University of the Pacific

Final Project

CUTTING BLOCK OF GOLD

Bang Ngoc Pham

COMP 257 - Advanced Algorithms

16 December, 2022

1. **Corrected project proposal:**

   https://github.com/bang128/COMP257-Final-Project/blob/fea87689a262124eec790370f21a9317f4178674/COMP%20257%20-%20Project%20Proposal.pdf

2. **Github username:** bang128

3. **Github repository with code and test files:**

   https://github.com/bang128/COMP257-Final-Project

   Run: `python3 main_test.py`

4. The output of code

```
Test case 1:
Size = 10
        Brute_Force:
                Result: (26, [1, 3, 3, 3])
                Elapsed time: 0.0010020732879638672
        Dynamic:
                Result: (26, [1, 3, 3, 3])
                Elapsed time: 5.7697296142578125e-05
        Greedy:
                Result: (20, [5, 5])
                Elapsed time: 1.5020370483398438e-05


Test case 2:
Size = 13
        Brute_Force:
                Result: (26, [1, 2, 2, 2, 2, 2, 2])
                Elapsed time: 0.0073299407958984375
        Dynamic:
                Result: (26, [1, 2, 2, 2, 2, 2, 2])
                Elapsed time: 7.200241088867188e-05
        Greedy:
                Result: (20, [13])
                Elapsed time: 1.1920928955078125e-05


Test case 3:
Size = 15
        Brute_Force:
                Result: (30, [1, 7, 7])
                Elapsed time: 0.020627975463867188
        Dynamic:
                Result: (30, [1, 7, 7])
                Elapsed time: 8.893013000488281e-05
        Greedy:
                Result: (24, [15])
                Elapsed time: 9.059906005859375e-06


Test case 4:
Size = 18
        Brute_Force:
                Result: (24, [6, 12])
                Elapsed time: 0.1548159122467041
        Dynamic:
                Result: (24, [6, 12])
                Elapsed time: 9.012222900390625e-05
        Greedy:
                Result: (24, [10, 6, 2])
                Elapsed time: 1.1920928955078125e-05


Test case 5:
Size = 20
        Brute_Force:
                Result: (78, [3, 17])
                Elapsed time: 0.6264371871948242
        Dynamic:
                Result: (78, [3, 17])
                Elapsed time: 0.00010275840759277344
        Greedy:
                Result: (78, [17, 3])
```

5. **Explanation of why the greedy algorithm has a different solution than the other algorithms on some test cases**

In most cases, the greedy algorithm results in different solutions than brute force and dynamic algorithms. This is because the greedy algorithm always takes the optimal solution at each step, meaning that it always cuts the block into a piece with the highest cost at a certain moment. However, the optimal cost at a certain moment does not guarantee that the final solution is optimal. Fox example, test 1 is:

block_weight = 10

weight_cost = {1: 2

3: 8

5: 10

10: 2}

As shown in the above screenshot, the greedy algorithm cuts the block into 2 pieces of weight 5 because a piece of weight 5 costs the highest (it costs 10). So the total cost resulting from the greedy algorithm is 20. However, the brute force and dynamic algorithms prove that there is a solution that costs 26 although each individual piece chosen by these two algorithms does not have the highest cost. Thus, the actual optimal solution is (26, [1, 3, 3, 3])

6. **Explanation of limiting the test size.**

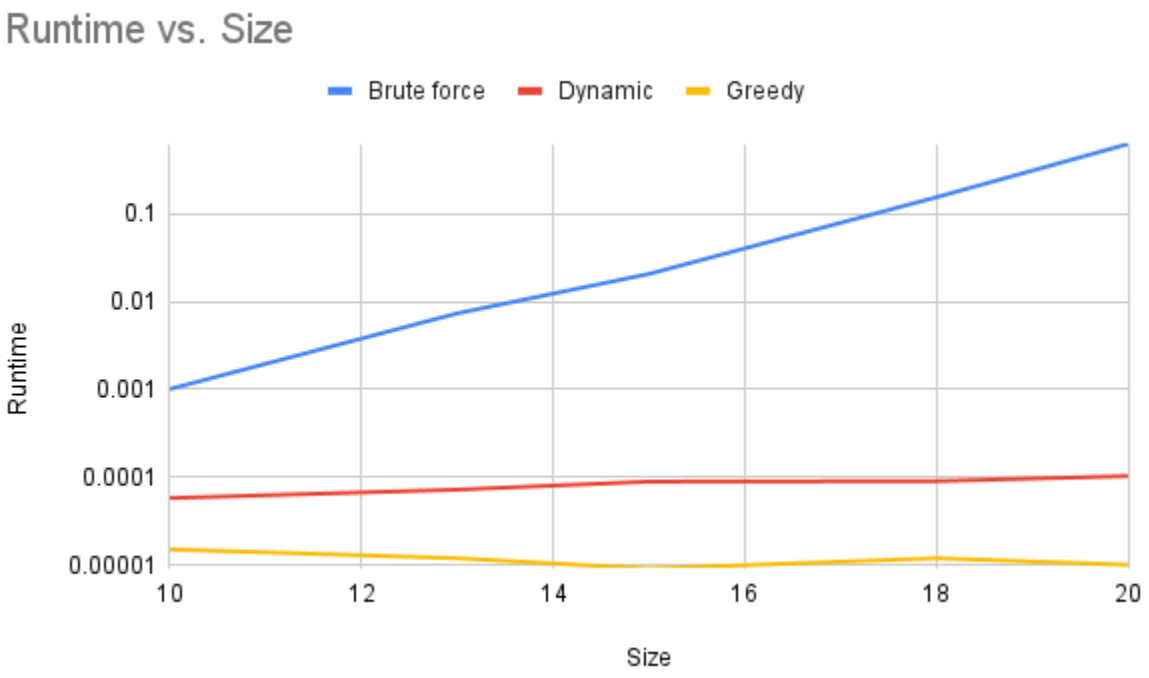
In all test cases, the block weight is never larger than 20 because the brute force algorithm takes very long to cut blocks of weight larger than 20. As analyzed in the proposal, the brute force algorithm takes $O(2^n)$ in which n is the block_weight. Thus the

computer can hardly handle this exponential runtime if the block_weight (or n) is too large.

7. **Plot of the amount of time for each test case.**

Size of test case = block_weight

Size	Brute force	Dynamic	Greedy
10	0.0010020732879638672	5.7697296142578125e-05	1.5020370483398438e-05
13	0.0073299407958984375	7.200241088867188e-05	1.1920928955078125e-05
15	0.020627975463867188	8.893013000488281e-05	9.059906005859375e-06
18	0.1548159122467041	9.012222290039062e-05	1.1920928955078125e-05
20	0.6264371871948242	1.0275840759277344e-05	1.0013580322265625e-05



8. **Recommended algorithm.**

I recommend using dynamic programming algorithm. Dynamic programming and greedy algorithms solve each test case much significantly faster than brute force algorithm.

Although greedy algorithm is faster than dynamic programming, greedy algorithm cannot guarantee an optimal solution. On the other hand, dynamic programming can both satisfy a fast runtime and an optimal solution.