SÜMEYYE DALGA

22010 4004091

CSE 222 HW2:

1.a) $\lim_{n\to\infty} \frac{2^{2n}+1}{2^{2n}} = \lim_{n\to\infty} \frac{2^{2n}}{2^{2n}} + \lim_{n\to\infty} \frac{1}{2^{n}}$

$= \lim_{n\to\infty} 1 + \lim_{n\to\infty} \frac{1}{\infty}$

$= 1 + 0 = 1 \Rightarrow$ constant

so $f(n) = \Theta(2^{2n})$

$\boxed{\text{It is false}}$

1.b) $\lim_{n\to\infty} \frac{n^3-4n^2+7}{2^n} = 0$, because $n^3 < 2^n$.

so $f(n) = O(2^n)$

$\boxed{\text{It is false}}$

1.c) $\lim_{n\to\infty} \frac{n^3(1+\sqrt{n})}{n^3 \log n} = \lim_{n\to\infty} \frac{1+\sqrt{n}}{\log n} = \infty$, because $\log n < \sqrt{n}$

so $f(n) = \Omega(n^3 \log n)$

$\boxed{\text{It is false}}$

1.d) $\lim_{n\to\infty} \frac{28n}{7n^2} = \lim_{n\to\infty} \frac{4}{n} = 0$

so $f(n) = O(7n^2)$

$\boxed{\text{It is true}}$

1.e) $\lim_{n\to\infty} \frac{n+\log n+21}{7n^2} = \underbrace{\lim_{n\to\infty} \frac{1}{7n}} + \underbrace{\lim_{n\to\infty} \frac{\log n}{7n^2}} + \underbrace{\lim_{n\to\infty} \frac{21}{7n^2}}$

$= 0 + 0 + 0$

$= 0$ so $f(n) = O(7n^2)$

$\boxed{\text{It is true}}$

SÜMEYYE DALGA
220104004091

1.f) $\lim_{n\to\infty} \frac{n^2+9n-13}{n^2} = \lim_{n\to\infty} 1 + \lim_{n\to\infty} \frac{9}{n} - \lim_{n\to\infty} \frac{13}{n^2}$

$= 1 + 0 - 0$

$= 1 \Rightarrow$ constant so

$f(n) = \Theta(n^2)$

$\boxed{\text{It is true}}$

2.)

1. Logarithmic functions are the slowest growing functions. $\Rightarrow \log n$
2. Linear functions grow faster than logarithmic functions but slower than polynomials.
$\Rightarrow \log n < 7n <$
3. n^4 grows faster than n^2
$\Rightarrow \log n < 7n < 2n^2 < 3n^4$
4. Exponential functions grow faster than polynomials.
$\Rightarrow \log n < 7n < 2n^2 < 3n^4 < 3^n$
5. Factorial functions grow faster than exponentials.
$\Rightarrow \log n < 7n < 2n^2 < 3n^4 < 3^n < n!$

This is the order: $\log n < 7n < 2n^2 < 3n^4 < 3^n < n!$

SÜMEYYE DALGA
220104004091

3.b) static void someFunction (int a, int b){

```
int sum =0;
for ( int i =0 ; i<a; i++)   // runs a times
    for (int j=0 ; j<b ; j++) // runs b times
        sum += i+b;}
```

$\sum_{i=0}^{a-1} \sum_{j=0}^{b-1} 1 \Rightarrow a.b$,, $O(a.b) \Rightarrow$ worst case

3.b) static void another Function (int a) {

```
int sum=0
for (int i=0 ; i<a ; i++)
    sum += i;
    i= i*2; }
```

$i = 0, 1, 3, 7, 15, \ldots$ until $i \geq a$

$2^k - 1 \geq a$

$2^k \geq a+1$

$k \geq \log_2 a+1$

$\sum_{k=0}^{\log_2 a+1} 1 \Rightarrow \log_2 a+1$

$O(\log_2 a+1) = O(\log_2 a) = O(\log a) \Rightarrow$ worst case

SÜMEYYE DALGA

220104004081

3.c) static void differentFunction (int n){

```
for (int i=0; i<n; i++) //Runs n times
    for (int j=0; j<n; j++) //Runs n times
        for(int k=0; k<n; k++) //Runs n times
            System.out.println ("Hello, world!") }
```

$$\sum_{i=0}^{n-1}\sum_{j=0}^{n-1}\sum_{k=0}^{n-1} 1 = n.n.n = n^3$$

$O(n^3)$ ⇒ worst case.

4) We can use the "binary search" method. First, we go to the 50th floor, which is the middle of the building. If the toy does not break, we go up. If it breaks, we go down. Each time, we choose the middle floor and repeat the process. We continue until there are no more floors to check.