A2: Runtime Analysis

Overview

This assignment will explore some of the basics of runtime analysis. We need a strong understanding of how runtime works to analyze our data structure operations. This assignment should be done solo, and work should all be your own. This assignment is a total of 50 points.

Deliverables

- **A2.pdf** will be submitted to the A2 assignment on Autolab.
    - This will be a PDF with all of your solutions to the problem set.
    - We recommend typing this.
    - **If you submit a .docx or any other file format you will receive no credit. You can easily save a .docx file as a pdf. Any file that is not a pdf will not be graded.**
- **There is no autograded portion for this assignment.**

Due Date

DUE DATE:

A2.pdf: 3/7/2017, at 01:59 (this is Tuesday, Mar 7 at 1:59AM/or for those who stay up late, the continuation of Monday night).

***** Please be aware of this deadline. Autolab will also tell you how long until the deadline.*****

Objectives

In this assignment, you will apply the definitions of Big-O notation to concepts we have learned in C++ to gain an understanding of tradeoff decisions we have to make.

Instructions

Complete the following problems. **When computing Big-O values, they should be as tight as possible.** This means giving $O(2^{n!})$ as the solution to every problem will not receive credit (although technically true).

1. Consider the function $f(n) = n^3 + 3n^2 - 2n + 20$. In order to prove that $f(n)$ is $O(n^3)$, we need constants $c, n_0 > 0$ such that $f(n) \leq cn^3$ for every $n \geq n_0$. Show your work for each part. (4 points each for 16 total)
    a. Suppose we fix $c = 3$. What is the smallest integer value of n_0 that works?
    b. Suppose we fix $c = 1.01$. Now what is the smallest integer value of n_0 to satisfy the inequality?
    c. Suppose we want to fix $n_0 = 1$. Provide some value of c that satisfies the inequality.
    d. Provide a value of c such that no matter what value n_0 takes, the inequality cannot be satisfied.

2. Consider our CSEVector class from lecture. Assume I have created an instance of CSEVector<int> named vector and inserted n integers into vector, for some n>0. (8 points each for 16 total)
   a. Let foo be the following function:

```
bool foo (const CSEVector<int>& in) {
   if(in.size() > 0 && in[0] % 2 == 0) {
      return true;
   }
   return false;
}
```

   What is the runtime of calling `foo(vector);`?

   b. Now suppose I created a copy of the function foo without const or reference qualifiers, creating a new function foobar:

```
bool foobar (CSEVector<int> in) {
   if(in.size() > 0 && in[0] % 2 == 0) {
      return true;
   }
   return false;
}
```

   Does the runtime for `foobar(vector);` differ? Explain or why not. If changed, provide analysis for the new runtime.

3. Analyze the runtime for the following function pow. (8 points)

```
int pow (int a, int n) {
   if(n == 0) { return 1;}
   if(n % 2 == 1) {return pow(a,n/2)*pow(a,n/2)*a;}
   else { return pow(a,n/2)*pow(a,n/2);}
}
```

4. Recall the bubble_sort function from lecture. We originally assumed that T was a primitive type in our analysis. Suppose we wrote an operator to compare strings with the < operator as follows: (6 + 4 points for 10 total)

```
bool operator< (const std::string& lhs, const std::string& rhs) {
   return lhs.compare(rhs) < 0;
}
```

a. Provide a new runtime analysis for the function bubble_sort for n strings of length at most m.
b. Write the code for a new < operator to compare two strings that would allow our original analysis for bubble_sort to hold.

Submission

A2 Write-up (50 points):

You must submit all your solutions in one file A2.pdf to the A2 assignment on Autolab.

- Your submission must be a pdf. If you submit a file that is not of pdf format, you will not receive credit for the assignment.
- **Make sure after you upload your submission you can view it on the Autolab system.**

Start Early!!!!!!!!

Late Policy

The policy for late submissions is as follows:

- Before the deadline: 100% of what you earn (from your best submission).
- One day late: 25-point penalty (lower your earned points by 25, minimum of 0).
- > 1 days late: 0 points.

Keep track of the time if you are working up until the deadline. Submissions become late after the set deadline. If you are working late, keep in mind that **submissions will close 24 hours after the original deadline** and you will no longer be able to submit your code.

AI Policy Overview

As a refresher from what we stated in the first day of lecture, please re-read the academic integrity policy of the course. I will continue to remind you throughout the semester and hope to avoid and incidence.

What constitutes a violation of academic integrity? These first bullets should be obvious:

- Turning in your friend's code (obvious)
- Turning in solutions you found on google (should be obvious)
- Paying someone to do your work (just don't)
- Posting to forums asking someone to solve the problem (Yahoo answers will not save you)

You should know that seeking solutions to the assignment does not fall under solving the problem yourself. Things that may not be as obvious:

- Sending your code to a friend to help them (**if they use your code, you are also liable**)
- Reading your friend's code (it will most likely influence you directly or subconsciously to solve the problem identically)

The assignments should be solved individually with assistance from course staff. You may discuss and help one another with technical issues, such as how to get your compiler running, etc. It is not acceptable that you both worked together and have nearly identical code. If that is going to be a problem for you, don't solve the problems in that close of proximity.

There is also a gray area. How far can you discuss the problems. **I do encourage you to work with one another to solve problems**. That is the best way to learn and overcome obstacles, but at the same time you need to be sure you do not overstep. Talking out the solution from a high level is okay (I choose to use stack to store ___ and then went through things in thus and such order). Just remember that you shouldn't be showing code for **how** to do that. A good rule of thumb would be: Don't come away from discussions with your peers with any written work. Especially not written code.

If you have concerns that you may have overstepped or worked closely with someone, please address me prior to deadlines for the assignment. We will address options at that point.

All this said, please feel free to use any files that we provide directly in your code. Feel free to directly use anything from lecture or recitations. You will never be penalized for doing so. If you are in doubt, always cite where your code comes from. Just remember, if you are citing an algorithm that is not provided by us, then you are probably overstepping.