```cpp
#include <bits/stdc++.h>
#include <fstream>
#include <iostream>
#include <sstream>
#include <string>
#include <vector>


using namespace std;
```

```cpp
struct TreeNode{
int val;
TreeNode *left;
TreeNode *right;
TreeNode() : val(0), left(nullptr), right(nullptr){}
TreeNode(int x) : val(x), left(nullptr), right(nullptr){}
TreeNode(int x, TreeNode *left, TreeNode * right) : val(x),
left(left), right(right){}
};
```

```cpp
/*
* Complete the 'BSTdistance' function below.
*
```

```
* The function is expected to return an INTEGER.
* The function accepts following parameters:
* 1. INTEGER_ARRAY values
* 2. INTEGER nodeA
* 3. INTEGER nodeB
*/
```

```
TreeNode* insertIntoBST(TreeNode *root, int val) {
if (!root) {
return new TreeNode(val);
}
if (val < root -> val) {
root -> left = insertIntoBST(root -> left, val);
}
else {
root -> right = insertIntoBST(root -> right, val);
}
return root;
}
```

```
TreeNode* LCA(TreeNode* root, int p, int q) {
if (!root || root -> val == p || root -> val == q) {
return root;
```

```
}
TreeNode* left = LCA(root -> left, p, q);
TreeNode* right = LCA(root -> right, p, q);
if (!left && !right) {
return NULL;
}
if (left && right) {
return root;
}
return !left ? right : left;
}
```

```
int dist(TreeNode* root, int target, int travel) {
if (!root) {
return -1;
}
if (root -> val == target) {
return travel;
}
int leftDist = dist(root -> left, target, travel + 1);
if (leftDist == -1) {
return dist(root -> right, target, travel + 1);
```

```
}
return leftDist;
}
```

```
int findDistance(TreeNode* root, int p, int q) {
TreeNode* lca = LCA(root, p, q);
return dist(lca, p, 0) + dist(lca, q, 0);
}
```

```
int BSTdistance(std::vector<int> values, int nodeA, int
nodeB)
{
TreeNode* root = nullptr;
for (int val : values) {
root = insertIntoBST(root, val);
}
return findDistance(root, nodeA, nodeB);
}
```