Project 2: Dynamic programming

COT 4400, Summer 2017

Due July 16, 2017

1 Overview

For this project, you will develop an algorithm to optimize the behavior of a very impatient mover. Designing and implementing this solution will require you to model the problem using dynamic programming, then understand and implement your model.

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. These evaluations will form a major part of your grade on this project, so be sure that you respond to messages promptly, communicate effectively, and contribute substantially to your group's solution. Details for your team evaluations are in Section 7.2. You will submit the peer evaluations to another assignment on Canvas, labelled "Project 2 (individual)."

A word of warning: this project is team-based, but it is quite extensive and a nontrivial task. You are highly encouraged to start working on (and start asking questions about) this project early; teams who wait to start until the week before the due date may find themselves unable to complete it in time.

2 Problem Description

In the *impatient mover* problem, you have packed all of your belongings into boxes in your apartment, and your friend has come over to help you move. Your friend is going to move through your apartment, loading boxes onto the dolly, and take them out to your car; however, they are very impatient and will only make one pass through the apartment to pick things up. This means that they they will consider each of the boxes in order and decide whether or not to load them, but they will never go backwards to a box they have already seen. Moreover, heavier boxes must be picked up first; the mover won't load a heavier box on top of a lighter box. (Boxes of the same weight may be stacked on top of one another.)

Given the weights of all of the boxes (in the order that your friend will visit them), find the greatest number of boxes that your friend can pick up without backtracking or putting a heavier box on a lighter one.

3 Example

Suppose that you had arranged boxes of weight 29, 20, 30, 18, 35, 26, 28, 40, 8, and 39 lbs in your apartment (in that order). The impatient mover could pick up a maximum of 4 boxes: 29 lbs (bottom), 20 lbs, 18 lbs, and 8 lbs. Actually, this solution is unique; there is no other way for the mover to pick up 4 boxes in one pass through the apartment. Note, while the mover does pick up the leftmost box in this problem, that is not always the case—if there were a 15 lb box in front of all the others, the mover would do better to skip this and pick up the 29 lb box first.

4 Modelling the problem recursively

In order to write a dynamic programming algorithm for this problem, you will need to develop a recurrence to help solve the problem. I recommend that you develop a recurrence to solve the problem of the most boxes the mover can pick up if they pick box $\#i$ first, where i is some index in the range of 1–n. If you're able to solve this problem for all of the boxes, finding the best box to pick up first should be straightforward.

5 Project report

In your project report, you should include brief answers to 9 questions. Note that you must use dynamic programming to solve this problem; other solutions—for example, those involving sorting or binary search—will not receive credit.

1. How you can break down a problem instance when the mover is picking box $\#i$ first into one or more smaller instances? Your answer should include how the solution to the original problem is constructed from the subproblems.
2. What are the base cases of this recurrence?
3. What data structure would you use to recognize repeated problems? You should describe both the abstract data structure, as well as its implementation.
4. Give pseudocode for a memoized dynamic programming algorithm to find the greatest number of boxes that can be picked up when selecting box $\#i$ first.
5. What is the *worst-case* time complexity of your memoized algorithm?
6. Give pseudocode for an iterative algorithm to find the greatest number of boxes that can be picked up when selecting box $\#i$ first. This algorithm does not need to have a reduced space complexity relative to the memoized solution.
7. Can the space complexity of the iterative algorithm be improved relative to the memoized algorithm? Justify your answer.
8. Give pseudocode for an algorithm that solves the full problem—what is the maximum number of boxes that the mover can pick up in one pass? Your solution may involve iteration or recursion.
9. Give pseudocode for an algorithm that solves the problem of which boxes the mover picks up.

6 Coding your solutions

In addition to the report, you should implement a dynamic programming algorithm that can find the maximum number of boxes the impatient mover can pick up. Your code may be iterative or recursive, but it must be a dynamic programming algorithm. Also, you may code your solution in C++ or Java, but it must compile and run in a Linux environment. If you are using C++ and compiling your code cannot be accomplished by the command

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

you should include a Makefile that is capable of compiling the code via the `make` command.

If you choose to implement your code in Java, you should submit an executable jar file with your source. In either case, your source code may be split into any number of files.

Your code will not need to handle invalid input (e.g., negative weight boxes or problem instances with no boxes).

6.1 Input format

Your program should read its input from the file `input.txt`, in the following format. The first line of the file has a positive integer x specifying the number of problem instances. The rest of the file contains x pairs of lines, each separated by a blank line. The first line of each pair is the number of boxes that problem instance, and the second contains the weight of every box, in the order the mover will visit them. The weights will all be positive integers.

6.2 Output

Your program should write its output to the file `output.txt`. For each problem, your program should output the maximum number of boxes that the mover can pick up, one problem instance per line.

7 Submission

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

7.1 Group submission

The submission for your group should be a zip archive containing 1) your report (described in Section 5) as a PDF document, and 2) your code (described in Section 6). If your code requires more than a simple command to compile and run then you must also provide a Makefile and/or shell script. You should submit this zip archive to the "Project 2 (group)" assignment on Canvas.

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

7.2 Teamwork evaluation

The second part of your project grade will be determined by a peer evaluation. Your peer evaluation should be 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.

It's important that you be honest in your evaluation of your peers. In addition to letting your team members whether they do (or do not) need to work on their teamwork and communication skills, we will also evaluate your group submission in light of your team evaluations. For example, a team in which one member refused to contribute would be assessed differently than a team with three functioning members.

You should submit your peer evaluation to the "Project 2 (individual)" assignment on Canvas.

8 Grading

Report	**40 points**
Question 1	8 points
Questions 2 and 3	2 each
Questions 4 and 6	6 each
Questions 5 and 7–9	4 each
Code	**30 points**
Compiles	5
Uses correct input and output format	5
Computes correct answer	15
Good coding style	5
Teamwork	**30 points**

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 or greater than 30. In particular, if you do not contribute to your group's solution at all, you can expect to receive an overall grade of 0 on the project.