Tarih:......./......../201

1

a) $\lim\limits_{n\to\infty} \frac{f(n)}{g(n)}\ \frac{n^2+7n}{n^3+7} = \frac{n^2(1+7/n)}{n^2(n+7/n^2)} = \frac{1+\overbrace{7/n}^{0}}{n+\underbrace{7/n^2}_{0}}$

$\lim\limits_{n\to\infty} \frac{1+7/n}{n+7/n^2} = \frac{1}{\infty} = 0$ so $f(n) = O(g(n))$

b) $\lim\limits_{n\to\infty} \frac{12n+\log_2 n^2}{n^2+6n}$ } Derivative both sides

$= \frac{12 + \frac{2n}{n^2 \ln 2}}{2n+6} = \frac{12n^2\ln 2 + 2n}{(2n+6)\cdot(n^2\ln 2)}$ $\quad 2n^3\ln 2 + 6$

$= \frac{n(12n\ln 2 + 2)}{n(2n^2\ln^2 + 6n\ln 2)} = \frac{12n\ln 2 + 2}{2n^2\ln^2 + 6n\ln^2}$

$= \frac{\cancel{n}(12\ln^2 + 2/n)}{\cancel{n}(2n\ln 2 + 6\ln 2)} =$

$= \lim\limits_{n\to\infty} \frac{12\ln 2 + 2/n}{2n\ln 2 + 6\ln 2} = \frac{1}{\infty} = 0$

so $f(n) = O(g(n))$

!

c) $\lim\limits_{n\to\infty} \dfrac{n\log_2 3n}{n+\log_2(8n^3)}$ => Derivative both side

$$\frac{1.\log_2 3n + n.\frac{3}{3n\ln 2}}{1+\frac{24n^2}{8n^3\ln 2}} = \frac{\log_2 3n + 1/\ln 2}{1+\frac{3}{n\ln 2}}$$

$$= \frac{\frac{\log_2 3n.\ln 2+1}{\ln 2}}{\frac{n\ln 2+3}{n\ln 2}} = \frac{(\log_2 3n\ln 2+1)}{\cancel{\ln 2}}.\frac{n\cancel{\ln 2}}{n\ln 2+3} = \frac{n(\log_2 3n\ln 2+1)}{n(\ln 2+3/n)}$$

$$\Rightarrow \lim_{n\to\infty} \frac{\log_2(3n).\ln 2+1}{\ln 2+\underbrace{3/n}_{0}} = \infty$$

$\boxed{\text{So } f(n) = \Omega(g(n))}$

d) $\lim\limits_{n\to\infty} \dfrac{n^n+5n}{3.2^n}$ => Growth rate of $n^n$ is bigger than $2^n$. So result is $\infty$

So $f(n) = \Omega(g(n))$

other $g(n) = o(f(n))$

e) $\lim\limits_{n\to\infty} \dfrac{\sqrt[3]{2n}}{\sqrt{3n}} = \dfrac{2^{\frac13}.n^{\frac13}}{3^{1/2}\,n^{\frac12}}$ => Derivative both side

$$= \frac{\frac{1}{3}.2^{1/3}.n^{-2/3}}{\frac{1}{2}.3^{1/2}.n^{-1/2}} = n^{-2/3+1/2} = \frac{1}{n^6} = 0$$

$f(n) = O(g(n))$

Tarih:......./......../201

2)

a) Sysout $\Rightarrow T_1(n) = \Theta(1)$
for loop $\Rightarrow T_2(n) = \Theta(n)$

$$T(n) = T_1(n) * T_2(n) = \Theta(n)$$

b) string assignment $= \Theta(1) \Rightarrow T_1(n)$

for loop $= \Theta(n) \Rightarrow T_2(n)$

each calling of method A $= \Theta(n) \Rightarrow T_3(n)$

$$T(n) = T_1(n) + T_2(n) * T_3(n)$$
$$= \Theta(1) + \Theta(n.n)$$
$$= \Theta(n^2)$$

c) if there is not i increment this is finite loop.
I assume that end of the while loop i++;

assign of i $= \Theta(1) \Rightarrow T_1(n)$

While control $= \Theta(n+1) = \Theta(n) = T_2(n)$

Sys.out $= \Theta(1) \Rightarrow T_3(n)$

i++; $= \Theta(1) = T_4(n)$

$$T(n) = T_1(n) + T_2(n).T_3(n) + T_2(n)T_4(n)$$
$$= \Theta(1) + \Theta(n).\Theta(1) + \Theta(n).\Theta(1)$$

$$T_n = \Theta(n)$$

d) assignment of $i = \Theta(1) = T_1(n)$

while control → Best case $T_b(n) = \Theta(1)$

→ Worst case $T_w(n) = \Theta(n)$

Sys.out $= \Theta(1) \Rightarrow T_2(n)$

$i{+}{+};\ = \Theta(1) = T_3(n)$

$T_{best} = \Theta(1)$

$$T_{worst} = \Theta(1) + \Theta(n).(\Theta(1) + \Theta(1))$$
$$= \Theta(n)$$

3) First method

Assigment of $i = \Theta(1)$

Sys.out $= \Theta(1)$
$i{+}{+} = \Theta(1)$ } length of array times, which is $n$

$$T(n) = \Theta(1) + \Theta(2n) = \Theta(2n+1) = \Theta(n)$$

Second method

Assigment of $i = \Theta(1)$

length check $= \Theta(n+1)$

increment of $i = \Theta(n)$

Sys.out $= \Theta(1)$

$$T(n) = \Theta(1) + \Theta(n+1) + \Theta(n) + \Theta(n).\Theta(1)$$
$$= \Theta(3n+2) = \Theta(n)$$

Continue of 3.

Both method's complexity are same $\Theta(n)$
(n is length of array)

Second method's number of operation much more than first one.
So, first is more adventageous.

4) It can not be solved in constant time. In worst case, each element should be checked.

Since we don't know whether it is sequential or not, we should use linear search.

```
int i=0;
while (i<n)
{
    if( array[i] == spesific_integer)
        return true;
    i++;
}
return false.
```

$T_{worst} = \Theta(n)$

$T_{best} = \Theta(1)$

Tarih:......./......../201

To find min value, we should consider also negative numbers. So we should find min and max values of arrays.

```
int minOfA = A[0], minOfB = B[0], maxOfA = A[0], maxOfB = B[0];
int minOfProduct = minOfA * minOfB; i, j;

for(i=0; i<n; i++)
{
        if(A[i] < minOfA)            } Θ(1)
            minOfA = A[i];                          } Θ(n)
        if(A[i] > maxOfA)            } Θ(1)
            maxOfA = A[i];
}

for(j=0; j<m; j++)
{
        if(B[j] < minOfB)            } Θ(1)
            minOfB = B[j];                          } Θ(m)
        if(B[j] > maxOfB)            } Θ(1)
            maxOfB = B[j];
}

if(minOfA * minOfB < minOfProduct)
        minOfProduct = minOfA * minOfB;
if(minOfA * maxOfB < minOfProduct)                   } Θ(1)
        minOfProduct = minOfA * maxOfB;
if(maxOfA * minOfB < minOfProduct)
        minOfProduct = maxOfA * minOfB;
```

$$T(n,m) = \Theta(n).(\Theta(1)+\Theta(1)) + \Theta(m).(\Theta(1)+\Theta(1)) + \Theta(1)$$

$$T = \max(\Theta(n), \Theta(m))$$