CS2230 Computer Science II: Data Structures
Homework 4
Asymptotic Analysis

Due February 24, 2017, 11:59pm

Goals for this assignment

- Practice using Big-Oh notation
- Analyze the running times of some algorithms

Submission Checklist

You should submit a PDF file titled *hw4.pdf*. Upload it on ICON under Assignments > Homework 4. Physical paper copies are not acceptable.

Part 0: Read about Asymptotic Analysis and take Quiz 4

To help you better understand Big-Oh notation and running time of algorithms, it is recommended that you read Chapter 4 in your textbook.

Part 1: Experiments

1. Ryan and Brandon are arguing about the solution to your upcoming homework assignment on sorting algorithms. Ryan claims that his O(n log n)-time solution is *always* faster than Brandon's O(n^2) solution. However, Brandon claims that he ran several experiments on both algorithms on his laptop and *sometimes* his was faster. Explain what probably happened.

Part 2: Growth rate

2. Order the following functions by asymptotic growth rate:
    a. $5n \log n + 4n$ | $12n^2$ | 150 | $4\log n$
    b. $12n^4 + 5n$ | 2^{10} | $6\log n$ | $5n^3$
    c. 6^n | $7n \log n$ | $8n + 9$ | $60000 * n^6$
    d. 63 | $64n$ | $3\log n$ | 2^{n+2} | $10^{\log n}$

Part 3: Proof and Analysis

3. Give a *good* big-Oh characterization in terms of *n* of the running time of the following. Provide brief justification for your answer (in terms of finding a k and n_0).
    a. $4n^5 + 3n^3 + 7$
    b. $15n^{12} + 3n \log n + 2n$
    c. $3n \log n + 2\log n + n$
    d. 12n*3^n + 50n
4. Give a *good* big-***Omega*** characterization in terms of *n* of the running time of the following. Provide brief justification for your answer (in terms of finding a k and n_0).
    a. $5 \log n + 12n^2$
    b. $6n \log n + 5\log n + 4n$
5. Show that the following statements are true:
    a. 4^{n+5} is in $O(4^n)$
    b. $n \log n$ is in $\Omega(n)$

Part 4: Algorithm Analysis

6. Given the following algorithms below, give a big-Oh characterization of the *running time* in terms of the size of the input, *n*. Provide justification (description, equations, and/or diagrams) for your answer.

    a.

```java
public static boolean two_sum(int[] arr) {
    for (int i=0; i<arr.length; i++) {
        for (int j=i; j<arr.length; j++) {
            if (i!=j && arr[i]+arr[j]==0) {
                return true;
            }
        }
    }
    return false;
}
```

    b.

```java
public static int something(int n){
     for (int i=0; i<42; i++) {
        n += i;
     }
     return n;
}
```

c. First, find the big-Oh running time of inside, in terms of input sizes n_a and n_b.

```
private static double[] inside(double[] a, double[] b) {
 double[] c = new double[a.length];
 int i = 0, j = 0;
 for (int k = 0; k < c.length; k++) {
     if (i < a.length) {
         if (j < b.length) {
             if(a[i] <= b[j]) {
                 c[k] = a[i];
             } else {
                 c[k] = b[j];
             }
         } else  {
             c[k] = a[i];
             i++;
         }
     } else {
         if (j < b.length) {
             c[k] = b[j];
             j++;
         }
     }
 }
 return c;
}
```

Now, find the running time of outside, in terms of the size n, using your answer from above.

```
public static double[] outside(double[] list) {
     int x = list.length;
     if (x <= 1) return list;
     double[] a = new double[x/2];
     double[] b = new double[x - x/2];
     for (int i = 0; i < a.length; i++) {
         a[i] = list[i];
     }
     for (int i = 0; i < b.length; i++) {
         b[i] = list[i + x/2];
     }
     return outside(inside(a, b));
}
```

d.

```
int strange_sum(int[] arr) {
    if (arr.length == 1) {
        return arr[0];
    } else {
        int[] arrLeft = new int[arr.length/2+1];
        int[] arrRight = new int[arr.length/2];
        for (int i=0; i<arr.length/2+1; i++) {
            arrLeft[i] = arr[i];
        }
        for (int i=arr.length/2+1; i<arr.length; i++) {
            arrRight[i-(arr.length/2+1)] = arr[i];
        }
        return strange_sum(arrLeft) + strange_sum(arrRight);
    }
}
```

e.

```
public static void printSomething(int n){
    for(int i = 0; i < n; i++)
    {
        for(int j = n; j > 0; j/=2)
        {
            System.out.println("Something");
        }
    }
}
```