Homework #2: Analysis of Algorithms

CIS 121—Fall 2015
cis121-15fa-staff@googlegroups.com
Due: Tuesday, September 15th, 2015 at 10:30 AM
(paper submission and online submission)

1 Observe, Model, Predict
20 points

The table below contains observations of the running times of a program for different sized inputs.

<table>
<thead>
<tr>
<th>Problem size $N$</th>
<th>Running time $t(N)$</th>
</tr>
</thead>
<tbody>
<tr>
<td>10,000</td>
<td>0.063</td>
</tr>
<tr>
<td>20,000</td>
<td>0.2468</td>
</tr>
<tr>
<td>40,000</td>
<td>0.2766</td>
</tr>
<tr>
<td>80,000</td>
<td>1.14</td>
</tr>
<tr>
<td>160,000</td>
<td>4.3754</td>
</tr>
<tr>
<td>320,000</td>
<td>17.4726</td>
</tr>
<tr>
<td>640,000</td>
<td>66.24</td>
</tr>
</tbody>
</table>

Use this data to do the following:

1. **Observe**. Using whatever method you like, plot the observations in a graph with problem size on the $x$-axis and running time on the $y$-axis. Each row in the table should correspond to a point in the graph. Fit a line to the points and extend the line for larger values of $N$.

2. **Hypothesize**. Given the observations, create a model of the program’s behavior. Write an equation that mimics the behavior of the line from your plot. Explain how you came to this answer, and define all variables in your equation.

3. **Predict**. Predict the running time of this program for the following values:

<table>
<thead>
<tr>
<th>Problem size $N$</th>
<th>Running time $t(N)$</th>
</tr>
</thead>
<tbody>
<tr>
<td>500,000</td>
<td></td>
</tr>
<tr>
<td>1,000,000</td>
<td></td>
</tr>
<tr>
<td>2,000,000</td>
<td></td>
</tr>
</tbody>
</table>

2 Order of Growth
10 points

Label each of the plots below with the description that best matches the order of growth that it depicts. Choose one of the following:
(a) cubic
(b) logarithmic
(c) constant
(d) exponential
(e) linearithmic
(f) quadratic
(g) linear
3 Tilde Approximations
20 points

Let each cell in the function column represent a different $g(N)$ and each tilde approximation represent $f(N)$. Compare the ratio of $g(N)/f(N)$ for each $g(N) \sim f(N)$.

<table>
<thead>
<tr>
<th>$g(N)$</th>
<th>$f(N)$</th>
<th>$g(N)/f(N)$ for</th>
</tr>
</thead>
<tbody>
<tr>
<td>$N + 1$</td>
<td></td>
<td>$N = 1$</td>
</tr>
<tr>
<td>$1000 + \frac{1}{N}$</td>
<td></td>
<td>$N = 100$</td>
</tr>
<tr>
<td>$\left( \frac{1}{2} + \frac{1}{N} \right) \left( \frac{1}{2} + \frac{2}{N} \right)$</td>
<td></td>
<td>$N = 10,000$</td>
</tr>
<tr>
<td>$N(N - 1)(N - 2)/6$</td>
<td></td>
<td>$N = 1,000,000$</td>
</tr>
<tr>
<td>$\log N + N^2 + 1000$</td>
<td></td>
<td></td>
</tr>
<tr>
<td>$15N^2 + 2N^3 + N$</td>
<td></td>
<td></td>
</tr>
<tr>
<td>$\log 2^N + 2^{1000}$</td>
<td></td>
<td></td>
</tr>
<tr>
<td>$n^3 + \lfloor \log n \rfloor!$</td>
<td></td>
<td></td>
</tr>
<tr>
<td>$n + 2^\log n + 10000$</td>
<td></td>
<td></td>
</tr>
<tr>
<td>$N\log N^2 + N^2 + 10N$</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

4 Code analysis
15 points

1. For the code fragment below, how many times does the code increment the counter as a function of $N$?

```cpp
int count = 0;
for (int i = 1; i < N; i *= 2)
    for (int j = 0; j < N; j++)
        count++;
```
2. For the code fragment below, how many times does the code access the array as a function of $N$?

```java
int count = 0;
for (int i = 0; i < N; i++)
    for (int j = i + 1; j < N; j++)
        for (int k = j + 1; k < N; k++)
            if (a[i] + a[j] == a[k])
                count++;
```

5 Big $\mathcal{O}$, and Other Notations
30 points

1. What is the difference between $O(f(n))$ notation and tilde notation? Under what circumstances would one be preferred over the other? How about $\Omega(f(n))$ and $\Theta(f(n))$? Under what conditions can you describe an algorithm as optimal? Write a short paragraph explaining how these different notations are related.

2. Prove that if $d(n) \in O(f(n))$ and $e(n) \in O(g(n))$, then $d(n) \cdot e(n) \in O(f(n) \cdot g(n))$.

3. Prove or give a counterexample: if $f \in O(g)$, then $2^f \in O(2^g)$.

4. Suppose you are given $n \geq 3$ apples. All of them have the same weight except for 1 apple. You are allowed to perform experiments in which you compare the weights of two collections of apples with each other. Each experiment reveals whether or not the weight of two piles are equal. Give an algorithm that runs in $O(\log n)$ to find the odd apple that has a different weight.

6 Memory Analysis
5 points

How much memory does the `WeightedQuickUnionUF` implementation below use as a function of $N$?

```java
public class WeightedQuickUnionUF {
    private final int[] parent;
    private final int[] size;
    private int count;

    public WeightedQuickUnionUF(int N) {
        count = N;
        parent = new int[N];
        size = new int[N];
        for (int i = 0; i < N; i++) {
            parent[i] = i;
            size[i] = 1;
        }
    }
    /* ... */
}
```
7 Extra Credit

20 points

A function $f(n)$ is said to be polynomially bounded if and only if there exists a constant $\alpha$ such that $f(n) \in O(n^\alpha)$.

1. Determine whether the function $g(n) = \lceil \log n \rceil !$ is polynomially bounded. Give a proof for your answer.

2. Determine whether the function $h(n) = \lceil \log \log n \rceil !$ is polynomially bounded. Give a proof for your answer.

**Hint:** You may find it helpful to use the following characterization of polynomially bounded functions: A function $f(n)$ is polynomially bounded if and only if $\log f(n)$ is $O(\log n)$. 