University of the Pacific

Project Proposal

CUTTING BLOCK OF GOLD

Bang Ngoc Pham

COMP 257 - Advanced Algorithms

28 October, 2022

I. Description

The problem is to decide how to cut a block of gold into smaller pieces in order to maximize the profit when selling the gold, knowing that pieces with different weights cost differently. For this problem, I am provided with the weights of some pieces and their corresponding costs. If the weight of a certain piece is not given, I will not get any money for selling that piece. To solve this problem, I will develop three algorithms which are brute force, greedy and dynamic programming in Python. These algorithms take 2 inputs and return 2 outputs.

The inputs include:

1. block_weight: the entire weight of the given block of gold.
2. weight_cost: a dictionary whose keys are weights and the value of each key is the cost of a piece of gold of that weight. Missing weights (weights are not displayed in the dictionary) means their costs are equal to 0.

The outputs include:

1. max_profit: the maximum profit when selling the given block of gold after cutting.
2. pieces_list: the list of weights of pieces in order to achieve max_profit.

II. Pseudocode

1. Brute force algorithm

```
brute_force(block_weight, weight_cost):
      if block_weight <= 0:
            return 0
      max_profit = 0
      pieces_list = empty list
```

```
    for i from 1 to block_weight:
        if i is a key of weight_cost:
            cost = value of i in weight_cost
        else:
            cost = 0
        profit, pieces = brute_force(block_weight - i, weight_cost)
        max_profit = max(max_profit, cost + profit)
        if max_profit == the latter:
            append i to pieces
            pieces_list = pieces
    return max_profit, pieces_list
```

2. Greedy algorithm

```
greedy(block_weight, weight_cost):
    max_profit = 0
    pieces_list = empty list
    weights = list of weight_cost keys
    while block_weight >= min(weights):
        weight = the weight that is <= block_weight && have maximum cost
        cost = value of weight in weight_cost
        max_profit += cost
        block_weight -= weight
        append weight to pieces_list
    return max_profit, pieces_list
```

3. Dynamic programing

```
dynamic(block_weight, weight_cost):
    max_profit = list of 0's with len = block_weight + 1
    pieces_list = list of empty lists with len = block_weight + 1
    for i from 2 to block_weight + 1:
        max_val = 0
        pieces = empty list
        for j from 1 to i:
            max_val = max(max_val, weight_cost[j] + max_profit[i-j])
            if max_val == the latter:
                append j to pieces_list[i-j]
                pieces = pieces_list[i-j]
        max_profit[i] = max_val
        pieces_list[i] = pieces
    return max_profit[block_weight], pieces_list[block_weight]
```

III. Big-O Time Complexity

Assume n = block_weight, m = |weight_cost|

1. Brute Force algorithm

Since we need to generate all possible ways to cut the given block of weight into smaller pieces, and the weights of those pieces range from 1 to n

=> Runtime =$O(2^n)$

2. Greedy algorithm

The while loop runs in n times.

Finding the weight and cost such that that weight is equal or smaller than block_weight and has maximum cost takes m times.

=> Runtime = O(n*m)

3. **Dynamic programming**

   The runtime mostly depends on the nested for-loop. The outer and inner loops both take n times.

   => Runtime = O(n*m)