

Our Solution(s)

Run Code

Your Solutions

Run Code

Solution 1

```
1 // Copyright © 2020 AlgoExpert, LLC. All rights reserved.
2
3 using System.Collections.Generic;
4
5 public class Program {
6     public class BinaryTree {
7         public int value;
8         public BinaryTree left;
9         public BinaryTree right;
10
11         public BinaryTree(int value) {
12             this.value = value;
13             this.left = null;
14             this.right = null;
15         }
16     }
17
18     // O(n) time | O(n) space - where n is the number of nodes in the Bi
19     public static List<int> BranchSums(BinaryTree root) {
20         List<int> sums = new List<int>();
21         calculateBranchSums(root, 0, sums);
22         return sums;
23     }
24
25     public static void calculateBranchSums(BinaryTree node, int runningS
26         if (node == null) return;
27
28         int newRunningSum = runningSum + node.value;
29         if (node.left == null && node.right == null) {
30             sums.Add(newRunningSum);
31             return;
32         }
33     }
```

Solution 1

Solution 2

Solution 3

```
1 using System.Collections.Generic;
2
3 public class Program {
4     // This is the class of the input root. Do not edit it.
5     public class BinaryTree {
6         public int value;
7         public BinaryTree left;
8         public BinaryTree right;
9
10         public BinaryTree(int value) {
11             this.value = value;
12             this.left = null;
13             this.right = null;
14         }
15     }
16
17     public static List<int> BranchSums(BinaryTree root) {
18         // Write your code here.
19         return null;
20     }
21 }
22
```

Our Tests

Custom Output

Submit Code

```
1 using System.Collections.Generic;
2 using System.Collections.Generic;
3 public class Program {
4     public class BinaryTreeNode {
5         public int value;
6         public BinaryTreeNode left;
7         public BinaryTreeNode right;
8     }
9 }
```

```
1 // Write your code here.
2 return null;
```

```

18
19
20     def __init__(self, input_size, hidden_size):
21         self.input_size = input_size
22         self.hidden_size = hidden_size
23
24     def forward(self, input):
25         # Compute the hidden state
26         hidden = torch.tanh(torch.matmul(input, self.W) + self.b)
27
28         # Compute the output
29         output = torch.matmul(hidden, self.V) + self.c
30
31         return output
32
33     def loss(self, input, target):
34         # Compute the loss
35         output = self.forward(input)
36         loss = torch.nn.functional.binary_cross_entropy(output, target)
37
38         return loss
39
40     def train(self, data_loader):
41         # Training loop
42         for epoch in range(1, 100):
43             for input, target in data_loader:
44                 loss = self.loss(input, target)
45                 self.optimizer.zero_grad_()
46                 loss.backward()
47                 self.optimizer.step()

```

Run or submit code when you're ready.