Problem Set 1

MIT 6.0002
Introduction to Computational Thinking and Data Science
as Taught in Fall 2016

John L. Jones IV

December 13, 2019

Problem A.1

What were your results from `compare_cow_transport_algorithms`? Which algorithm runs faster? Why?

Using `ps1_cow_data.txt` the results were:

```
greedy_cow_transport:
length = 6 trips
time = .000145 seconds

brute_force_cow_transport:
length = 5 trips
time = 0.48294 seconds
```

The algorithm `greedy_cow_transport` does not iterate through every possible combination of trips like `brute_force_cow_transport`. In my implementation of `greedy_cow_transport`, first the input dictionary of cows is copied to a list of cow names `sorted` from largest to smallest weight. The python `sorted` function utilizes a Timsort which is $\mathcal{O}(n \log n)$. Then `greedy_cow_transport` removes any cow which is larger than `limit`, an $\mathcal{O}(n)$ operation. The sorted list is then utilize to select the cows which can fit on the ship. Starting at the big end of the list, iterate and select cows that can fit onto the ship without exceeding the payload `limit`. Since the list has been sorted, this is an $\mathcal{O}(n \log n)$ operation. The python `sorted` function's Timsort and selecting cows operation dominate the run time, making `greedy_cow_transport` $\mathcal{O}(n \log n)$. In comparison, `brute_force_cow_transport` must first create all permutations of the possible trips, $\mathcal{O}(n^2)$. Then evaluate each of these trips, $\mathcal{O}(n^2)$. This emphasizes Professor John Guttag's quote from Lecture 1, "many optimization problems are inherently exponential. What that means is there is no algorithm that provides an exact solution to this problem whose worst case running time is not exponential in the number of items."

Problem A.2

Does the greedy algorithm return the optimal solution? Why/why not?

No, the greedy algorithm *does not* return the optimal solution. The nature of this "knapsack" problem is $\mathcal{O}(n^2)$. However, a reasonable solution can be solve in $\mathcal{O}(n \log n)$ with `greedy_cow_transport`. With `ps1_cow_data.txt` `greedy_cow_transport` returns a solution 1000 times faster than `brute_force_cow_transport`.

Problem A.3

Does the brute force algorithm return the optimal solution? Why/why not?

Yes, the brute force algorithm does return the optimal solution. All possible solutions are found then evaluated. The optimal solution is guaranteed to be produced and returned. However, this comes at a great cost to run-time speed. With `ps1_cow_data.txt` `brute_force_cow_transport` produces a solution 1000 times slower than the greedy algorithm.

Problem B.1

Explain why it would be difficult to use a brute force algorithm to solve this problem if there were 30 different egg weights. You do not need to implement a brute force algorithm in order to answer this.

Brute force requires all possible solutions to be solved then evaluated. The width of the search tree grows linearly with the number of egg weights. The number of nodes grows exponentially with number of egg weights. Therefore, the computational complexity grows exponentially with the number of egg weights. Observe: 1. This problem can be separated into smaller similar sub-problems therefore a recursive solution is possible. 2. It is possible to have re-occurring sub-problems. Since these two conditions are true, a dynamic programming approach, which uses a memo, allows the optimal solution to be found in much less run time than a typical brute-force approach. The same recursive solution without a memo is so slow, I did not have the patience to see if it worked on $n = 99$.

Problem B.2

If you were to implement a greedy algorithm for finding the minimum number of eggs needed, what would the objective function be? What would the constraints be? What strategy would your greedy algorithm follow to pick which eggs to take? You do not need to implement a greedy algorithm in order to answer this.

Always take the largest egg available, if and only if the largest egg does not exceed the available weight.

Problem B.3

Will a greedy algorithm always return the optimal solution to this problem? Explain why it is optimal or give an example of when it will not return the optimal solution.

No the greedy algorithm will not always return the optimal solution. For example, if the egg weights are (1,5,7) and the target weight is 10. The optimal solution is 2 (2 * 5 = 10). The greedy solution would return 4 (1 * 7 + 3 * 1 = 10).