Our Solution(s) Run Code

Solution 1

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

class Program {
    // O(nc) time | O(nc) space
    func knapsackProblem(_ items: [[Int]], _ capacity: Int) -> (Int,
        var knapsackValues = [[Int]]()

        for _ in 0 ..< items.count + 1 {
            let row = Array(repeating: 0, count: capacity + 1)
            knapsackValues.append(row)
        }

        for currentItemIndex in 1 ..< items.count + 1 {
            let currentValue = items[currentItemIndex - 1][0]
            let currentWeight = items[currentItemIndex - 1][1]

            for currentCapacity in 0 ..< capacity + 1 {
                if currentWeight <= currentCapacity {
                    knapsackValues[currentItemIndex][currentCapacity]
                } else {
                    knapsackValues[currentItemIndex][currentCapacity]
                }
            }
        }

        return (knapsackValues[items.count][capacity], getKnapsackItem
    }

    func getKnapsackItems(_ items: [[Int]], _ knapsackValues: [[Int]])
        var sequence = [Int]()

        var currentItemIndex = knapsackValues.count - 1
        var currentCapacity = knapsackValues[0].count - 1
```

Your Solutions Run Code

Solution 1 Solution 2 Solution 3

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

```

Our Tests

Custom Output Submit Code

Run or submit code when you're ready.