COMP5511 - Assignment 1

Group 7

THI NGOC HOP NGUYEN, ID 29729283

WEIWEI XIAO, ID 40069298

ADNAN ALI, ID 40181614

PATRICK DRUMMOND, ID 40185198

1a) $log_2 n = 1,000,000$ => $n = 2^{1,000,000}$

1b) $\sqrt{n} = 1,000,000$ => n = 1,000,000,000,000

1c) n = 1,000,000

1d) n^2 = 1,000,000 => n = 1,000

1e) $(log_2 n)$^($log_2 n$) = 1,000,000

Because 7^7 < 1,000,000 < (7.1)^(7.1)

Then: 7 < $log_2 n$ < 7.1 => 128 < n < 137.18

Try all values from 129 to 137, we found that n = 133 is the largest value satisfying the above condition.

Therefore: n = 133

2a) ***Prove that $(n + 25)^2$ is O(n^2)

We have: $(n + 25)^2$ = n^2 + 50n + 25^2

Choose c = 3, $n_0 = 50$, we have: c n^2 = 3 n^2

We consider if the following is correct:

$$n^2 + 50n + 25^2 \leq 3n^2$$

$$\iff 50n + 25^2 \leq 2n^2$$

We can see that for every n $\geq n_0$, the above inequation satisfies.

So, with c = 3, $n_0 = 50$, $(n + 25)^2 \leq$ c n^2 for every n $\geq n_0$

We conclude that $(n + 25)^2$ is O(n^2)

*** Prove that n^2 is O($(n + 25)^2$)

We can see that with c = 1, n_0 = 0, for every n $\geq n_0$, the following inequation satisfies: $n^2 \leq$ c ($(n + 25)^2$)

We conclude that n^2 is O($(n + 25)^2$)

2b) Prove that n^3 is not O(n^2)

Assuming that n^3 is O(n^2), so there exists c $\geq$ 0, and n_0 such that $n^3 \leq$ c n^2 for every n $\geq n_0$

Take n_1 such that n_1 > c and $n_1 > n_0$

With n = n_1, $n^3 \geq$ c n^2 (because n_1 > c)

This contradicts to the assumption that $n^3 \leq$ c n^2 for every n $\geq n_0$.

Therefore, n^3 is not O(n^2)

2c) We have: f_1(n) x f_2(n) = $(n + 25)^2$ (n^3)

We choose c = 4, n_0 = 25, consider the following inequation for every n $\geq n_0$:

$$(n + 25)^2 (n^3) \leq c\, n^5$$

$$\Leftrightarrow \quad (n + 25)^2 (n^3) \leq 4\, n^5$$

$$\Leftrightarrow \quad (n + 25)^2 \leq 4\, n^2$$

$$\Leftrightarrow \quad (n + 25)^2 \leq (2n)^2$$

It's straight-forward that for every n $\geq n_0$, the above inequation satisfies.

So, f_1(n) x f_2(n) is O(n^5)

3) We add line numbers to the code for easier code analysis:

```
(1)  x = 0;
(2)  for (i=0; i<=n-1; i++) {
(3)          for (j=i; j<=n-1; j++) {
(4)                  x = x + A[j];
(5)          }
(6)          for (k=0; k<= n-1; k++) {
(7)                  for (j=0; j< =n-1; j++) {
(8)                          x = x + A[j]*A[k];
(9)                  }
(10)         }
(11) }
```

- The initialization of “x = 0” executes in O(1) time
- The “for” loop at line (2) controlled by the counter “i”, it takes O(n) executing time to initialize the value of “i”, then divided to 2 branches:
  ** The first branch is the “for” loop at line (3), controlled by the counter j, this inner loop executes in O(n) time, combining with the outer loop we have executing time of O(n^2)
  ** The second branch is the “for” loop at line (6), controlled by the counter “k”, followed by the “for” loop at line (7), controlled by the counter j, these two inner loops executes in O(n^2) time, combining with the outer loop at line (2) we have the executing time of O(n^3)

In summary, the total execution time is O(1) + O(n^2) + O(n^3) , take the dominant one, the complexity of the algorithm is O(n^3)

4) Programming question: in a separate folder, with source codes, input files, answer file: explanation and results produced by the source codes