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);
  int rightMaxSumAsBranch = rightMaxSumArray[0];
  int rightMaxPathSum = rightMaxSumArray[1];

  int maxChildSumAsBranch = max(leftMaxSumAsBranch, rightMaxSumAsBranch);
  int maxSumAsBranch = max(maxChildSumAsBranch + tree->value, tree->value);
  int maxSumAsRootNode = max(
      leftMaxSumAsBranch + tree->value + rightMaxSumAsBranch, maxSumAsBranch);
  int maxPathSum = max(leftMaxPathSum, max(rightMaxPathSum, maxSumAsRootNode));

  return vector<int>{maxSumAsBranch, maxPathSum};
}

```