

Our Solution(s)

Run Code

Your Solutions

Run Code

Solution 1

```
1 // Copyright © 2020 AlgoExpert, LLC. All rights reserved.
2
3 class Program {
4     // O(nc) time | O(nc) space
5     func knapsackProblem(_ items: [[Int]], _ capacity: Int) -> (Int,
6         var knapsackValues = [[Int]]())
7
8     for _ in 0 ..< items.count + 1 {
9         let row = Array(repeating: 0, count: capacity + 1)
10        knapsackValues.append(row)
11    }
12
13    for currentItemIndex in 1 ..< items.count + 1 {
14        let currentValue = items[currentItemIndex - 1][0]
15        let currentWeight = items[currentItemIndex - 1][1]
16
17        for currentCapacity in 0 ..< capacity + 1 {
18            if currentWeight <= currentCapacity {
19                knapsackValues[currentItemIndex][currentCapacity]
20            } else {
21                knapsackValues[currentItemIndex][currentCapacity]
22            }
23        }
24    }
25
26    return (knapsackValues[items.count][capacity], getKnapsackItem
27 }
28
29 func getKnapsackItems(_ items: [[Int]], _ knapsackValues: [[Int]])
30     var sequence = [Int]()
31
32     var currentItemIndex = knapsackValues.count - 1
33     var currentCapacity = knapsackValues[0].count - 1
```

Solution 1   Solution 2   Solution 3

```
1 class Program {
2     func knapsackProblem(_ items: [[Int]], _ capacity: Int) -> (Int, [
3         // Write your code here.
4         // Replace the return blow.
5     return (
6         10, // total value
7         [1, 2] // item indices
8     )
9 }
10 }
11 }
```

Our Tests

Custom Output

Submit Code

```
1 class Program {
2     func knapsackProblem(_ items: [[Int]], _ capacity: Int) -> (Int, [
3         // Write your code here.
4         // Replace the return blow.
5     return (
6         10, // total value
7         [1, 2] // item indices
8     )
9 }
10 }
11 }
```

```

10
11
12 #define TEST_CASE 271 // 271 #pragma omp test
13
14 int expected = 1000; // 10, 10, 10, 10
15 int output = program_output(1000, 1000, 1000, 10)
16
17 // assert(expected == output)
18
19
20 #define TEST_CASE 271 // 271 #pragma omp test
21
22 int expected = 1000; // 10, 10, 10, 10
23 int output = program_output(1000, 1000, 1000, 10)
24
25 // assert(expected == output)
26
27
28 #define TEST_CASE 271 // 271 #pragma omp test
29
30 int expected = 1000; // 10, 10, 10, 10
31 int output = program_output(10, 10, 100, 100, 10)
32
33 // assert(expected == output)

```

Run or submit code when you're ready.