Project 4: Exhaustive vs. Dynamic Programing

Spring 2022 CPSC 335 - Algorithm Engineering

Instructor: Dr. Sampson Akwafuo

Abstract

In this project, you will design, implement and analyze an exhaustive optimization and a dynamic programming algorithm for solving the same problem.

The Stock Purchase Maximization Problem

Given a fixed amount of purchasing power or financial resources, the problem seeks to maximize the number of stocks an investor may purchase. Future values of these stocks are not considered at the time of purchase.

This problem can be mathematically represented as:

Stock purchase maximation problem

input*:* a set of items numbered 1 to n, each having a number of stocks s_i and a value v_i, n= (s, v). M is the total available investment sum.
output: a list of items *P* from n, such that the total number of stocks in *P* is at most *M*, and the total value of all items in *P* is maximized

$$max \sum_{i=1}^{n} s_i v_i.$$

such that

$$P = \sum_{i=1}^{n} v_i \leq M$$

Assume you are given a set of stocks as an array of arrays. Each entry in the subarray represents a company, with the number of stocks available and the cumulative value of all available stocks. You are also given a specific sum of money to invest. Buying a fraction of any item is not allowed. Your task is to design an algorithm to determine the maximum number of stocks you are able to purchase. You solution should give the indices of the array(stocks) to be selected. The cumulative value of all selected stocks must be equal or less than the given sum. The problem is similar to a well-known algorithmic or optimization problem.

A sample is given below. It contains 4 items corresponding to company 0, 1, 2 and 3. Each company has $[x, y]$ attributes, where x = numbers of trading stocks and y = monetary value of the trading stocks.

Sample input
Stocks_and_values = [[1, 2], [3, 3], [5, 6], [6, 7]].
Amount = 10

Sample Output = [10, [1,3]
where 1 is the index of [4, 3] and 3 is the index of [6, 7]

The Exhaustive Search Approach (Part A)

An exhaustive search algorithm can be used to solve this problem. The approach evaluates the number of stocks and value of all possible subsets, then selects the subset with the highest value that is still under the available fund limit. It recomputes combination at each state. This approach provides an efficient, but expensive solution.

```
def stock_maximization (M, items):
    best = None
    for candidates in <stocks_combinations>(items):
        if verify_combinations(M, items, candidate):
            if best is None or total_value(candidate) > total_value(best)
                best = candidate
     return best
```

This approach was extensively discussed in class.

The Dynamic Programming Approach (Part B)

This problem can also be solved using the dynamic programming approach. It uses a top-down dynamic programming to handle the overlapping subproblems. Since there are two changing values, the resultant amount and the current index, a two-dimensional array can be used to store the results of all the solved sub-problems.

To calculate the maximum value obtainable with the selection of item i, a comparison of its cost is made with the total purchase capacity. If item i cost more than the available investment sum, it cannot be used. If a new candidate potentially increases the value of purchase and is less or equal

to the maximum available sum, it is selected. The approach evaluates the highest value of all possible subsets, then selects the subset with the highest value that is still under the weight limit.

To Do

1. Create PDF file and include your name(s) and email address(es). The first section of your report should contain instructions on how to run your program.
2. Study the sample input and output above.
3. Develop and implement the algorithm in part A using either Python or C++.
4. Develop a complete and clear pseudocode (for Part B) for an algorithm to solve this problem. Implement your algorithm in either Python or C++.
5. Your code should be able to solve the problem for any number of items (companies).
6. Mathematically analyze each pseudocode, your algorithms and observations. Compare the dynamic algorithm with the Exhaustive approach. Organize and present your report in the same PDF file.

Grading Rubric

The suggested grading rubric is below.

1. Understanding the problem and submitting a solution= 20 points

2. Part A. Exhaustive Implementation Algorithm = 35 points, divided as follows:
    a. Successful compilation = 15 points
    b. Produces accurate result = 12 points
    c. Complete and clear instruction on how to run your program= 3 points

3. Part B: Algorithm design and implementation = 45 points, divided as follows:
    a. Clear and complete Pseudocode = 10 points
    b. Successful compilation = 15 points
    c. Produces accurate result = 10 points
    d. Comparison of performance with exhaustive algorithm = 10 points

Ensure your submissions are your own works. Your submissions will be checked for similarities using a software.

Submitting your code

Submit your files to the Project 4 assignment on Canvas. It allows for multiple submissions. Upload your submissions as separate unzip files.

Deadline

The project deadline is **May 13, 11:59 pm** on Canvas.
Penalty for late submission is as stated in the syllabus. Projects submitted more than 48 hours after the deadline will not be accepted.