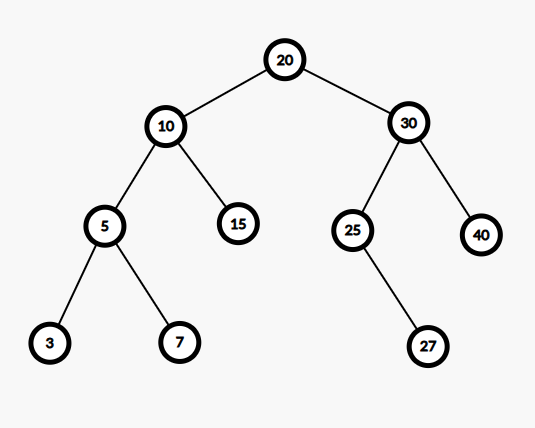
**Question:**

Given the root of a Binary Search Tree and two node values x and y, find the number of nodes in the shortest path between the given two nodes.

**Note:**

* If either node is not found in the BST, return -1.
* For the same node (x == y), return 1.
* Solve the problem using recursion.
* Can use as many helper functions you want.
* For python and java, can use array, ArrayList, reverse and pop / remove if needed.

**Tree:**



| **Sample Input** | **Sample Output** |
| --- | --- |
| distance\_in\_nodes(root, 5, 15) | 3  Explanation: path is 5 -> 10 -> 15 |
| distance\_in\_nodes(root, 3, 27) | 7 Explanation: path is 3 -> 5 -> 10 -> 20 -> 30 -> 25 -> 27 |
| distance\_in\_nodes(root, 10, 10) | 1  Explanation: Same node — path has only one node 10 |
| distance\_in\_nodes(root, 7, 99) | -1  Explanation: Node not found |

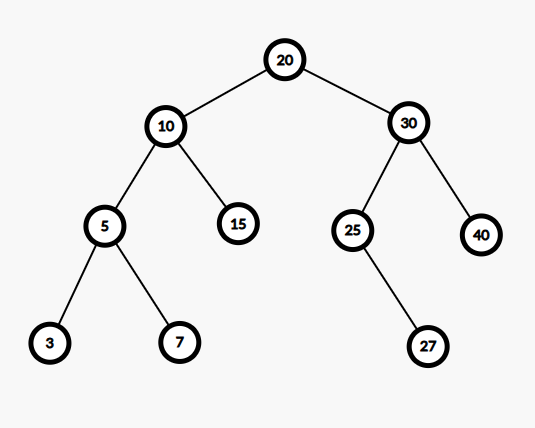
**Question:**

Given the root of a Binary Search Tree and two node values x and y, find the nodes in the shortest path between the given two nodes.

**Note:**

* If either node is not found in the BST, return “No Path Found”.
* For python, can use array, reverse and pop if needed.
* For java, can use ArrayList, reverse and remove if needed. Otherwise return as string.
* Solve the problem using recursion.
* Can use as many helper functions you want.

**Tree:**



| **Sample Input** | **Sample Output** |
| --- | --- |
| path\_between\_nodes(root, 5, 15) | [5, 10, 15] |
| path\_between\_nodes(root, 3, 27) | [3, 5, 10, 20, 30, 25, 27] |
| path\_between\_nodes(root, 10, 10) | [10] |
| path\_between\_nodes(root, 7, 99) | No Path Found |