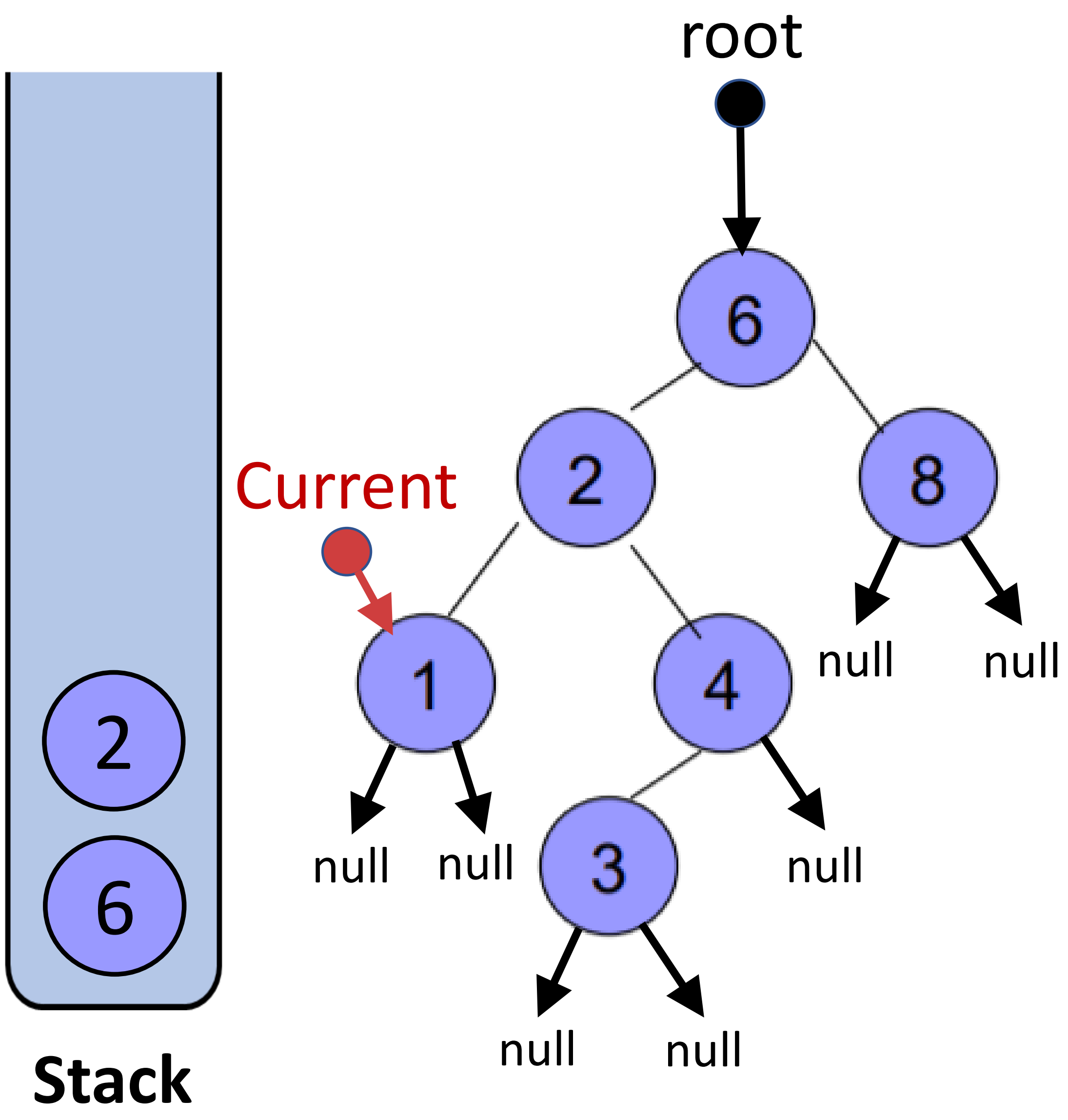


Stack



BST - Inorder Traversal using Stack

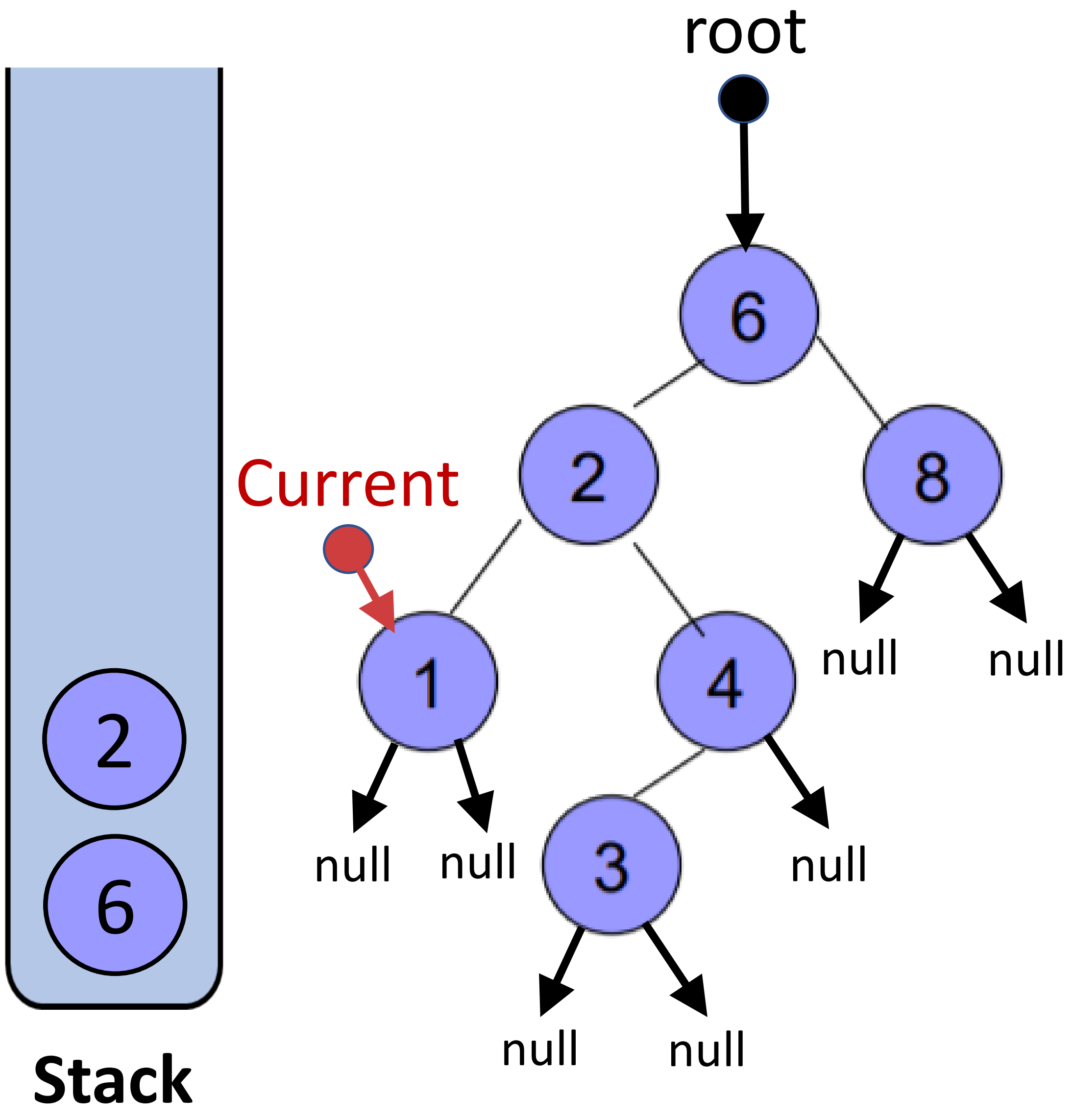
```
void  inorder()
 {
       node* current=root;

       while(1)

           if (current != NULL)
                  stack.push(current)
                  current = current->left
           else if  (!stack.empty() )
                  current = stack.top()
              ⇨  stack.pop()
                  cout<< current->data;
                  current = current->right
           else
                  return
  }
```



BST - Inorder Traversal using Stack

```
void  inorder()
 {
        node* current=root;

        while(1)

           if (current != NULL)
                  stack.push(current)
                  current = current->left
           else if  (!stack.empty() )
                  current = stack.top()
                  stack.pop()
              ➡  cout<< current->data;
                  current = current->right
           else
                  return
 }
```

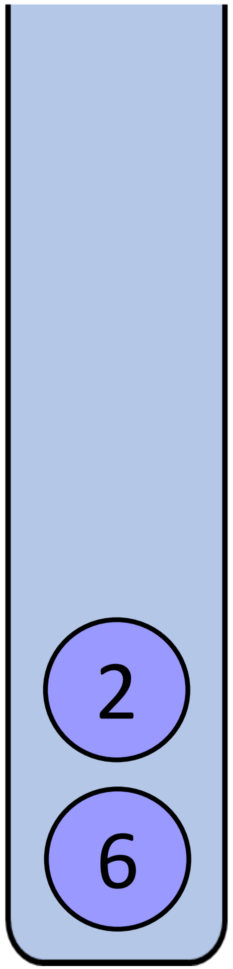
1



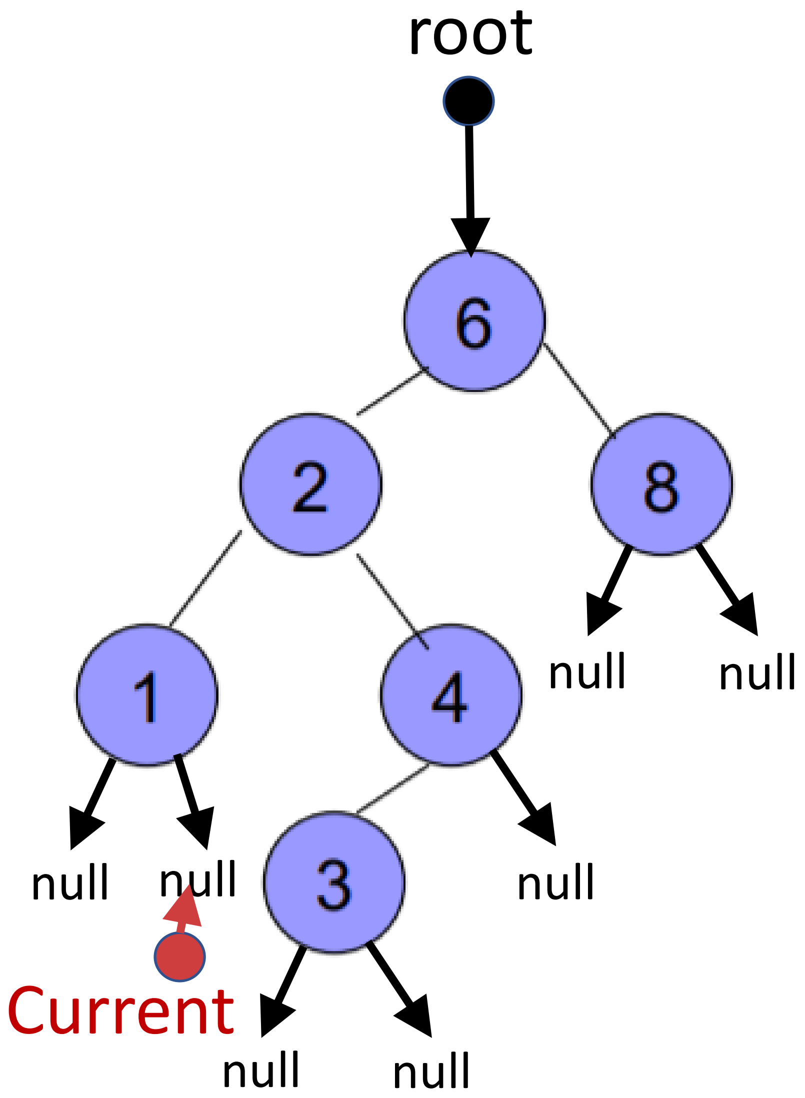
BST - Inorder Traversal using Stack

```
void  inorder()
 {
        node* current=root;

        while(1)

            if (current != NULL)
                    stack.push(current)
                    current = current->left
            else if  (!stack.empty() )
                    current = stack.top()
                    stack.pop()
                    cout<< current->data;
                    current = current->right
            else
                    return
 }
```

1



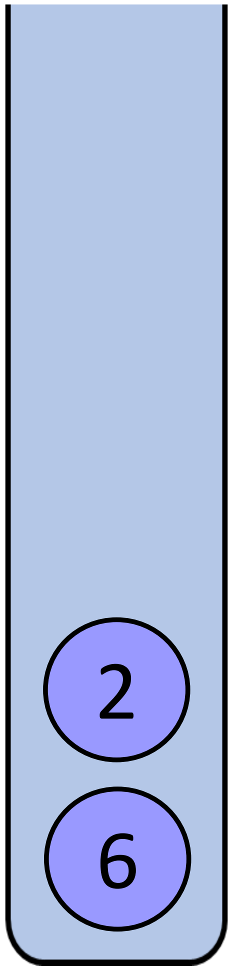
Stack



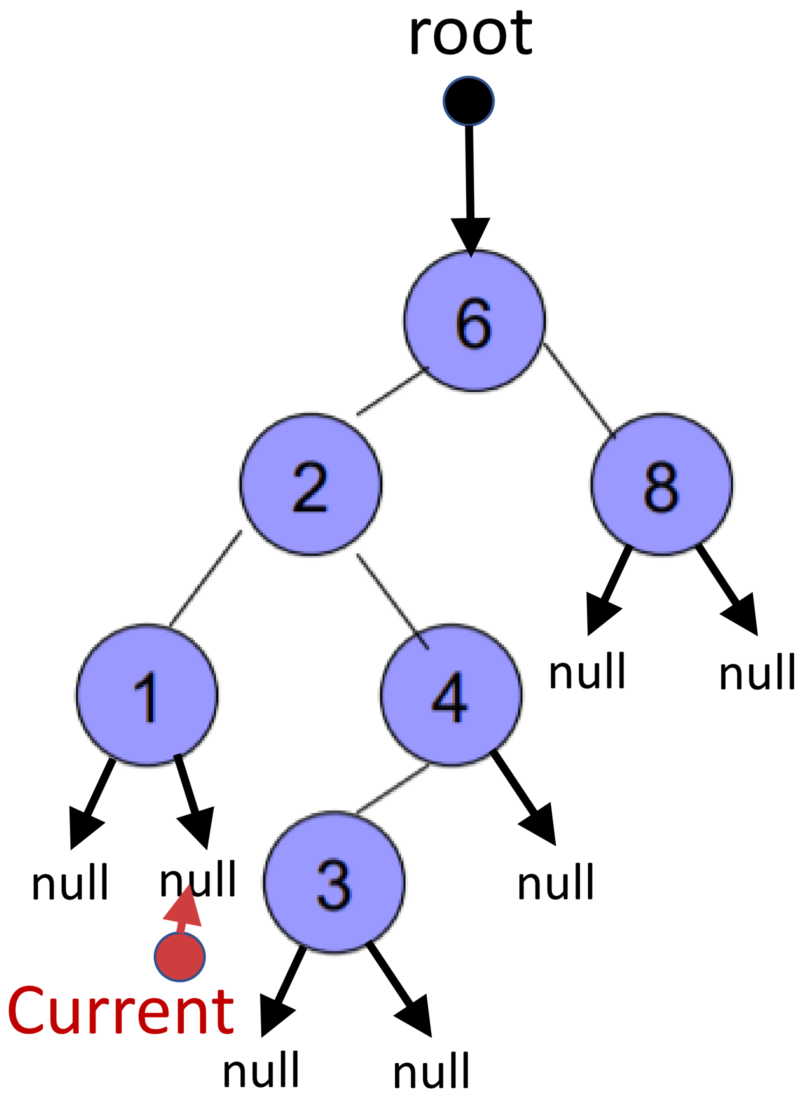
BST - Inorder Traversal using Stack

```
void  inorder()
 {
      node* current=root;

      while(1)

        if (current != NULL)
              stack.push(current)
              current = current->left
        else if  (!stack.empty() )
              current = stack.top()
              stack.pop()
              cout<< current->data;
              current = current->right
        else
              return
 }
```

1



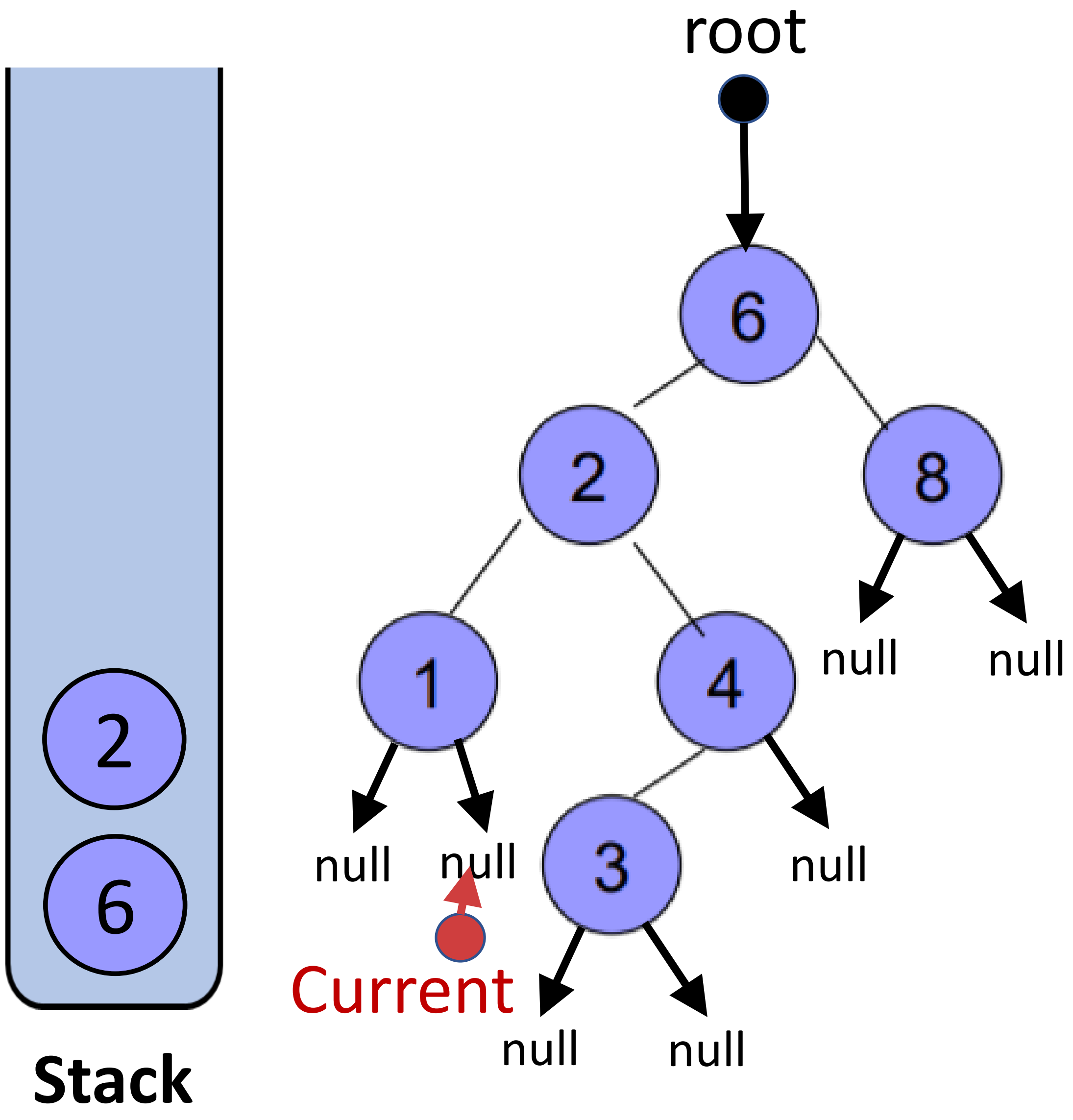
Stack



BST - Inorder Traversal using Stack

```
void  inorder()
 {
      node* current=root;

      while(1)

          if (current != NULL)
                stack.push(current)
                current = current->left
          else if  (!stack.empty() )
                current = stack.top()
                stack.pop()
                cout<< current->data;
                current = current->right
          else
                return
 }
```

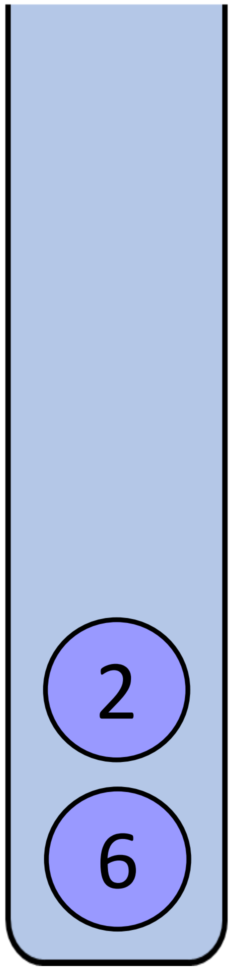
1



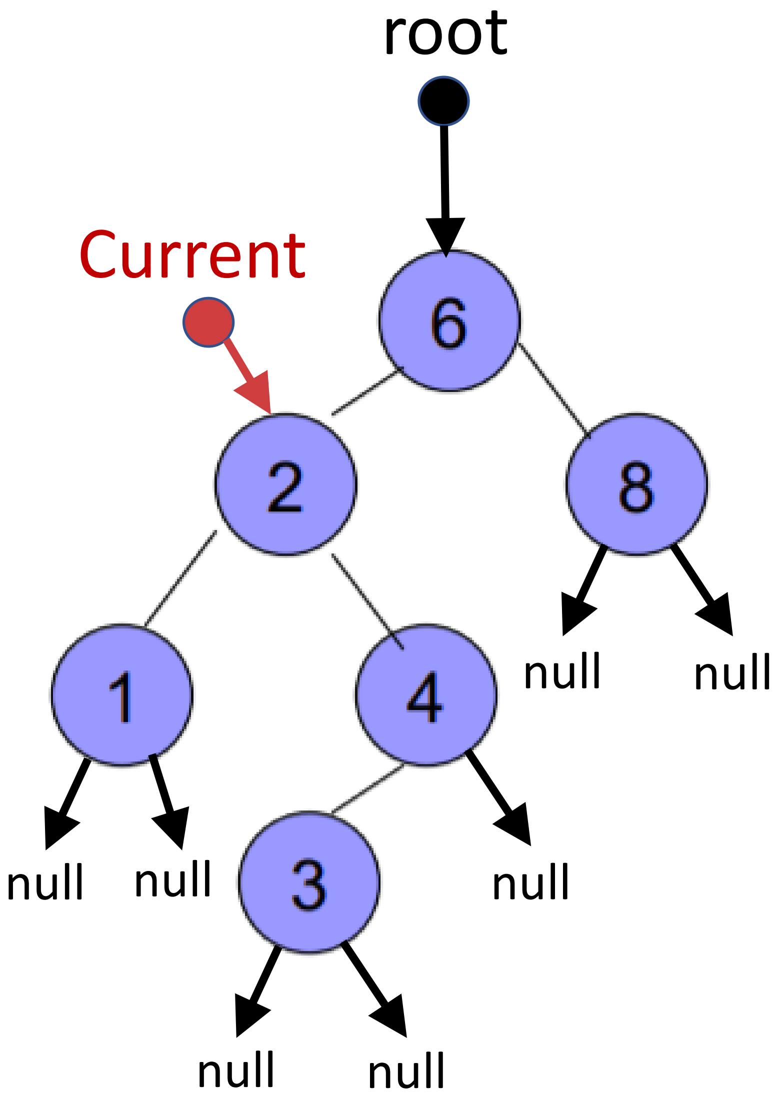
BST - Inorder Traversal using Stack

```
void  inorder()
 {
      node* current=root;

      while(1)

        if (current != NULL)
              stack.push(current)
              current = current->left
        else if  (!stack.empty() )
      ➡       current = stack.top()
              stack.pop()
              cout<< current->data;
              current = current->right
        else
              return
 }
```

1



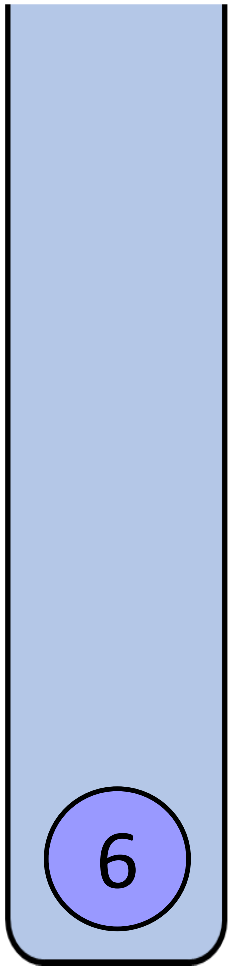
Stack



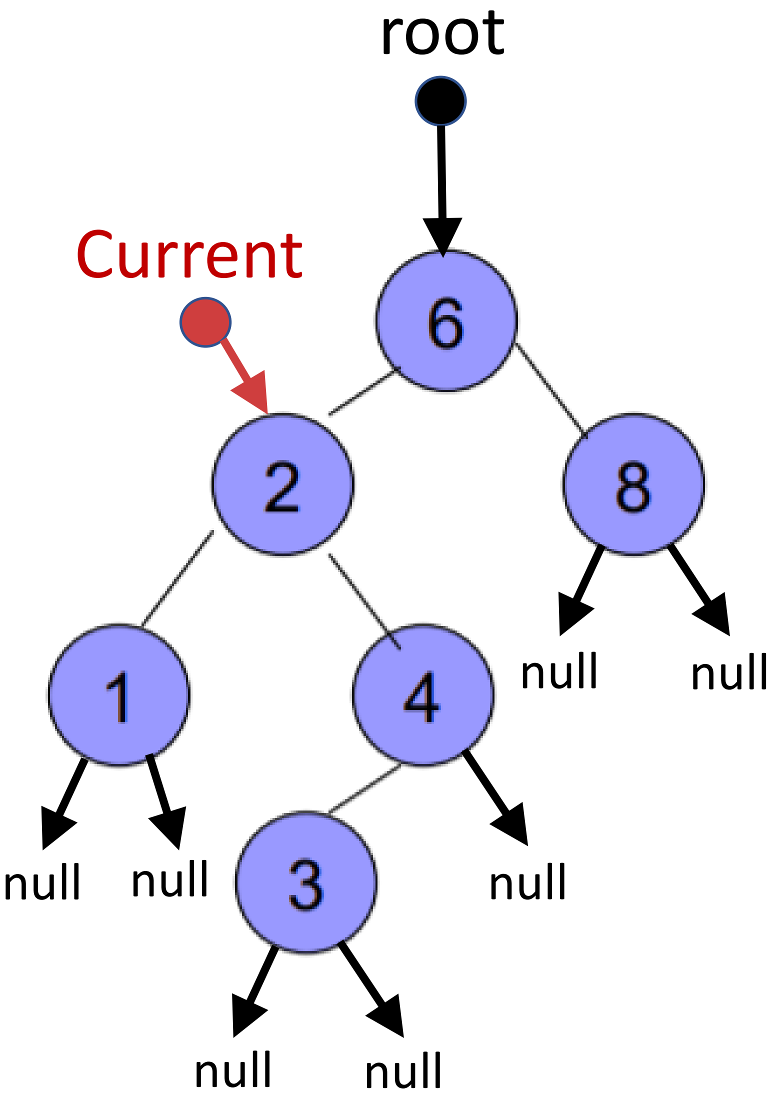
BST - Inorder Traversal using Stack

```
void  inorder()
 {
      node* current=root;

      while(1)

          if (current != NULL)
                stack.push(current)
                current = current->left
          else if  (!stack.empty() )
                current = stack.top()
             ⇨  stack.pop()
                cout<< current->data;
                current = current->right
          else
                return
 }
```

1



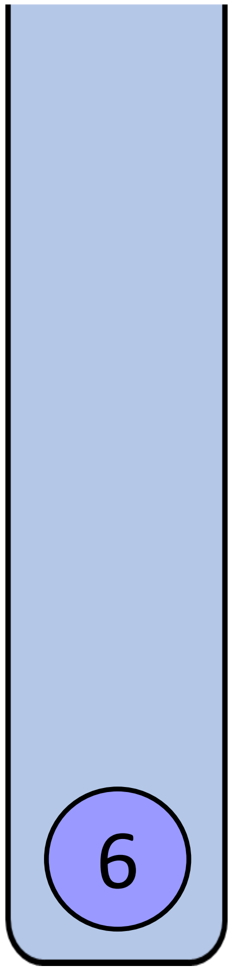
Stack



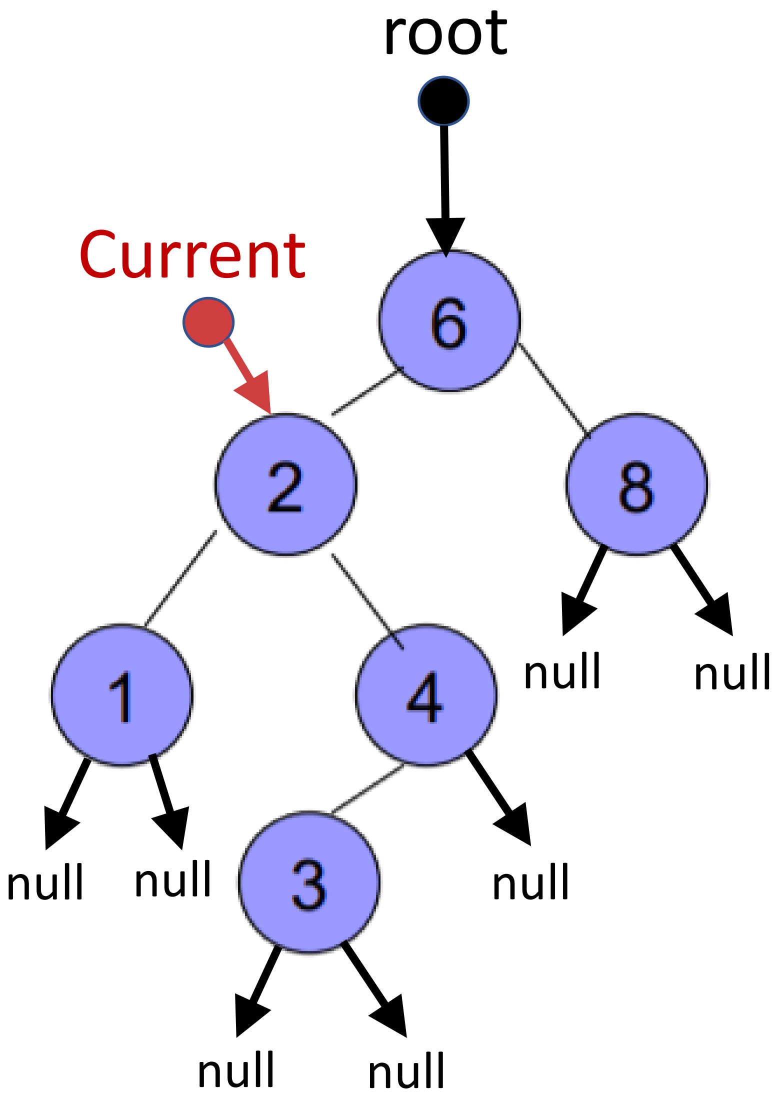
BST - Inorder Traversal using Stack

```
void  inorder()
 {
       node* current=root;

       while(1)

          if (current != NULL)
                stack.push(current)
                current = current->left
          else if  (!stack.empty() )
                current = stack.top()
                stack.pop()
          ➡    cout<< current->data;
                current = current->right
          else
                return
 }
```

1   2



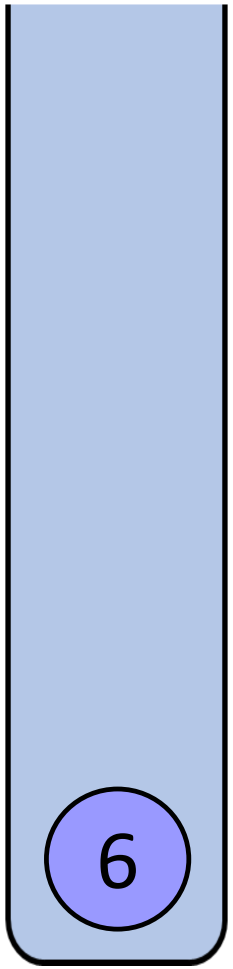
Stack



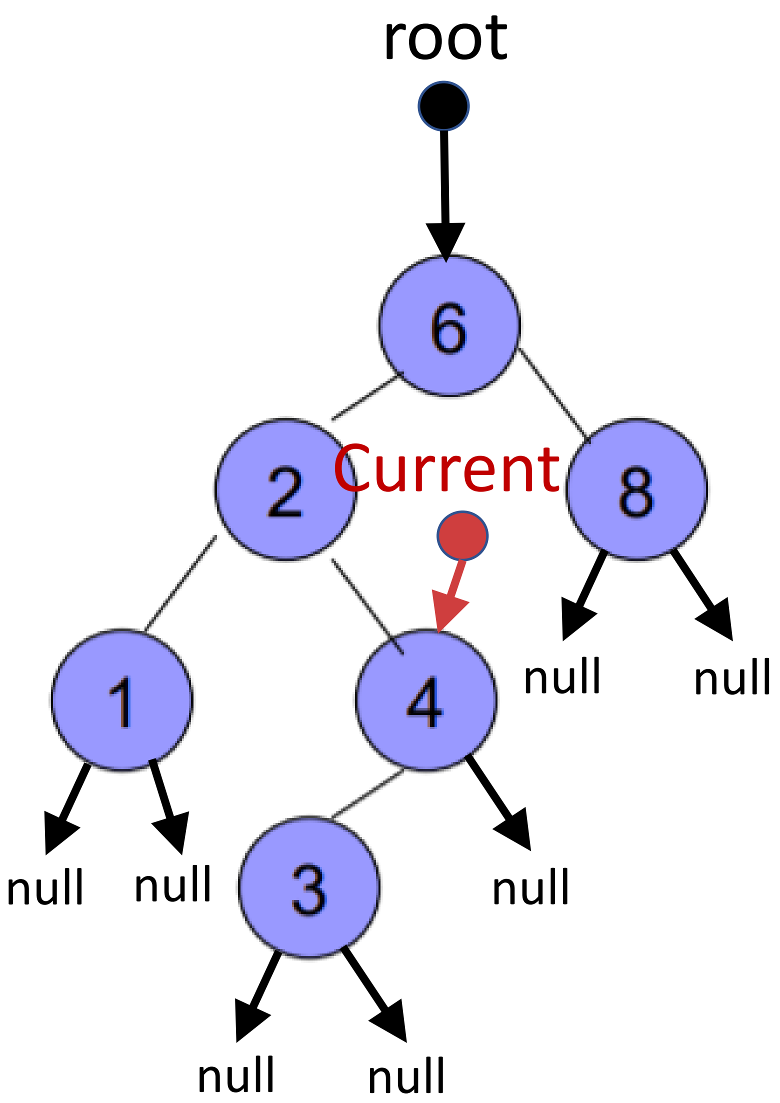
BST - Inorder Traversal using Stack

```
void  inorder()
 {
        node* current=root;

        while(1)

            if (current != NULL)
                  stack.push(current)
                  current = current->left
            else if  (!stack.empty() )
                  current = stack.top()
                  stack.pop()
                  cout<< current->data;
          ➡     current = current->right
            else
                  return
 }
```

1   2



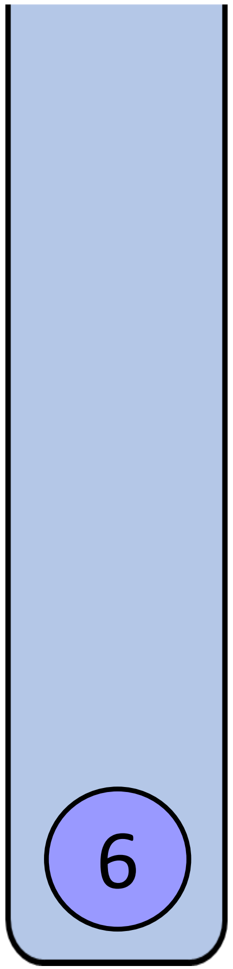
Stack



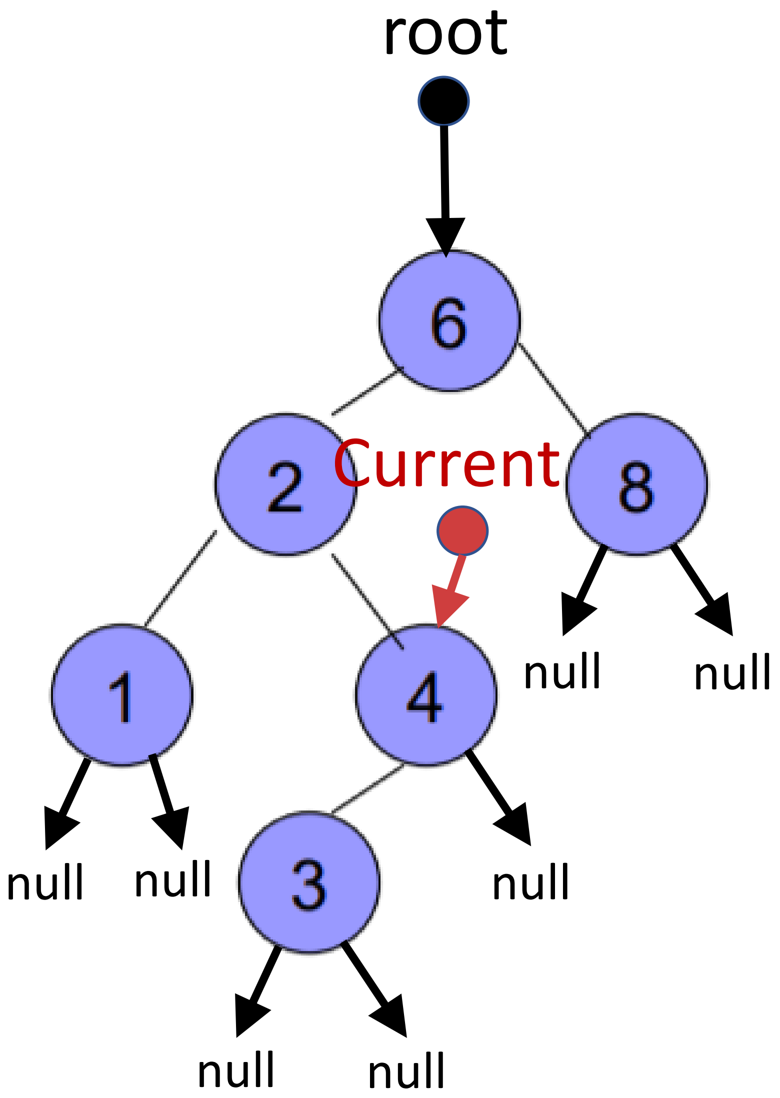
BST - Inorder Traversal using Stack

```
void  inorder()
 {
        node* current=root;

        while(1)

          if (current != NULL)
                stack.push(current)
                current = current->left
          else if  (!stack.empty() )
                current = stack.top()
                stack.pop()
                cout<< current->data;
                current = current->right
          else
                return
 }
```

1   2



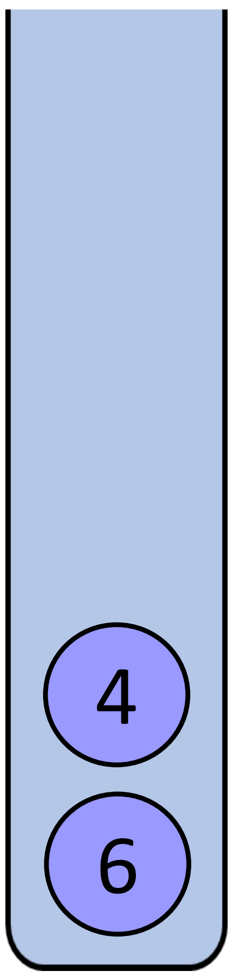
Stack



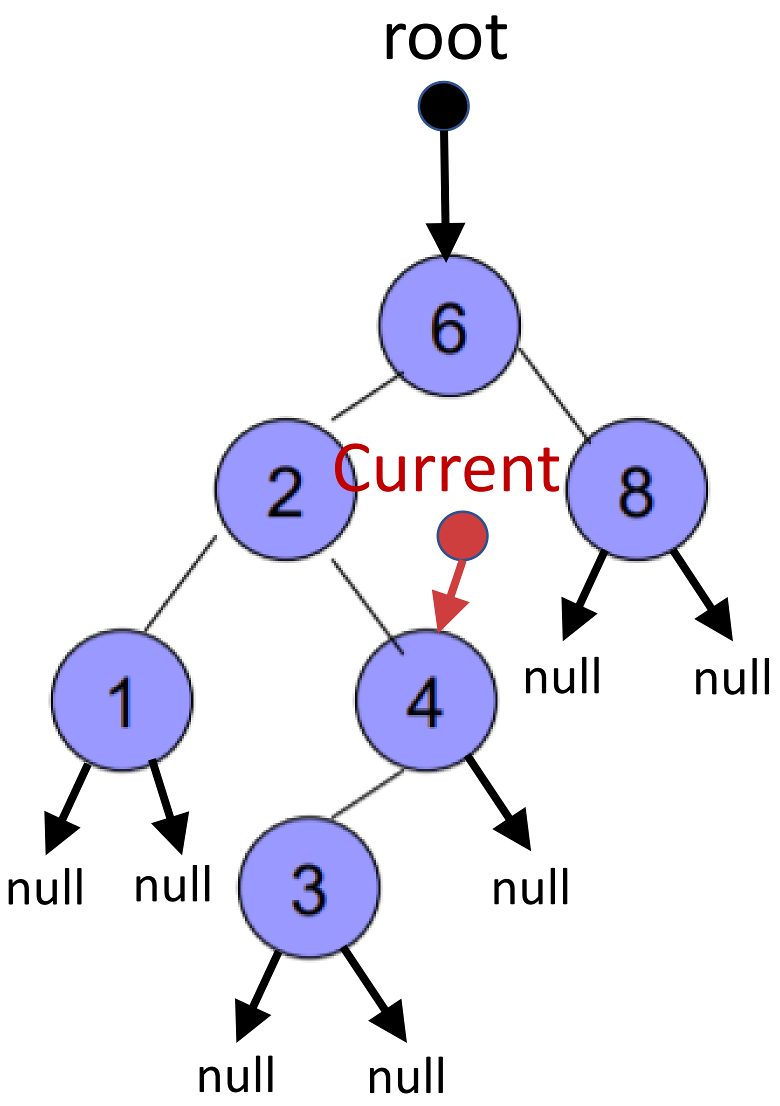
BST - Inorder Traversal using Stack

```
void  inorder()
 {
      node* current=root;

      while(1)

          if (current != NULL)
                stack.push(current)
                current = current->left
          else if  (!stack.empty() )
                current = stack.top()
                stack.pop()
                cout<< current->data;
                current = current->right
          else
                return
 }
```

1   2



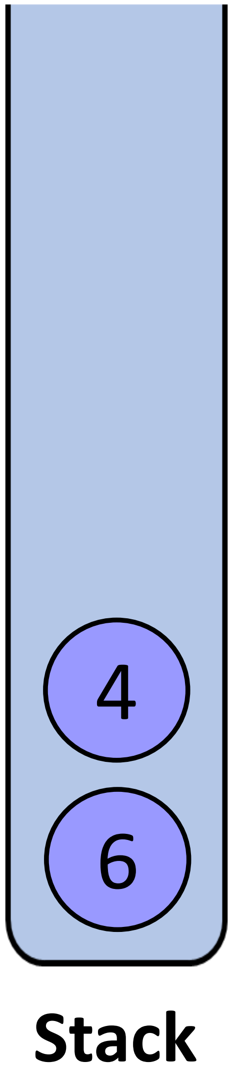
Stack

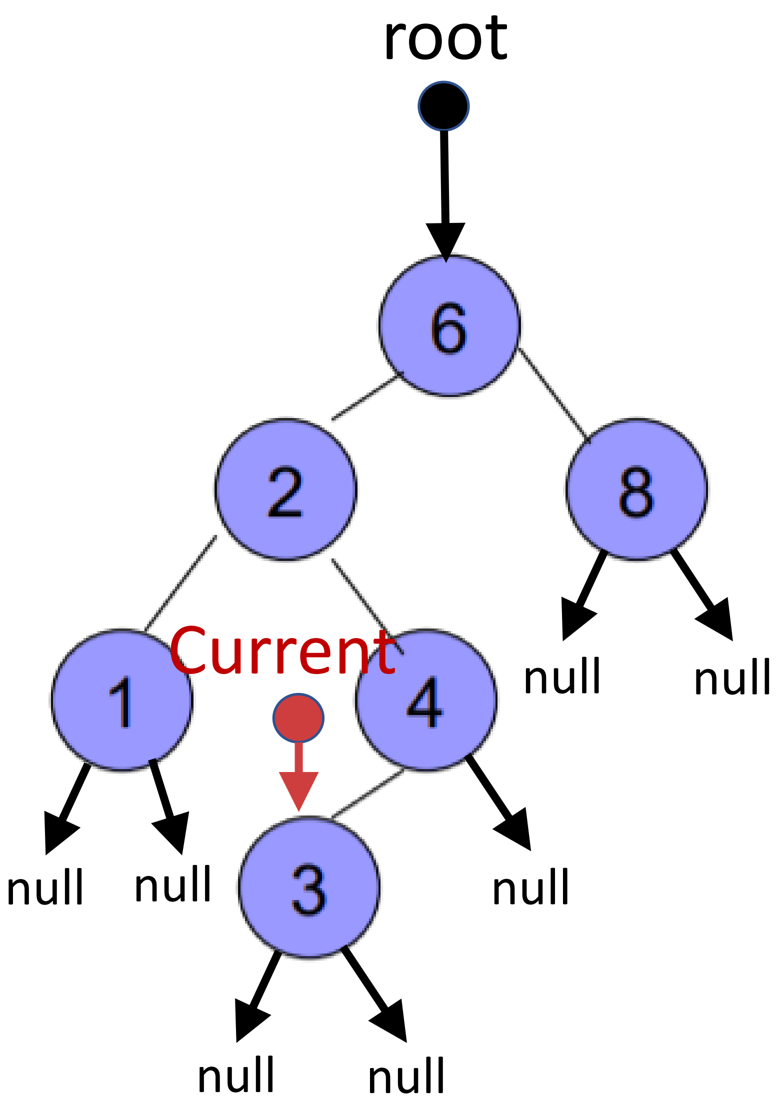


BST - Inorder Traversal using Stack

```
void  inorder()
 {
        node* current=root;

        while(1)

           if (current != NULL)
                 stack.push(current)
         ➡      current = current->left
           else if  (!stack.empty() )
                 current = stack.top()
                 stack.pop()
                 cout<< current->data;
                 current = current->right
           else
                 return
 }
```

1   2

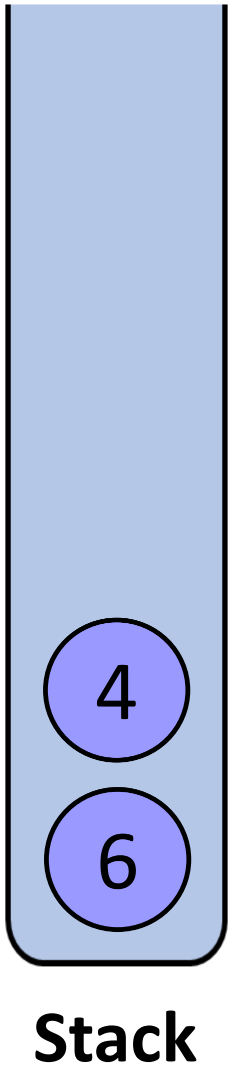


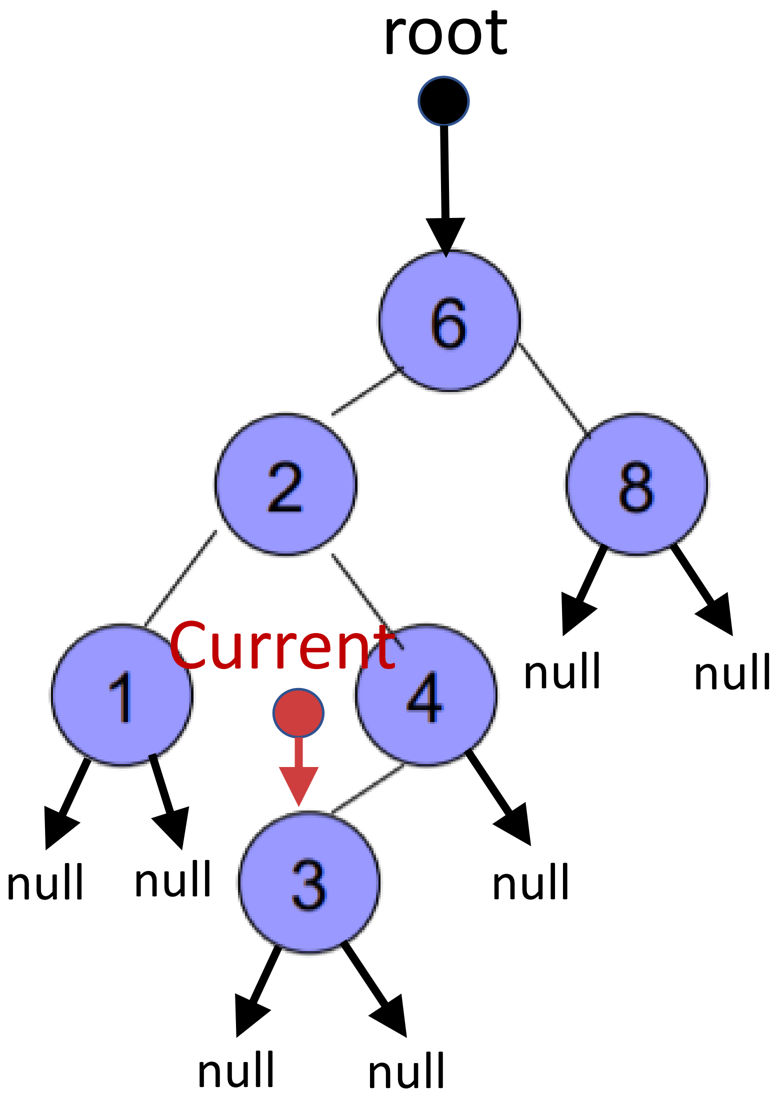


BST - Inorder Traversal using Stack

```
void  inorder()
 {
      node* current=root;
      while(1)
         if (current != NULL)
               stack.push(current)
               current = current->left
         else if  (!stack.empty() )
               current = stack.top()
               stack.pop()
               cout<< current->data;
               current = current->right
         else
               return
 }
```

1 2

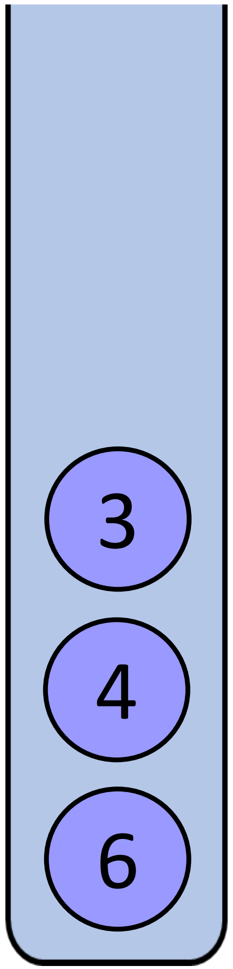




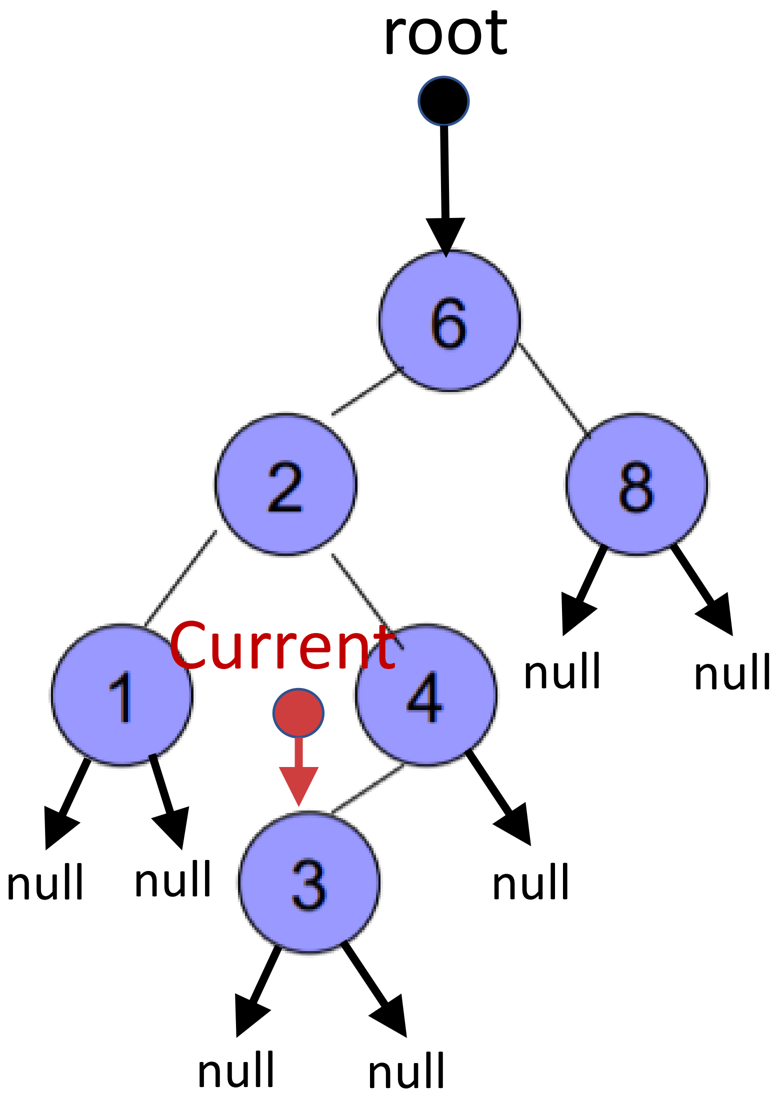
BST - Inorder Traversal using Stack

```
void  inorder()
 {
        node* current=root;

        while(1)

            if (current != NULL)
                  stack.push(current)
                  current = current->left
            else if  (!stack.empty() )
                  current = stack.top()
                  stack.pop()
                  cout<< current->data;
                  current = current->right
            else
                  return
  }
```

1   2



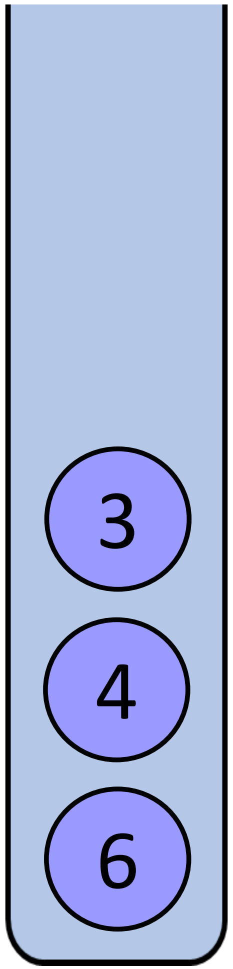
Stack



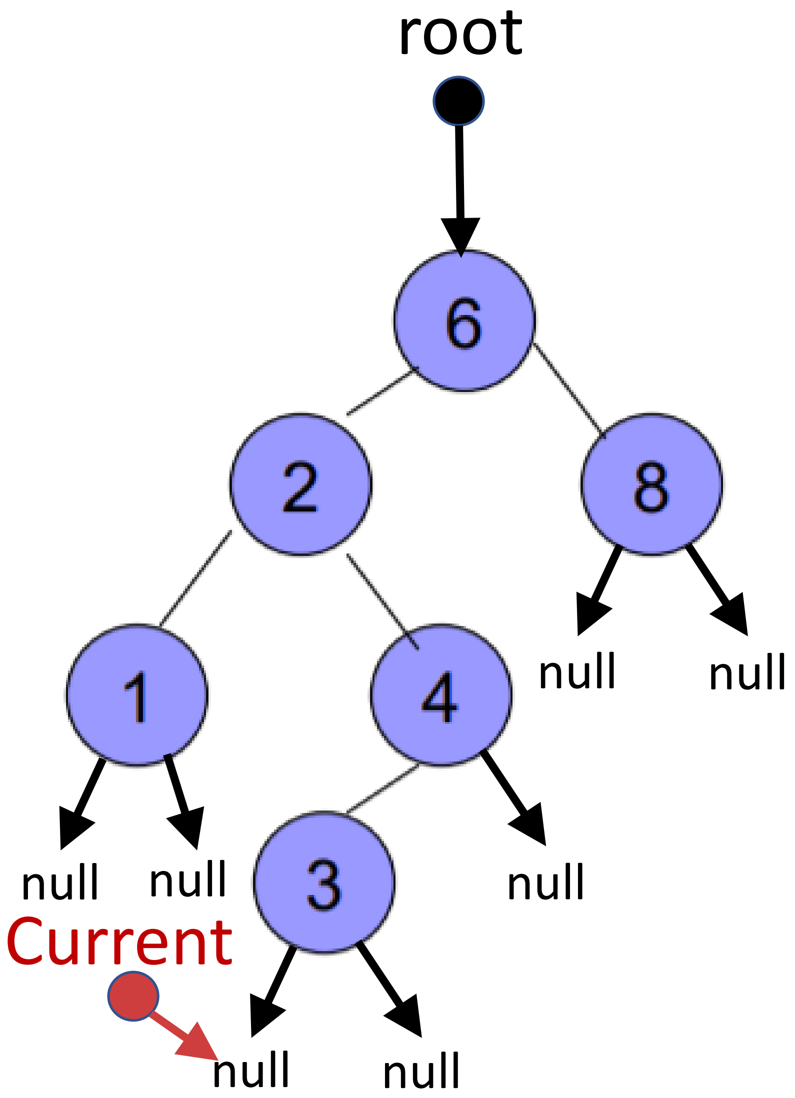
BST - Inorder Traversal using Stack

```
void  inorder()
 {
        node* current=root;

        while(1)

            if (current != NULL)
                  stack.push(current)
                  current = current->left
            else if  (!stack.empty() )
                  current = stack.top()
                  stack.pop()
                  cout<< current->data;
                  current = current->right
            else
                  return
 }
```

1   2



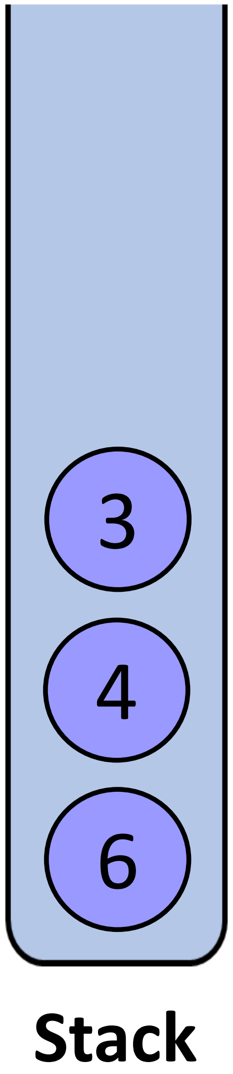
Stack

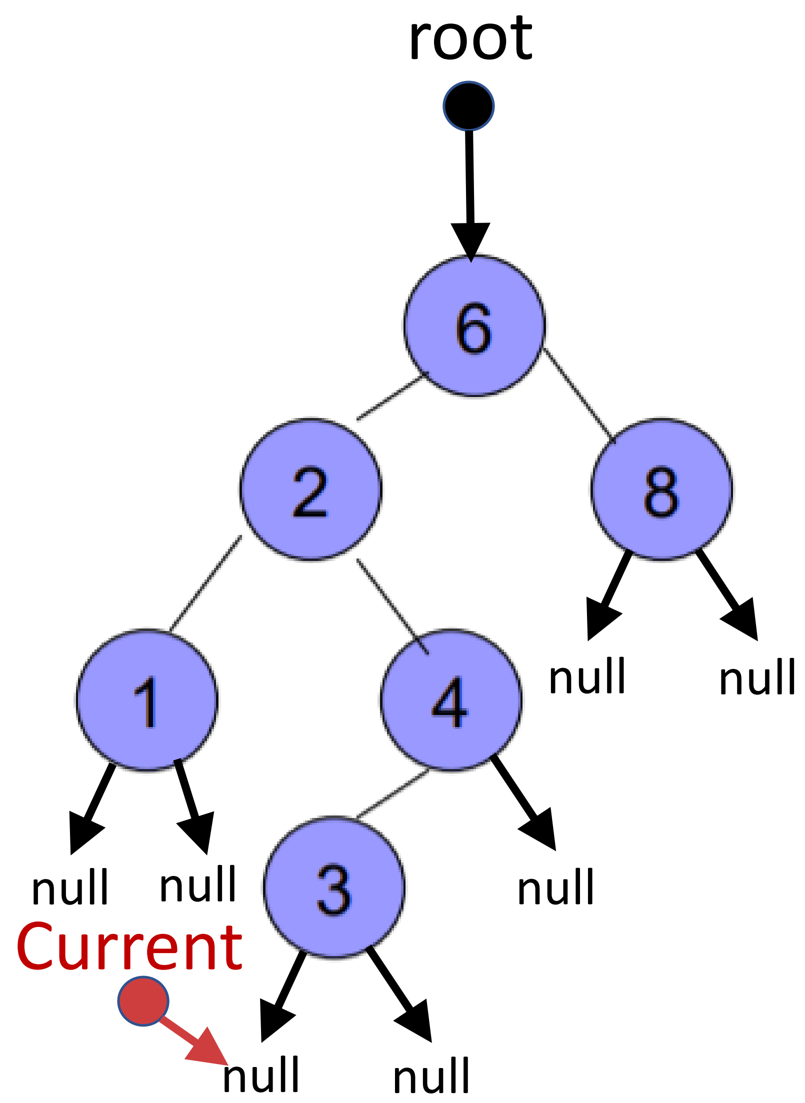


BST - Inorder Traversal using Stack

```
void  inorder()
 {
        node* current=root;

        while(1)

          if (current != NULL)
                 stack.push(current)
                 current = current->left
          else if  (!stack.empty() )
                 current = stack.top()
                 stack.pop()
                 cout<< current->data;
                 current = current->right
          else
                 return
 }
```

1   2

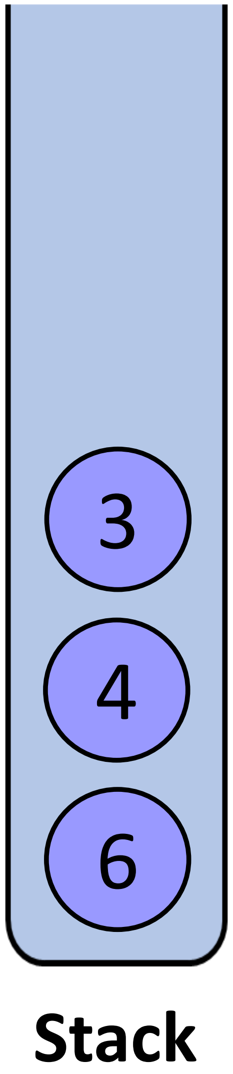


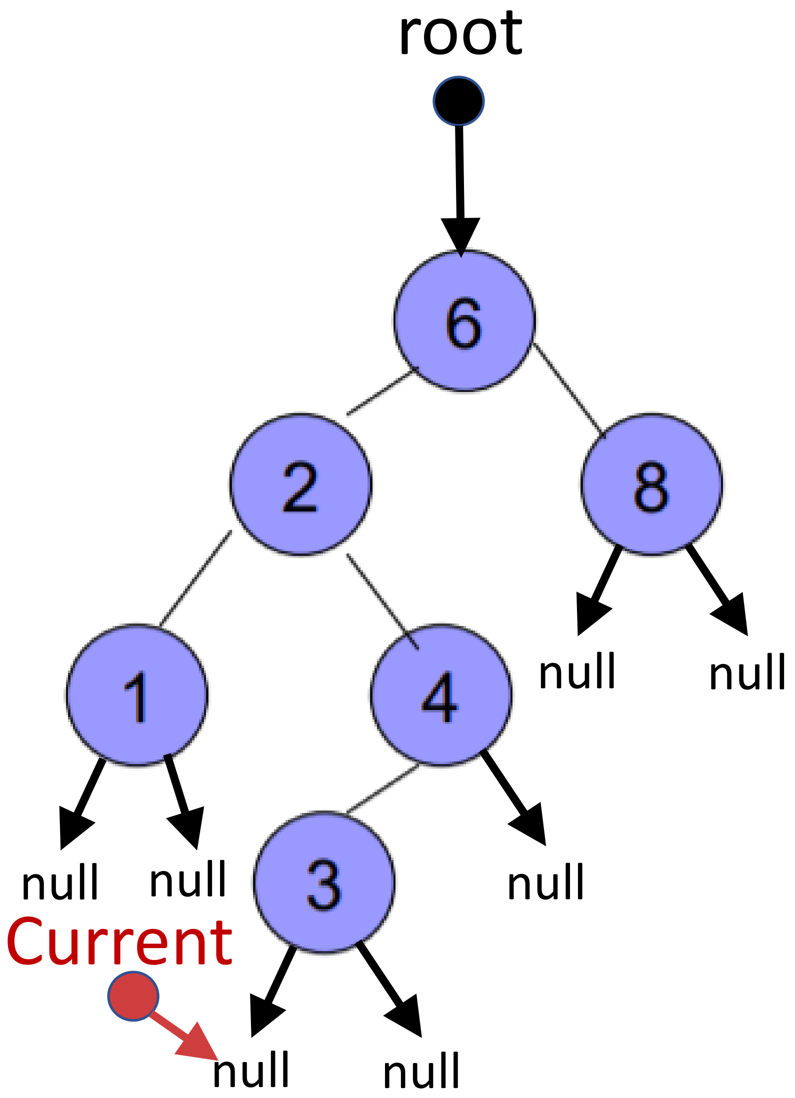


BST - Inorder Traversal using Stack

```
void  inorder()
 {
        node* current=root;

        while(1)

            if (current != NULL)
                   stack.push(current)
                   current = current->left
            else if  (!stack.empty() )
                   current = stack.top()
                   stack.pop()
                   cout<< current->data;
                   current = current->right
            else
                   return
 }
```

1   2

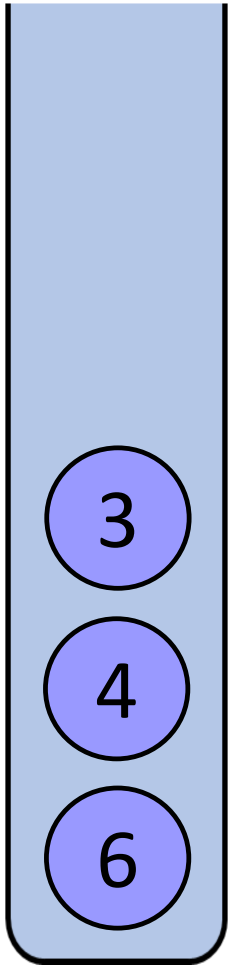




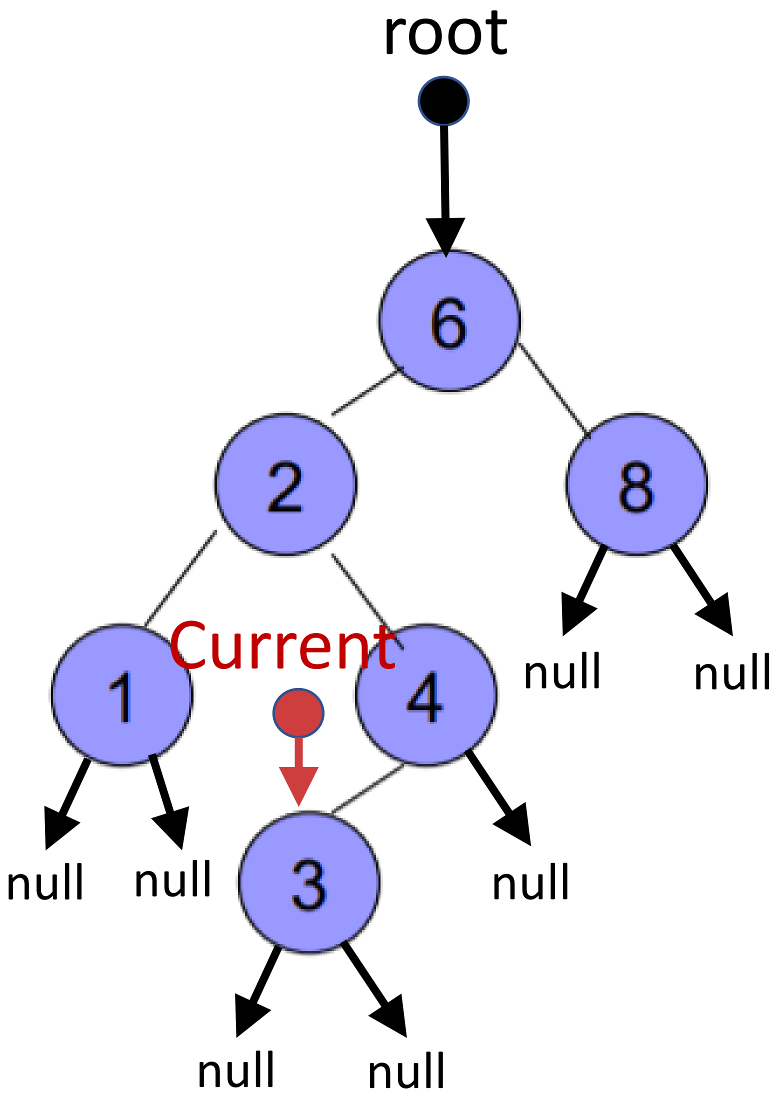
BST - Inorder Traversal using Stack

```
void  inorder()
 {
      node* current=root;

      while(1)

         if (current != NULL)
                stack.push(current)
                current = current->left
         else if  (!stack.empty() )
                current = stack.top()
                stack.pop()
                cout<< current->data;
                current = current->right
         else
                return
 }
```

1   2



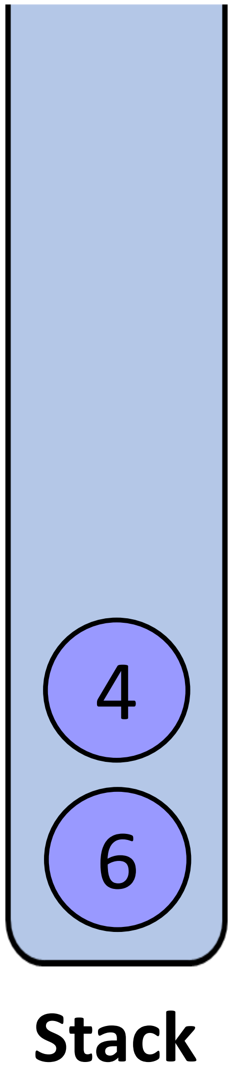
Stack

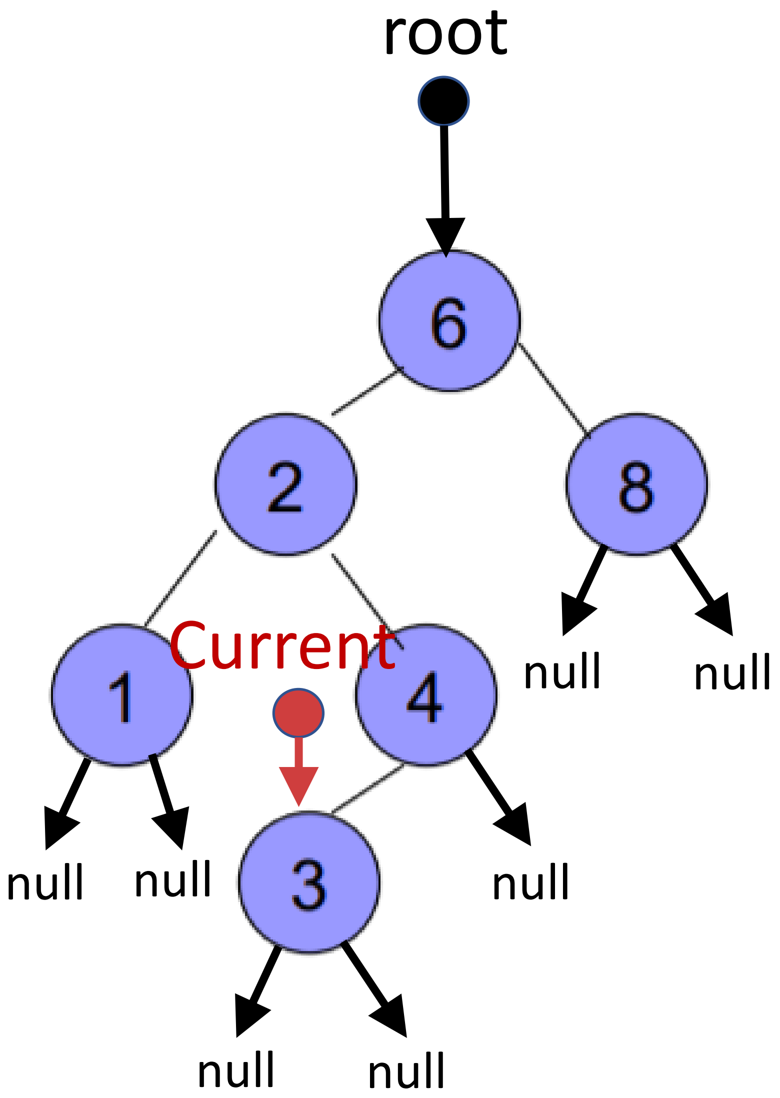


BST - Inorder Traversal using Stack

```
void  inorder()
 {
       node* current=root;

       while(1)

           if (current != NULL)
                  stack.push(current)
                  current = current->left
           else if  (!stack.empty() )
                  current = stack.top()
                  stack.pop()
                  cout<< current->data;
                  current = current->right
           else
                  return
  }
```

1   2

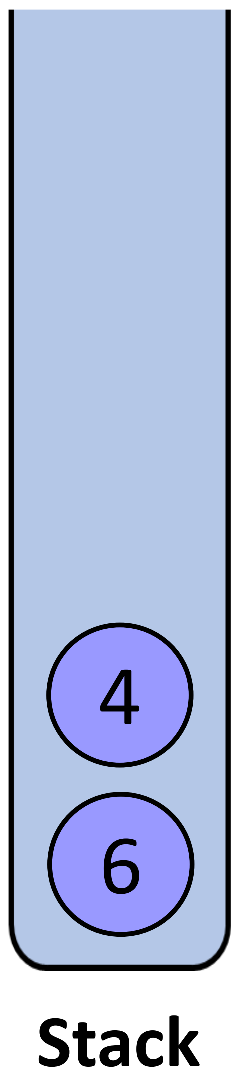


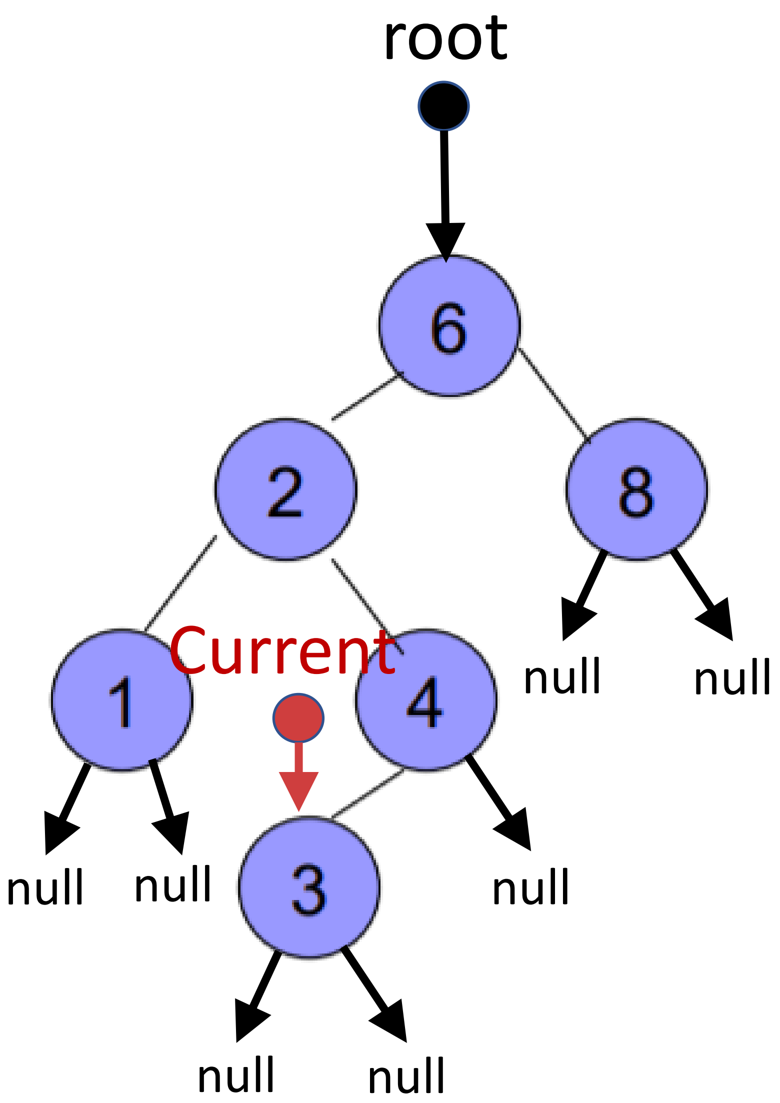


BST - Inorder Traversal using Stack

```
void  inorder()
 {
       node* current=root;

       while(1)

              if (current != NULL)
                     stack.push(current)
                     current = current->left
              else if  (!stack.empty() )
                     current = stack.top()
                     stack.pop()
                     cout<< current->data;
                     current = current->right
              else
                     return
 }
```

1   2   3

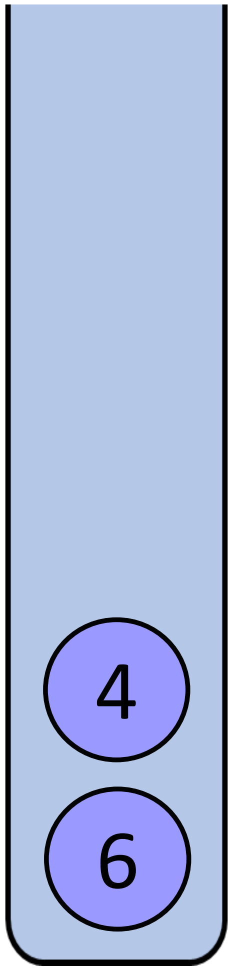




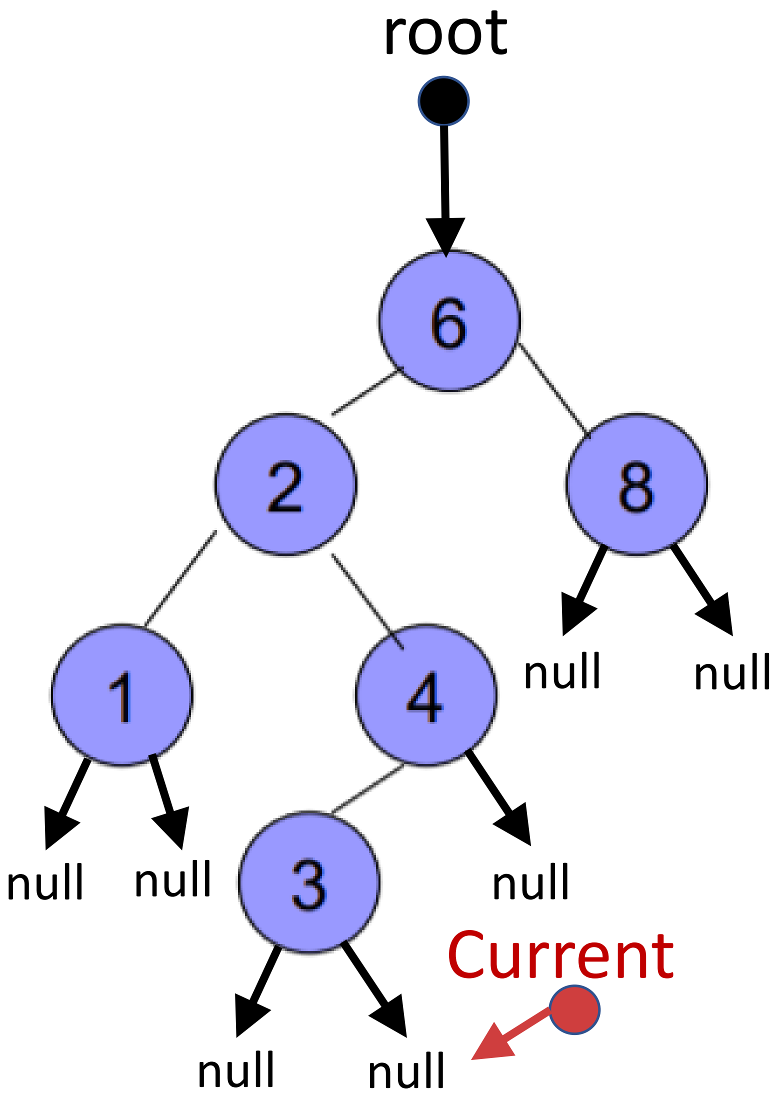
BST - Inorder Traversal using Stack

```
void  inorder()
 {
      node* current=root;

      while(1)

        if (current != NULL)
              stack.push(current)
              current = current->left
        else if  (!stack.empty() )
              current = stack.top()
              stack.pop()
              cout<< current->data;
              current = current->right
        else
              return
 }
```

1   2   3



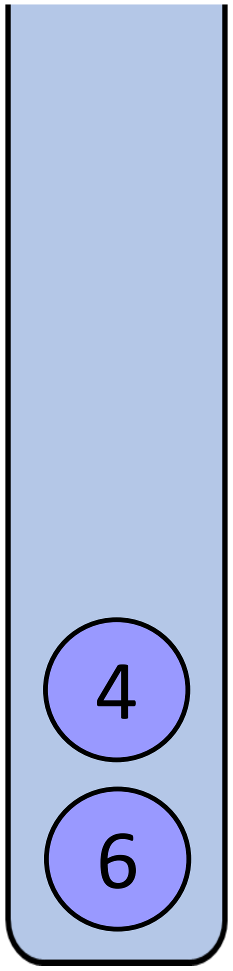
Stack



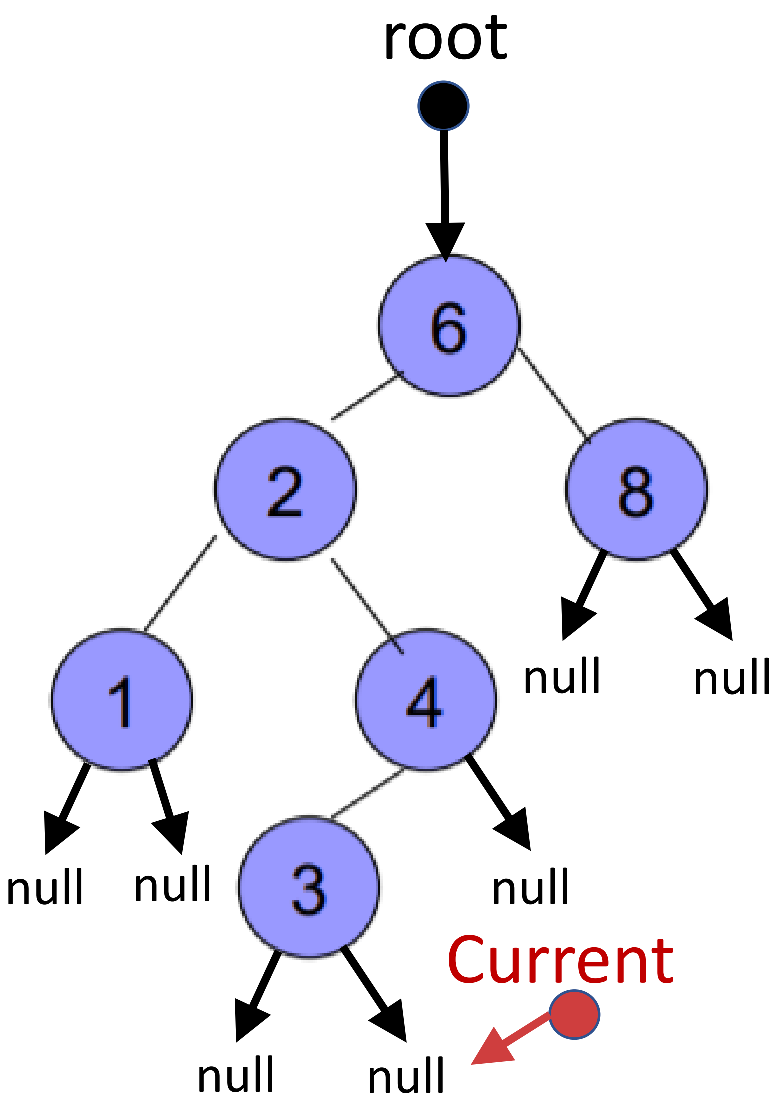
BST - Inorder Traversal using Stack

```
void  inorder()
 {
       node* current=root;

       while(1)

           if (current != NULL)
                 stack.push(current)
                 current = current->left
           else if  (!stack.empty() )
                 current = stack.top()
                 stack.pop()
                 cout<< current->data;
                 current = current->right
           else
                 return
  }
```

1   2   3



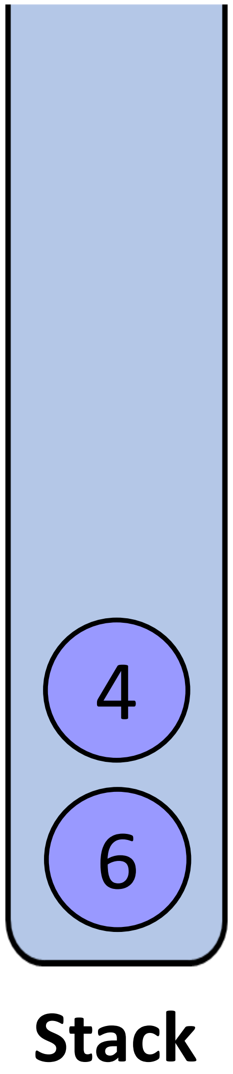
Stack

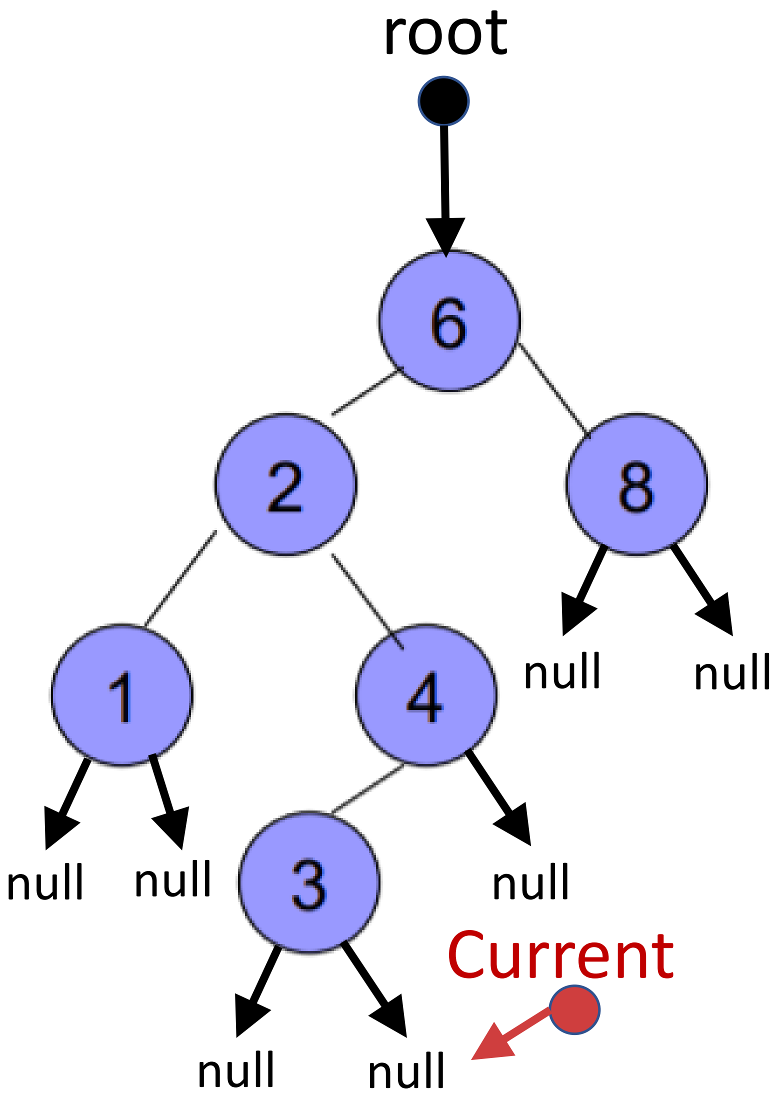


BST - Inorder Traversal using Stack

```
void  inorder()
 {
      node* current=root;

      while(1)

        if (current != NULL)
              stack.push(current)
              current = current->left
        else if  (!stack.empty() )
              current = stack.top()
              stack.pop()
              cout<< current->data;
              current = current->right
        else
              return
 }
```

1   2   3

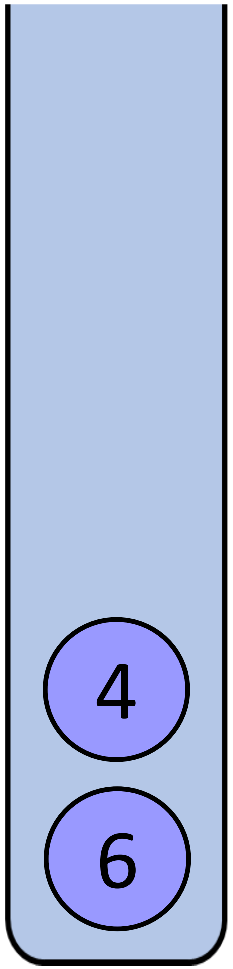




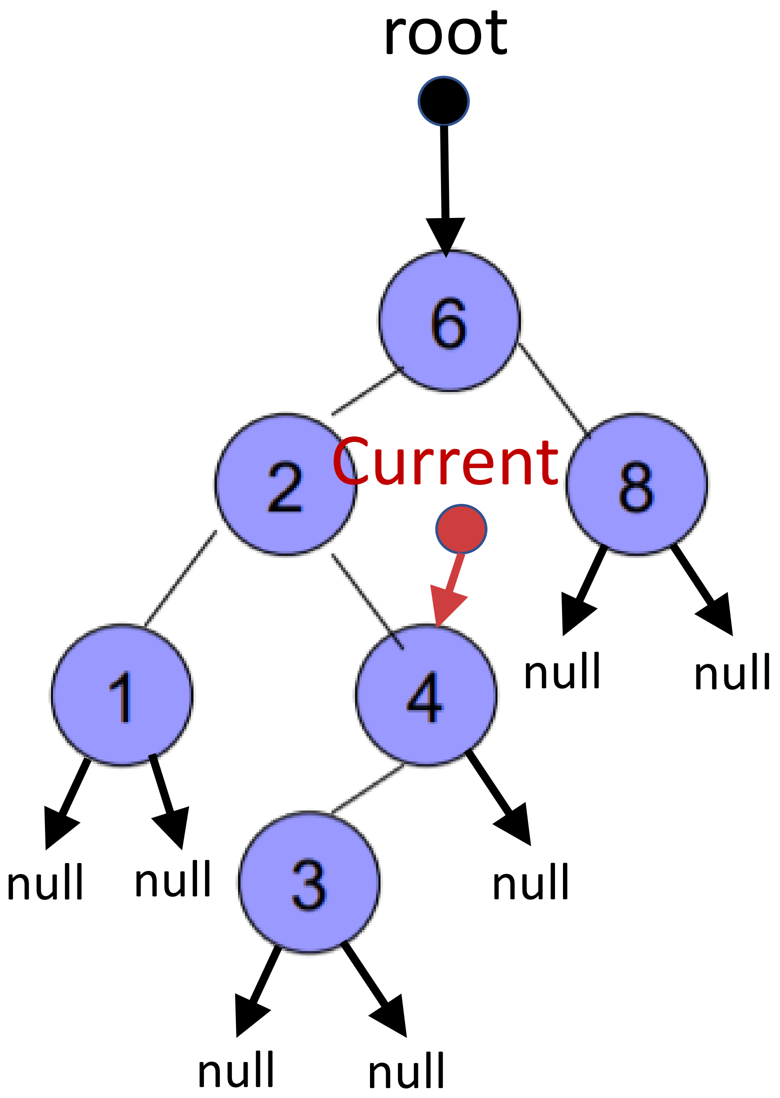
BST - Inorder Traversal using Stack

```
void  inorder()
 {
        node* current=root;

        while(1)

            if (current != NULL)
                    stack.push(current)
                    current = current->left
            else if  (!stack.empty() )
                    current = stack.top()
                    stack.pop()
                    cout<< current->data;
                    current = current->right
            else
                    return
 }
```

1   2   3



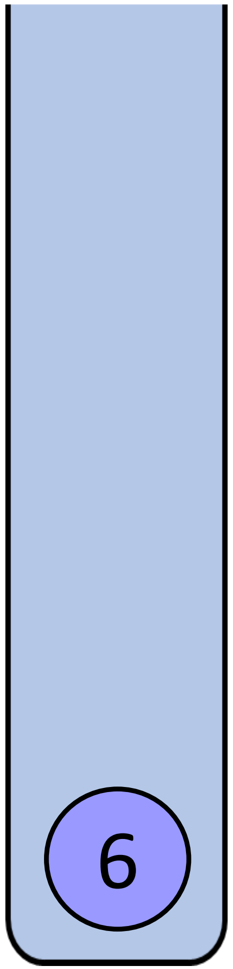
Stack



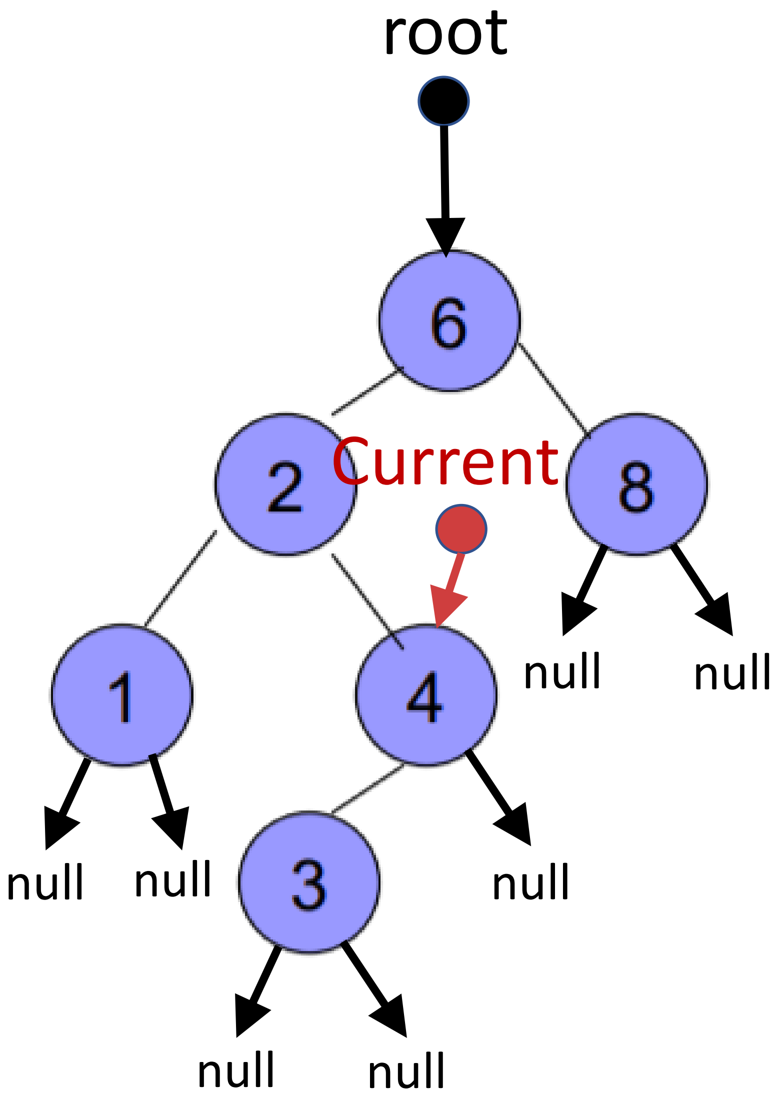
BST - Inorder Traversal using Stack

```
void  inorder()
 {
        node* current=root;

        while(1)

            if (current != NULL)
                  stack.push(current)
                  current = current->left
            else if  (!stack.empty() )
                  current = stack.top()
          ⇨       stack.pop()
                  cout<< current->data;
                  current = current->right
            else
                  return
 }
```

1   2   3



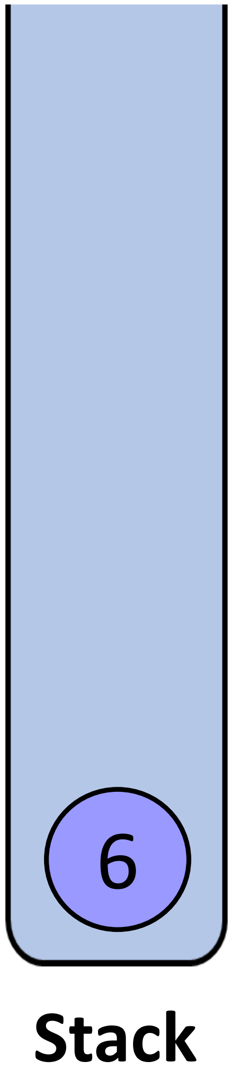
Stack

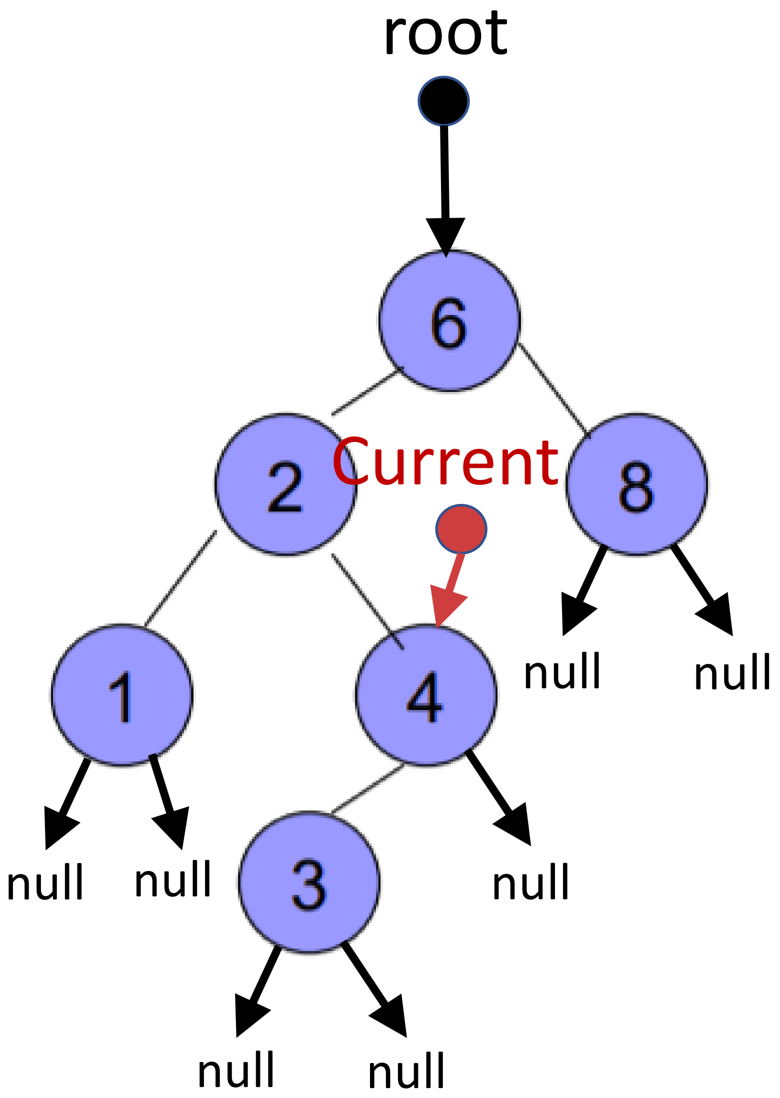


BST - Inorder Traversal using Stack

```
void  inorder()
 {
        node* current=root;

        while(1)

            if (current != NULL)
                    stack.push(current)
                    current = current->left
            else if  (!stack.empty() )
                    current = stack.top()
                    stack.pop()
                    cout<< current->data;
                    current = current->right
            else
                    return
 }
```

1   2   3   4

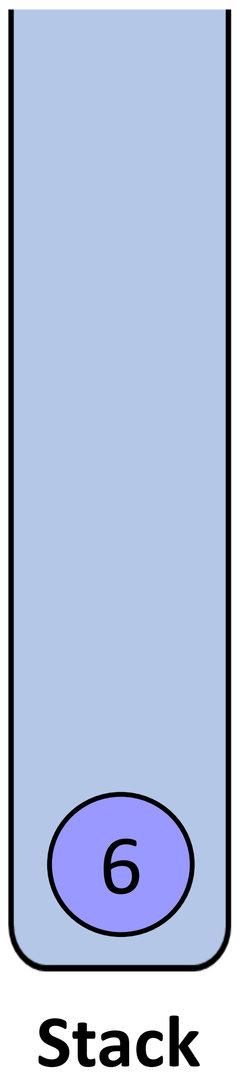


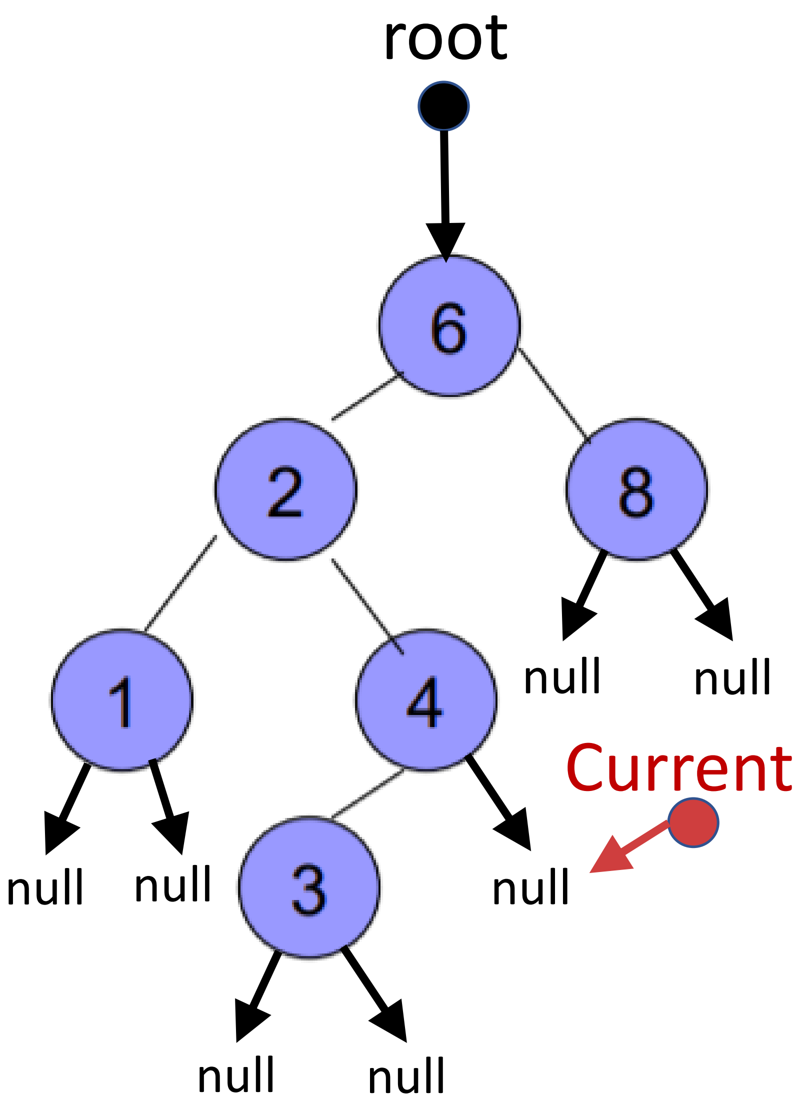


BST - Inorder Traversal using Stack

```
void  inorder()
 {
       node* current=root;

       while(1)

            if (current != NULL)
                  stack.push(current)
                  current = current->left
            else if  (!stack.empty() )
                  current = stack.top()
                  stack.pop()
                  cout<< current->data;
                  current = current->right
            else
                  return
 }
```

1   2   3   4

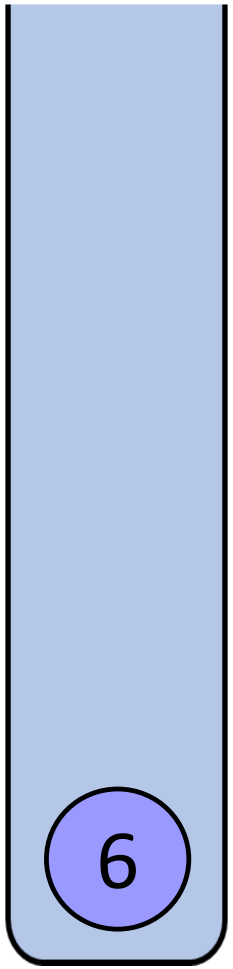




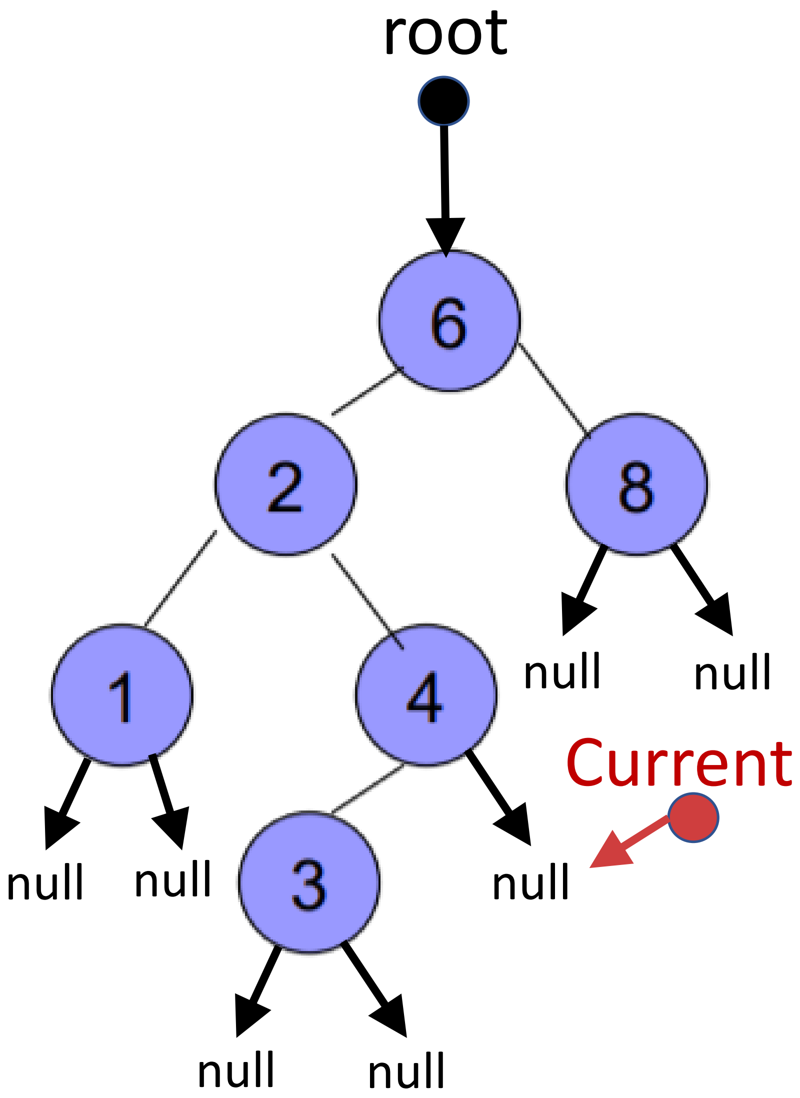
BST - Inorder Traversal using Stack

```
void  inorder()
 {
        node* current=root;

        while(1)

          if (current != NULL)
                 stack.push(current)
                 current = current->left
          else if  (!stack.empty() )
                 current = stack.top()
                 stack.pop()
                 cout<< current->data;
                 current = current->right
          else
                 return
 }
```

1   2   3   4



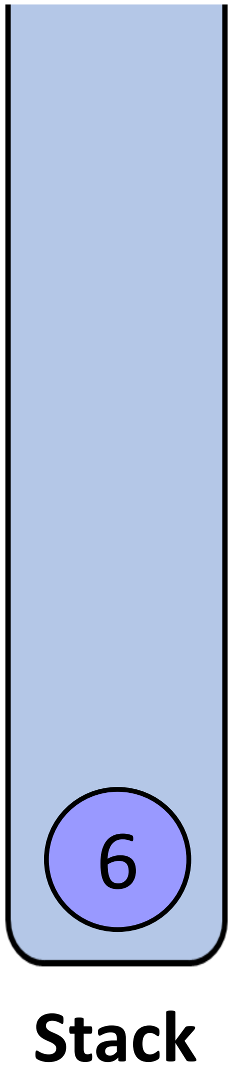
Stack

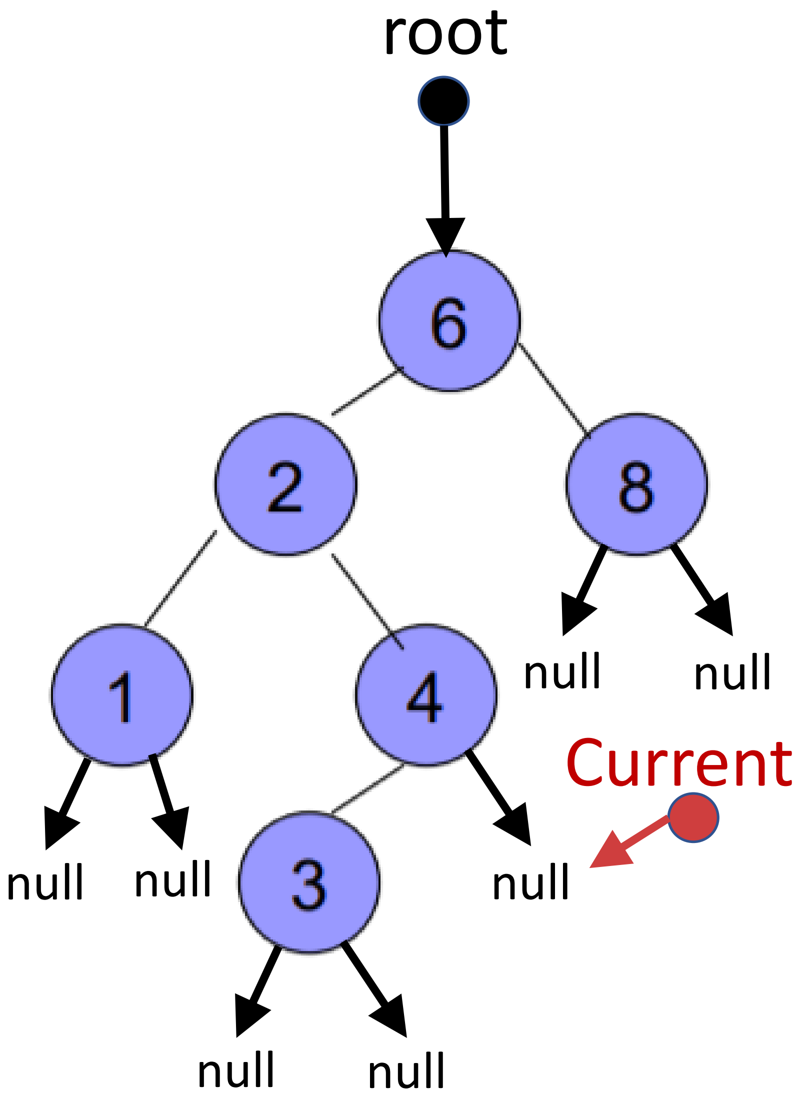


BST - Inorder Traversal using Stack

```
void  inorder()
 {
        node* current=root;

        while(1)

            if (current != NULL)
                  stack.push(current)
                  current = current->left
            else if  (!stack.empty() )
                  current = stack.top()
                  stack.pop()
                  cout<< current->data;
                  current = current->right
            else
                  return
 }
```

1   2   3   4

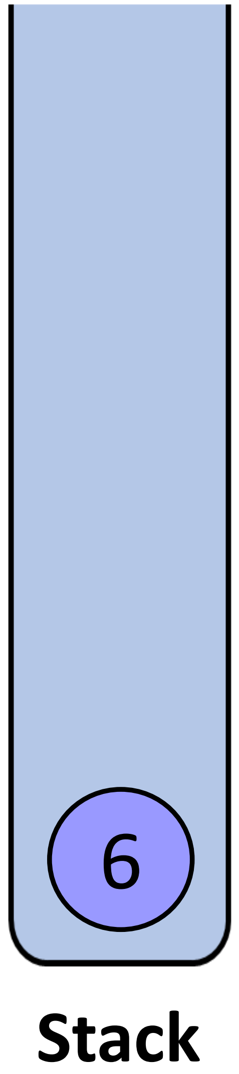


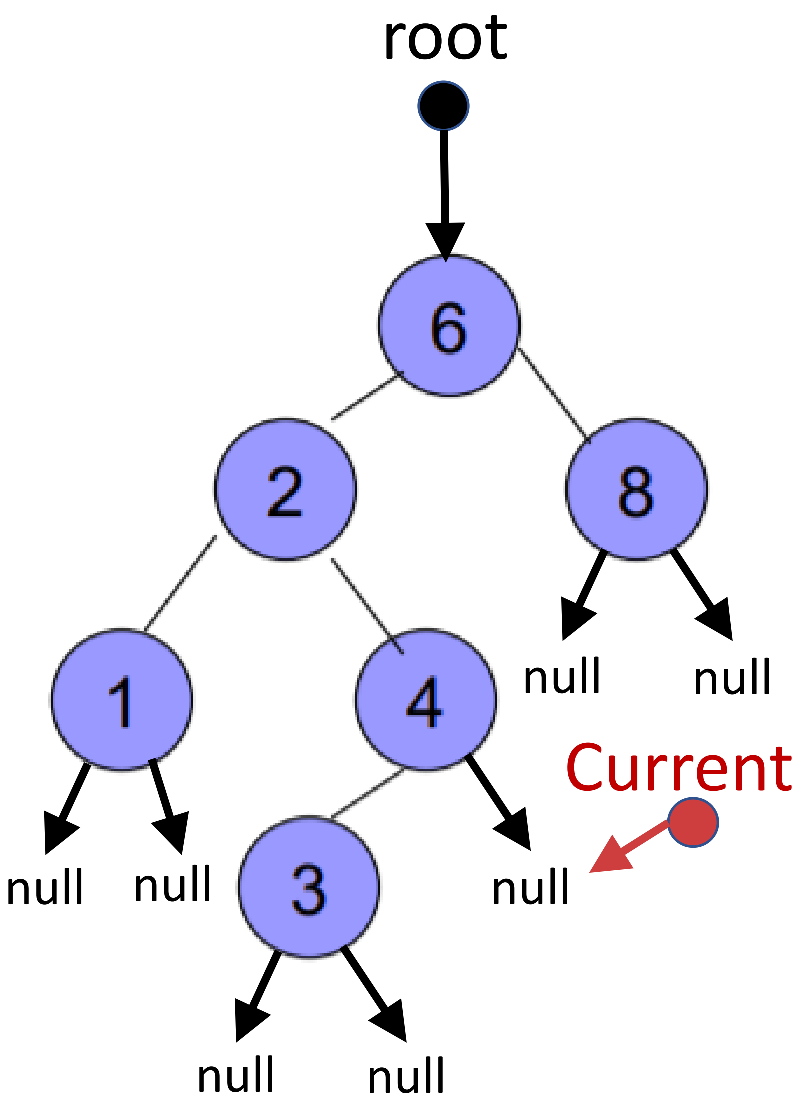


BST - Inorder Traversal using Stack

```
void  inorder()
 {
        node* current=root;

        while(1)

          if (current != NULL)
                 stack.push(current)
                 current = current->left
          else if  (!stack.empty() )
        ⇨        current = stack.top()
                 stack.pop()
                 cout<< current->data;
                 current = current->right
          else
                 return
 }
```

1   2   3   4

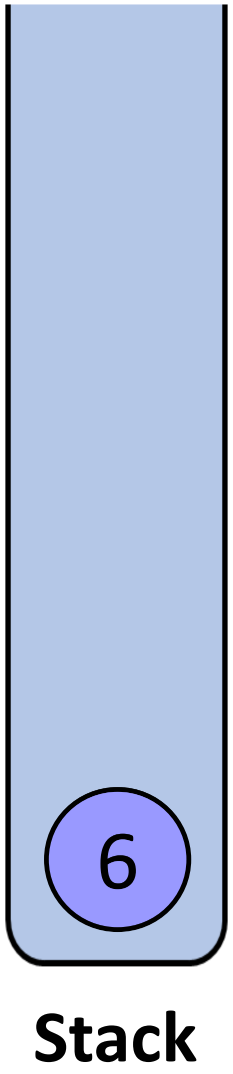


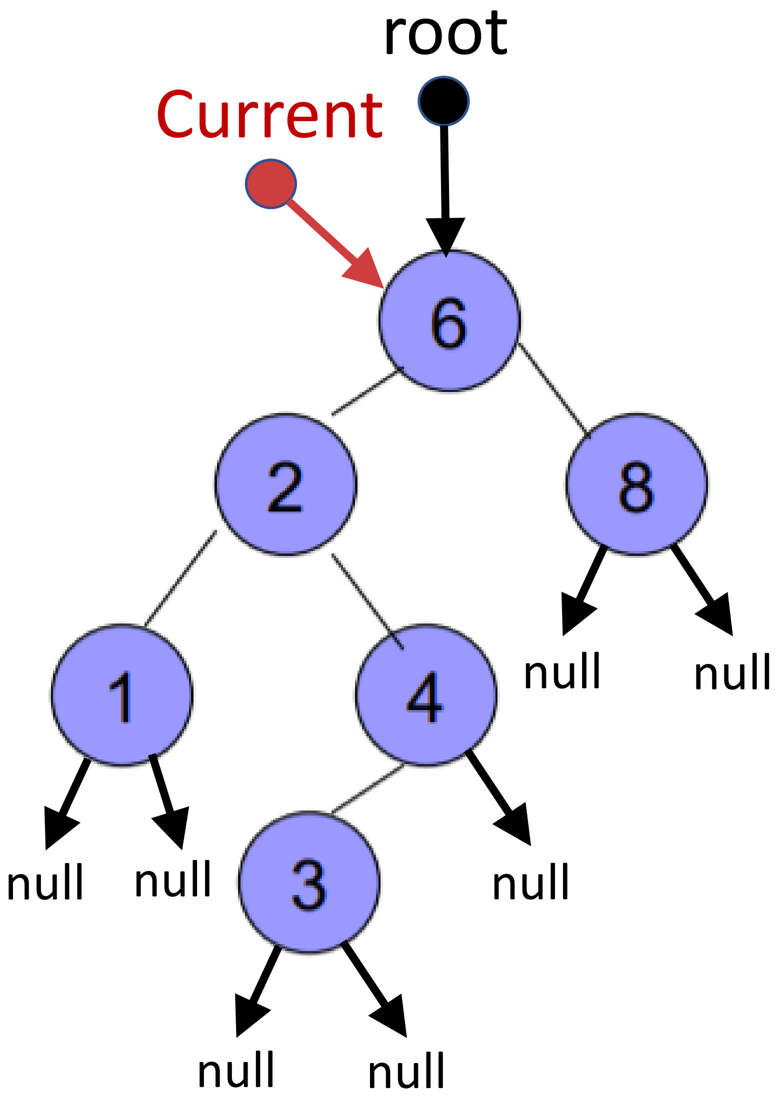


BST - Inorder Traversal using Stack

```
void  inorder()
 {
        node* current=root;

        while(1)

           if (current != NULL)
                  stack.push(current)
                  current = current->left
           else if  (!stack.empty() )
                  current = stack.top()
                  stack.pop()
                  cout<< current->data;
                  current = current->right
           else
                  return
 }
```

1 2 3 4

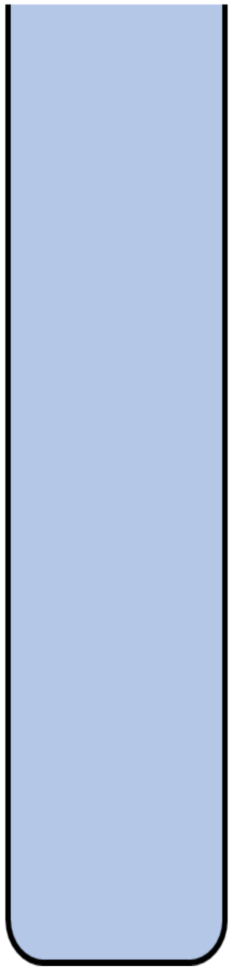




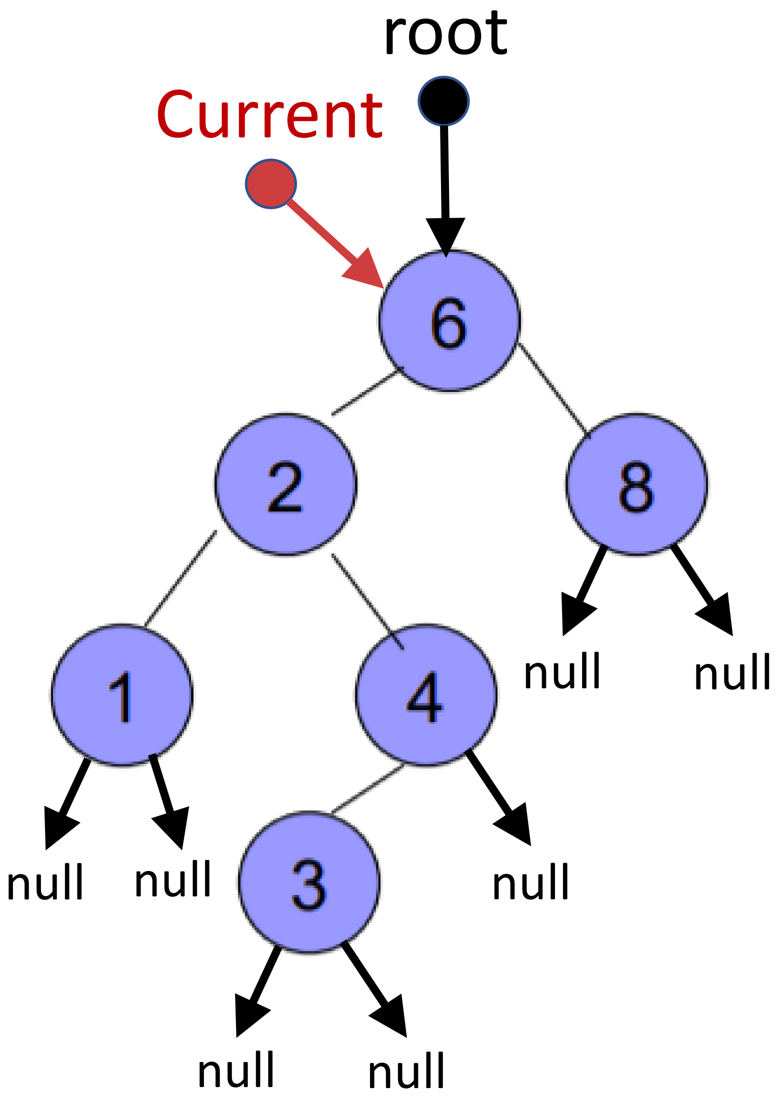
BST - Inorder Traversal using Stack

```
void  inorder()
 {
        node* current=root;

        while(1)

            if (current != NULL)
                    stack.push(current)
                    current = current->left
            else if  (!stack.empty() )
                    current = stack.top()
            ➡      stack.pop()
                    cout<< current->data;
                    current = current->right
            else
                    return
 }
```

1   2   3   4



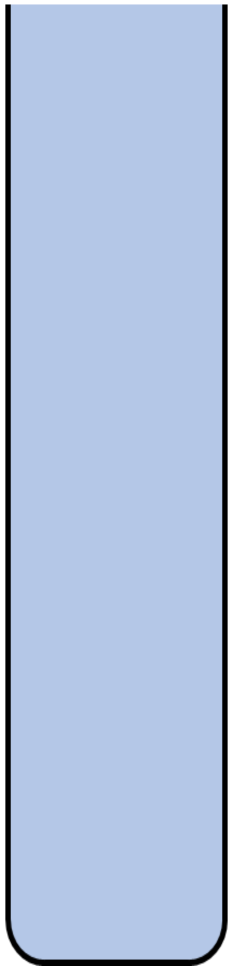
Stack



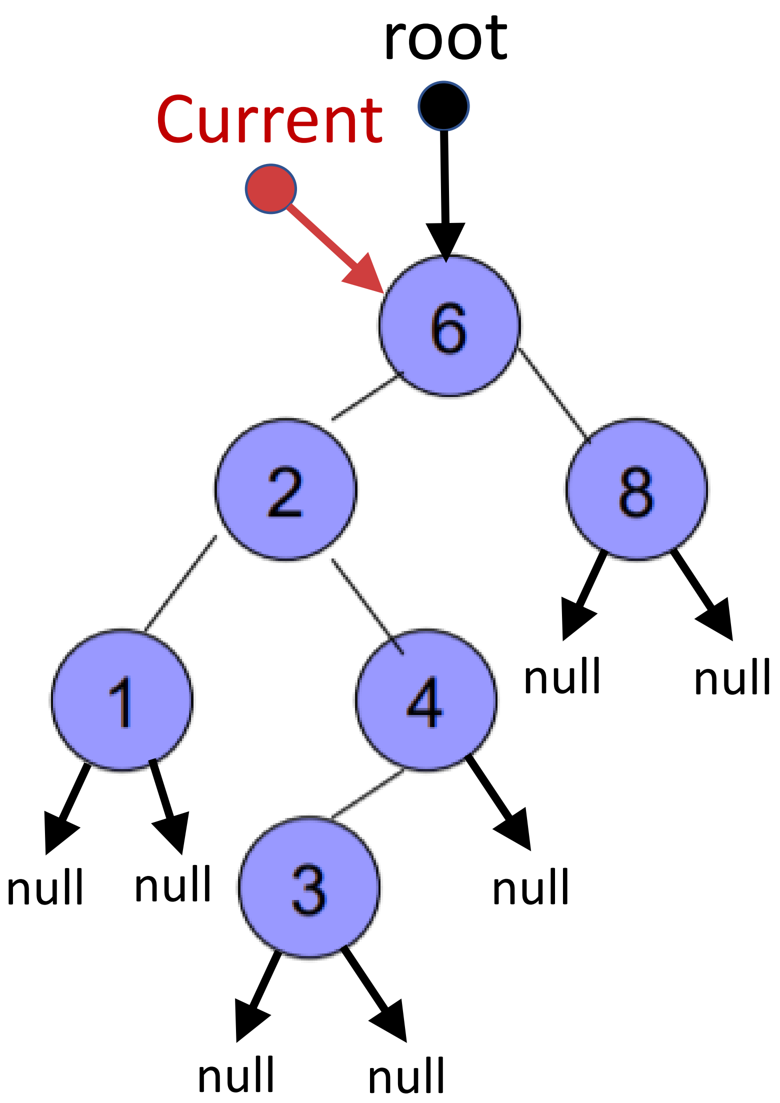
BST - Inorder Traversal using Stack

```
void  inorder()
 {
        node* current=root;

        while(1)

            if (current != NULL)
                    stack.push(current)
                    current = current->left
            else if  (!stack.empty() )
                    current = stack.top()
                    stack.pop()
            ⇨       cout<< current->data;
                    current = current->right
            else
                    return
  }
```

1 2 3 4 6



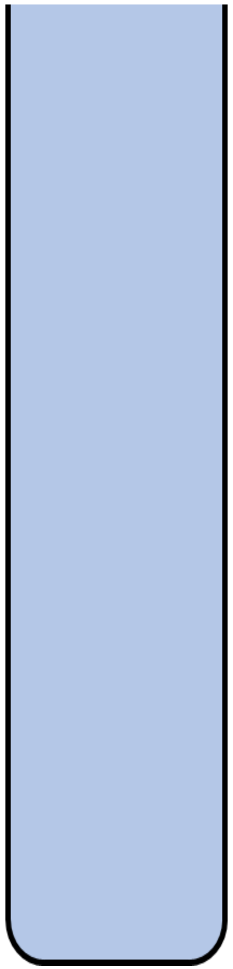
Stack



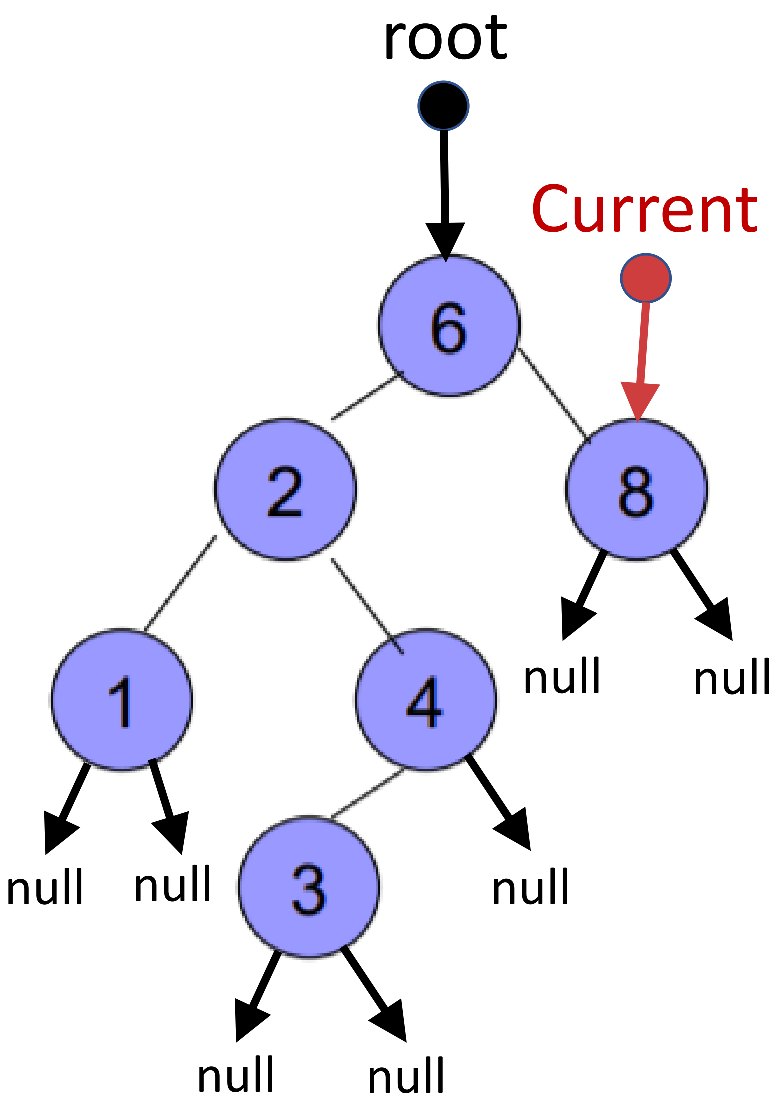
BST - Inorder Traversal using Stack

```
void  inorder()
 {
      node* current=root;

      while(1)

         if (current != NULL)
                stack.push(current)
                current = current->left
         else if  (!stack.empty() )
                current = stack.top()
                stack.pop()
                cout<< current->data;
                current = current->right
         else
                return
 }
```

1   2   3   4   6



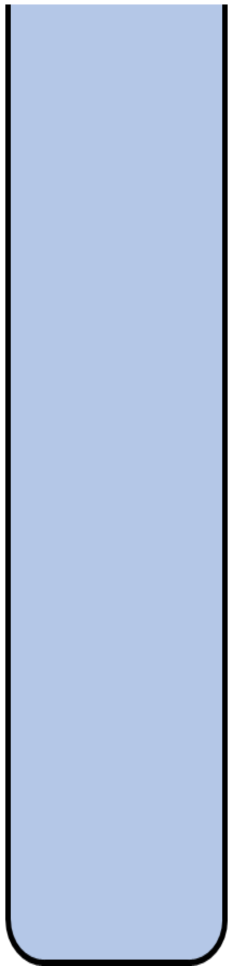
Stack



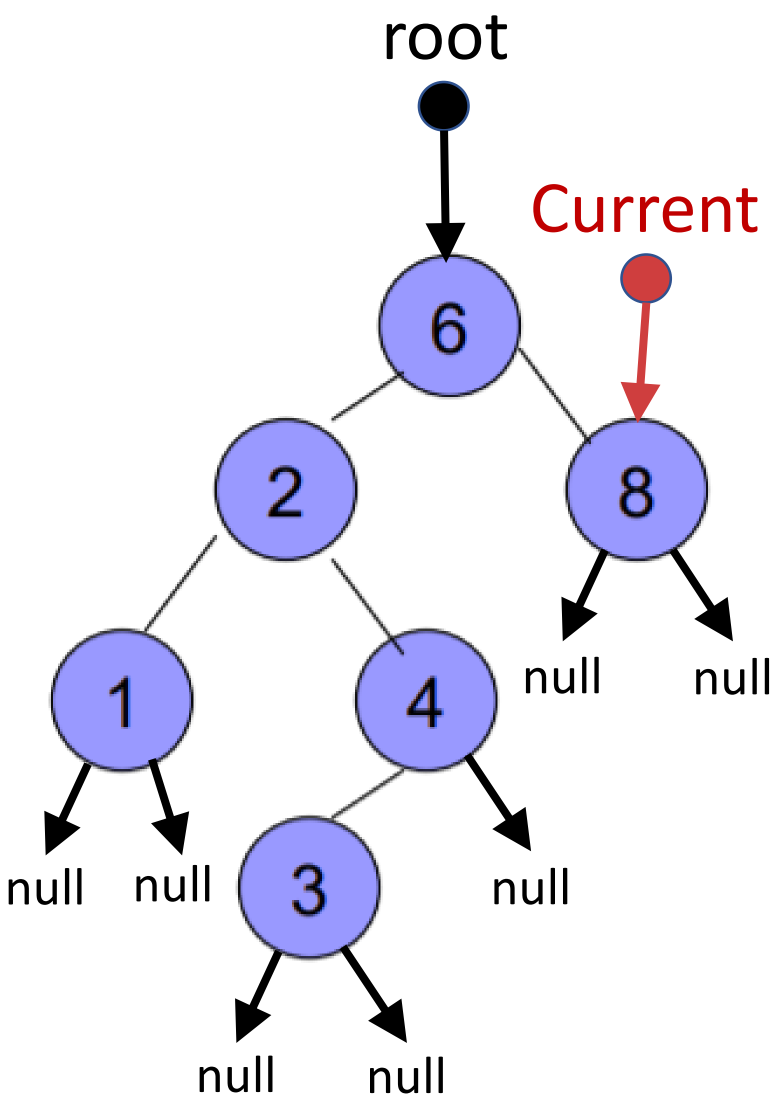
BST - Inorder Traversal using Stack

```
void  inorder()
 {
        node* current=root;

        while(1)

          if (current != NULL)
                stack.push(current)
                current = current->left
          else if  (!stack.empty() )
                current = stack.top()
                stack.pop()
                cout<< current->data;
                current = current->right
          else
                return
 }
```

1 2 3 4 6



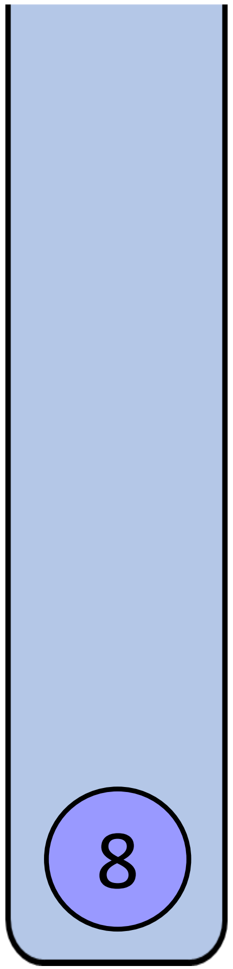
Stack



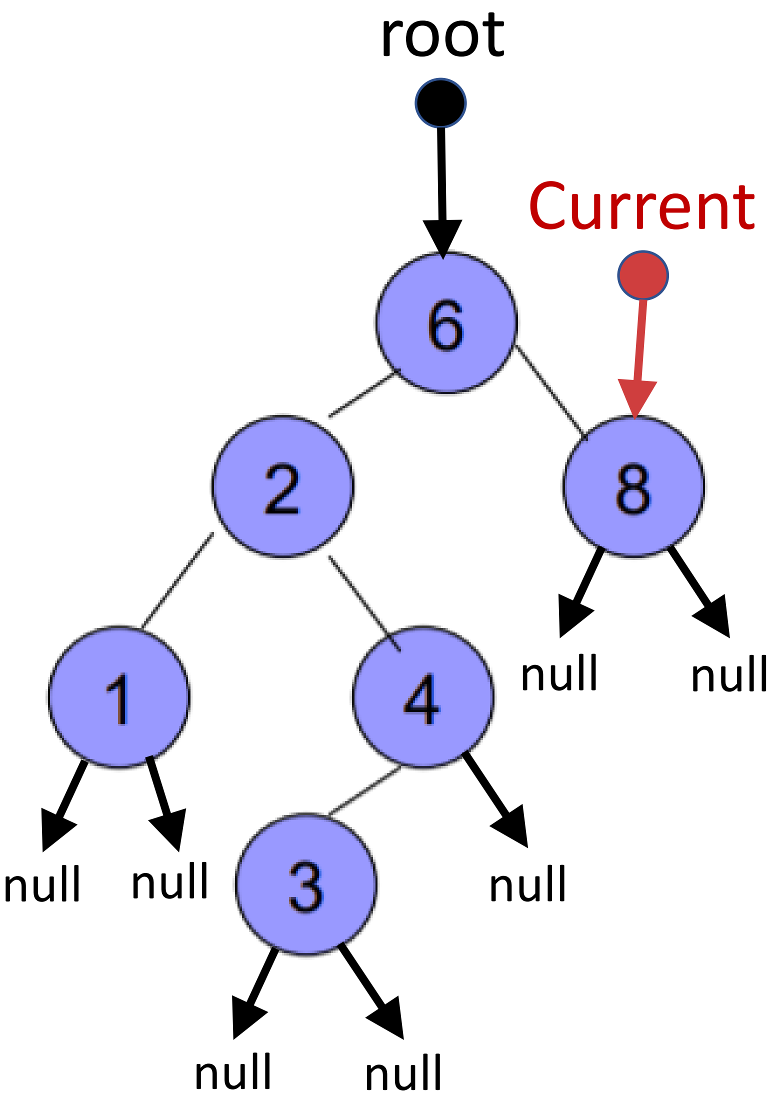
BST - Inorder Traversal using Stack

```
void  inorder()
 {
      node* current=root;

      while(1)

           if (current != NULL)
                stack.push(current)
                current = current->left
           else if  (!stack.empty() )
                current = stack.top()
                stack.pop()
                cout<< current->data;
                current = current->right
           else
                return
 }
```

1   2   3   4   6



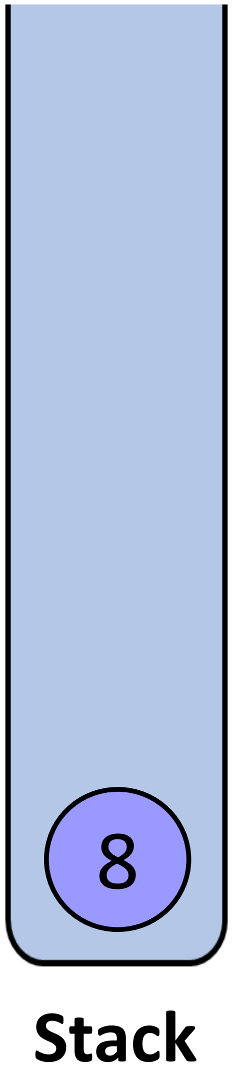
Stack

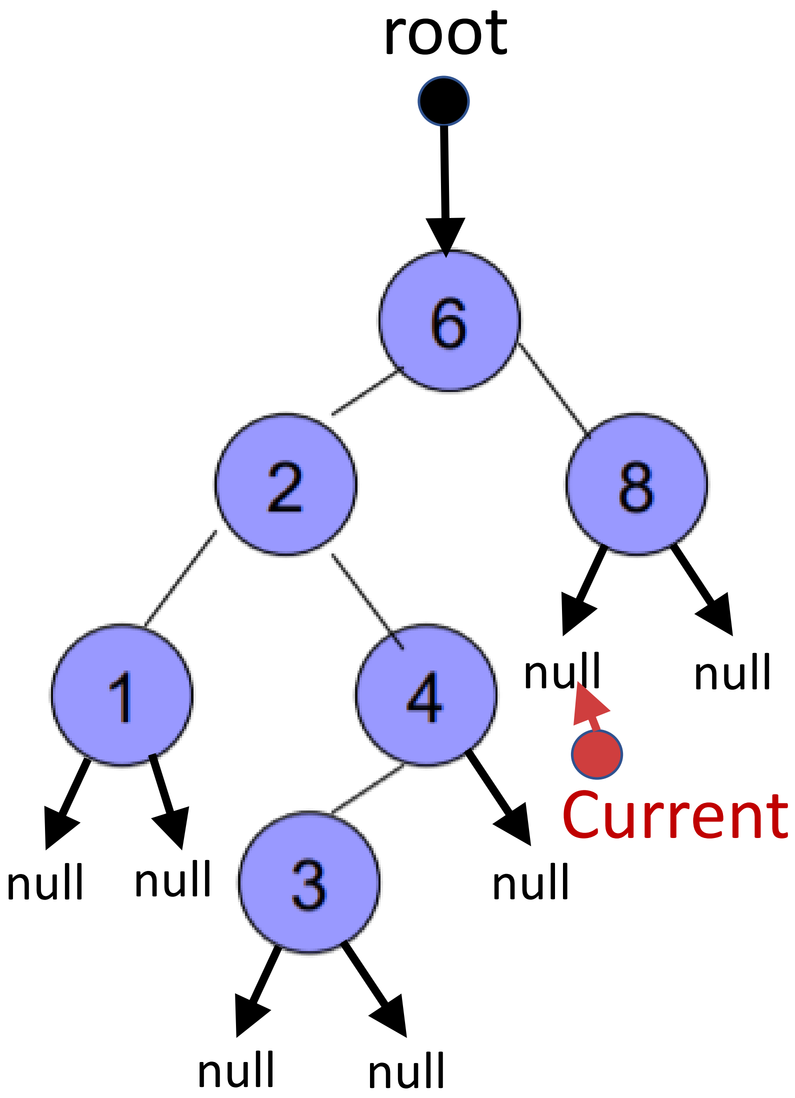


BST - Inorder Traversal using Stack

```
void  inorder()
 {
        node* current=root;

        while(1)

          if (current != NULL)
                stack.push(current)
                current = current->left
          else if  (!stack.empty() )
                current = stack.top()
                stack.pop()
                cout<< current->data;
                current = current->right
          else
                return
 }
```

1   2   3   4   6

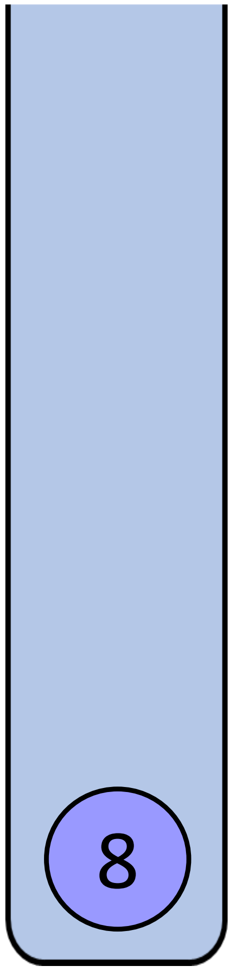




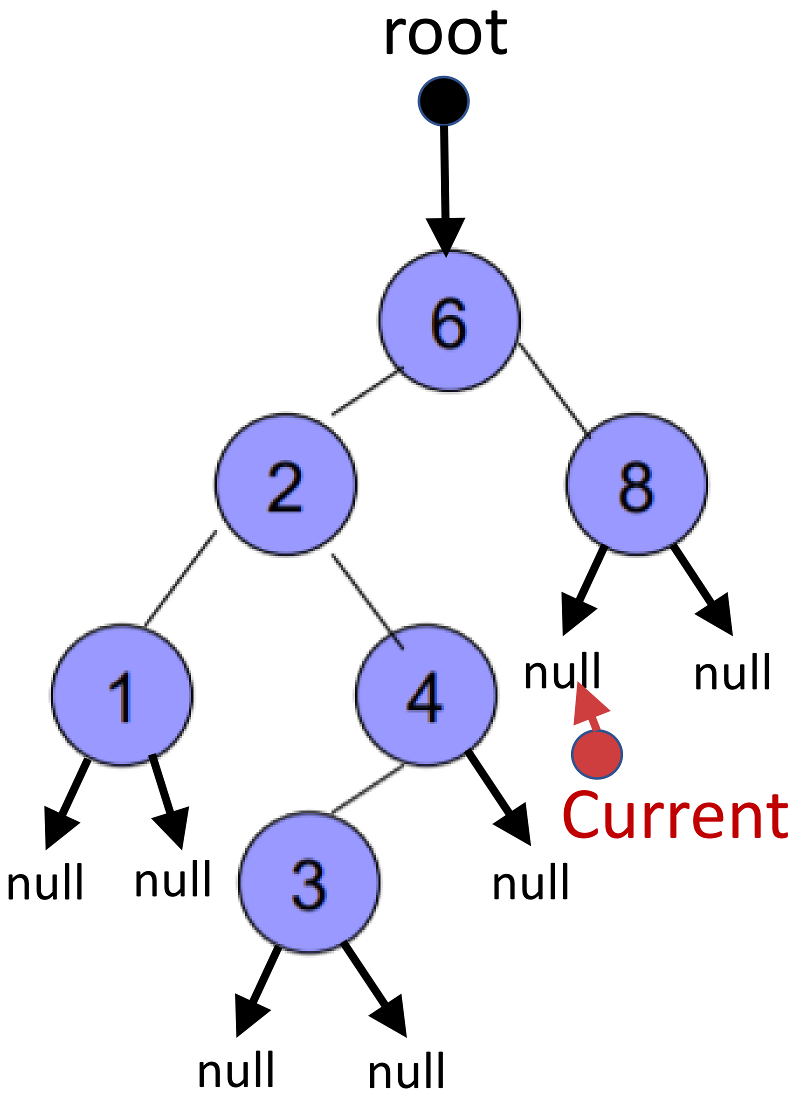
BST - Inorder Traversal using Stack

```
void  inorder()
 {
        node* current=root;

        while(1)

          if (current != NULL)
                stack.push(current)
                current = current->left
          else if  (!stack.empty() )
                current = stack.top()
                stack.pop()
                cout<< current->data;
                current = current->right
          else
                return
 }
```

1   2   3   4   6



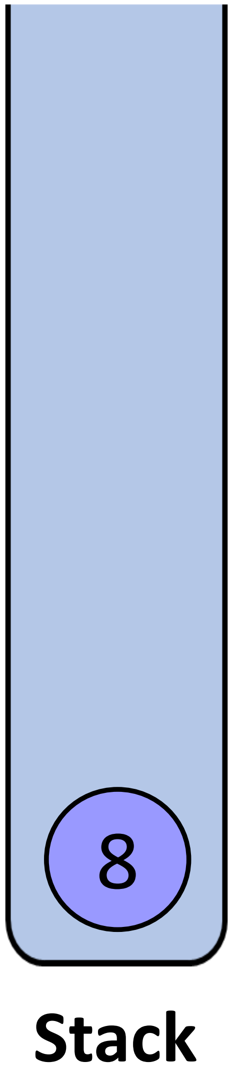
Stack

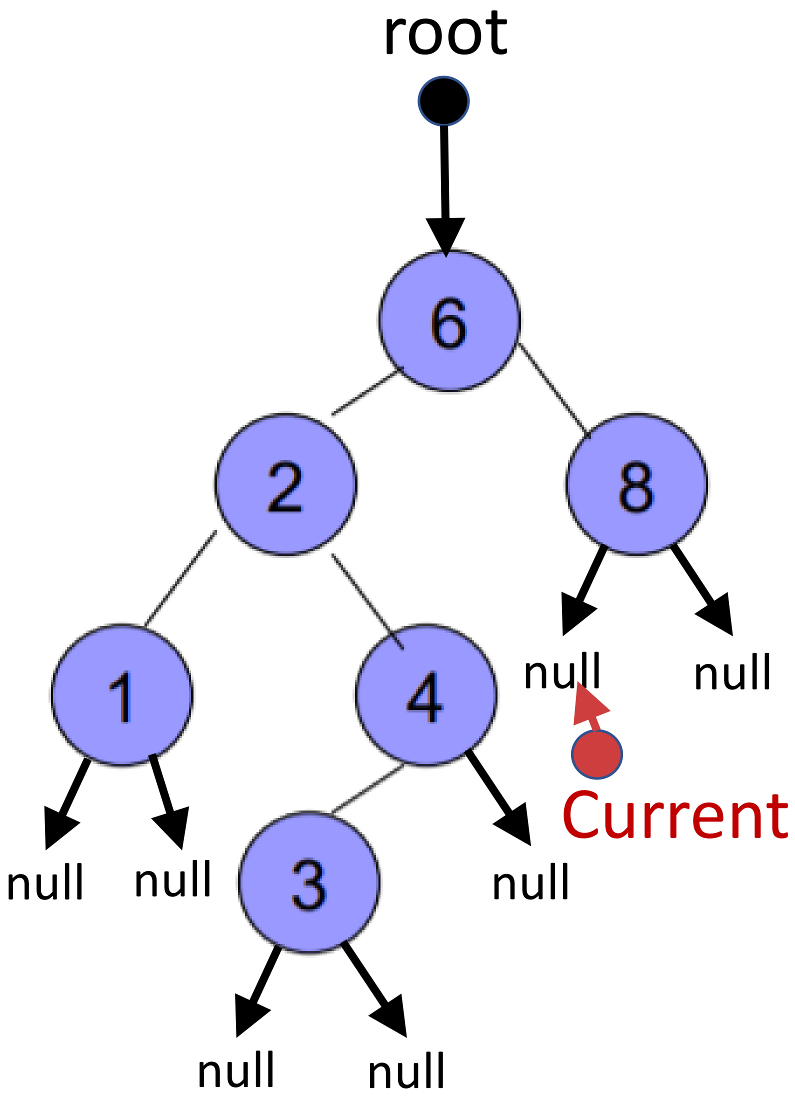


BST - Inorder Traversal using Stack

```
void  inorder()
 {
      node* current=root;

      while(1)

         if (current != NULL)
                stack.push(current)
                current = current->left
         else if  (!stack.empty() )
                current = stack.top()
                stack.pop()
                cout<< current->data;
                current = current->right
         else
                return
 }
```

1   2   3   4   6

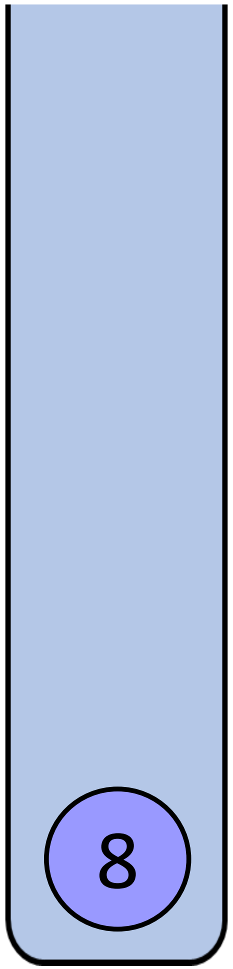




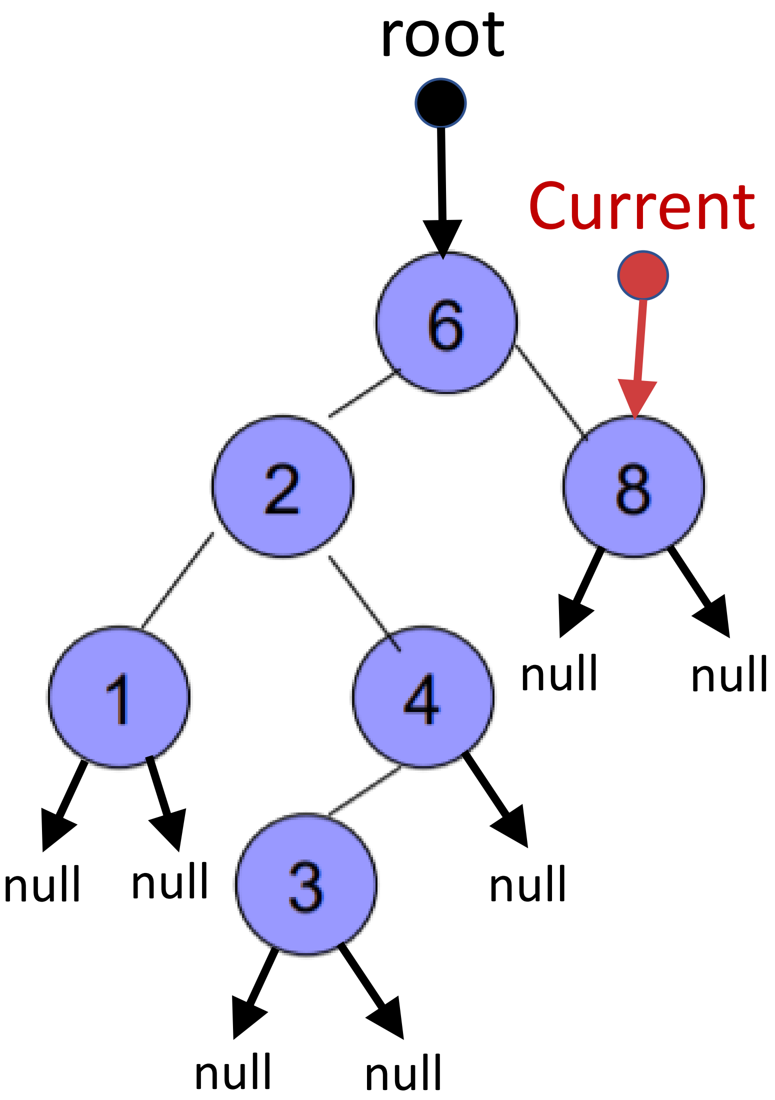
BST - Inorder Traversal using Stack

```
void  inorder()
 {
       node* current=root;

       while(1)

          if (current != NULL)
                 stack.push(current)
                 current = current->left
          else if  (!stack.empty() )
                 current = stack.top()
                 stack.pop()
                 cout<< current->data;
                 current = current->right
          else
                 return
 }
```

1   2   3   4   6



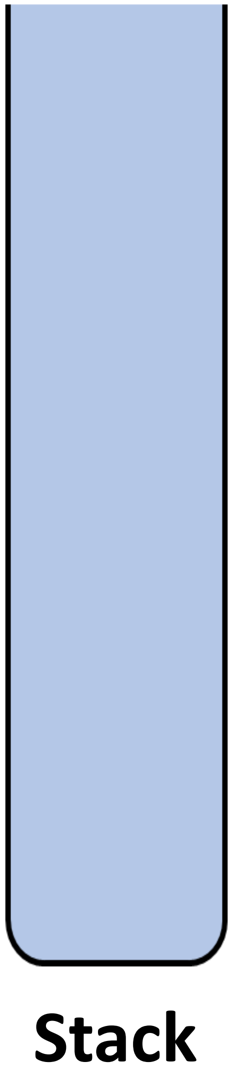
Stack

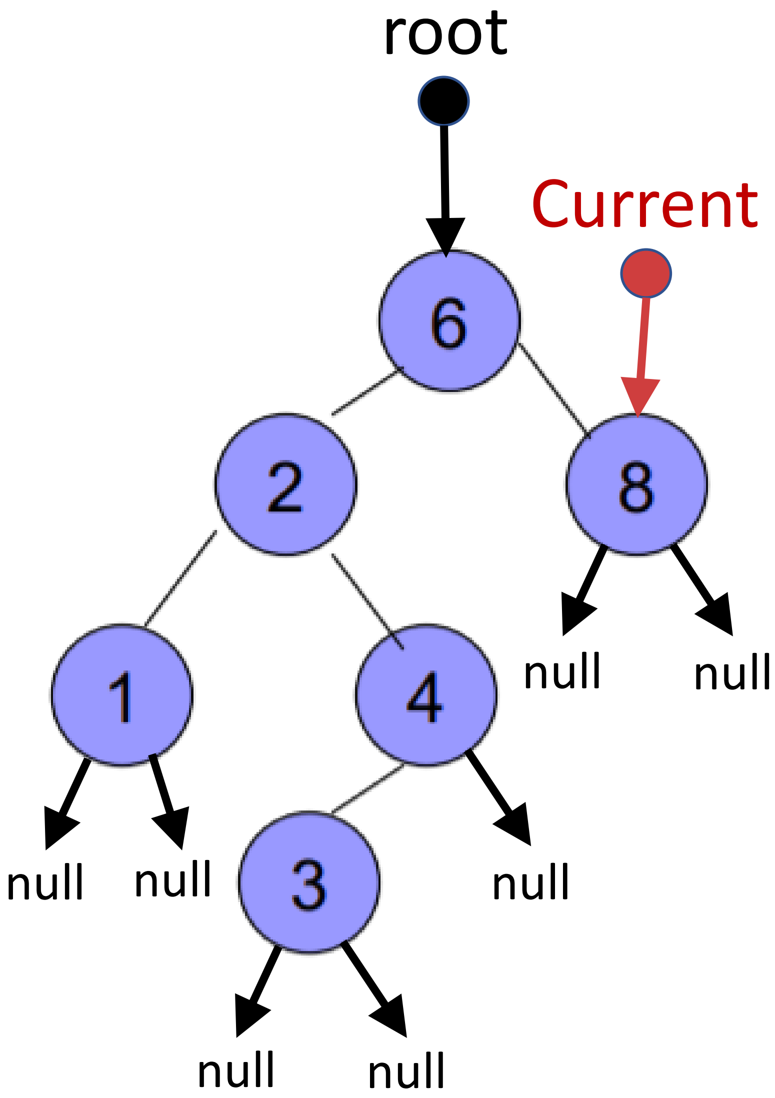


BST - Inorder Traversal using Stack

```
void  inorder()
 {
        node* current=root;

        while(1)

            if (current != NULL)
                    stack.push(current)
                    current = current->left
            else if  (!stack.empty() )
                    current = stack.top()
                    stack.pop()
                    cout<< current->data;
                    current = current->right
            else
                    return
  }
```

1   2   3   4   6

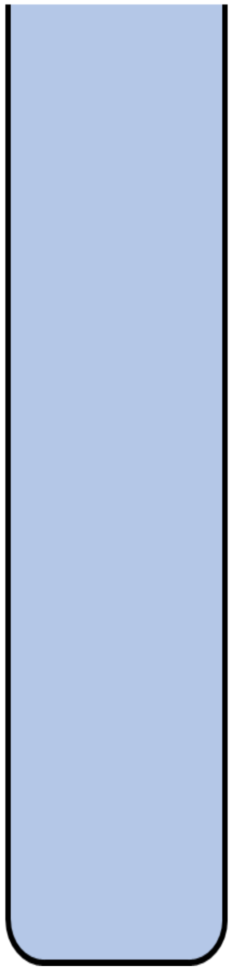




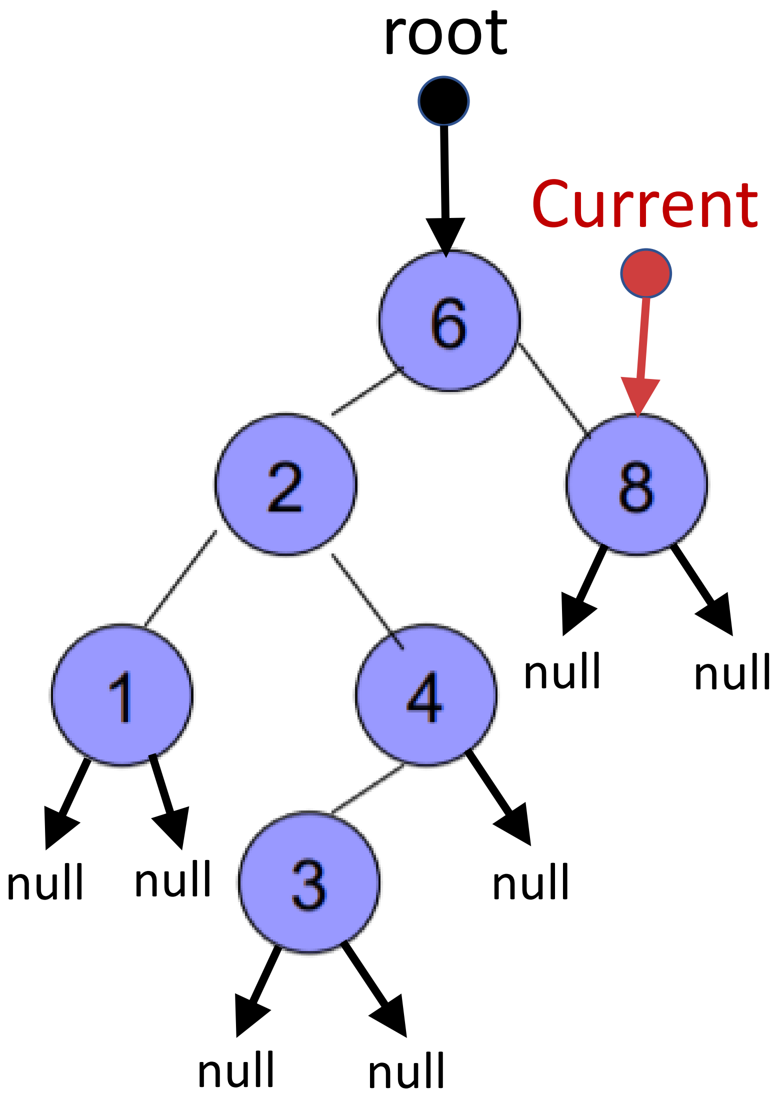
BST - Inorder Traversal using Stack

```
void  inorder()
 {
        node* current=root;

        while(1)

            if (current != NULL)
                   stack.push(current)
                   current = current->left
            else if  (!stack.empty() )
                   current = stack.top()
                   stack.pop()
                   cout<< current->data;
                   current = current->right
            else
                   return
 }
```

1   2   3   4   6   8



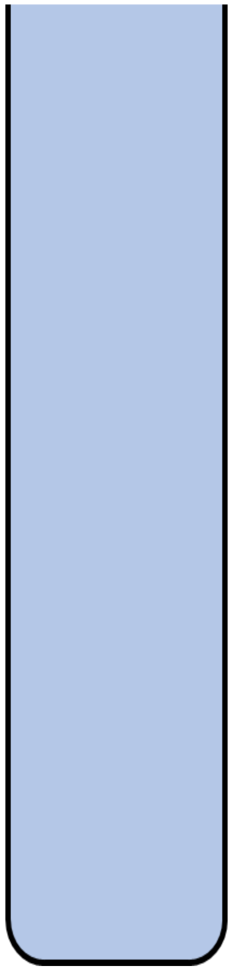
Stack



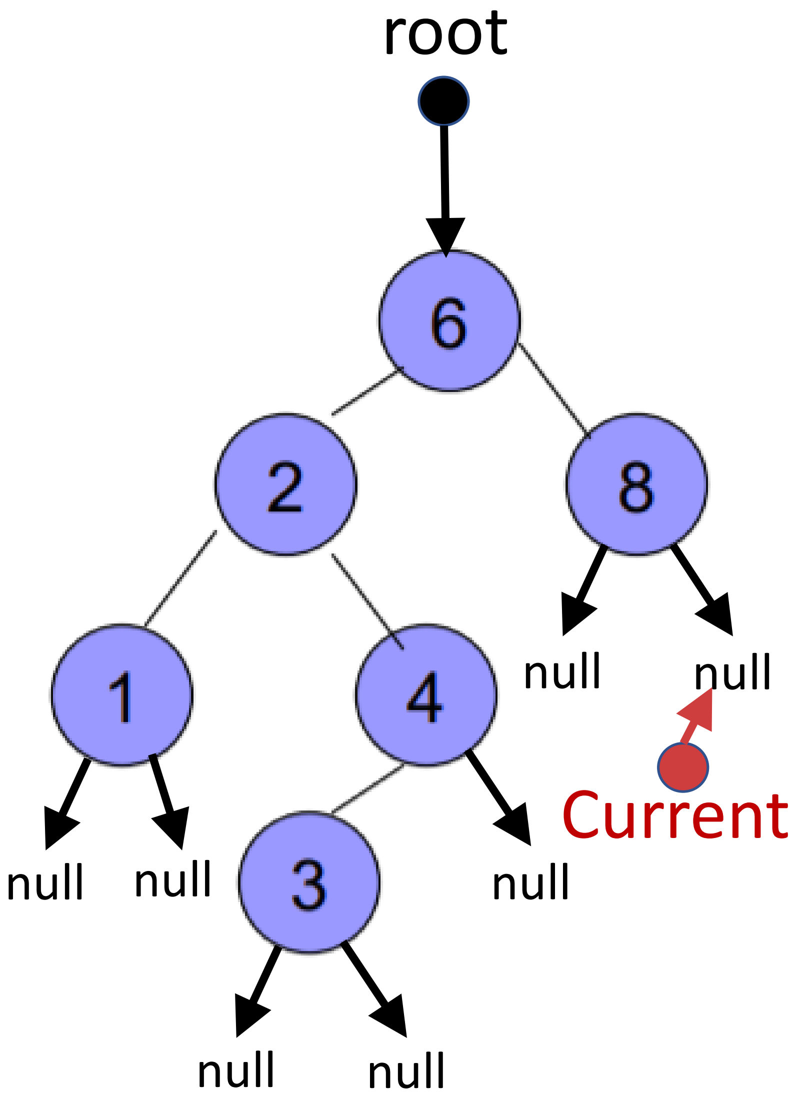
BST - Inorder Traversal using Stack

```
void  inorder()
 {
      node* current=root;

      while(1)

          if (current != NULL)
                stack.push(current)
                current = current->left
          else if  (!stack.empty() )
                current = stack.top()
                stack.pop()
                cout<< current->data;
                current = current->right
          else
                return
 }
```

1 2 3 4 6 8



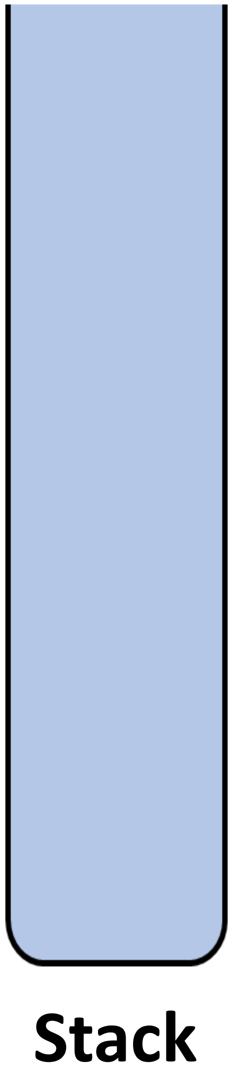
Stack

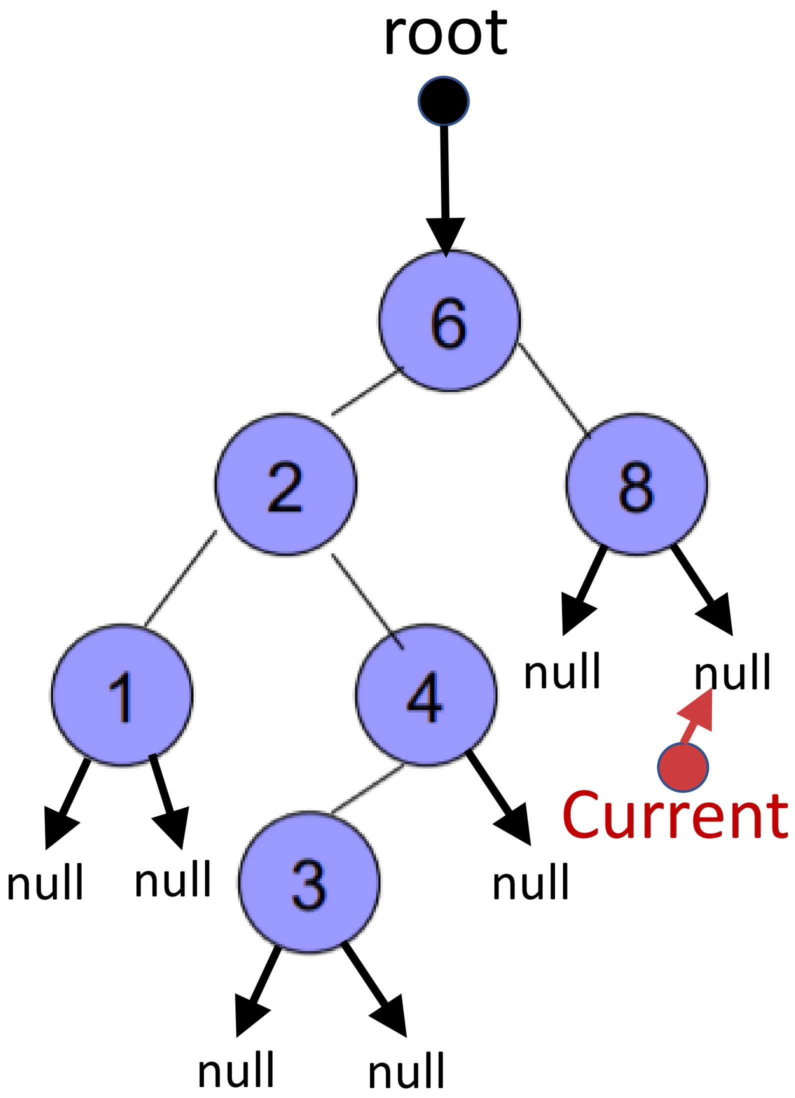


BST - Inorder Traversal using Stack

```
void  inorder()
 {
        node* current=root;

        while(1)

          if (current != NULL)
                  stack.push(current)
                  current = current->left
          else if  (!stack.empty() )
                  current = stack.top()
                  stack.pop()
                  cout<< current->data;
                  current = current->right
          else
                  return
 }
```

1   2   3   4   6   8

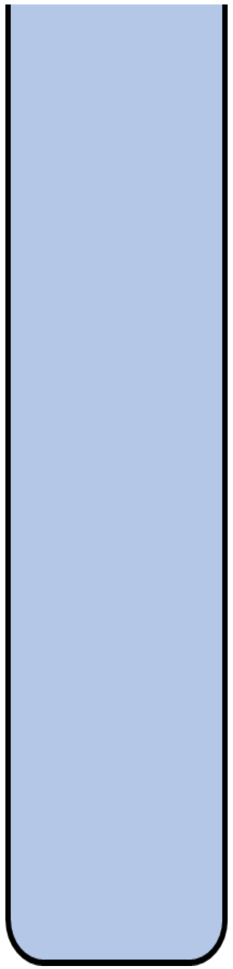




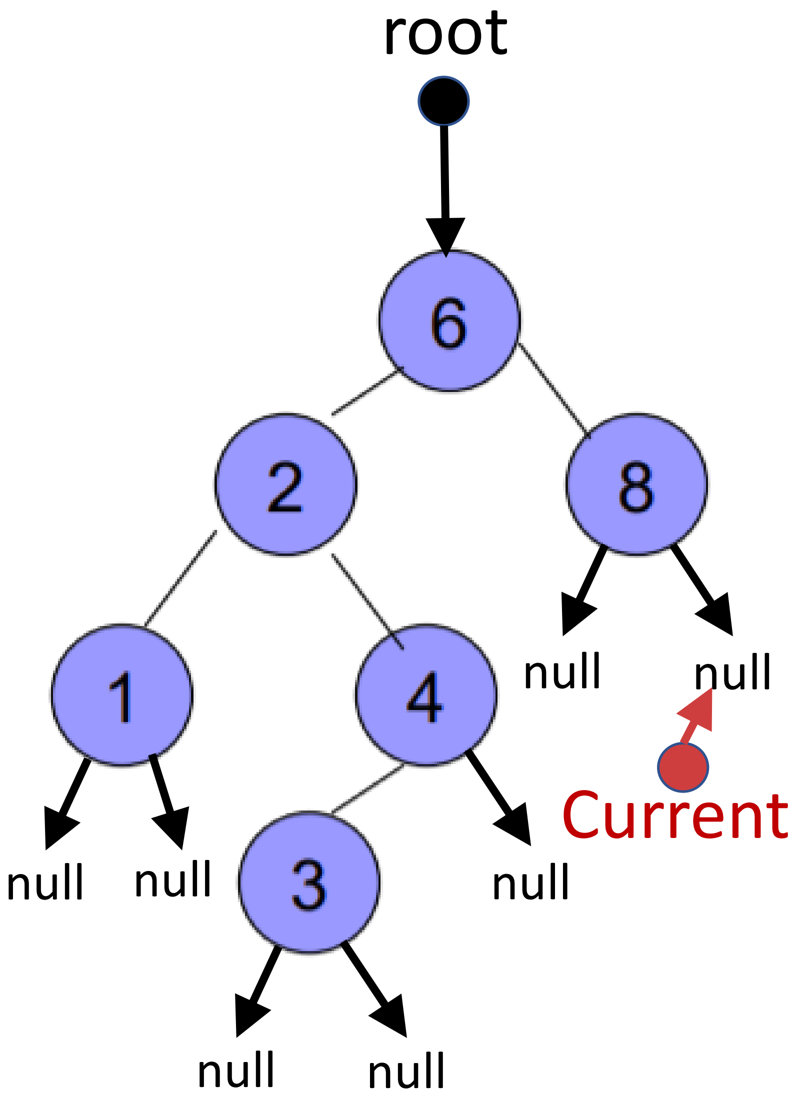
BST - Inorder Traversal using Stack

```
void  inorder()
 {
        node* current=root;

        while(1)

           if (current != NULL)
                  stack.push(current)
                  current = current->left
           else if  (!stack.empty() )
                  current = stack.top()
                  stack.pop()
                  cout<< current->data;
                  current = current->right
           else
                  return
 }
```

1   2   3   4   6   8



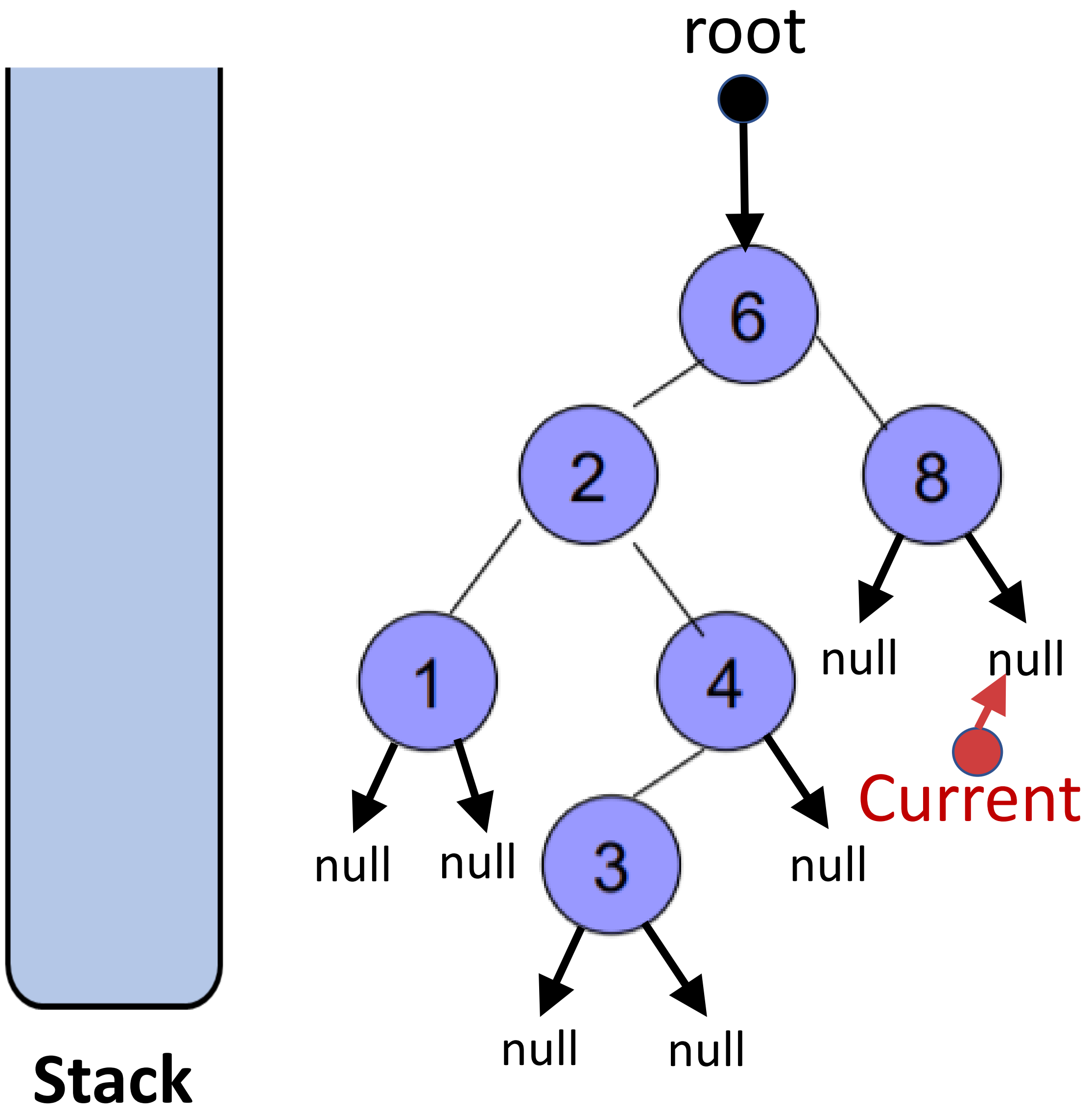
Stack



BST - Inorder Traversal using Stack

```
void  inorder()
 {
        node* current=root;

        while(1)

            if (current != NULL)
                    stack.push(current)
                    current = current->left
            else if  (!stack.empty() )
                    current = stack.top()
                    stack.pop()
                    cout<< current->data;
                    current = current->right
            else
                    return
 }
```

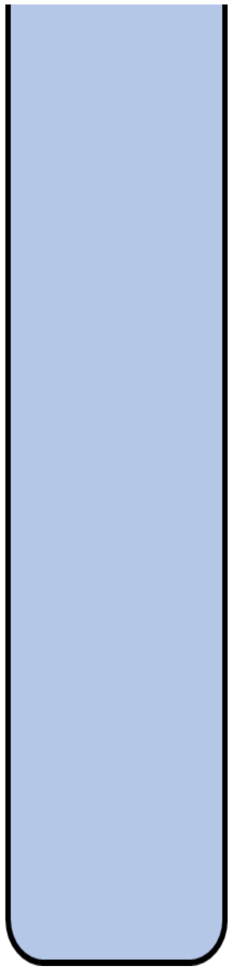
1   2   3   4   6   8



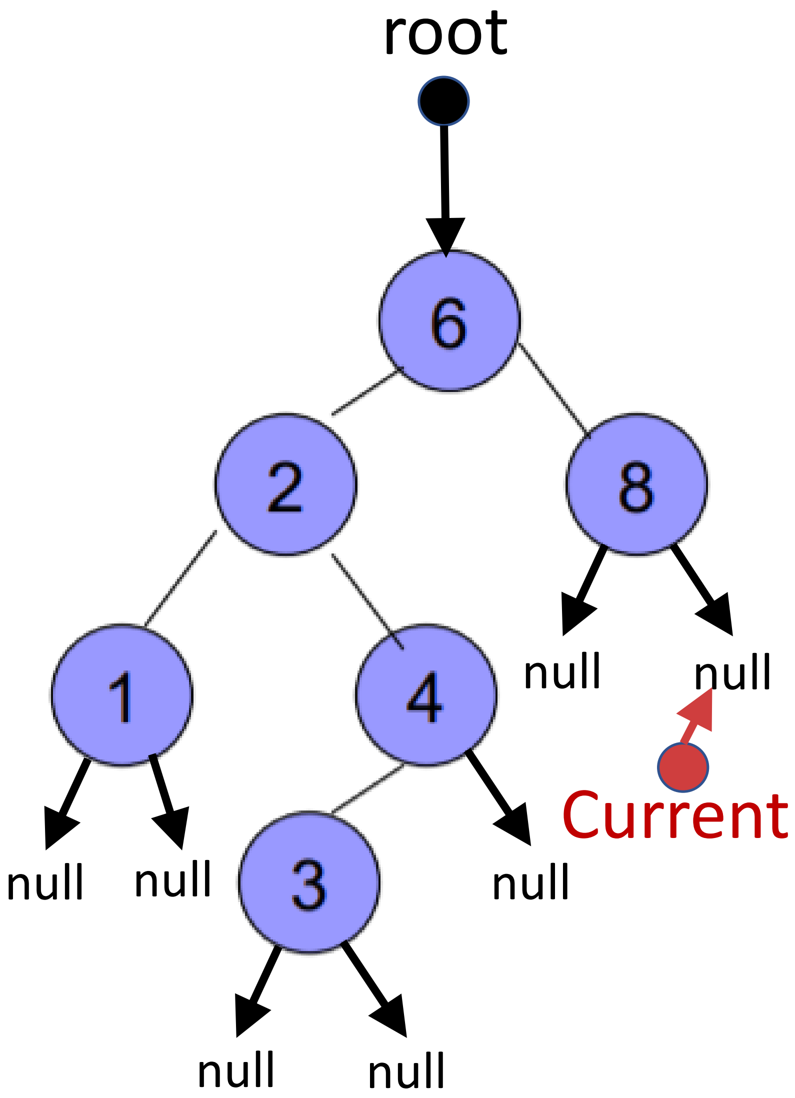
BST - Inorder Traversal using Stack

```
void  inorder()
 {
        node* current=root;

        while(1)

            if (current != NULL)
                    stack.push(current)
                    current = current->left
            else if  (!stack.empty() )
                    current = stack.top()
                    stack.pop()
                    cout<< current->data;
                    current = current->right
            else
          ➡       return
  }
```

1   2   3   4   6   8



Stack



Thanks a lot

If you are taking a Nap, **wake up**........Lecture Over