Project 2: Dynamic programming

COT 4400, Spring 2018

Due April 2, 2018

1 Overview

For this project, you will develop an algorithm to count the number of unique arrangements of pieces of candy. 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 5.2. You will submit the peer evaluations to another assignment on Canvas, labelled "Project 2 (individual)."

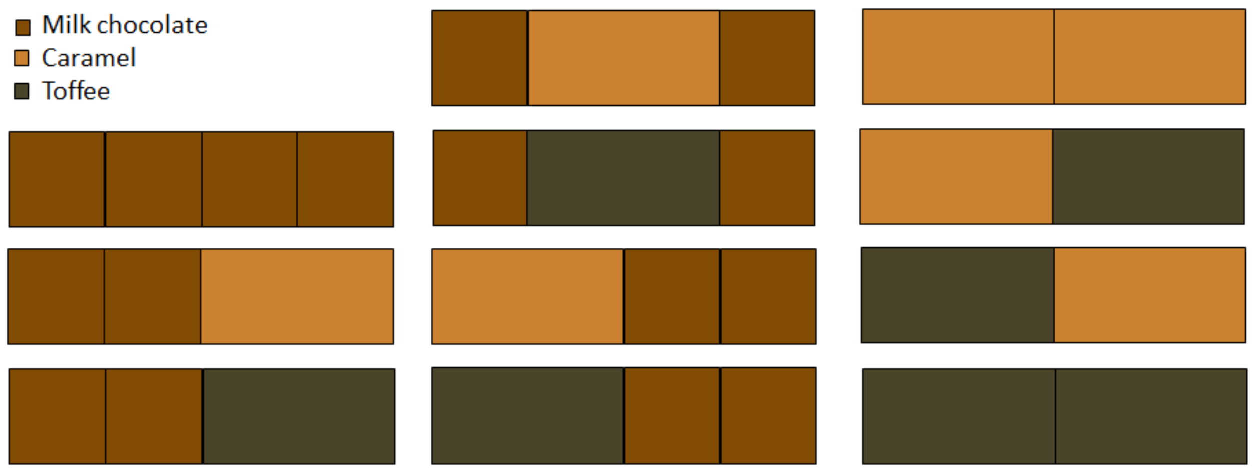
A word of warning: this project is team-based, but it is a nontrivial task. You are highly encouraged to start working on (and start asking questions about) this project early; teams who wait to start until the week before the due date may find themselves unable to complete it in time.

2 Problem Description

The Bitman's Confectionary Company (BCC) is a world-famous candy-making company, and their most popular product is the Bitman's Sampler, which has a variety of different chocolate candies arranged in a spiral pattern. BCC sells the samplers using the tagline "No two alike!" so it's important to know how many different samplers they can put together. The samplers contain a number of different types of candies, which can be one of two sizes, square or "long," which is a candy that is twice as long as a square but has the same width and height. BCC is always developing new types of candies and sells many different sizes of samplers, so we need to be able to answer this question for many different values.

Your algorithm should be able to take in three parameters and compute the number of samplers possible. The three parameters are n, the size of the sampler (given in units of squares); s, the number of varieties of square candies that BCC makes; and ℓ, the number of varieties of long candies.

For example, BCC would be able to make 11 samplers of size 4 using one square (milk chocolate) and two long candies (caramel cookie crunch and toffee surprise):



Note that we consider samplers with the same contents in a different order to be different. Also, this example is displayed in a line rather than a spiral for convenience.

3 Project report

In your project report, you should include brief answers to 8 questions. Note that you must use dynamic programming to solve these problems; purely combinatorial solutions will not receive significant credit.

1. How you can break down a large problem instance into one or more smaller instances? Your answer should include how the solution to the original problem is constructed from the subproblems and why this breakdown makes sense.
2. What recurrence can you use to model this problem using dynamic programming?
3. What are the base cases of this recurrence?
4. What data structure would you use to store the partial solutions to this problem? Justify your answer.
5. Describe a pseudocode algorithm that uses memoization to compute the number of samplers.
6. What is the time complexity of your memoized algorithm?
7. Describe an iterative algorithm for this problem.
8. Can the space complexity of the iterative algorithm be improved relative to the memoized algorithm? Justify your answer.

4 Coding your solutions

In addition to the report, you should implement an *iterative* dynamic programming algorithm that can solve the candy sampler counting problem. Your code may be in C++ or Java, but it must compile and run on the C4 Linux Lab machines. If compiling your code cannot be accomplished by the command

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

you should include a Makefile that 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 in addition to your source code. In either case, your source code may be split into any number of files.

Your code will not need to handle invalid input, such as non-numeric values, negative numbers, or lines with fewer than 3 values. The solution to these problems may potentially exceed 2^{32}, but they will be less than 2^{64}. You have been provided with sample input and output files to test your algorithm.

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 the three parameters on a line. The first parameter will be the size n, the second will be the number of squares s, and the last will be the number of long candies ℓ. All three parameters will be positive integers.

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, containing the number of candy samplers with size n using s squares and ℓ long candies.

5 Submission

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

5.1 Group submission

The submission for your group should be a zip archive containing 1) your report (described in Section 3) as a PDF document, and 2) your code (described in Section 4). 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.

5.2 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.

6 Grading

Report	**50 points**
Questions 1 and 5	10 each
Questions 2–4 and 6–8	5 each
Code	**20 points**
Compiles and is correct	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 a total grade of 0.