# BIL 505 / Data Structures and Algorithms
## Homework #2

Q1) a) $\lim_{n \to \infty} \frac{f(n)}{g(n)} = \lim_{n\to\infty} \frac{(n^2-3n)^2}{5n^3+n} = \frac{\infty}{\infty}$

So we can use L'Hospital's rule.

$$\frac{f'(n)}{g'(n)} = \lim_{n\to\infty} \frac{2.\cancel{(n^2-3n)}.(2n-3)}{\cancel{15n^2+1}} = \infty$$

They degree are same

Therefore $f(n) \in \Omega(g(n))$.

b) $\lim_{n\to\infty} \frac{f(n)}{g(n)} = \lim_{n\to\infty} \frac{n^3}{\log_2 n^4} = \frac{\infty}{\infty}$

$$\xrightarrow{\text{L'hospital}} \lim_{n\to\infty} \frac{3n^2}{\frac{4\cancel{n^3}}{\cancel{n^4}\,n} . \log_2 e} = \lim_{n\to\infty} \frac{3n^2}{\frac{4}{n} . \log_2 e}$$

$$= \lim_{n\to\infty} \frac{3n^3 . \ln 2}{4} = \infty$$

Therefore $f(n) \in \Omega(g(n))$.

c) $\lim_{n\to\infty} \frac{f(n)}{g(n)} = \lim_{n\to\infty} \frac{5n . \log_2 4n}{n . \log_2 5^n} \xrightarrow{\text{L'hospital}}$

$$f'(n) = 5n . \frac{1}{n . \ln 2} + \log_2(4n) . 5 = \frac{5}{\ln 2} + 5(\log_2 n + 2)$$

$$= \frac{5}{\ln 2} + 5\log_2 n + 10$$

$g(n) = n . n . \log_2 5 = n^2 . \log_2 5$ $\quad g'(n) = 2n . \log_2 5 + 0$

$= 2n . \log_2 5$

$$\lim_{n\to\infty} \frac{f'(n)}{g'(n)} = \lim_{n\to\infty} \frac{\frac{5}{\ln 2} + 5\log_2 n + 10}{2n . \log_2 5} = \frac{\infty}{\infty} \xrightarrow{\text{L'hospital}}$$

$$\lim_{n \to \infty} \frac{\frac{1}{n} \cdot \overset{\text{constant}}{\cancel{\log_e \frac{5}{2}}}}{\underset{\text{constant}}{\cancel{2 . \log \frac{5}{2}}}} = \lim_{n \to \infty} \frac{1}{n} = 0$$

Therefore $f(n) \in o(g(n))$.

d) $\lim_{n \to \infty} \frac{f(n)}{g(n)} = \lim_{n \to \infty} \left(\frac{n}{10}\right)^n \longrightarrow$ 10 is constant

so we can say $\lim_{n \to \infty} n^n = \infty$

Therefore $f(n) \in \Omega(g(n))$.

e) $\lim_{n \to \infty} \frac{f(n)}{g(n)} = \frac{8n . \sqrt[5]{2n}}{n . \sqrt[3]{n}} = \frac{8 \cancel{n} . 2n^{1/5}}{\cancel{n} \quad n^{1/3}} = n^{1/5 - 1/3}$

$= \lim_{n \to \infty} n^{-2/5} = 0$

Therefore $f(n) \in O(g(n))$.

Q2) a)

```
static void methodA (String str_array[]) {
    for (int i=0 ; i < str_array.length ; i++)
        str_array[i] = "";
}
```

$\Big\} O(n)$, $\quad$ `str_array[i] = "";` $\rightarrow O(1)$

- The for loop with i as index will execute n times.
- The growth rate has an order of $\underline{\underline{n \text{ or } O(n)}}$ for the worst case

$\nearrow O(n.1)$



CamScanner ile tarandı

b) static void methodB (String str_array[]) {

So we can say like its inner loop $n^2$

```
for (int i=0 ; i< str_array.length ; i++)   -> Loop works n times
    methodA(str_array); -> also it works n times
for (int j=0 ; j< str_array.length ; j++)   } This loop also work n times
    sout(...);  -> O(1), its const operation
}
```

- The for loop i as index will execute n times. Then method inside of it will execute and this method work n times. For this part it takes $n^2$.
- The for loop j as index will execute n times.
- $n^2 + n$ times total execution
- The growth rate has an order of $n^2$ or $O(n^2)$ for the worst case

c) static void methodC (String str_array[]) {

$n \cdot n \cdot (n^2+n) = n^4 + n^3$

```
for (int i=0 ; i< str_array.length ; i++)  -> Loop works n times
    for (int j=0 ; str_array.length ; j++)  -> Loop works n times
        methodB(str_array);  -> This method takes n^2+n times
}
```

- Each loop will execute n times.
- methodB takes $n^2+n$ times.
- $n^4+n^3$ times total execution.
- The growth rate has an order of $n^4$ or $O(n^4)$ for the worst case



CamScanner ile tarandı

d)
```
static void methodD(String str_array[]) {
    for (int i=0; i< str_array.length; i++) {
        Sout(str_array[i]);
        str_array[i--] = "";
    }
}
```

- Normally, this loop works n times but there is a conflict error in this loop. When for loop increased number, the last operation i-- decreased the index value and it cause infinite loop. Thus $O(\infty)$ is the big-O notation.

e)
```
static void methodE(String str_array[]) {
    for (int i=0; i< str_array.length; i++)      -> Loop execute n times if array not contain " "
        if(str_array[i] == " ")   -> O(1)
            break;
}
```

$O(1.n) = O(n)$

- For the best case if first element of array equals " ", it only execute one time and loop will break.

- For the worst case there is no element equals " " and it execute n times.

- So we can say $O(n)$ is the big-O notation of worst case. Growth rate has an order of $O(n)$.

Q3) a) If array is sorted in ascending order, we know first element is the min number and last element is the maximum number.

Therefore we can directly set max and min number and take differences between them. Time complexity will be constant for this process.

Pseudo Code:

```
FUNCTION findMaxDiff (a)
    IF length(a) < 1
        return 0;
    SET max = a[n-1]
    SET min = a[0]
    return max - min;
```

O(4)

There are O(1) operations so its constant operation for my algorithm. Thus, big-O notation of this pseudo code is O(1).

b) If array is not sorted, max and min element in array need to find in linear. Then finded elements are setted and need differences to find max diff.

Pseudo code:

```
FUNCTION findMaxDiff (a)
    IF length(a) < 1
        return 0;
    SET max = a[0]
    SET min = a[0]
```

O(1)
constant time

```
FOR i from 1 to length(a)-1:
    IF a[i] > max:
        max = a[i]
    IF a[i] < min:
        min = a[i]
return max-min
```

→ This loop execute n-1 times

First of all first element of array assigned to max and min variable then in a for loop all elements are checked and variables assign if any if statement is true. Then difference between them returned. As a result, this execution works in linear time. So, big-O notation of this pseudo-code is O(n).

This approach find min and max number and take their differences can be used also for all of this question variances.

Mehmet Can OLGAY
235008003006