Homework 1 : Notation

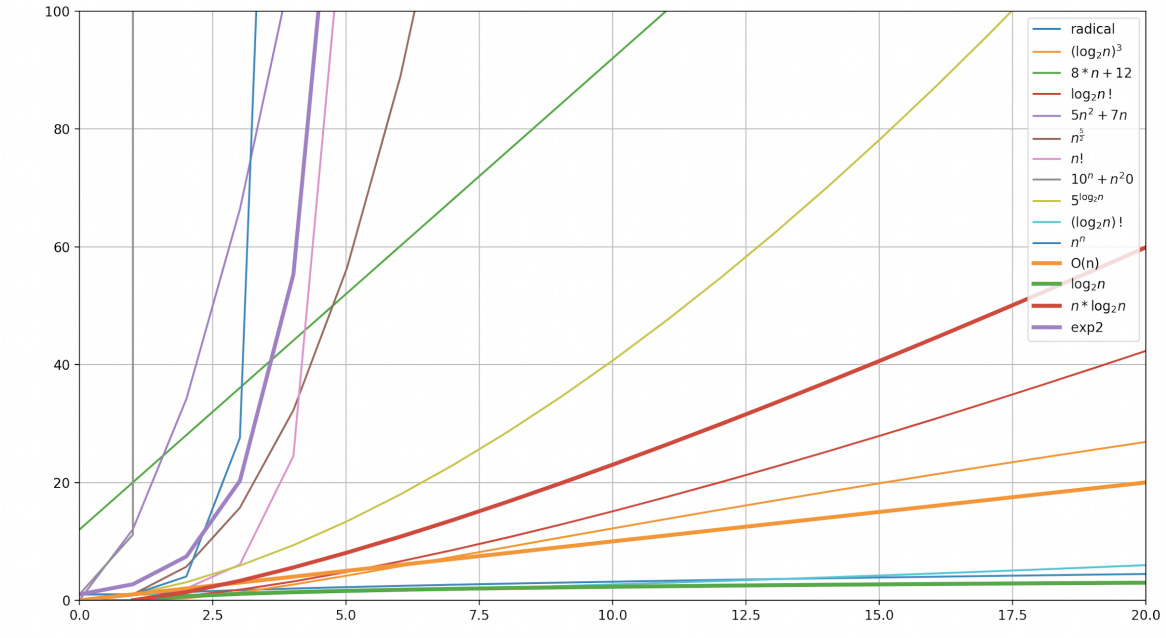
Solution:

1. quick way to compare growth of two given functions f(x) and g(x) :

   $\lim_{x \to \infty} \frac{f(x)}{g(x)} = if\ 0 \leftrightarrow growth\ of\ f(x)\ is\ less\ than\ g(x)\ ,\ \infty \leftrightarrow\ growth\ of\ f(x)\ is\ more\ than\ g($
   $,\ k\ \neq\ 0 \leftrightarrow they\ both\ grow\ equally$

   - $\lim_{n \to \infty} \frac{log(n)}{n} = 0 \rightarrow log(n) \in O(n)$
   - $\lim_{n \to \infty} \frac{n}{nlog(n)} = 0 \rightarrow n \in O(nlog(n))$
   - $nlog(n) \leq n^2\ N = 0\ ,\ c = 1,\ n \geq 0$
   - $2^n \geq 5^{ln(n)}\ \ N = 0\ ,\ c = 1\ ,\ n \geq 0$

2. $logn < \sqrt{n} < (logn)^3 < 8n + 12 < (logn)! < log(n!) < n.logn < 5n^2 + 7n < n^{\frac{5}{2}} < n^3 < 5$
   $10n + n^{20} < n! < n^n < n^n + ln(n)$



3. The first loop is going from 0 to n , so repetition of “i<n” will be n times . second loop is going to execute for $2^k = n$, so the “j>1” will be executed $logn$ times. In the end the hole code is going to run in $O(nlog(n))$order .
4. Answers:
   a. 25+32 = 57
   b. Each loop is going to execute n times and they’re not nested so the function runs in O(n) order .
   c. We can do the k’s multiplication in the first loop .
5.

```java
static int minDiffSubArray(int arr[], int n){
    // To store prefix sums
    int[] prefix_sum = new int[n];

    // Generate prefix sum array
    prefix_sum[0] = arr[0];

    for (int i = 1; i < n; i++)
        prefix_sum[i]
            = prefix_sum[i - 1] + arr[i];

    // To store suffix sums
    int[] suffix_sum = new int[n];

    // Generate suffix sum array
    suffix_sum[n - 1] = arr[n - 1];

    for (int i = n - 2; i >= 0; i--)
        suffix_sum[i]
            = suffix_sum[i + 1] + arr[i];

    // Stores the minimum difference
    int minDiff = Integer.MAX_VALUE;

    // Traverse the given array
    for (int i = 0; i < n - 1; i++) {

        // Calculate the difference
        int diff
            = Math.abs(prefix_sum[i]
                       - suffix_sum[i + 1]);

        // Update minDiff
        if (diff < minDiff)
            minDiff = diff;
    }

    // Return minDiff
    return minDiff;
}
```

6. the first computer will solve the problem of n=1000 size in one minute . the new computer will solve n=1000 size problem in $\frac{1}{1000}$minute , so it can solve a problem of size n = 10^6 in one minute .

    a) T(n)$\in\theta(n)$ n=1000 n = 1000*1000 = 1000000 n* = 10^6
    b) T(n)$\in\theta(n^3)$ n=100 n = 100*100*100 = 1000000 n* = 10^6
    c) T(n)$\in\theta(10^n)$ n=6 n=10^6 10^n = 10^6