# CSC 232: Data Structures and Algorithms

**Lab7: Big-O Problems**

**Due: See BB**

*Submit your program file(s) through BB before midnight on the due date. Email your programs to me* ***as a last resort*** *if you experience problems with BB.*

**Exercises**

1. **The number of operations executed by algorithms A and B is and , respectively. Determine such that A is better than B for . Hint: simplify the equation as much as possible, then try different values to solve it.**

|  |  |  |
| --- | --- | --- |
| n | log n | n/4 |
| 13 | 3.70044 | 3.25 |
| 14 | 3.807355 | 3.5 |
| 15 | 3.906891 | 3.75 |
| 16 | 4 | 4 |
| 17 | 4.087463 | 4.25 |
| 18 | 4.169925 | 4.5 |
| 19 | 4.247928 | 4.75 |
| 20 | 4.321928 | 5 |

A is a better algorithm for .

A and B are equivalent at 16.

1. **Explain why the plot of the function is a straight line with slope *c* on a log scale.**

To plot on a logarithmic scale, we are plotting , or the points (c, y) where y is an exponential value based on c. Therefore for each step up in c, the y value will grow consistently, presenting as a straight line on a log scale.

1. **Order the following functions by asymptotic growth rate.**

(5) **210** (1)(8)

(3)(2)(9)

(6)(7) (4)

Note::

(a constant), , , , , , , ,

1. **Show that if is , then is for any constant .**

Given for some (by definition), then Since , then ac\*f(n). Let , then

1. **Show that if is and is , then the product is .**

Given and for some constants ,

then . Combing like terms, .

Since and are constants and both and are functions, then let and . By substitution, .

1. **An algorithm is used to find an element in an array with rows and columns. Algorithm iterates over the rows of , then calls the algorithm findIndex() on each row until is found or it has searched all rows of . What is the worst-case running time of:**
   1. **in terms of *n*?**
   2. **in terms of *N*, if *N* is the total size of ?**
   3. **Can you say that is a linear time algorithm? Justify.**

No, T cannot be considered a linear time algorithm as it has a for loop with a nested while loop, each of which in the worst case runs to n, thus T can be considered quadratic.

**Algorithm** findIndex(*x*, *R*):

***Input:*** An element *x* and an *n*-element array, *R*.

***Output:*** The index if *x* = *R*[*idx*]. Returns −1 if no element of *R* is equal to *x*.

*idx* ← 0

**while** *idx* < *n* **do**

**if** *x* = *R*[*idx*] **then**

**return** *idx*

**else**

*idx* ← *idx* + 1

**return** −1

1. **Show that if is a polynomial in , then is .**

Ifis in , the we can express .

Then (since there are m+1 terms and is the largest degree)

(log product rule – add two logs)

Since m and () are constants, then we can allow .

By substitution, then . Remembering that m is a constant, we can then say that .

1. **Show that is .**
2. **Show that *n* is .**

Since , then it can be said that

1. **An array contains *n* elements. Algorithm *L* calls Algorithm *M* on each element . Algorithm *M* runs in time for element . What is the worst-case running time of Algorithm *L*?**

- since L runs through all n elements, and M runs in , then the worst case running time of L is , but is a constant, so the running time is .