

# CIS 675 (Fall 2018) Disclosure Sheet

**Name:** Wentan Bai  
**HW #** 1

☒ **Yes**    **No**    Did you consult with anyone on parts of this assignment, including other students, TAs, or the instructor?

☒ **Yes**    **No**    Did you consult an outside source (such as an Internet forum or a book other than the course textbook) on parts of this assignment?

If you answered **Yes** to one or more questions, please give the details here:

I consulted all questions with teaching assistant, Arash Sahebolamri, to confirm my understanding of all questions are correct. I also consult question 3, question 5, question 6 and extra question with another student, Wentian Bai. For question 3, question 5 and question 6, we discussed our ideas and I finish my algorithms independently. For the extra question, we discussed our understandings and I write my explanation independently.

For the question 1, I view some mathematical formulas online.  
<http://tutorial.math.lamar.edu/Classes/CalcI/LHospitalsRule.aspx>

By submitting this sheet through my Blackboard account, I assert that the information on this sheet is true.

## Question 1:

(a)  $f(n) = n$        $g(n) = n^2 - n$

$$L = \lim_{n \rightarrow \infty} \frac{f(n)}{g(n)} = \lim_{n \rightarrow \infty} \frac{n}{n^2 - n} = \lim_{n \rightarrow \infty} \frac{n}{n(n-1)} \\ = \lim_{n \rightarrow \infty} \frac{1}{n-1} = 1$$

If  $0 < L < \infty$ ,  $f(n) = \theta(g(n))$  and  $f(n) = O(g(n))$

(b)  $f(n) = \log 5n + 2$        $g(n) = \log 2n + 5$

In Big O notation,  $n$  is considered as infinite, so we should only compare  $\log 5n$  with  $\log 2n$ .

$$L = \lim_{n \rightarrow \infty} \frac{f(n)}{g(n)} = \lim_{n \rightarrow \infty} \frac{\log 5n + 2}{\log 2n + 5} = \lim_{n \rightarrow \infty} \frac{\log 5n}{\log 2n}$$

Based on L'Hopital Rule,

$$f_1(n) = \lim_{n \rightarrow \infty} \log 5n = \infty$$

$$g_1(n) = \lim_{n \rightarrow \infty} \log 2n = \infty.$$

$$\text{Therefore, } \lim_{n \rightarrow \infty} \frac{\log 5n}{\log 2n} = \lim_{n \rightarrow \infty} \frac{f'_1(n)}{g'_1(n)}$$

$$f'_1(n) = \frac{d}{dn} \log 5n = \frac{1}{n}$$

$$g'_1(n) = \frac{d}{dn} \log 2n = \frac{1}{n}$$

$$\text{Thence, } L = \lim_{n \rightarrow \infty} \frac{\log 5n}{\log 2n} = \lim_{n \rightarrow \infty} \frac{f'_1(n)}{g'_1(n)} = 1$$

Since,  $\lim_{n \rightarrow \infty} \frac{\log 2n + 5}{\log 5n + 2}$  can be proved as the same way and get the same answer. If  $0 < L < \infty$ ,  $f(n) = \theta(g(n))$  so  $f(n) = O(g(n))$  and  $g(n) = \theta(f(n))$  so  $g(n) = O(f(n))$ .

(c)  $f(n) = 10 \log n$        $g(n) = \log n^4$

$$L = \lim_{n \rightarrow \infty} \frac{f(n)}{g(n)} = \lim_{n \rightarrow \infty} \frac{10 \log n}{\log n^4} = \lim_{n \rightarrow \infty} \frac{10 \log n}{4 \log n}$$

$$L = \lim_{n \rightarrow \infty} \frac{\log n}{\log n} = 1$$

Since,  $\lim_{n \rightarrow \infty} \frac{g(n)}{f(n)}$  can be proved as the same way and get the same answer. Therefore,  $f(n) = O(g(n))$  and  $g(n) = O(f(n))$  since  $0 < L < \infty$ .

(d)  $f(n) = 100n + (\log n)^2$        $g(n) = 100n + \log n$

Firstly, I need to compare  $100n$  with  $\log n$ .

$$\lim_{n \rightarrow \infty} \frac{100n}{\log n} = 100 \cdot \lim_{n \rightarrow \infty} \frac{n}{\log n} = 100 \cdot \lim_{n \rightarrow \infty} \frac{1}{\frac{1}{n}} = 100 \cdot \infty = \infty$$

This means  $\log n = o(100n)$  and when  $n \rightarrow \infty$ , we should only consider about  $100n$  for  $g(n)$ .

Then, I need to compare  $100n$  with  $(\log n)^2$ .

$$\lim_{n \rightarrow \infty} \frac{(\log n)^2}{100n} = \frac{1}{100} \cdot \lim_{n \rightarrow \infty} \left( \frac{\log n}{\sqrt{n}} \right)^2 = \frac{1}{100} \cdot \lim_{n \rightarrow \infty} \left( \frac{\ln n}{\sqrt{n}} \right)^2 \\ = \frac{1}{100} \cdot \lim_{n \rightarrow \infty} \left( \frac{\frac{1}{n}}{\frac{1}{2\sqrt{n}}} \right)^2 = \frac{1}{100} \cdot \lim_{n \rightarrow \infty} \left( \frac{2}{\sqrt{n}} \right)^2 = 0$$

This means  $(\log n)^2 = o(100n)$  and when  $n \rightarrow \infty$ , we should only consider about  $100n$  for  $f(n)$ .

$$\text{Therefore, } L = \lim_{n \rightarrow \infty} \frac{f(n)}{g(n)} = \lim_{n \rightarrow \infty} \frac{100n}{100n} = 1$$

Since,  $\lim_{n \rightarrow \infty} \frac{g(n)}{f(n)}$  can be proved as the same way and get the same answer. Therefore,  $f(n) = O(g(n))$  and  $g(n) = O(f(n))$  since  $0 < L < \infty$ .

(e)  $f(n) = n!$        $g(n) = 2^n$

Based on Big-O Notation:  $g(n) = O(f(n))$  iff there exists a constant  $c$  and a fixed  $n_0$  such that for all  $n > n_0$ ,  $g(n) \leq cf(n)$ .

When  $n \geq 4$ ,  $g(n) < f(n)$ . However, whatever constant  $c$  is, there is not a fixed  $n_0$  to satisfy the Big-O notation. Therefore,  $g(n) = O(f(n))$ .

## Question 2:

1. Recurrence Relation:

$$T(n) = 3 \cdot T\left(\frac{n}{3}\right) + O(n^0).$$

2. Running time:

Based on Master Method,  $T(n) = a \cdot T\left(\frac{n}{b}\right) + O(n^d)$ ,  
where  $a = 3$ ,  $b = 3$  and  $d = 0$ .

The  $\log_3 3 = 1 > 0$ .

Therefore the result is  $O(n \log_3 3) = O(n)$ .

### Question 3:

At beginning, we can put a thin book or a normal book or a wide book. Then for each selection, we also need to choose to put a thin book or a normal book or a wide book. Therefore, the pseudo code is:

---

```
function WAYS(int  $N$ )  
  if  $N == 0$ :  
    return 0  
  else if  $N == 1$ :  
    return 1  
  else if  $N == 2$ :  
    return 2  
  else if  $N == 3$ :  
    return 4  
  else:  
    return ways( $N - 1$ ) + ways( $N - 2$ ) + ways( $N - 3$ )  
end function
```

---

The Recurrence Relation is:

$$T(0) = C_0;$$

$$T(1) = C_1;$$

$$T(2) = C_2;$$

$$T(3) = C_3$$

$$T(N) = T(N-1) + T(N-2) + T(N-3)$$

#### Question 4:

Claim: if  $f(x) = o(g(x))$ , then we must have that  $f(x) = O(g(x))$ .

Proof: By contradiction, suppose  $f(x) = o(g(x))$  and  $f(x) \neq O(g(x))$  for some  $f(x)$  and  $g(x)$ . Based on Big-O Notation,  $f(n) \neq O(g(n))$  means there does **not** exist a constant  $c$  and a fixed  $n_0$  such that for all  $n > n_0$ ,  $f(n) \leq cg(n)$ . However, in Little-o Notation,  $f(x) = o(g(x))$  means, for **every**  $c > 0$ , there exists a fixed  $n_0$  such that for all  $n > n_0$ ,  $f(n) \leq cg(n)$ . There is contradiction, so my supposition is incorrect. Therefore, original proposition is true.

## Question 5:

---

```
function FIND(A, left, right)
    while right == 1:
        left = right
        right = right * 2
    midpoint = (left + (right - left)/2)
    if A[midpoint] == 1 and A[midpoint + 1] == "error message":
        return midpoint
    else if A[midpoint] == 1:
        return find(A, midpoint, right)
    else:
        return find(A, left, midpoint)
end function
```

---

I use two pointers, *left* and *right*, to traverse the given array. Initialize *left* = 0 and *right* = 1. Then use a while loop to double *left* and *right* each time until *right* returns an error message. When *right* first returns an error message, it means *right* is out off the bounds of array *A*. And since *left* always equals to the previous *right*, the *n* will be in the range from *left* to *right*. Then we can use binary search to find *n*. Declare a variable *midpoint* which is the middle point of *left* and *right*. If the *midpoint* returns 1 and its next returns error message, the *midpoint* is the bounds and return it. If *midpoint* still returns 1 and its next is not error message, the bounds is in range from *midpoint* to *right*. Then recursively call function and limit the range from *midpoint* to *right*. Otherwise, the *midpoint* exceeds the bound of array and we recursively call function and limit the range from *left* to *midpoint*.

For the while loop, since it double each time and the first iteration takes the longest time, the worst running time of while loop is  $\log_2 n$ . For the recurrence part, I shrink half of current range each time, so using the Master Method, the *b* is 2 and *a* is 1. The loop part only works when we need to find bounds, so the *d* is 0. The Recursion Relation is  $T(n) = T(\frac{n}{2}) + O(n^0) + O(\log_2 n) \log_b a = 0 = d$ . Therefore, the running time is  $O(n^d \cdot \log n) = O(\log n)$ .

## Question 6:

---

```
function CHECK(L, K, start, sum result):  
    if result == False:  
        return result  
    if  $sum < (len(K))^2$ :  
        result = False  
        return result  
    for i in range(start, len(L)):  
        K.add(L[i])  
        result && check(L, K, i + 1, sum+L[i], result)  
    return result  
end function
```

---

For this question, I design a function `check()` to find and check all subsets  $K \subseteq L$ . At beginning, initialize K as an empty set,  $start = 0$ ,  $sum = 0$  and  $result = True$ . The variable `sum` is the sum of elements in current  $K$ . If there exists a subset  $K$  that  $sum < |K|^2$ , then the claim is false and returns the result as False. Otherwise, check other possible subsets. The  $L[start]$  is the first element of current  $K$ . Recursively find all possible subsets with first element  $L[start]$ . After that, move `start` to next and recursively find all possible subsets with first element  $L[start + 1]$ . For each recursion, let `result &&` with return of recursion. As long as there exists a result that is False, the final result will be False. Otherwise, the claim is True.

The running time is  $O(2^n)$  since there are  $2^n$  subsets which we need to check.



## Extra:

If  $f(x) = O(g(x))$ , it must be the case that  $\lim_{x \rightarrow \infty} f(x)/g(x) = c$ , for some finite  $c$ . Let  $L = \lim_{x \rightarrow \infty} f(x)/g(x)$ . If  $L = \infty$ , then  $g(x) = o(f(x))$  and  $f(x) = \omega(g(x))$ . Based on Little-o-notation, if  $g(x) = o(f(x))$ , then for every  $c > 0$ , there exists a fixed  $x_0$  such that for all  $x > x_0$ ,  $g(x) \leq cf(x)$ . Therefore, there does **not** exist a constant  $c$  and a fixed  $n_0$  such that for all  $n > n_0$ ,  $f(n) \leq cg(n)$ . So based on Big-O-notation,  $f(n) \neq O(g(n))$ . Hence, when  $\lim_{x \rightarrow \infty} f(x)/g(x) = c$ , for some finite  $c$ ,  $f(x) = O(g(x))$ .