CSE 222 Programming Assignment #5

February 28, 2020

1 Introduction

In this assignment, you're going to write a C program that uses recursion and dynamic programming to solve the *egg drop analysis problem* (EDAP). EDAP is a classic problem with a clever recursive solution whose effective implementation requires dynamic programming.

2 Problem Statement

The Question

Given a number of identical eggs and a building with a number of floors, one wishes to answer the question**:** what is the highest floor from which an egg can be dropped without it breaking.

Approaches

There are many ways to do this:

- given as many eggs as there are floors, one can simply drop an egg from each floor to discover the answer;
- given only a single egg, one needs to start from floor 1 and work towards higher floors, since, once an egg is broken, it cannot be re-used
- The cases in-between are more interesting. Given two eggs, one can try the middle floor. If the egg survives, you only need to investigate the higher floors (and you still have 2 eggs); whereas if the egg breaks, you need to investigate the lower floors (and now only have one egg left).

The goal of EDAP is to determine *worst case* how many experiments must be run to answer **The Question**.

Note that you are not coming up with the algorithm for running these experiments (since the steps of the experiment depend on whether or not the egg survives any given drop).

3 Algorithm

You are to design and implement a function

```
int egg(int floors, int eggs);
```

which will return the smallest number of experiments *guaranteed to be sufficient* to answer **The Question** for the given number of floors and eggs. One may define egg() recursively as follows:

$$egg(floors, eggs) = \begin{cases} 0 & floors \leq 0 \\ 1 & floors = 1 \\ floors & eggs = 1 \\ min(1 + max(egg(f-1, eggs-1), egg(floors-f, eggs)) & otherwise \\ 1 \leq f \leq floors & \end{cases}$$

This has the familiar repeated-subcase structure of other algorithms we've been discussing (knapsack, rod cutting, etc.) and, similar to those, the performance is abysmal if coded as shown above. Therefore, you must employ **dynamic programming** to speed up the calculation of the egg function. You can do this by using a 500x500 array of ints:

```
int save[500][500]={0};
```

where save[f][e] stores the value of eggs(f,e).

See your notes from lecture for various simplifying assumptions and other relevant details.

4 Main Program

Your submission should include a main file named "eggtest.c" and a makefile for creating an executable file named "eggtest" eggtest should simply:

1. ask the user for the number floors;
2. ask for the number of eggs; and then
3. print the minimum guaranteed-sufficient trials required, and exit.

Please don't print anything else, ask for any other output, or perform any other actions. **See /tmp/eggtest on the server for a sample executable.**

5 Submission/Groups

- You may work on this in groups of up to 3 people. If you do so, **one** person should submit via GitLab.
- Submit via GitLab by creating a repository named "EggTest" and adding me as a reported by the due date (which is Monday, 9 March 2020 at 8:00 am).

- You must include a **README** file (named "README") that indicates whether you worked on this alone or in a group, and, if you worked in a group, the **full name of each group member (first name + last name) as listed in Canvas**
- Include your source code, a makefile, and the README file only: no other files please.
- I am not asking groups to present in class this quarter: I will run your code myself.
- For this assignment, I will not accept any late submissions. Once I download the repositories (sometimes after 8 am on Monday 9 March 2020), that is it: anything not successfully downloaded by then will not be graded, and missing assignments will receive a 0. You have the entire time between now and the deadline to upload what you have for partial credit. Use GIT to keep things up to date, so that when the deadline comes, I can grade your latest work.