Project 2: Dynamic programming

COT 4400, Fall 2016

Due November 11, 2016

1 Overview

This project requires you to solve a dynamic programming problem and write a program that implements your solution.

You are only allowed to consult the class slides, the textbook, the TAs, and the professor. In particular, you are not allowed to use the Internet. This is a group project. The only people you can work with on this project are your group members. This policy is strictly enforced.

In addition to the group submission, you will also evaluate your teammates' cooperation and contribution. You will submit these evaluations to another assignment on Canvas, labelled "Project 2 (individual)."

2 Problem Description

For this project, you will solve a variant of the integer partition problem described in Section 8.5 of the required text. For this problem, you will be given an array of positive integers that you need to divide as evenly as possible without rearranging the array. Unlike the problem described in the textbook, you are given a limit on how large these partitions can be, rather than how many you can use. Specifically, you should divide the integers to minimize the "inequality score," calculated as the sum of the squares of the unused capacity; i.e.,

$$\sum_{i=1}^{k}(t - p_i)^2,$$

where t is the maximum size of a partition and $p_1, p_2, \ldots, p_k$ are the sums of the values in each of the k partitions.

3 Modeling the problem

Before you can write a program to solve this problem, you must first write a report describing how you will solve this problem. This report should address the following:

1. Describe in English how you can break down the larger problem into one or more smaller problem(s). This description should include how the solution to the larger problem is constructed from the subproblems.

2. What recurrence can you use to model the problem using dynamic programming?
3. What are the base cases of this recurrence?
4. Describe a pseudocode algorithm that uses memoization to compute the inequality score (i.e., the sum of the squares of the unused capacity) for the optimal (least unequal) solution.
5. Describe an iterative algorithm for the same purpose.
6. Describe how to modify and extend your iterative algorithm to identify the optimal partition.
7. Analyze the time and space complexity of your improved iterative algorithm.
8. Can the space complexity be improved relative to the memoized algorithm? Justify your answer.

For full credit, your pseudocode must be clear enough that any competent programmer will understand how your algorithm works and could implement your algorithm in their preferred programming language.

4 Coding your solutions

In addition to the report, you should implement the *improved iterative* version of your algorithm so that it can solve the max-capacity linear partitioning problem. Your code may be in C++ or Java, but it must compile and run on the C4 Linux Lab machines. Your code may be split into any number of files.

Also, your code will not need to handle incorrectly formatted or invalid input, nor problems that are unsolvable (containing an integer larger than the max capacity) or containing values greater than 10000.

4.1 Input format

Your program should read its input from the file `input.txt`, in the following format. This file may contains multiple instances. It begins with a single positive integer on a line by itself indicating the number of problem instances in the file, p. Each problem instance will specify value of n and t on a line, and the next line will contain n integers representing the array to partition. Problem instances will be separated by blank lines.

4.2 Output format

Your program should write its output to the file `output.txt`. You should write one line for each instance in the input file, and each line should contain k, the number of partitions in the optimal solution, followed by k integers representing the size of each partition, in order. All of these values should be separated by spaces.

4.3 Example

Consider problem instance where we are trying to partition the array below with a max capacity of $t = 10$:

5	2	3	4	10	5	3

The optimal solution in this case is to divide the array as follows, for a total inequality of 22:

5 2	3 4	10	5 3
$(10-7)^2$	$(10-7)^2$	$(10-10)^2$	$(10-8)^2$

In particular, while you could use all of the available capacity of the first partition by combining 5, 2, and 3, the second partition would only be able to hold 4, for an inequality score of $(10-4)^2 = 36$, which far exceeds the optimal solution on its own. The correct output for this problem would be:

```
4 2 2 1 2
```

5 Submission

Your submission for this project will be in two parts, the group submission and your individual peer evaluations.

The submission for your group should be a zip archive containing 1) your report (described in Section 3) as a PDF document, 2) your code (described in Section 4), and 3) a README file describing how to compile and run your code to Canvas. If your code requires more than a simple command to compile and run then you must also provide a Makefile and/or shell script. A simple command might be something like:

```
g++ *.cpp -o basket
```

Be aware that your project report and code will be checked for plagiarism.

6 Teamwork evaluation

The second part of your submission is a text file that includes 1) the names of all of your teammates (including yourself), 2) the team member responsibilities, 3) whether or not your teammates were cooperative, 4) a numeric rating indicating the proportional amount of effort each of you put into the project, and 5) other issues we should be aware of when evaluating your and your teammates' relative contribution. The numeric ratings must be integers that sum to 30.

7 Grading

Report	**45 points**
Dynamic programming model	20
Memoized pseudocode	5
Iterative pseudocode	5
Improved algorithm	5
Iterative complexity	10
Code	**25 points**
README file	5
Compiles and is correct	15
Good coding style	5
Teamwork	**30 points**
Follows the given format	5
Participation	25

Note that if your algorithm is inefficient, you may lose points for both your pseudocode and your submission. Also, in extreme cases, the teamwork portion of your grade may become negative. In particular, if you do not contribute to your group's solution at all, you can expect to receive a 0 overall.