Our Solution(s) Run Code

Solution 1

```cpp
// Copyright © 2020 AlgoExpert, LLC. All rights reserved.

#include <vector>
using namespace std;

class BinaryTree {
public:
  int value;
  BinaryTree *left;
  BinaryTree *right;

  BinaryTree(int value);
  void insert(vector<int> values, int i = 0);
};

vector<int> findMaxSum(BinaryTree *tree);

// O(n) time | O(log(n)) space
int maxPathSum(BinaryTree tree) {
  vector<int> maxSumArray = findMaxSum(&tree);
  return maxSumArray[1];
}

vector<int> findMaxSum(BinaryTree *tree) {
  if (tree == NULL) {
    return vector<int>{0, 0};
  }

  vector<int> leftMaxSumArray = findMaxSum(tree->left);
  int leftMaxSumAsBranch = leftMaxSumArray[0];
  int leftMaxPathSum = leftMaxSumArray[1];

  vector<int> rightMaxSumArray = findMaxSum(tree->right);
```

Your Solutions Run Code

Solution 1 Solution 2 Solution 3

```cpp
#include <vector>
using namespace std;

class BinaryTree {
public:
  int value;
  BinaryTree *left;
  BinaryTree *right;

  BinaryTree(int value);
  void insert(vector<int> values, int i = 0);
};

int maxPathSum(BinaryTree tree) {
  // Write your code here.
  return -1;
}
```

Our Tests

Custom Output Submit Code

```
void BinaryTree::insert(vector<int> values, int i) {
  if (i >= values.size()) {
    return;
  }
  queue<BinaryTree *> queue;
  queue.push_back(this);
  while (queue.size() > 0) {
    BinaryTree *current = queue.front();
    queue.pop_front();
    if (current->left == NULL) {
      current->left = new BinaryTree(values[i]);
      break;
    }
    queue.push_back(current->left);
```

Run or submit code when you're ready.