Assignment 3

Algorithms, Spring 2024

Honor code: *Work on this assignment with one partner (highly encouraged) or, if you must, work alone*[1]*. Partner policy: You and your partner will work together on the assignment throughout the whole process, you will write it and review it together, and will submit one assignment. Between different teams, collaboration is at level 1 (verbal collaboration only). It is not allowed to search online for the specific problems in this assignment—doing so violates academic honesty for the class.*

---

1. For the algorithm below, write the recurrence for its worst case running time and find a tight bound for it.

```
MYSTERY(n)
1  If (n == 1)
2       do something that takes O(1) time and return
3  Else
4       Do something that takes O(1)
5       Mystery(n/2)
6       Do something that takes Θ(n)
7       Mystery(n/2)
8       Do something that takes Θ(n^2)
9       Mystery(n/2)
```

   We expect: (1) the recurrence (2) two steps of the iteration (3) the expression for $T(n)$ after i steps of iteration; (4) derivation of the recursion depth; (5) the final $\Theta()$ bound

2. **Analyzing the Towers of Hanoi problem:** The Towers of Hanoi problem is one of the classic examples shown when learning recursion, because it is a hard problem that has a beautifully simple and short solution via recursion. The problem is the following: 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. This would be a

[1] Working with a partner will improve your learning—you are highly encouraged to find a partner, you are also welcome to message me and I'll help connect you

straightforward task, but there are the following constraints: (1) We can move only one disk at a time, and (2) it is forbidden to place a larger disk on top of a smaller one.

The problem has an elegant recursive solution: to move n disks from peg1 to peg3, first move the top $n-1$ disks from peg1 to peg2, then move the nth disk from peg1 to peg3, and then move the top $n-1$ disks from peg2 to peg3. At the end, all disks are on peg3, task accomplished! Written in pseudo-code, the solution looks something like this:

```
MOVE(n, fromPeg, toPeg, auxPeg)
1  // Move the top n disks from fromPeg to toPeg using auxPeg
2  If (n == 1)
3        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) Let $T(n)$ be the running time of calling MOVE to move n disks. Assume that the base case (ie.moving one disk) runs in $\Theta(1)$ time. Write a recurrence relation for $T(n)$ and find its asymptotic complexity.

(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 which was introduced in 1202 by Leonardo Fibonacci as a solution to a problem about the size of a rabbit population; many more examples of Fibonacci-like numbers have been since discovered.

   We'll denote by $F(n)$ the nth Fibonacci number. The first two numbers, $F(0)$ and $F(1)$ are both 1: $F(0) = 1, F(1) = 1$. For $n > 1$, $F(n)$ is the sum of the two numbers before it

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

   The Finonacci sequence is the following:

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

   Here is a straightforward (recursive) function to compute the nth Fibonacci number:

```
FIB(n)
1  Input: A nonnegative number n
2  Output: Returns the nth Fibonacci number
3  If n ≤ 1:
4        return n
5  Else
6        return FIB(n − 1)+FIB(n − 2)
```

(a) Let $T(n)$ be the running time of computing Fib(n). Write a recurrence for $T(n)$ and show that $T(n) = \Omega(2^{n/2})$.

(Hint: Note that the problem is asking for a lower bound, not a tight bound; basically we want to show that $T(n)$ is lower bounded by an exponential. This is a good time to remember examples of exponential recurrences–check out lab3. After you write the recurrence for $T(n)$, use that $T(n)$ is an increasing function, which means that $T(n-1) > T(n-2)$; use this to show that $T(n) > 2T(n-2)$, and from here derive an exponential lower bound for $T(n)$).

As we know, an algorithm that runs in exponential time is basically infeasible on anything except very small values of n. Can we do better? In this case we can obtain a much faster algorithm for computing $F(n)$ by simply computing the elements of the Fibonacci sequence and saving them in an array, as follows:

```
FIB2(n)
1  Input: A nonnegative number n
2  Output: Returns the nth Fibonacci number
3  create an array F[ ] of size n + 1 and set F[0] = F[1] = 1
4  For i = 2....n
5        set F[i] = F[i − 1] + F[i − 2]
6  return F[n]
```

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

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

(d) 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).

[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.

Evaluation

The assignment will be evaluated along three criteria: correctness, justification and style.
Style guidelines:

- Space: Write each problem on a separate page or leave plenty of space between problems so that we can write comments.
- Neat and legible: We expect your answers to be clearly written and easily human-readable, with complete sentences and well-organized logic, and should definitely not be your first draft. Ideally answers should be typed.
- Clear: 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.