Our Solution(s) Run Code

Solution 1

```python
# Copyright © 2020 AlgoExpert, LLC. All rights reserved.

# O(nc) time | O(nc) space
def knapsackProblem(items, capacity):
    knapsackValues = [[0 for x in range(0, capacity + 1)] for y in ra
    for i in range(1, len(items) + 1):
        currentWeight = items[i - 1][1]
        currentValue = items[i - 1][0]
        for c in range(0, capacity + 1):
            if currentWeight > c:
                knapsackValues[i][c] = knapsackValues[i - 1][c]
            else:
                knapsackValues[i][c] = max(
                    knapsackValues[i - 1][c], knapsackValues[i - 1][c
                )
    return [knapsackValues[-1][-1], getKnapsackItems(knapsackValues,


def getKnapsackItems(knapsackValues, items):
    sequence = []
    i = len(knapsackValues) - 1
    c = len(knapsackValues[0]) - 1
    while i > 0:
        if knapsackValues[i][c] == knapsackValues[i - 1][c]:
            i -= 1
        else:
            sequence.append(i - 1)
            c -= items[i - 1][1]
            i -= 1
        if c == 0:
            break
    return list(reversed(sequence))

```

Your Solutions Run Code

Solution 1 Solution 2 Solution 3

```python
def knapsackProblem(items, capacity):
    # Write your code here.
    # return [
    #   10, // total value
    #   [1, 2], // item indices
    # ]
    pass

```

Our Tests

Custom Output Submit Code

Run or submit code when you're ready.