Homework 4: Dynamic Programming

CS 4040/5040

Due: Wednesday, November 15th at 5:00 pm

This assignment will include programming work as well as a project report. Make sure you include all files in your submission. Answer all questions from the prompt in your report.

1. Consider the Rod Cutting algorithm discussed in class. This algorithm makes an important assumption: Every length < n has a price associated with it. This assumption may not always be true. A company could decide not to stock certain sizes (when was the last time you saw an 11 ft board for sale?) and could start off with a rod much longer than the longest length we sell.
    a. Adapt the rod-cutting algorithm to be the "Sparse Price Rod Cutting Algorithm" which would allow your algorithm to work with a list of prices that is not continuous (such as the example below).

Length	Price
1	1
2	5
4	9
6	17
8	20
10	30
12	36
16	48

    b. Adapt the "Sparse Price Rod Cutting Algorithm" to also work with initial rod lengths that are longer than the longest length sold.
    c. In your report, describe how you made changes to the algorithm to allow this, and any impacts on space and time complexity that may have (you do not need formal proofs, just a justification of any changes)
    d. In your code, time the length it takes for one call to your algorithm to complete. Then run the following experiments, and produce a graph comparing the run times for the adapted "Sparse Price Rod Cutting Algorithm" for each set of inputs. Use the price table provided in 1a and lengths provided below for n.

        Lengths = 4, 8, 16, 32, 64, 128, 256, 512, 1024, 2048, 4096

        In your report, include the graph as well as analysis if these run times match your expectations.
2. Consider the Coin Change problem and its dynamic programming solution discussed in class. Implement this algorithm. This algorithm gives us the number of unique combinations we can make with the currency available to us to create the specified amount in change. It does not provide the actual combinations. Create an algorithm that can create the unique combinations of coins that provide that amount of change (i.e. reconstruct the solution). You will need to adapt the Coin Change Algorithm to save more information about the solution. What is the time complexity of reconstructing the solutions? You adapted the coin change algorithm, how will these changes affect the time and space complexity of the algorithm?

a. Run the following experiments on the original coin change algorithm. In your code, collect the run time for each experiment, and produce a graph of these times in your report. Use the following list of US Currency amounts for coins = [1, 5, 10, 25, 50, 100, 200, 500, 1000, 2000]. Run experiments for the following amounts: [10, 50, 100, 500, 1000, 1500, 2000, 3000, 5000]
b. The wizarding world uses Knuts, Sickles, and Galleons as their currency. There are 29 Knuts to a Sickle, and 17 sickles to a Galleon. To represent this as a list of coin values, we would say coins = [1, 29, 493]. Repeat the experiments in 2a, but using wizard currency instead of US Currency.
c. Repeat the experiments in 2a and 2b, but instead of the original coin change algorithm, use the one you adapted to be able to reconstruct the solutions. Run the experiments for these amounts, not the ones provided in 2a and 2b: [10, 25, 50, 100]
d. Compare and analyze these results in your report.

CS 5040 Students ONLY:

3. Consider our Rod Cutting Algorithm. It has a loop that determines the first cut to make. A rod of length 4 can be cut into (1, 3) and (3, 1), but the revenue values for those would be equivalent. Our algorithm currently spends time considering both combinations. Modify the sparse rod cutting algorithm we created in problem 1 to no longer consider those equivalent cuts. Perform the experiments in 1d again with this new modified algorithm. In your report describe the changes you made, what impacts that will have on time complexity, and then present and analyze the results of your experiment.
4. Consider our coin change algorithm. We can safely assume that any useful currency value would have the smallest unit be 1. Modify the coin change algorithm to consider and use this assumption to perform fewer calculations. Repeat the experiments for 2a and 2b with your new modified coin change algorithm. In your report, explain the changes you made to the algorithm, what impacts this will have on time complexity, and then present and analyze the results of your experiment.

Program Requirements:

- Written in C++ and can compile and run on the school Linux machines
- Makefile is provided to compile the code
- TAs should be able to unzip/uncompress your submission and be able to type “make” and “make run” to compile and run your code.
- Code should be well documented and follow best practices for readability (good variable names, comments, broken into meaningful functions)
- Each algorithm/variation of an algorithm should exist in it’s own function
- Output to the screen should be well organized and labeled to make it clear which results are being presented
- Do NOT have your code create the graphs for the report in C++. Just have it output the data and use excel, matplotlib, or your favorite data visualization program to create the graphs and charts. Doing so in C++ is unnecessary for the scope of this class.

- No changes should have to be made to the code to re-run the experiments. We should not be changing variables in the code to see different results. Running the code should run all experiments.

Report Requirements:

- Report should be a PDF to ensure it can be opened on the grader's computer.
- Report should be well organized and formatted and be free of major grammatical errors.
- All graphs/charts should be well labeled and easy to read