

Lab Interlude: Algorithm Analysis

CSCI 2101 – Fall 2018

Due: Tuesday, October 2, 11:59 pm (**firm; no flex days may be used**)

Collaboration Policy: Level 1

Group Policy: Individual

In lieu of a regular programming lab, this week's exercise is to explore the fundamentals of algorithm analysis and Big-O notation. You may either type or handwrite (neatly!) your solutions. Upload your solutions electronically (e.g., by scanning) to Blackboard.

1. Show that 2^{n+1} is $O(2^n)$ by finding c and n_0 to satisfy the big-O requirement. Explain why your chosen values work.
2. Show that 2^{2n} is *not* $O(2^n)$ by showing that it is not possible to find c and n_0 to satisfy the big-O requirement. Note that $2^{2n} = (2^n)^2$.
3. Give a big-O characterization (and brief justification) of the running time, in terms of n , of each of the following five loops. Think in terms of the number of loop iterations that will be required. Note that the sum of the arithmetic sequence $1, 2, 3, \dots, k$ is $\frac{k}{2}(1+k)$.

Algorithm Loop1 (n):

```
s ← 0
for i ← 1 to n do
  s ← s + i
```

Algorithm Loop2 (n):

```
p ← 1
for i ← 1 to 2n do
  p ← p * i
```

Algorithm Loop3 (n):

```
p ← 1
for i ← 1 to n2 do
  p ← p * i
```

Algorithm Loop4 (n):

```
s ← 0
for i ← 1 to 2n do
  for j ← 1 to i do
    s ← s + i
```

Algorithm Loop5 (n):

```
s ← 0
for i ← 1 to n2 do
  for j ← 1 to i do
    s ← s + i
```

4. Given an `ArrayList` of initial size n , give a big-O characterization (and justification) of the running time of the following Java function, in terms of n :

```
public void doubleList(ArrayList<Integer> myList) {
    int size = myList.size();
    for (int i = 0; i < size; i++) {
        int pos = rand.nextInt(myList.size()); // rand is a Random object
        myList.add(pos, i);
    }
}
```

Would your answer change if the the fourth line instead read “`int pos = myList.size();`”? If so, what would be the new running time and why?

5. Explain whether the following statement is true or false:

“If choosing between an $O(n \log n)$ algorithm and an $O(n^2)$ algorithm to solve a problem on a specific input, it is always better to use the $O(n \log n)$ algorithm.”

Assume that the two algorithms use equivalent space and that the algorithms are already implemented (so you do not need to worry about the difficulty of implementation, for instance).