EECS 3101 Fall 2021 (B & E) – Assignment 2

Remember to write your **full name** and **student number** prominently on your submission. To avoid suspicions of plagiarism: at the beginning of your submission, **clearly state any resources (people, print, electronic) outside of the course materials, and the course staff, that you consulted**. You must submit a PDF file with the required filename. The PDF file could be generated using LaTeX(recommended), Microsoft Word (saved as PDF, **not** the `.docx` file), or other editor/tools. The submission must be **typed**. Handwritten submissions are **not** accepted.

Due October 22, 2021, 10:00 PM (on eClass); required file(s): a2sol.pdf

Answer each question completely, always justifying your claims and reasoning. Your solution will be graded not only on correctness, but also on clarity. Answers that are technically correct that are hard to understand will not receive full marks. Mark values for each question are contained in the [square brackets]. Your submitted file does **not** need to include/repeat the questions — just write your solutions.

The assignment must be completed individually.

1. **[10] Slot machine design and analysis**

   Alice works for the casino as a **senior slot machine designer**. Her slot machine has three reels, namely A, B and C. Each of the reels, when stops, reveals an integer value chosen from the set $\{1, \ldots, m\}$ (for some $m \geq 1$) independently and uniformly at random. The slot machine is programmed to run the following algorithm, which takes the values of the three reels as input and returns the amount of money gained by the gambler.

```
// pre: 1 <= A, B, C <= m
int SlotMachine(int A, int B, int C) {
    int money = -2;            // pay 2 dollars to play
    if (A == B == C == 1) {
        for (i in [1, 2, 3, ..., 500]) {    // Jackpot!
            print "cha-ching";
            money = money + 1;
        }
    }
    else if(A == B == C) {
        for (i in [1, 2, 3, ..., 20]) {     // Not bad!
            print "cha-ching";
            money = money + 1;
        }
    }
    else if (A == B) {
        for (i in [1, 2, 3, 4, 5]) {        // Better than nothing!
            print "cha-ching";
            money = money + 1;
        }
    }
    return money;
}
```

   Alice's job as a senior slot machine designer sounds very simple: decide a proper value of m. To realize how technical this job is, you will answer a series of questions below.

   In the following questions, we measure the running time of the `SlotMachine` algorithm by counting the number of times that a **print** line is executed. The "best case" means the case with the smallest number of **print** lines executed (the gambler gains the least amount of money). Similarly, the "worst case" means the case with the largest number of **print** lines executed. In other words, the "best" and "worst" are from the casino's point of view, not from the gambler's.

   (a) In the **worst case**, how many "cha-ching"'s are printed by `SlotMachine`? Justify your answer.

(b) What is the probability that the worst case occurs? Justify carefully.

(c) In the **best case**, how many "cha-ching"s are printed by `SlotMachine`? Justify your answer.

(d) What is the probability that the best case occurs? Justify carefully.

(e) What is the **expected** return value (money) of `SlotMachine`? Give a *detailed* analysis based on the probability distribution described above, and simplify your final answer.

(f) Now the casino's boss wants the gambler to be **expected to lose** between 0.8 and 1.0 dollar per play. What is the proper value of m that Alice should choose? Justify your answer.

2. [10] **Computing numbers in a sequence**

Consider a sequence S_n of numbers defined by the following recurrence.

$$S_n = \begin{cases} 1 & n = 1 \\ 2 & n = 2 \\ 2S_{n-1} + 3S_{n-2} & n > 2 \end{cases}$$

Our job in this question is to design an efficient algorithm `GetS(n)` that returns S_n given a natural number $n \geq 1$.

(a) Below is a naive implementation of the `GetS` function.

```
// pre: n >= 1
// post: return S_n
int GetS(int n) {
    if (n <= 2) {
        return n;
    }
    return 2 * GetS(n-1) + 3 * GetS(n-2);
}
```

Use the **recursion tree** method to show that the above algorithm's runtime is in $\Omega(2^n)$, i.e., it has the exponential runtime that is unacceptably slow. In your PDF, you may include, in additional to your arguments, a picture of the tree that you draw by hand or with a diagram creating tool.

(b) Design a much more efficient algorithm for `GetS` that satisfies the following requirements.

- The algorithm must have a runtime of $\Theta(\log n)$.
- The design of the algorithm must use divide-and-conquer somewhere.

In your answer, describe the high-level idea of your algorithm and then write the **pseudocode** of your algorithm. **Explain** clearly and concisely why it returns the correct answer. A formal proof of correctness is not required.

Hint: The following matrix multiplication is a critical observation behind the idea of the algorithm.

$$\begin{bmatrix} x \\ y \end{bmatrix} \cdot \begin{bmatrix} 2 & 3 \\ 1 & 0 \end{bmatrix} = \begin{bmatrix} 2x + 3y \\ x \end{bmatrix}$$

(c) Justify the runtime of your $\Theta(\log n)$ algorithm using the master theorem.