

Recursive Binary Tree Search

Trace the following code

```
1 def binary_tree_search(node, search_item):
2     path.append(values[node])
3     if values[node] == search_item:
4         return "Value in Tree. Path: "+str(path)
5     elif values[node] < search_item:
6         if tree_right[node] == -1:
7             return "Value not in Tree"
8         return binary_tree_search(tree_right[node], search_item)
9     elif values[node] > search_item:
10        if tree_left[node] == -1:
11            return "Value not in Tree"
12        return binary_tree_search(tree_left[node], search_item)
13
14 path = [ ]
15 #       node[ 0,  1,  2,  3,  4,  5,  6,  7,  8,  9]
16 values  = [10,  1, 17,  4, 11,  8, 14,  5, 12, 16]
17 tree_left = [ 1, -1,  4, -1, -1,  7,  8, -1, -1, -1]
18 tree_right= [ 2,  3, -1,  5,  6, -1,  9, -1, -1, -1]
19 print(binary_tree_search(0, 5))
```

call	node	search_item	path	Output
			[]	

Stack Heap

Iterative Binary Tree Search

Trace the following code

```

1  def binaryTreeSearch(values,tree_right,tree_left,search_item):
2      node=0
3      path = []
4      while values[node] != search_item and node!=-1:
5          path.append(values[node])
6          if values[node] > search_item:
7              node=tree_left[node]
8          else:
9              node=tree_right[node]
10         if node!= -1:
11             path.append(values[node])
12             return "vistied nodes",path
13         else:
14             return "Value not found in tree"
15     values    =  [10,1,17,4,11,8,14,5,12,16]
16     tree_left = [1,-1,4,-1,-1,7,8,-1,-1,-1]
17     tree_right=[2,3,-1,5,6,-1,9,-1,-1,-1]
18     search_item=8
19     print(binaryTreeSearch(values,tree_right,tree_left,search_item))
  
```

node	search_item	path

