

**CIS121 - Fall 2009****Homework 2** Friday, Sep 25

Due: Wednesday, September 30 at 10:00am in Levine 459

You don't need to type up this assignment if you don't want to but keep in mind that illegible answers will not be graded. Please write your name and lab section at the top of each page, then staple your pages together securely. Turn in your assignment to the course administrator, Charity Payne, in Levine 459.

**Problem 1: 25 points**

Solve the recurrence relation  $T(n) = T(\sqrt{n}) + 1$  for  $n = 2^k \geq 2$  and  $T(2) = 1$ .

**Problem 2: 25 points**

Solve the recurrence relation and calculate a  $\Theta$ -bound.

$$T(n) = T(n/2) + n^2 \text{ and } T(1) = 1$$

**Problem 3: 25 points**

Compute the big- $\Theta$  for the following methods. If recursive, then write the recurrence relation and solve it. Pay attention to operations on strings (which are really character arrays, concatenation takes  $\Theta(\text{string.length})$ ). State any assumptions you make about the running time of basic operations.

```
public static String method1(int n) {
    String s = "a";
    for (int i = 0; i < n; i++)
        s = s + s;
    return s;
}
```

## Homework 2

cis121

```
public static String method2(int n) {
    if (n == 0) return "a";
    else return method2(n-1) + method2(n-1);
}
```

## Problem 4: 25 points

Analyze the following code and give a big-Theta characterization of the running time of `mystery`.

```
static void mystery(int[] a) {
    aux(a, 0, a.length-1);
}

static void aux(int[] a, int bot, int top) {
    if (bot >= top) return;
    int x = a[top];
    int j = bot;
    int k = top-1;
    while (j <= k) {
        while (j<=k && a[j]<=x) j++;
        while (k>=j && a[k]>=x) k--;
        if (j < k) {
            int tmp = a[j]; a[j] = a[k]; a[k] = tmp;
        }
    }
    a[top] = a[j]; a[j] = x;
    aux(a, bot, j-1);
    aux(a, j+1, top);}
}
```