Q1:

Part 1:

Step1: initialization of count.

$$T_1(n) = c_1 => T_1(n)\ is\ O(1)$$

Step2: the first for loop consists of initiating i, comparing i and (n-1), and increment.

$$T_2(n) = c_2 n => T_2(n)\ is\ O(n)$$

Step3: initialization of sum.

$$T_3(n) = c_3 n => T_3(n)\ is\ O(n)$$

Step4: the second for loop consists of initiating j, comparing j and (n-1), and increment.

$$T_4(n) = c_4 n \cdot n = c_4 n^2 => T_4(n)\ is\ O(n^2)$$

Step5: the additive operation and the assignment to sum.

$$T_5(n)\ is\ O(n^2)$$

Step6: the third for loop consists of initiating k, comparing k and j, and increment.

$$T_6(n) = c_6 n(1 + 2 + 3 + \cdots + n - 1) = c_6 n \cdot \frac{n(n-1)}{2} => T_6(n)\ is\ O(n^3)$$

Step7: the additive operation and the assignment to sum.

$$T_7(n)\ is\ O(n^3)$$

Step8: the if statement and executive statement contained. Their time complexity won't surpass the time complexity of the first loop. So $T_8\ is\ O(n)$

Step9: return statement.

$$T_9(n) = c_9 => T_9(n)\ is\ O(1)$$

Therefore, T$(n)\ is\ O(n^3)$.

Part 2:

1) n = 4; count = 1; i = 0; i < 4; sum = 0.
   A) j = 0; j < 4; A[0] = 1; sum = 1.
      a) k = 1; k > j; end of the third loop; j = 1.
   B) j < 4; A[0] = 1; sum = 2.
      a) k = 1; k ≤ j; A[1] = 2; sum = 4; j = 2.
      b) k > j; end of the third loop; j = 2.
   C) j < 4; A[0] = 1; sum = 5.
      a) k = 1; k ≤ 2; A[1] = 2; sum = 7; k = 2.
      b) k ≤ 2; A[2] = 5; sum = 12; k = 3.
      c) k > 2; end of the third loop; j =3
   D) j < 4; A[0] = 1; sum = 13.
      a) k = 1; k ≤ 3; A[1] = 2; sum = 15; k = 2.

b) $k \leq 3$; A[2] = 5; sum = 20; k = 3.
c) $k \leq 3$; A[3] = 9; sum = 29; k = 4.
d) k > 3; end of the third loop. j = 4.

E) j > 3; end of the second loop;

2) B[0] = 2 $\neq$ sum = 29.
3) i = 0 + 1 = 1.
4) i < 4; sum = 0.
   A) repeats the steps form 1)A) to 1)E).
5) B[1] = 29 = sum.
   count = 0 + 1 = 1.
6) i = 1+1 = 2.
7) i < 4; sum = 0.
   A) repeats the steps from 1)A) to 1)E).
8) B[2] = 40 $\neq$ sum.
9) i = 2+1 = 3.
10) i < 4; sum = 0.
   A) repeats the steps from 1)A) to 1)E).
11) B[3] = 57 $\neq$ sum.
12) i = 3 + 1 = 4.
13) i =4. end of the first loop.
14) return count = 1.

In conclusion, the final input: count = 1.

Q2

a)

Step1: the first for loop

$$T_1(n) = c_1 \frac{n}{c} => T_1(n)\ is\ O(n).$$

Step2: the second for loop

$$T_2(n) = c_2(log_2 1024)(\frac{n}{c}) => T_2(n)\ is\ O(n).$$

Hence, T(n) $is\ O(n)$.

b)

Step1: the first for loop

$$2^k = n => k = \log(n)$$

$$T_1(n) = c_1 \log(n) => T_1(n)\ is\ O(\log(n)).$$

Step2: the second for loop

$T_2(n) = c_2(1 + 2 + 4 + \cdots + 2^k)/2 = c_2\left(\frac{2^k-1}{2-1}\right)/2 = c_2\frac{2^k-1}{2}$. Since $2^k = i \leq n$, $T_2(n) = c_2\frac{n-1}{2}$. $This\ is, T_2(n)\ is\ O(n)$.

Therefore, T(n) $is\ O(n)$.

c)

Step1: the first for loop

$2^k = n \Rightarrow k = \log(n)$

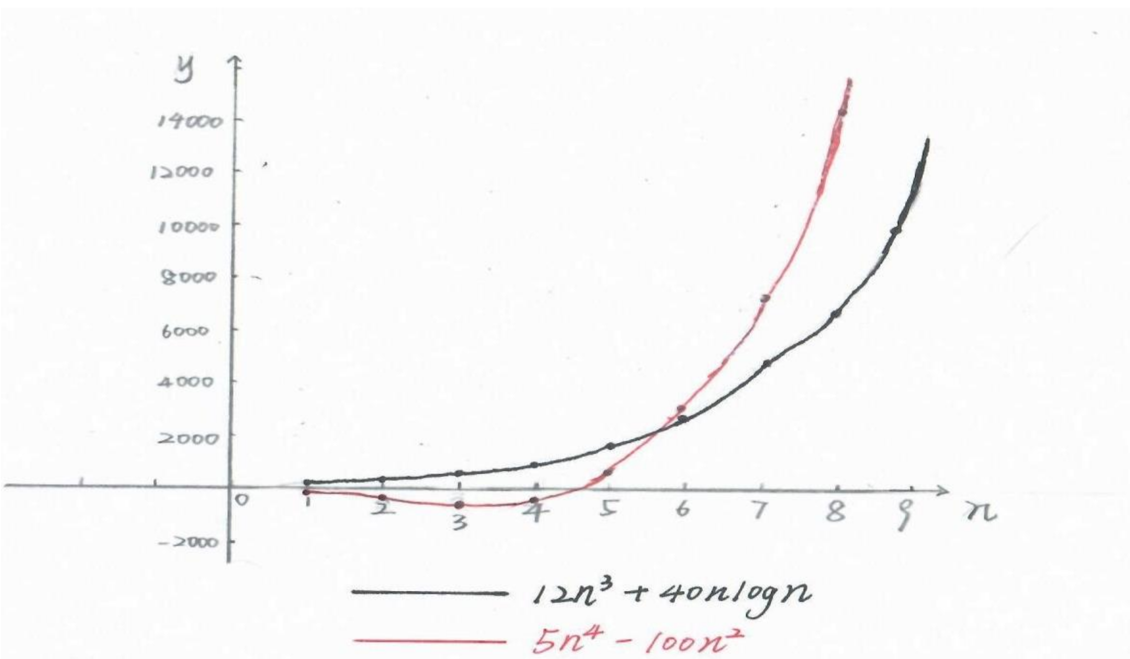
$T_1(n) = c_1 \log(n) \Rightarrow T_1(n)\ is\ O(\log(n))$.

Step2: the second for loop

$T_2(n) = c_2(\log(2) + \log(4) + \log(8) + \cdots + \log(2^k))$. Since $2^k = i \leq n, T_2(n) = c_2(1 + 2 + 3 + \cdots + \log(n)) = c_2\frac{(1+\log(n))\log(n)}{2} \Rightarrow T_2(n)\ is\ O((\log n)^2)$.

Therefor, T(n) is $O((\log n)^2)$.

Q3



By observing the two lines, we can choose n_0=6 such that B is greater than A for n≥6.

Q4

a)

Proof:

d(n) $is\ O(f(n))$ => $d(n) \le af(n)\ for\ n \ge n_0'$, a is a constant;

$e(n)\ is\ O(g(n)) => e(n) \le bg(n)\ for\ n \ge n_0'', b\ is\ a\ constant;$

then, $d(n) + e(n) \le af(n) + e(n) \le af(n) + bg(n)$. Let $c = \max(a, b)$, $n_0 = \max(n_0', n_o'')$; then $af(n) + bg(n) = c(f(n) + g(n))\ for\ n \ge n_0$, where c is constant. Therefor, $d(n) + e(n)\ is\ O(f(n) + g(n))$.

As desired.

b)

Proof:

To prove $2^{n+1} + n^3\ is\ O(2^n)$, $we\ should\ prove\ 2^{n+1}\ is\ O(2^n)\ and\ n^3$ is $O(2^n)$ respectively.

Firstly, choosing $c = 2, n_0 = 1, we\ can\ get\ 2^{n+1}\ is\ O(2^n)\ since\ 2^{n+1} \le 2 \cdot 2^n$.

Next, Since $\lim_{n\to+\infty} \frac{n^3}{2^n} = \lim_{n\to+\infty} \frac{3n^2}{2^n \cdot \ln 2} = \lim_{n\to+\infty} \frac{6n}{2^n (\ln 2)^2} = 0$, it is obvious that the growing rate of $n^3$is less than the growing rate of 2^n. This is, there exist a constant c and n_0 that make $n^3 \le c \cdot 2^n$. One possible solution is choosing c = 16, n_0 = 1.

According to the Question (a) we have proved, we can get $2^{n+1} + 3^n \le 16 \cdot (2^n + 2^n) = 32 \cdot 2^n\ for\ n \ge 1. This\ is, 2^{n+1} + 3^n\ is\ O(2^n). As\ desired.$

c)

Proof:

$2^n = 2 \times 2 \times 2 \times ... \times 2$; $2 \cdot n! = 2 \times 1 \times 2 \times 3 \times 4 \times ... \times n = 2 \times 2 \times 3 \times 4 \times ... \times n$. It is clear that every single factor of n! is greater or equal to that of $2*2^n$. So we can conclude that $2^n \le 2 \cdot n!\ for\ n \ge 1$; this is, $2^n\ is\ O(n!)$. As desired.

d)

Proof:

$$\log(n!) = \log(1 \times 2 \times 3 \times ... (n-1) \times n) = log1 + log2 + log3 + \cdots + logn$$

$$\le \text{logn} + \text{logn} + \cdots + \text{logn} = \log(n^n) = nlogn\ for\ n \ge 1.$$

Therefore, $\log(n!)\ is\ O(\text{nlogn})$, as desired.