CS91T Lab Assignment 1

This homework is due at 11:59PM on Wednesday, February 8. For the written portion of this assignment, Write your solution using LaTeX. Submit this homework using **github**.

This is a partnered assignment. You should primarily be discussing this assignment with your partner. It's ok to discuss approaches at a high level. However, you should not reveal specific details of a solution, nor should you show your written solution to anyone else.

When you submit homework assignments this semester, please keep the following in mind:

- Don't forget to fill out the **homework submission poll** using Pollster.
- Don't submit a .pdf – just the .tex will do.
- I will compile your .tex file using pdflatex. It is your responsibility to make sure the LaTeX compiles.
- For the coding portion of this (and other) assignment, feel free to use Python or C++. Submit your program along with the .tex file.

The main **learning goals** of this assignment are to practice working with discrete probability and to familiarize yourself with coding randomized algorithms.

Part 1: Written Problems

1. Show it is possible to color the numbers $\{1, \ldots, 100\}$ using three colors such that no arithmetic progression of length 8 is monochromatic.
2. (Shoup, Exercise 8.1) Show that for all events A, B, the following inequality holds:
$$\Pr[A \cap B] \cdot \Pr[A \cup B] \leq \Pr[A] \cdot \Pr[B] \ .$$
3. (Shoup, Exercise 8.2) Suppose that A, B, C are events such that $A \cap \bar{C} = B \cap \bar{C}$. Show that
$$|\Pr[A] - \Pr[B]| \leq \Pr[C] \ .$$

Part 2: QuickSort Implementation. In CS35, you implemented QuickSort, a sorting algorithm which works fast in practice ($O(n \log n)$ for "typical" inputs). Unfortunately, its worst-case runtime is $O(n^2)$, which can happen when the input array is sorted (or nearly sorted). One application of randomized algorithms is to use randomness to make the worst-case behavior match the typical average case behavior. QuickSort is a good example.

In this Part, you will implement two versions of the QuickSort function:

1. `quicksort` – the standard QuickSort algorithm, using the last element as the pivot element.
2. `rquicksort` – a randomized QuickSort algorithm, which selects a random element from the range of elements to be sorted, swaps this element with the last element, and uses this random element as the pivot element

You will then write a test program `testSort.py` or `testSort.cpp` that runs both algorithms (quicksort and rquicksort) on the same input. How does the runtime of rquicksort scale with the problem size?

Implementation Requirements

- Your program should read in the input from standard input and write output to standard output.
- The first line of input should containg a single integer N, which is the number of items in the list you'll sort.
- The second input line should contain N integers separated by whitespace. These are the integers that need to be sorted.
- The directory `/home/brody/public/cs91T/sorting` contains several input files you can use to test your program. To use these files, you'll need to pipe the content of the files to standard input, e.g. as follows: `python3 testSort.py < /home/brody/public/cs91T/in10k`.
- Your randomized quicksort implementation must run in approximately $O(n \log n)$ time on any input, just like the standard quicksort should run on typical inputs.

As mentioned above, you're welcome to use Python or C++. Either way, I encourage you to consult the "implementation details" post on EdSTEM for tips and tricks for using randomness, running timing experiments, etc.

Part 3: Extra Challenge Problems In many of the assignments, there will be a few extra challenge problems. These problems are completely optional.

1. **(extra challenge problem)** Give an explicit coloring of $\{1, \ldots, 100\}$ using three colors such that no length-8 arithmetic progression is monochromatic.
2. **(extra challenge problem)** Let $f(c, k)$ be the smallest n such that for any coloring of $\{1, \ldots, n\}$ using at most c colors, there is an arithmetic progression that is monochromatic. **Note:** Exercise 1 shows that $f(3, 8) > 100$.

   Give an improved lower bound for $f(3, 8)$. What is the best lower bound you can achieve?
3. **(extra challenge problem)** Choosing the pivot element uniformly as you did in this lab is not the only optimization trick for QuickSort. Come up with your own quicksort algorithm that is as effecient as possible. Can you guarantee the pivot element will always be the median of the list? If not, can you aim for the pivot element beign *close* to the median?