Homework 3

Algorithms, Fall 2022

Honor code: *Work on this assignment alone or with one partner. Between different teams, collaboration is at level 1 [verbal collaboration only]. There are lots of resources online, such as animations, visualizations, practice problems, videos, and solutions— which you are encouraged to explore to deepen your understanding. However, you must be careful not to search for the specific problems in the assignment with the intent of getting hints for the solutions. Searching for the assignment problems on the internet violates academic honesty for this class.*

---

1. For the algorithm below, give the runtime recurrence and solve it.

   We expect: (1) the recurrnece (2) two steps of iteration (3) the pattern after i steps of iteration; (4) derivation of the recursion depth; (5) the final $\Theta()$ bound

   AlgorithmB(n):
   - Do something that takes $O(1)$
   - AlgorithmB($n/2$)
   - Do something that takes $O(n)$
   - AlgorithmB($n/2$)
   - Do something that takes $O(n^2)$
   - AlgorithmB($n/2$)

2. Analyzing the Towers of Hanoi problem:[1] In this puzzle, we have n disks of different sizes and three pegs. Initially all the disks are on the first peg, in order of size, the largest on the bottom and the smallest on top. The problem is to move all the disks to the third peg, using the second one as auxiliary, if necessary. We can move only one disk at a time, and it is forbidden to place a larger disk on top of a smaller one. The problem has an elegant recursive solution:

[1] You may remember this from Data Structures

```
MOVE(n, fromPeg, toPeg, auxPeg)
1  // Move the top n disks from fromPeg to toPeg using auxPeg
2  If (n == 1)
3        move one disk: move the top disk of fromPeg to the top of toPeg
4  Else
5        MOVE (n - 1, fromPeg, auxPeg, toPeg)
6        MOVE(1, fromPeg, toPeg, auxPeg)
7        MOVE(n- 1, auxPeg, toPeg, fromPeg)
```

(a) Analyze the running time of MOVE(n) (i.e. write a recurrence relation for its running time and find its asymptotic complexity). Assume that the base case (ie.moving one disk) runs in $\Theta(1)$ time.

(b) How many actual moves (of one disk) are necessary to move $n = 5$ disks? What about $n = 100$ disks?

(c) Assuming we could do 1,000,000 (apprx. 2^{20}) moves a second, how long would it take us to move 100 disks?

3. Fibonacci numbers: In this problem we consider the Fibonacci numbers, a famous sequence.

$$0, 1, 1, 2, 3, 5, 8, 13, 21, 34, ...$$

That can be defined by the simple recurrence

$$F(n) = F(n-1) + F(n-2), \text{ for } n > 1$$

and two initial conditions:

$$F(0) = 1, F(1) = 1$$

The Fibonacci numbers were introduced by Leonardo Fibonacci in 1202 as a solution to a problem about the size of a rabbit population. Many more examples of Fibonacci-like numbers have been since discovered.

Here is a straightforward way to compute the nth Fibonacci number:

```
FIB(n)
1  // Compute the nth Fibonacci number using the definition
2  Input: A nonnegative number n
3  Output: The nth Fibonacci number
4  If n ≤ 1 return n
5  Else return FIB(n − 1)+FIB(n − 2)
```

(a) Analyze the running time of $Fib(n)$ and show that it is at least exponential.

(Hint: Note that the problem is asking for a lower bound, not a tight bound. Write a recurrence for its running time and show that it is lower bounded by an exponential. Use that $T(n)$ is an increasing function, which means that $T(n-1) > T(n-2)$, and use it to show that $T(n) > .....$).

We can obtain a much faster algorithm by simply computing the successive elements of the Fibonacci sequence iteratively, as is done in the following algorithm:

```
FIB2(n)
1  // Compute the nth Fibonacci number iteratively by using the definition
2  Input: A nonnegative number n
3  Output: The nth Fibonacci number
4  create an array F[] of size n + 1 and set F[0] = F[1] = 1
5  For i = 2 to i = n
6        set F[i] = F[i − 1] + F[i − 2]
7  return F[n]
```

(b) What is the running time of FIB2(n)?

(c) How much space does FIB2(n) use?

(d) Show how to improve FIB2(n) to use only $\Theta(1)$ space.

(e) It can be shown[2] that the nth Fibonacci number is $F(n) = \frac{1}{\sqrt{5}}(\phi^n - \hat{\phi}^n)$, where $\phi = (1+\sqrt{5})/2$ and $\hat{\phi} = -1/\phi = -0.61803$. The maximum value of the Java primitive type `int` is $2^{31} - 1$ Find the smallest n for which the nth Fibonacci number is not going to fit in a variable of type int (You can approximate the answer).

Evaluation

This assignment will be evaluated along several criteria:

1. **Correctness**: Is your answer correct?
2. **Justification**: Is your answer justified?
3. **Style**: Does it look professional and neat? Is the explanation written carefully in complete sentences, and well-organized logic? Is it easily human-readable? Is it easy to understand?

- For this assignment, if your handwriting is legible, you do not need to type as there are too many formulas. If your handwriting is hard to read, please type.
- Write each problem on a separate page or leave plenty of space between problems so that we can write comments.

[2]Constant ϕ is known as the golden ratio. Since antiquity it has ben considered the most pleasing ratio of a rectangle's two sides to the human eye and might have been consciously used by ancient architects and sculptors.

- Try to put yourself in the position of the reader. If you hadn't been thinking of this problem for 3 hours, would your answers make sense to you?
- Try to finish the assignment early, then step away for a day or two, and then come back to it and read it again. Chances are you'll find something you can write more clearly.
- Look at posted solutions for style advice (if solutions are not posted, ask).