

Our Solution(s)

Run Code

Your Solutions

Run Code

Solution 1

```
1 // Copyright © 2020 AlgoExpert, LLC. All rights reserved.
2
3 using System;
4 using System.Collections.Generic;
5
6 public class Program {
7     // O(nc) time | O(nc) space
8     public static List<List<int>> KnapsackProblem(int[,] items, int capacity) {
9         int[,] knapsackValues = new int[items.GetLength(0) + 1, capacity + 1];
10         for (int i = 1; i < items.GetLength(0) + 1; i++) {
11             int currentWeight = items[i - 1, 1];
12             int currentValue = items[i - 1, 0];
13             for (int c = 0; c < capacity + 1; c++) {
14                 if (currentWeight > c) {
15                     knapsackValues[i, c] = knapsackValues[i - 1, c];
16                 } else {
17                     knapsackValues[i, c] = Math.Max(knapsackValues[i - 1, c],
18                         knapsackValues[i - 1,
19                             c -
20                             currentWeight] +
21                             currentValue);
22                 }
23             }
24         }
25         return getKnapsackItems(knapsackValues, items,
26             knapsackValues[items.GetLength(0), capacity]);
27     }
28
29     public static List<List<int>> getKnapsackItems(int[,] knapsackValues,
30         int weight) {
31         List<List<int>> sequence = new List<List<int>> >();
32         List<int> totalWeight = new List<int>();
33         totalWeight.Add(weight);
```

Our Tests

```
1 using System;
2 using System.Collections.Generic;
3 public class Program {
4     // O(nc) time | O(nc) space
5     public static List<List<int>> KnapsackProblem(int[,] items, int capacity) {
6         int[,] knapsackValues = new int[items.GetLength(0) + 1, capacity + 1];
7         for (int i = 1; i < items.GetLength(0) + 1; i++) {
8             int currentWeight = items[i - 1, 1];
9             int currentValue = items[i - 1, 0];
10             for (int c = 0; c < capacity + 1; c++) {
11                 if (currentWeight > c) {
12                     knapsackValues[i, c] = knapsackValues[i - 1, c];
13                 } else {
14                     knapsackValues[i, c] = Math.Max(knapsackValues[i - 1, c],
15                         knapsackValues[i - 1,
16                             c -
17                             currentWeight] +
18                             currentValue);
19                 }
20             }
21         }
22         return getKnapsackItems(knapsackValues, items,
23             knapsackValues[items.GetLength(0), capacity]);
24     }
25
26     public static List<List<int>> getKnapsackItems(int[,] knapsackValues,
27         int weight) {
28         List<List<int>> sequence = new List<List<int>> >();
29         List<int> totalWeight = new List<int>();
30         totalWeight.Add(weight);
```

Solution 1 Solution 2 Solution 3

```
1 using System.Collections.Generic;
2
3 public class Program {
4     public static List<List<int>> KnapsackProblem(int[,] items, int capacity) {
5         // Write your code here.
6         // Replace the code below.
7         List<int> totalValue = new List<int> {
8             10
9         };
10        List<int> finalItems = new List<int> {
11            1, 2
12        };
13        var result = new List<List<int>> >();
14        result.Add(totalValue);
15        result.Add(finalItems);
16        return result;
17    }
18 }
19
```

Custom Output

Submit Code

```
1 using System;
2 using System.Collections.Generic;
3 public class Program {
4     public static List<List<int>> KnapsackProblem(int[,] items, int capacity) {
5         // Write your code here.
6         // Replace the code below.
7         List<int> totalValue = new List<int> {
8             10
9         };
10        List<int> finalItems = new List<int> {
11            1, 2
12        };
13        var result = new List<List<int>> >();
14        result.Add(totalValue);
15        result.Add(finalItems);
16        return result;
17    }
18 }
19
```

