CS 141, Fall 2018 Assignment 1

Posted: September 28th, 2018 Due: October 12th, 2018, 11:59pm

Notice

- Include your full name and student ID in your solution
- You are expected to work on this assignment on your own
- Use pseudocode, Python-like or English to describe your algorithms. Absolutely no C++/C/Java
- When designing an algorithm, you are allowed to use any algorithm or data structure we explained in class, without giving its details, unless the question specifically requires that you give such details
- Always remember to analyze the time complexity of your algorithms
- Homework has to be submitted electronically on iLearn by the deadline. Late submission allowed for 20% penalty for a calendar day.

Problem 1. (20 points) The Fibonacci numbers are the numbers in the following integer sequence.

0, 1, 1, 2, 3, 5, 8, 13, 21, 34, 55, 89, 144,...

In mathematical terms, the sequence F n of Fibonacci numbers is defined by the recurrence relation:

$F_n = F_{n-1} + F_{n-2}$, where $F_0 = 0, F_1 = 1$

Discuss the correctness of the following algorithms that calculate Fibonacci numbers:

1. Algorithm 1

```
fib1(n)
{
  if(n!=1)
      return fib1(n-1) + fib1(n-2);
  else
      return n;
}
```

2. Algorithm 2

```
fib2(n)
{
  integer array f[n+1];
  f[0] = 0;
  f[1] = 1;

  for (i = 2 to n)
    f[i] = f[i-1] + f[i-2]
  return f[n];
}
```

3. Algorithm 3

```
fib3(n)
{
  a = 0
  b = 1
  if(n = 0)
    return a
  i = 1
  while (i < n)
  {
        c = a + b
        b = c
        a = b
  }
  return b;
}
```

Problem 2. (20 points) Is the following statement true or false?

$$n^4 = \Omega(n^3 \log n)$$

Justify your answer using the basic definition of the Ω-notation.

Problem 3. (20 points) Give a tight bound (using the big-theta notation) on the time complexity of following method as a function of n. For simplicity, you can assume n to be a power of two.

Algorithm WeirdLoop (n : **integer**)
  for ($i = n/2;\ i \leq n;\ i = i + 1$)
    for ($j = 1;\ j \leq n;\ j = 2j$)
      $k \leftarrow 1$

while $k \leq n$ **do**
  $k \leftarrow 2k$

Problem 4. (20 points) Order the following list of functions by the big-Oh notation, i.e., rank them by order of growth. Group together (for example, by underlining) those functions that are big-Theta of one another. Logarithms are base two unless indicated otherwise. g

$3n+2$	n^3+n^2	$\log^2 n$	$n!$	2^{2^n}	4^{n-1}
$2^{\log n}$	$2\sqrt{2\log n}$	$\sqrt{n}$	n^3	1	$3^{n/3}$
$(n+1)!$	$2^{2^{n+1}}$	e^n	$n^{\log\log n}$	$\log n$	$(\log n)^2$
2^n	$n3^n$	$4^{\log n}$	$4n^2/\sqrt{n}$	$\sqrt{\log n}$	$n\log n$

Problem 5. (20 points) Given n non-negative integers $a_1, a_2, ..., a_n$, where each represents a point at coordinate (i, a_i). n vertical lines are drawn such that the two endpoints of line i is at (i, a_i) and $(i, 0)$. Find two lines, which together with x-axis forms a container, such that the container contains the most water.

1. Solve the problem with brute-force approach. What is the time complexity of this approach?

2. Is there any better solution? If so, describe your solution with time complexity analysis.

Note: You may not slant the container and n is at least 2.