

## **Title: Range sum of binary search tree**

**EX. NO : 04**

**Name: Parth Langalia**

**DATE : 13-02-2023**

**Reg No. : RA2011033010033**

### **AIM :**

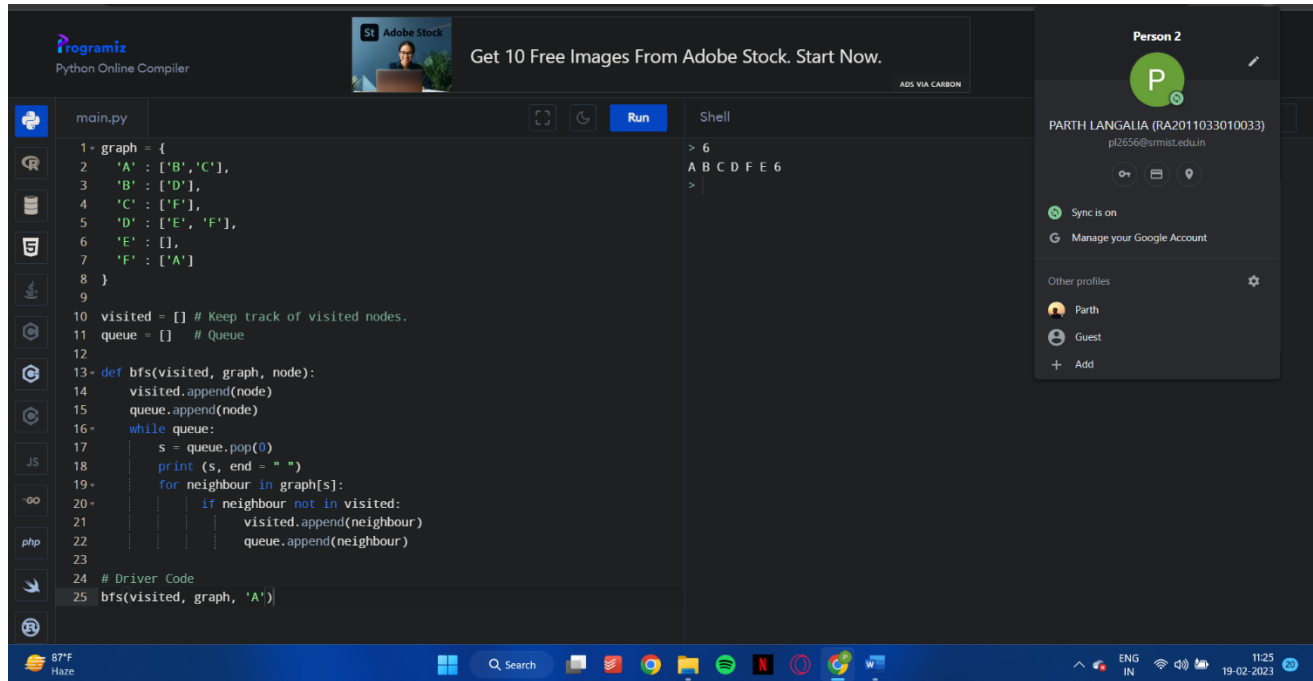
Implementation of Breadth first and depth first search in the travelling salesman problem.

### **PSEUDO CODE :**

1. We traverse the tree using a depth first search.
2. If node.value falls outside the range  $[L, R]$ , (for example  $\text{node.val} < L$ ), then we know that only the right branch could have nodes with value inside  $[L, R]$ .
3. We showcase two implementations - one using a recursive algorithm, and one using an iterative one.
4. Time Complexity:  $O(N)O(N)$ , where  $NN$  is the number of nodes in the tree.
5. Space Complexity:  $O(N)O(N)$
6. For the recursive implementation, the recursion will consume additional space in the function call stack. In the worst case, the tree is of chain shape, and we will reach all the way down to the leaf node.
7. For the iterative implementation, essentially we are doing a BFS (Breadth-First Search) traversal, where the stack will contain no more than two levels of the nodes. The maximal number of nodes in a binary tree is  $N/2$ .
8. Therefore, the maximal space needed for the stack would be  $O(N)O(N)$ .

## PROGRAM :

### BFS :



Programiz Python Online Compiler

Get 10 Free Images From Adobe Stock. Start Now.

main.py

```
1- graph = {
2  'A' : ['B','C'],
3  'B' : ['D'],
4  'C' : ['F'],
5  'D' : ['E', 'F'],
6  'E' : [],
7  'F' : ['A']
8 }
9
10 visited = [] # Keep track of visited nodes.
11 queue = [] # Queue
12
13 def bfs(visited, graph, node):
14     visited.append(node)
15     queue.append(node)
16     while queue:
17         s = queue.pop(0)
18         print(s, end = " ")
19         for neighbour in graph[s]:
20             if neighbour not in visited:
21                 visited.append(neighbour)
22                 queue.append(neighbour)
23
24 # Driver Code
25 bfs(visited, graph, 'A')
```

Shell

```
> 6
A B C D F E 6
>
```

Person 2

PARTH LANGALIA (RA2011033010033)  
pt2656@srmstedu.in

Sync is on  
Manage your Google Account

Other profiles

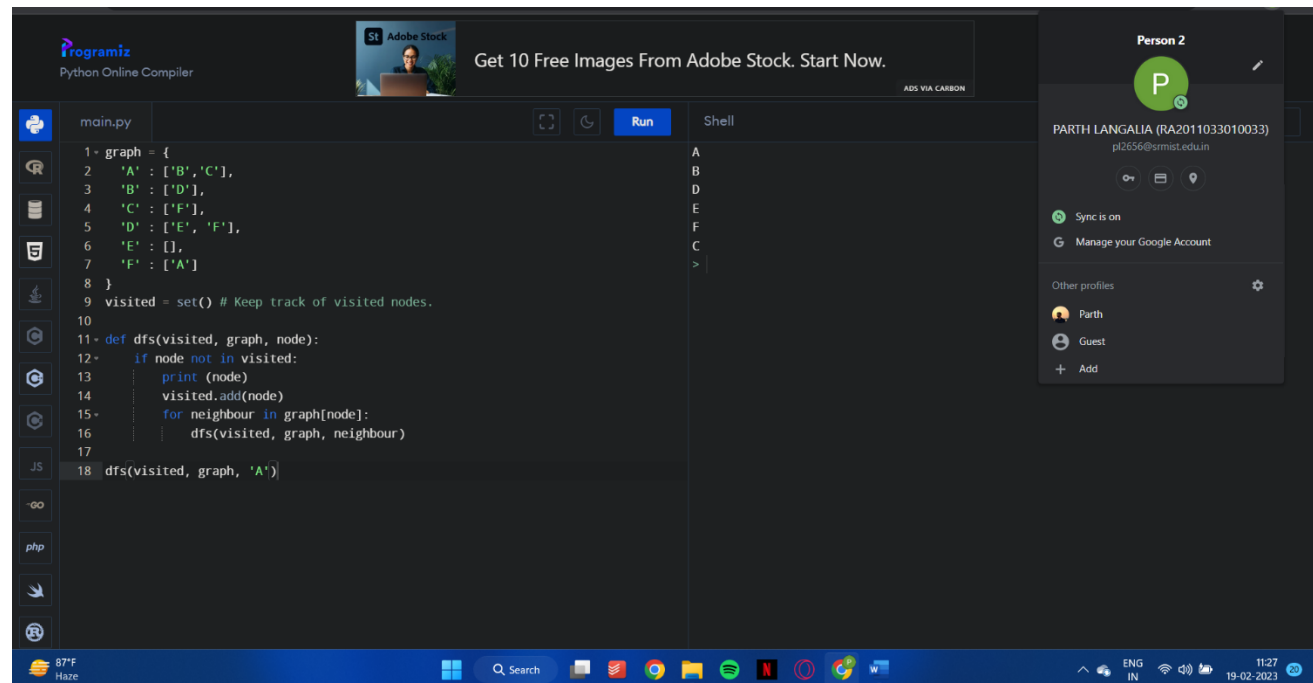
Parth  
Guest  
+ Add

87°F Haze

Search

ENG IN 11:25 19-02-2023

### DFS :



Programiz Python Online Compiler

Get 10 Free Images From Adobe Stock. Start Now.

main.py

```
1- graph = {
2  'A' : ['B','C'],
3  'B' : ['D'],
4  'C' : ['F'],
5  'D' : ['E', 'F'],
6  'E' : [],
7  'F' : ['A']
8 }
9
10 visited = set() # Keep track of visited nodes.
11
12 def dfs(visited, graph, node):
13     if node not in visited:
14         print(node)
15         visited.add(node)
16         for neighbour in graph[node]:
17             dfs(visited, graph, neighbour)
18
19 dfs(visited, graph, 'A')
```

Shell

```
A
B
D
E
F
C
>
```

Person 2

PARTH LANGALIA (RA2011033010033)  
pt2656@srmstedu.in

Sync is on  
Manage your Google Account

Other profiles

Parth  
Guest  
+ Add

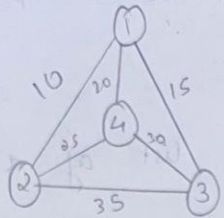
87°F Haze

Search

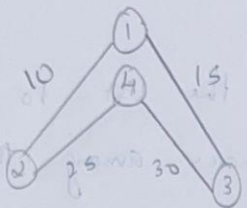
ENG IN 11:27 19-02-2023

## Manual Calculations :

Manual Calculations / Output :



← Graph



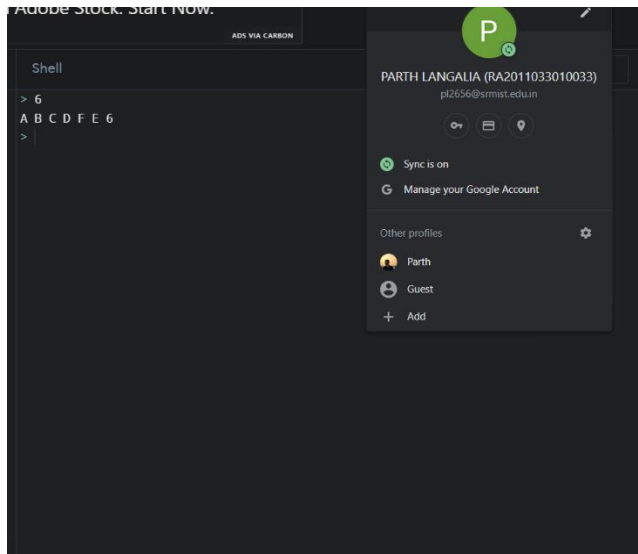
← Path / Tree

\* Input :  $\{20, 10, 15, 20\}$ ,  $\{10, 0, 35, 20\}$ ,  $\{15, 35, 0, 20\}$ ,  
 $\{20, 25, 30, 0\}$ ;

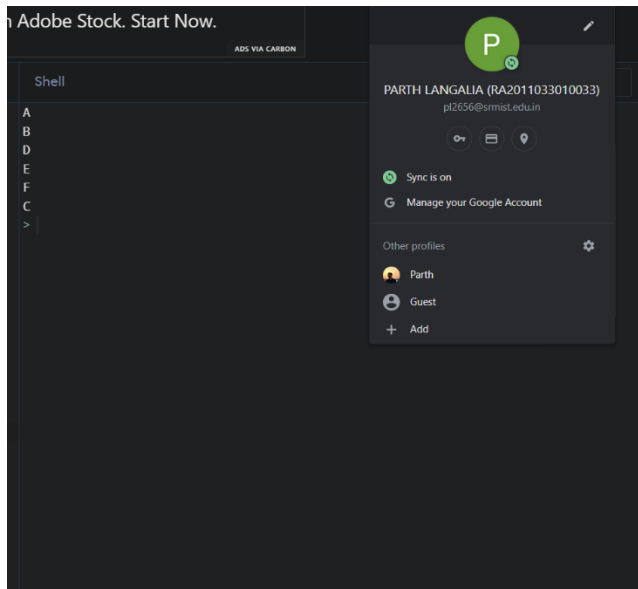
\* Output : 80

## **OUTPUT :**

BFS :



DFS :



## **RESULT :**

Successfully found the sum of nodes in a binary search tree between any given range (min, max) using both depth first search and breadth first search approach.