CSE 222 Homework 2

Oğuz MUTLU
1801042624

1-a)

$$\lim_{n\to\infty} \frac{n^2+7n}{n^3+7} \to \lim_{n\to\infty} \frac{n(n+7)}{n^3+7} \to \lim_{n\to\infty} \left(\frac{n(n+7)}{n^3(1+\frac{7}{n^3})}\right)$$

$$= \lim_{n\to\infty} \left(\frac{n+7}{n^2(1+\frac{7}{n^3})}\right) = \lim_{n\to\infty} \left(\frac{n^{-2}(n+7)}{1+\frac{7}{n^3}}\right) \quad (1+\frac{7}{n^3}) = 1$$

$$= \lim_{n\to\infty} \left(\frac{n^{-2}(n+7)}{1}\right) = \underbrace{\lim_{n\to\infty} \frac{1}{n}}_{0} + \underbrace{\lim_{n\to\infty} \frac{7}{n^2}}_{0} = \lim_{n\to\infty} \frac{n^2+7n}{n^3+7} = 0$$

$\star \lim_{N\to\infty} \frac{f(N)}{g(N)}$ { if 0 then $f(N)$ is in $O(g(N))$ ✓
if $L = c$ then $f(N)$ is in $\Theta(g(N))$
if ∞ then $f(N)$ is in $\Omega(g(N))$

$f(N) = O(g(N))$

1-b)

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

$$= \lim_{n\to\infty} \frac{n^{-1}\left(12+\frac{\log_2 n^2}{n}\right)}{1+\frac{6}{n}} = \underbrace{\lim_{n\to\infty} \frac{12}{n}}_{0} + \frac{\log_2 n^2}{n^2}$$

$$= \lim_{n\to\infty} \frac{12}{n} = 0 \qquad \lim_{n\to\infty} \frac{\log_2 n^2}{n^2}$$ => we use l'hospital rule

$$\frac{\frac{d}{dn}(\log_2 n^2)}{\frac{d}{dn}(n^2)} \Rightarrow \frac{\frac{\cancel{2n}}{\ln 2 . n^2}}{\cancel{2n}} = \frac{1}{\ln 2 . n^2} \to 0$$

$$\lim_{n\to\infty} \left(\frac{12n+\log_2 n^2}{n^2+6n}\right) = 0$$ so $f(N)$ is in $O(g(N))$

$f(N) \in O(g(N))$

1-c) $\lim_{n\to\infty}\left(\frac{n.\log_2 3n}{n+\log_2(8n^3)}\right) = \frac{n.\log_2 3n}{n\left(\frac{\log_2 8+\log_2 n^3}{n}+1\right)}$

$= \frac{n.\log_2 3n}{n\left(\frac{3}{n}+\frac{3\log_2 n}{n}+1\right)} = \frac{\log_2 3+\log_2 n}{\left(\frac{3}{n}+\frac{3\log_2 n}{n}+1\right)}$

$= \frac{\lim_{n\to\infty}(\log_2 3+\log_2 n)}{\lim_{n\to\infty}\left(\frac{3}{n}+\frac{3\log_2 n}{n}+1\right)} \rightarrow \frac{\log_2 3+\infty}{\underbrace{\lim_{n\to\infty}\frac{3}{n}}_{0}+\lim_{n\to\infty}\frac{3\log_2 n}{n}+\underbrace{1}_{1}}$

$\lim_{n\to\infty}\frac{3\log_2 n}{n} \Rightarrow$ l'hospital $= \frac{3.\frac{1}{n.\ln 2}}{1} \rightarrow \lim_{n\to\infty}\frac{3}{n.\ln 2} = 0$

$= \frac{\log_2 3+\infty}{1+0+0} = \frac{\infty}{1} \Rightarrow \lim_{n\to\infty}\left(\frac{n.\log_2 3n}{n+\log_2(8n^3)}\right) = \infty$ $\quad f(N) = \Omega(g(N))$

1-d) $\lim_{n\to\infty}\left(\frac{n^n+5n}{3.2^n}\right) \Rightarrow \frac{1}{3}\cdot\left(\lim_{n\to\infty}\frac{n^n+5n}{2^n}\right)$

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

$= \lim_{n\to\infty}\left(\frac{5n}{2^n}\right) \Rightarrow \frac{\frac{d}{dx}(5n)}{\frac{d}{dx}(2^n)} = \frac{5}{2^n.\ln 2} \rightarrow \lim_{n\to\infty}\frac{5}{2^n\ln 2} \rightarrow 0$

$= \frac{1}{3}(\infty+0) = \infty$ $\quad f(N) = \Omega(g(N))$

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

$\lim_{n\to\infty} n^{\left(-\frac{1}{6}\right)} \cdot \sqrt[6]{\frac{4}{27}} \rightarrow \lim_{n\to\infty} \frac{1}{n^{\frac{1}{6}}}$ (Constant)

0

$\lim_{n\to\infty} \frac{\sqrt[3]{2n}}{\sqrt{3n}} \Rightarrow 0 \quad . \quad f(N) = O(g(N))$

2.a)

```
static void methodA (String names [ ]) {
    for (int i = 0 ; i < names.length ; i++)
        System.out.println (names [i]);
}
```

→ Thus for loop loops as the length of names array complexity $(O(n))$

→ System.out.println (names[i]) → reaches specific part in $O(1)$ time

= worst case of this function is $O(n)$

2.b)

```
static void methodB () {
    String myArray = new String [ ] {"CSE 222",
"CSE 505", "L+W2"};
    for (int i = 0; i < myArray.length ; i++)
        methodA (myArray);   → O(n)
}
```

→ lenght of myArray → n (3) for loop loops lenght of my Array inside the loop methodA's complexity $O(n)$ so $O(n*n) = \underline{O(n^2)}$

2-c) Static void methodC (int numbers [ ]) {

```
    int i = 0;
    while (i < numbers.length)
        System.out.println(numbers[i]);
}
```

-> If we increment i by one then this methods worst-case have to be O(n) but we do not increment "i" so loop will run indefinately so function will never terminate.

2-d) static void methodD (int numbers [ ]) {

```
    int i = 0;
    while (numbers [i] < 4)
        System.out.println( numbers [i++ ]);
}
```

-> Worst case of this function when all elements are less then 4 the loop iterate through all elements which length is "n" O(n). There is no break statement so if all elements in the array are less than 4 function will never terminate or may throw exception in some point.

3.) I think using for loop instead of writing "n" times is not much change on time. But in assembly side we overhood to our computer with loop instruction. If we just write with many times we do not have to use for loop instruction in assembly by doing this we gain may be ~~two~~ three clock cycle of CPU it is negligible for computers.

In conclusion printing one by one is better for complexity but in today's computer speed if we thought, this speed is negligible.

4) No we cannot solve this problem in constant time. We can found specific integer in constant time when that integer is the first element of the array but we do not have any information about array's elements, sorted or not! in worst case we have to search all "n" elements of the array so our time complexity on worst case O(n) we cannot solve this problem in linear time.

CamScanner ile tarandı

5-

```
int findmin (int[] A, int[] B){

  int findminA = A[0];
  int findminB = B[0];

  for i in A.length {
     if A[i] < findminA {     = O(1)
        findminA = A[i]       = O(1)      } O(n)
     }
  }
  for i in B.length {
     if B[i] < findminB {     = O(1)
        findminB = B[i];      = O(1)      } O(m)
     }
  }

  return findminA * findminB;  = O(1)
```

Total complexity = O(n+m)

PSEUDO CODE

```
public static int findmin (int[] A,
                           int[] B )){

  int findMinA = A[0];
  int findMinB = B[0];

  for(int i=0; i<A.length ; i++){
     if(A[i] < findMinA)
        findMinA = A[i];
  }

  for (int j=0; j<B.length; j++){
     if(B[j] < findMinB){
        findMinB = B[i];
  }

  return findminA * findMinB;
}
```

Java code.

- We can find minimum element of an array in linear time. First of all we assign the first element of the array to the integer variable that we defined at the beginning. Then starting of the first element of the array, the indexes of array are compared. If the element in the new index is smaller than the element (variable) we defined as min at the beginning, this index will be our new minimum value and this process is continued until we reach end of the array. After performing this operation for both arrays the product of these two result is our main result. Because product of the minimum elements in the arrays always gives the smallest value. Since we iterate our array for length of array to find min value. Our best case and worst case is O(n+m) which "n" and "m" are the length of arrays.