# CSE 222 – HW2

Emre Oytun
2001040040 99

## 1)

a) $\lim_{n\to\infty} \frac{f(n)}{g(n)} = \lim_{n\to\infty} \frac{n^2+7n}{n^3+7} = \lim_{n\to\infty} \frac{n^3(1/n + 7/n^2)}{n^3(1+7/n^3)} = 0$

(0 over 1 + 0; 1 over 1 + 0)

* Since $\lim_{n\to\infty} \frac{f(n)}{g(n)} = 0$, $f(n) = O(g(n))$.

b) $\lim_{n\to\infty} \frac{f(n)}{g(n)} = \lim_{n\to\infty} \frac{12n + \log_2 n^2}{n^2+6n} = \lim_{n\to\infty} \frac{n^2\left(\frac{12}{n} + \frac{\log_2 n^2}{n}\right)}{n^2\left(1+\frac{6}{n}\right)}$

(1 over 0 + 0; 1 over 1 + 0)

$= \lim_{n\to\infty} \frac{\log_2 n^2}{n} \overset{L'}{=} \lim_{n\to\infty} \frac{\frac{2x}{n^x \cdot \ln 2}}{1} = \lim_{n\to\infty} \frac{2}{n \cdot \ln 2} \to \infty$

$(\infty/\infty)$

$= 0$

* Since $\lim_{n\to\infty} \frac{f(n)}{g(n)} = 0$, $f(n) = O(g(n))$.

c) $\lim_{n\to\infty} \frac{n \cdot \log_2(3n)}{n + \log_2(8n^3)} \overset{L'}{=} \lim_{n\to\infty} \frac{\log_2(3n) + \frac{3}{(3n)\ln 2} \cdot n}{1 + \frac{24n^2}{(8n^3)\cdot \ln 2}}$

$(\infty/\infty)$

$= \lim_{n\to\infty} \frac{\log_2(3n) + \frac{1}{\ln 2}}{1 + \frac{3}{n \cdot \ln 2}} = \lim_{n\to\infty} \frac{(\log_2(3n)\cdot \ln 2 + 1)\cdot n}{n\cdot \ln 2 + 3}$

$= \lim_{n\to\infty} \frac{\frac{\ln(3n)}{\ln 2} \cdot \ln 2 \cdot n + n}{n\cdot \ln 2 + 3} = \lim_{n\to\infty} \frac{\ln(3n)\cdot n + n}{n\cdot \ln 2 + 3}$

$= \lim_{n\to\infty} \frac{n\cdot(\ln(3n)+1)}{n\cdot(\ln 2 + 3/n)} = \frac{\infty + 1}{\ln 2 + 0} = \infty$

(3/n → 0)

c) Continues ...

* Since $\lim_{n\to\infty} f(n)/g(n) = \infty$, $f(n) = \Omega(g(n))$.

d) $\lim_{n\to\infty} \frac{n^n + 5n}{3 \cdot 2^n} = \lim_{n\to\infty} \frac{1}{3}\left(\frac{n}{2}\right)^n + \lim_{n\to\infty} \frac{1}{3}\left(\frac{5n}{2^n}\right)$

$\overset{L'}{=} \infty + \lim_{n\to\infty} \frac{5}{2^n \ln(2)} = \infty + 0$

$= \infty$ //

* Since $\lim_{n\to\infty} f(n)/g(n) = \infty$, $f(n) = \Omega(g(n))$.

e) $\lim_{n\to\infty} \frac{(2n)^{1/3}}{(3n)^{1/2}} = \lim_{n\to\infty} \frac{2^{1/3} \cdot n^{1/3}}{3^{1/2} \cdot n^{1/2}} = \lim_{n\to\infty} \frac{\sqrt[3]{2}}{\sqrt{3}} \cdot n^{-1/6}$

$= 0$

* Since $\lim_{n\to\infty} f(n)/g(n) = 0$, $f(n) = O(g(n))$

---

2)

a) Worst-time complexity is $O(n)$.

Since it iterates through the elements in the names array once and each time it only prints them. So, there are total "$n * 1 = n$" processes.

b) Worst-time complexity is $O(n2)$.

Since outer loop iterates n times (myArray's length) outer loop takes $O(N)$ time, and in every cycle it

calls methodA which has $O(N)$ time complexity with the same array. Therefore; there are total "$1+n*n = 1+n^2$" processes and time-complexity is $O(N^2)$.

c) Worst-time complexity is not-defined since the loop does not iterate at all if array's length is 0, but it iterates infinite times otherwise.
An algorithm should terminate to analyze complexity.

d) Worst-time complexity is $O(n)$.
Since it iterates through all elements if there are no numbers bigger or equal than 4 until there are no elements in the array. In this case, the loop iterates "n" times, and in every iteration it prints the element in $O(1)$ time. At the beginning it makes assignment in $O(1)$ time. There are total "$1 + n*1 = 1+n$" processes.

---

3) In withoutLoop method, printing takes $O(1)$ time and there are number of printing statements equal to n. Therefore, it has $O(n)$ time complexity.

In withLoop method, it iterates through all the elements so it takes $O(n)$ time and each time it only prints elements. Therefore, it has $O(n)$ time complexity.

So, their time complexities are equal, but withLoop method is better in term of readability and its easy to write.

---

4) We cannot solve this problem in constant time since we don't know if the array is sorted, we should treat it as it's unsorted. It takes O(n) time to search through all elements. Therefore, it's not possible.

---

5) Input: array A with n elements ($n \in \mathbb{Z}^+$),
array B with m elements ($m \in \mathbb{Z}^+$).

Output: The minimum product of two elements coming from array A and B.

```
Program Start
Initialize variable min_a = A[0]
Initialize variable max_a = A[0]
Initialize variable min_b = B[0]
Initialize variable max_b = B[0]

FOR each element in array A
    IF the element is greater than max_a THEN
        SET max_a to the element.
    ELSE IF the element is smaller than min_a THEN
        SET min_a to the element.
    END IF
END FOR

FOR each element in array B
    IF the element is greater than max_b THEN
        SET max_b to the element.
    ELSE IF the element is smaller than min_b THEN
        SET min_b to the element.
    END IF
END FOR

Initialize variable product1 = min_a * min_b
Initialize variable product2 = min_a * max_b
Initialize variable product3 = max_a * min_b
Initialize variable product4 = max_a * max_b
```

Initialize variable res1 = minimum value of product 1 and product 2

Initialize variable res2 = minimum value of product 3 and product 4

Return the minimum value of res1 and res2.

PROGRAM END

=> The program whose pseudo-code i write above first finds minimum and maximum values of arrays A and B. Secondly, finds the 4 possible products using these elements. Finally, it returns the minimum value of these products.

=> Complexity Analysis:

* In first loop, it iterates through each elements in array A so it works N times, in each time it checks 2 conditions, and makes 1 assignment in worst-case. So, total number of processes in worst-case is "N * 3 = 3N" which makes the loop's complexity O(N).

* Second loop does exactly the same thing for array B, so it's complexity O(m) having "3M" processes.

* It has 11 more processes to initialize variables and return.

* Total process number T(n) = 4n + 4m + 11; so time complexity is O(n+m).