Ad = Yahya Kemal Kuyumcu No : 220104004406 Y.K.

CSE-222 HW-2

$Q_1) \rightarrow a)$ $f(n) = (n^2-3n)^2$ and $g(n) = 9n^3+n$

$$\lim_{n\to\infty} \frac{f(n)}{g(n)} = \lim_{n\to\infty} \frac{(n^2-3n)^2}{9n^3+n} \Rightarrow \lim_{n\to\infty} \frac{n^4-6n^3+9n^2}{9n^3+n} \Rightarrow \lim_{n\to\infty} \frac{n\cancel{n^3}(1-\frac{6}{n}+\frac{9}{n^2})}{\cancel{n^3}(9+\frac{1}{n^2})}$$

$$\Rightarrow \lim_{n\to\infty} \frac{n(1-\frac{6}{n}\to 0+\frac{9}{n^2}\to 0)}{9+\frac{1}{n^2}\to 0} \Rightarrow \infty \qquad \underline{f(n) = \Omega(g(n))}$$

$\rightarrow b)$ $f(n) = n^3$ and $g(n) = \log_2 n^4$

$$\lim_{n\to\infty} \frac{f(n)}{g(n)} = \lim_{n\to\infty} \frac{n^3}{\log_2 n^4} = \frac{\infty}{\infty} \rightarrow \text{L'Hôpital} \rightarrow \lim_{n\to\infty} \frac{3n^2}{4\frac{1}{n\ln(2)}} \Rightarrow \lim_{n\to\infty} \frac{3n^3\ln(2)\to\infty}{4} = \infty$$

$$= \underline{f(n) = \Omega(g(n))}$$

$\rightarrow c)$ $f(n) = 5n\log_2(4n)$ and $g(n) = n.\log_2(5^n)$

$$\lim_{n\to\infty} \frac{f(n)}{g(n)} = \lim_{n\to\infty} \frac{5n\log_2(4n)}{n\log_2(5^n)} = \frac{\infty}{\infty} \rightarrow \text{L'Hôpital} \Rightarrow 4n = 4 \; = \lim_{n\to\infty} \frac{5 \cdot \frac{1.4'}{4\ln(2)}}{(\log_2(5)+n\cdot\frac{1}{5\ln(2)})}$$

$$\rightarrow \frac{20}{4n\cancel{\ln(2)}} \cdot \frac{\cancel{5\ln(2)}}{5\ln(2)\log_2(5)+n} \Rightarrow \lim_{n\to\infty} \frac{25 \to 0}{n(5\ln(2)\log_2(5)+n)} = 0 \Rightarrow \underline{f(n) = O(g(n))}$$

$d)$ $f(n) = n^n$ and $g(n) = 10^n$

$$\lim_{n\to\infty} \frac{f(n)}{g(n)} = \lim_{n\to\infty} \frac{n^n}{10^n} \Rightarrow \lim_{n\to\infty} \left(\frac{n}{10}\right)^n \Rightarrow \lim_{n\to\infty} e^{n\ln(\frac{n}{10})} = \lim_{n\to\infty} e^{\infty\cdot\infty} = e^{\infty} = \infty$$

$$= \underline{f(n) = \Omega(g(n))}$$

$e)$ $f(n) = 8n\sqrt[9]{2n}$ and $g(n) = n.\sqrt[3]{n}$

$$\lim_{n\to\infty} \frac{f(n)}{g(n)} \Rightarrow \lim_{n\to\infty} \frac{8\cancel{n}(2n)^{1/9}}{\cancel{n}.(n)^{1/3}} \Rightarrow \lim_{n\to\infty} \frac{8.2^{1/9}}{n^{2/9}} \to 0 = 0$$

$$\underline{f(n) = O(g(n))}$$

Q2)

a) Lets say str-array has n elements this method sets every element in this str-array to an empty string so this method has a time complexity of O(n) in every case.

b)
```
method B (str-array[])
    for (int i=0; i<str-array.length i++)
        method A (str_array)
    for( int J=0; J<str-array.length ;J++)
        Print ("....")
```

1 This line gets called n times because of first loop. and we know that method A has time complexity of O(n) this makes this lines time complexity n^2 itself.

2 This print statement has constant time complexity (1) and gets called n times so this line has time complexity of n.

3 final. So because of first loop which calls method A n times this whole method has time complexity of $O(n^2)$.

c)
```
method C(string[] str-array)
    for (int i=0; i<str-array.length; i++)
        for (int J=0; J<str-array.length; J++)
            methodB(str-array)
```

This loop gets called n times because of outer loop.

This method gets called n^2 times because of outer nested loops. And we know that method B has time complexity of n^2. This makes whole methodC time complexity of $O(n^4)$.

d)
```
methodD(string[] str-array)
    for(i=0; i<str-array.length; i++)
        Print(arr[i]);
        arr[i--]="";
```

Because of this line i gets decremented after being incremented by for loop, so this method has an infinite loop. Time complexity can't be calculated.

e)
```
method E(string[] str-array)
    for(int i=0; i<str-array.length; i++)
        if(str_array[i]=="")
            break;
```

In the best case scenario if searched value is the first element of str-array this method takes constant time. But we look for worst case scenario and the worst case scenario happens when the searched value is happen to be last element of the array. In that case this method has linear time complexity O(n).

Q3)

a) For ascending array:

```
function methodA(array)
        return array[n-1] - array[0];
```

<u>Time Complexity</u>

this method has constant time complexity O(1). Because running time of algorithm doesn't change based on input size.

<u>Exploration</u>

Because array is sorted in ascending order, for finding max difference we substract last and first element of the array this operation has constant time complexity.

b) for non sorted array

```
FUNCTION methodB (int array)
    int max = -infinity
    int min = +infinity
    for index < array.length increase index
        if max < array[index] then
            max = array[index]
        if min > array[index] then
            min = array[index]
    end for
    RETURN max-min;
```

<u>Time Complexity</u>

this method has linear time complexity O(n) because of for loop.

<u>exploration</u>: The array is not sorted because of that we have to check every pair and calculate the max difference, first algoritm comes to mind is checking everyother element in array for element n in array using nested for loops, but this algoritm has time complexity of $O(n^2)$.
Instead of that algoritm we use temporary variables max and min to store max and min values in the array with this way we dont have to checkh pairs in array and we only go through the array one time.
With this algoritm we reduce our time complexity to O(n).