CSE 222 Homework 2

Name : Tuana Melisa Aksoi

No: 230104004903

Q1)

a) $2^{2n}+1 = O(2^n)$

Step 1:

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

Step 2:

- 2^n goes to ∞.
- $\frac{1}{2^n}$ goes to 0.

So the limit goes to ∞. Which means that statement is false.

$\therefore\ 2^{2n}+1 \neq O(2^n)$

b) $n^3-4n^2+7 = \Theta(2^n)$

Step 1:

$$\lim_{n\to\infty} \frac{n^3-4n^2+7}{2^n} = 0$$

Exponential grows faster than polynomial so the limit is 0.

Step 2:

Because of the limit is zero, the statement is false.

$\therefore\ n^3-4n^2+7 \neq \Theta(2^n)$

c) $n^3(1+\sqrt{n}) = O(n^3\log n)$

Step 1:

$$\lim_{n\to\infty} \frac{\cancel{n^3}(1+\sqrt{n})}{\cancel{n^3}\log n} = \lim_{n\to\infty} \frac{1+\sqrt{n}}{\log n} = \infty$$

Sqareroot grows faster than logarithm so the limit is ∞.

Step 2:

Because the limit is ∞, statement is false.

$\therefore\ n^3(1+\sqrt{n}) \neq O(n^3\log n)$

230104004903

d)

d) $28n = O(7n^2)$

Step 1:

$$\lim_{n \to \infty} \frac{28\cancel{n}}{7n^2} = \lim_{n \to \infty} \frac{4}{n} = 0$$

Step 2:

The limit is 0, so the statement is true.

$\therefore\ 28n = O(7n^2)$

e) $n + \log n + 21 = O(7n^2)$

Step 1:

$$\lim_{n \to \infty} \frac{n + \log n + 21}{7n^2} = \lim_{n \to \infty} \left(\frac{n}{7n^2} + \frac{\log n}{7n^2} + \frac{21}{7n^2} \right) \Rightarrow$$

$$\Rightarrow \underbrace{\lim_{n \to \infty} \frac{n}{7n^2}}_{0} + \underbrace{\lim_{n \to \infty} \frac{\log n}{7n^2}}_{\text{n^2 grows faster, limit is 0.}} + \underbrace{\lim_{n \to \infty} \frac{21}{7n^2}}_{0} = 0 + 0 + 0 = 0.$$

Step 2: The limit is 0, so the statement is true.

$\therefore\ n + \log n + 21 = O(7n^2)$

f) $n^2 + 9n - 13 = \Theta(n^2)$

Step 1:

$$\lim_{n \to \infty} \frac{n^2 + 9n - 13}{n^2} = \lim_{n \to \infty} \frac{\cancel{n^2}\left(1 + \frac{9}{n} - \frac{13}{n^2}\right)}{\cancel{n^2}} = \lim_{n \to \infty} 1 + \underbrace{\frac{9}{n}}_{0} - \underbrace{\frac{13}{n^2}}_{0} = 1$$

Step 2:

The limit is 1, so the statement is true.

$\therefore\ n^2 + 9n - 13 = \Theta(n^2)$

230104004903

Q2)

I will compare the functions using limits.

1) $\log n$ vs. $7n$

$\lim_{n\to\infty} \frac{\log n}{7n} = 0$ $7n$ grows faster than $\log n$, $\log n < 7n$

2) $7n$ vs. $2n^2$

$\lim_{n\to\infty} \frac{7n}{2n^2} = 0$ $2n^2$ grows faster than $7n$, $7n < 2n^2$

3) $2n^2$ vs. $3n^4$

$\lim_{n\to\infty} \frac{2n^2}{3n^4} = 0$ $3n^4$ grows faster than $2n^2$, $2n^2 < 3n^4$

4) $3n^4$ vs. 3^n

$\lim_{n\to\infty} \frac{3n^4}{3^n} = 0$ 3^n grows faster than $3n^4$, $3n^4 < 3^n$

5) 3^n vs. $n!$

$\lim_{n\to\infty} \frac{3^n}{n!} = 0$ $n!$ grows faster than 3^n, $3^n < n!$

$$\boxed{\log n < 7n < 2n^2 < 3n^4 < 3^n < n!}$$

230104004903

Q3)

a)

```
Static void someFunction(int a, int b) {
    int sum = 0;
    for (int i = 0; i < a; i++)              } O(a)
        for (int j = 0; j < b; j++)          } O(b)
            sum += i+b;
```

The outer loop runs a timer, for each operation of a inner loop runs b, so the total operations a.b

Worst-Case Complexity: $T(n) = O(a.b)$

b)

```
static void anotherFunction(int a) {
    int sum = 0;
        for (int i = 0; i < a; i++)
            sum += i;
            i = i * 2;
```

i doubles in every iteration so it is exponential growth, therefore loop stops when $i \geq a$.

$2^T \approx a$

$T = \log_2 a$

Worst-Case Complexity: $T(n) = O(\log a)$

c)

```
static void differentFunction (int n) {
    for (int i = 0; i < n; i++)                      } O(n)
        for (int j = 0; j < n; j++)                  } O(n)
            for (int k = 0; k < n; k++)              } O(n)
                System.out.println(" Hello, world! ");
```

Each loop running n times, so the total operations n.n.n

Wort-Case Complexity: $T(n) = O(n^3)$

230104004903

Q4)

The fastest way in my opinion would be using binary search, to find the lowest floor where the toy would break.

- Dropping at floor 50
  - => if breaks, search in 1-49.
  - => if dont break, search in 51-100.
- If searching in 1-49.
  - => Drop at 25th floor.
    - if breaks, search 1-25.
    - if dont break, search 26-49.
- if searching in 51-100.
  - => Drop at 75th floor.
    - if breaks, search 51-74.
    - if dont break, search 76-100.
- if searching in 1-24.
  - => Drop at 12th floor.
    - if breaks, search 1-11.
    - if dont break, search 13-24.
- if searching 51-74.
  - => Drop at 62th floor.
    - If breaks, search 51-61.
    - if dont break, search 63-74.

=> Continue this, always drop at the middle of the search range. this method requres at most 7 drops.

The time complexity is $\boxed{T(n) = O(\log n)}$

230104004903