ICS 691: Advanced Data Structures Spring 2016

Pre-requisite Assessment Exam

Prof. Nodari Sitchinava *Due: 5pm, Thursday, January 14, 2016*

Instructions

The purpose of this take-home exam is to give me an idea of your level of comprehension of the pre-requisites for this class. The grade on this exam will not be part of your overall course grade. But you have to give honest effort, because based on your answers, I will be able to recommend you specific material to review before we jump into more advanced topics.

While taking the exam you may consult the course webpage and CLRS textbook ("Introduction to Algorithms" by T. Cormen, C. Leiserson, R. Rivest, C. Stein), but no other resources. You may NOT discuss this exam with anyone in or outside the class and should solve all problems by yourself.

Important: For each problem, please list how long it took you to solve it (including the time to review book chapters and/or course webpage)?

1 Analysis of Algorithms

Analyze the runtime of the following algorithm. Justify your answer.

```
Foo(n)
1  if (n < 2)
2     return 1
3  else
4     x = 0
5     for i = 1 to 5
6        x = x + Foo(n/2)
7     for i = 1 to n
8        for j = 1 to n
9           x = x + 1
10    return x
```

2 Solving Recurrences

Solve the following recurences using any method you like, but you must justify your answers (i.e. show the proofs).

(a) $T(n) = \begin{cases} T(n/2) + O(\log n) & \text{if } n > 1 \\ O(1) & \text{otherwise.} \end{cases}$

(b) $T(n) = \begin{cases} T(n-1) + O(\log n) & \text{if } n > 1 \\ O(1) & \text{otherwise.} \end{cases}$

(c) $T(n) = \begin{cases} 2T(\sqrt{n}) + O(1) & \text{if } n \geq 4 \\ O(1) & \text{otherwise.} \end{cases}$

(d) $T(n) = \begin{cases} \sqrt{n}T(\sqrt{n}) + O(n) & \text{if } n \geq 4 \\ O(1) & \text{otherwise.} \end{cases}$

(e) $T(n) = \begin{cases} 2T(n/2) + O(n) & \text{if } n > K \\ O(1) & \text{otherwise.} \end{cases}$
K is NOT a constant and should appear in your solution.

(f) $T(n) = \begin{cases} T(n/K) + O(n) & \text{if } n > K \\ O(1) & \text{otherwise.} \end{cases}$
K is NOT a constant and should appear in your solution.

3 Binary counter

An array $A[0 \ldots k-1]$ of bits (each array element is 0 or 1) stores a binary number $x = \sum_{i=0}^{k-1} A[i] \cdot 2^i$. To add 1 (modulo 2^k) to x, we can use the following procedure:

```
Increment(A,k)
  i = 0
  while (i < k and A[i] = 1) do
    A[i] = 0
    i = i+1
  if i < k
    then A[i] = 1
```

Let $n < 2^k$ and assume the array A had been initialized to all 0's. Perform the analysis of the following algorithm (which simply calls `Increment(A,k)` n times).

```
Foo(n, A, k)
  for j = 1 to n
    Increment(A,k)
```

Provide the tightest possible analysis you can. Your runtime should be a function of n (using big-O notation).

4 Sorting integers

(a) Design an algorithm that sorts n integers in $O(n)$ time, if each integer can take any value in the range $[0, n^3 - 1]$. Analyze your algorithm.

(b) What is the best algorithm you can come up with for sorting n integers without any bound on possible values. What is its runtime?

5 Syllabus

Read the course syllabus on the course webpage before answering these questions.

(a) Describe the grading scheme for homeworks.

(b) **True or False:** It is an instance of academic dishonesty to raise your hand when asked if you solved a particular problem on the homework if you hadn't solved it and are simply hoping not to be called to explain the solution on the whiteboard.

(c) What type of project would you be most likely interested in?

(d) List the dates of all lectures in order of preference of whether you would like to scribe the notes. (You can find the dates of lectures on the class website).