CSC 212: Data Structures and Abstractions

Spring 2018

University of Rhode Island

Weekly Problem Set #4

Due Wednesday 2/21 before lab. Please turn in neat, and organized, answers hand-written on standard-sized paper **without any fringe**. At the top of each sheet you hand in, please write your name, and ID. The only library you're allowed to use in your answers is `iostream`.

1. Rank the following functions by their asymptotic growth rate in ascending order. In your solution, group those functions that are big-Theta of one another (all log functions are base 2):

$$\begin{array}{ccccc} 6 \cdot n \log n & 2^{100} & \log \log n & \log^2 n & 2^{\log n} \\ 2^{2^n} & \lceil \sqrt{n} \rceil & n^{0.01} & 1/n & 4n^{3/2} \\ 4^n & n^3 & n^2 \log n & 4^{\log n} & \sqrt{\log n} \end{array}$$

2. Algorithm `algo1` uses $10n \log n$ operations, while algorithm `algo2` uses n^2 operations. What is the value of n_0, such that `algo1` is better than `algo2` for $n \geq n_0$.

3. For each of the following, give both a big-Oh characterization in terms of n, and an exact characterization (count additions and multiplications):

(a)
```
EX: For the following, the big-Oh characterization is: O(n),
the exact characterization is n.
s = 1
for i = 1 to n do
    s = s * i
```

(b)
```
s = 1
for i = 1 to 4n do
    s = s * i
```

(c)
```
s = 1
for i = 1 to n*n*n do
    s = s * i
```

(d)
```
s = 0
for i = 1 to 4n do
    for j = 1 to i do
        s = s + i
```

(e)
```
s = 0
for i = 1 to n*n do
    for j = 1 to i do
        s = s + i
```

(f)
```
s = 1
for i = 1 to n do
    for j = 1 to n do
        for k = 1 to n do
            s = s * i
```

4. Suppose you run two algorithms, `P` and `Q`, on many randomly generated data sets. `P` is an $O(n \log n)$-time algorithm and `Q` is an $O(n^2)$-time algorithm. After your experiments you find that if $n < 100$, `Q` actually runs faster, and only when $n \geq 100$, `P` is faster. Explain why this scenario is possible, including numerical examples.

The following is considered optional:

1. Given an array `A`, of n integers, describe a method to find the longest subarray of `A` such that all the numbers in that subarray are in sorted order. What is the running time of your algorithm?