

**COM S 311 SPRING 2021
HOMEWORK 1**

Due: February 26 11:59 p.m.

Early submission: February 25, 11:59 p.m., (5% bonus)

Late Submission Due: February 27, 11:59 A.M. (25% penalty)

GUIDELINES

- For each problem, if you write the statement “I do not know how to solve this problem” (and nothing else), you will receive 20% credit for that problem. If you do write a solution, then your grade could be anywhere between 0% to 100%. To receive this 20% credit, you must explicitly state that you do not know how to solve the problem.
 - You are allowed to discuss with classmates, but you must work on the homework problems on your own. You should write the final solutions alone, without consulting anyone. Your writing should demonstrate that you understand the proofs completely.
 - When proofs are required, you should make them both clear and rigorous. Do not hand-waive.
 - Please submit your assignment via Canvas.
 - You **must** type your solutions. Please submit a PDF version.
 - Please make sure that the file you submit is not corrupted and that its size is reasonable (e.g., roughly at most 10-11 MB).
- If we cannot open your file, your homework will not be graded.*
- Any concerns about grading should be expressed within one week of returning the homework.

PROBLEMS

(1) Prove or disprove the following statements.

(a) $n^2 - 10n + 2 = O(n^2)$.

$$f(n) \leq c * g(n)$$

$$c > 0 \text{ for all } n > 1$$

$$c = 2, n = 1$$

$$1^2 - 10(1) + 2 \leq 2 * 1^2$$

$$-7 \leq 2$$

$$\text{Let } c = 2 \text{ for all } n > 1$$

$$1 - \frac{10}{n} + \frac{2}{n^2} < 2$$

$$\lim_{n \rightarrow \infty} \frac{f(n)}{g(n)} \rightarrow 1 - 0 - 0 < 2$$

(b) $2^{n^2} = O(2^{2n})$.

For $n > 1$, $c = 1$, $f(n) \leq c * g(n)$ is false

$$n = 1 \rightarrow 2^{2^2} = 16 = 2^{2*2} \text{ Which satisfies the statement, however}$$

$$n = 4 \rightarrow 2^{4^2} = 65536 > 256 = 2^{2*4}$$

2^{n^2} is not $O(2^{2*n})$ because there exists no value greater than $n = 2$ that satisfies the relationship when $c = 1$.

Disproven

(c) $n \log_2(n) = O(n \log_{10}(n))$.

$$n = 2, c = 1$$

$$2 \log_2(2) = 2 \log_{10}(2) * 1$$

$$1.0 \leq 0.3$$

$$\lim_{n \rightarrow \infty} \frac{f(n)}{g(n)} \rightarrow \lim_{n \rightarrow \infty} \frac{n \log_2(n)}{n \log_{10}(n)}$$

For any c there is no value for n that will make $n \log_{10}(n)$ greater than or equal to $n \log_2(n)$.

Disproven

(d) $n \log_2(n) = O(n)$.

For $n > 2$ let $c = 1$

$$\log_2(n) > 1$$

$$n \log_2(n) = n * c$$

$$\log_2(n) = c$$

As long as $\log_2(n) > c$ then $n \log_2(n) > O(n)$

\therefore Disproven for all $n > 2$

Disproven

(e) $n 2^n = O(2^{2n})$.

$$\text{For } n = 2$$

$$2 * 2^2 \leq 2^{2*2}$$

$$8 \leq 16$$

$$\lim_{n \rightarrow \infty} \frac{f(n)}{g(n)} = \frac{n * 2^n}{2^{2*n}} = 0$$

$g(n)$ grows faster than $f(n)$ and $f \in O(g)$

\therefore for all $n \geq 1$ and all $c \geq 1, n * 2^n = O(2^{2*n})$

(2) Consider the following algorithms (written in pseudocode):

```

1 Alg1(A)
  | Input: Array of integers of length  $n$ 
2  | constant number of operations
3  | for  $i = 1$  to  $n$  do
4  |   | constant number of operations
5  | for  $j = n$  to  $1$  do
6  |   | for  $k = j$  to  $1$  do
7  |     | constant number of operations

```

```

1 Runtimes per line
  | Input: Array of integers of length  $n$ 
2  |  $c$ 
3  |  $n + 1$ 
4  |  $c * n$ 
5  |  $n + 1$ 
6  |  $n^2 + 1$ 
7  |  $c * n^2$ 

```

$$(c + 1)n^2 + (c + 2)n + 3 = O(n^2)$$

```

1 Alg2(A)
  | Input: Array of integers of length  $n$ 
2   for  $i = n$  to 1 do
3   |   constant number of operations
4   |    $i = i/2$ 

```

$$O(\log(n)) \Leftarrow \begin{cases} \text{for } i \rightarrow n \text{ to } 1 \text{ do} \\ \text{constant number of operations} \\ i \rightarrow i/2 \end{cases}$$

Formally analyze the runtime of Alg1 and Alg2, and give the runtime of each in big oh notation. You must show your work - clearly and rigorously derive the runtime, do not just give the big oh bound.

- (3) Given an array A of integers, we say that an integer k is the *majority element* of A if k occurs in A strictly more than $A.length / 2$ times. Give a Divide and Conquer algorithm which, given an array A , determines if A contains a majority element. If A does contain a majority element k , it outputs k , otherwise, it outputs "null". Formally analyze the runtime of your algorithm, giving a recurrence relation and a big oh bound on the runtime of your algorithm. You **must** use a divide and conquer strategy. You do not have to prove correctness.

```

int majElem(int [] arr , int left , int right){
    if(left == right) return null
    if(left+1 == right) return arr[left]
    lCnt = 0
    rCnt = 0
    lMajElem = majElem(arr , left , (right + left - 1)/2+1)
    rMajElem = majElem(arr , (right + left - 1)/2+1, right)

    for (i = left ; i < right ; i++){
        if(a[i]==lMajElem) lCnt++
        if(lCnt > (right-left)/2) return lMajElem
    }

    for (i = left ; i < right ; i++){
        if(a[i]==rMajElem) rCnt++
        if(rCnt > (right-left)/2) return rMajElem
    }
}

```

Sum of 1 through n of $\frac{n}{2}$ is $\log(n)$, Then 2 linearly based checks occur (the for loops) equating to a run time of $2n$, which can be rebased as just n . Thus the runtime of this algo is $n * \log(n)$.

- (4) Using the Master Theorem, bound the runtime $T(n)$ of the following recurrence.

$$T(n) = 3T(n/2) + n^2, \text{ where } T(1) = O(1).$$

Case 3

$$\frac{n}{B} = \frac{n}{2}$$

$$F(n) = n^2$$

$$\log_2(3) = 1.585 < 2$$

$$F(n) < n^{(Log_B A + \varepsilon)}$$

ε is a constant

$$3T_{\frac{n}{2}} + n^2 \Rightarrow \Theta(n^2)$$

- $$T(n) = 2T(n/2) + n \log_2(n), \text{ where } T(1) = O(1).$$

The base tree

$$n \log_2(n) \rightarrow \left(\frac{n \log_2(n)}{2} \right) \rightarrow \left(\frac{\left(\frac{n \log_2(n)}{2} \right)}{2} \right) \rightarrow \dots$$

Each section sums to $n \log_2(n)$ nicely.

depth: $1 = \frac{n}{2^i} \rightarrow i = \log_2(n)$

$$n \log_2(n) + n \log_2(n) + \dots \log_2(n) \text{ times.}$$

$$\sum_{i=0}^{\log_2(n)} n \log_2(n)$$

$$= n \log_2(n) * \sum_{i=0}^{\log_2(n)} 1$$

$$= n \log_2(n) * (\log_2(n) + 1)$$

- $a = ???$

$$b = 4$$

CASE 1

If $a > b$ then $T(n) = \Theta(n^{\log_b(a)}) \rightarrow \log_4(a)$ must be less than $\log_2(3) = 1.589$

$$\log_4(a) = \log_2(3)$$

$$q = 4^{\log_2(3)} = 9$$

a must be less than $4^{\log_2(3)}$ but also an integer $\therefore a = 8$