LI. Yuntong
20944800
ylino@connect.ust.hk

Q1.

(1) $n^3(\log n)^5 = O(n^{3.1})$

(2) $2^{\sqrt{\log n}} = O(n)$

(3) $10n^2-2n+5 = \Theta(100n^2)$

(4) $n^2 = \Theta(4^{\log_2 n})$

(5) $n\log^2 n + n^{1.5} = \Omega(n(\log_{10} n)^{1.5})$

Q2.

(a).
$$T(n) = T(n-1)+n^2$$
$$T(n-1) = T(n-2)+(n-1)^2$$
$$T(n-2) = T(n-3)+(n-2)^2$$
$$\vdots$$
$$T(2) = T(1)+2^2$$

Adding all the left and right of the equation we can have:

$T(n) = T(1)+2^2+3^2+\dots\dots+n^2$. Since: $T(1)=1$

$\therefore$ $T(n) = 1+2^2+3^2+\dots\dots+n^2 = \frac{n(n+1)(2n+1)}{6} \approx \frac{2n^3}{6} = \frac{n^3}{3}$

$\therefore$ $T(n) = O(n^3)$

(b) $T(n) = 4T(\frac{n}{2})+n^2$

$= 4\left[4T(\frac{n}{4})+(\frac{n}{2})^2\right]+n^2 = 4^2\cdot T(\frac{n}{4})+n^2+n^2$

$= 4^2\left[4T(\frac{n}{8})+(\frac{n}{4})^2\right]+n^2+n^2 = 4^3T(\frac{n}{8})+n^2+n^2+n^2$

$\vdots$

$= 4^i T(\frac{n}{2^i})+i\cdot n^2$

Since: $i=\log_2 n$. $T(1)=1$ we have: $T(n) = 4^{\log_2 n}T\left(\frac{n}{2^{\log_2 n}}\right)+n^2\cdot\log_2 n$

$= n^2T(1)+n^2\cdot\log_2 n$

$= n^2+n^2\cdot\log_2 n = O(n^2\log n)$

(c) $T(n) = 3T(\frac{n}{2}) + n$

$$= 3\left[3T(\frac{n}{4}) + \frac{n}{2}\right] + n = 3^2 T(\frac{n}{4}) + \frac{3}{2}n + n$$

$$= 3^2\left[3T(\frac{n}{8}) + \frac{n}{4}\right] + \frac{3}{2}n + n = 3^3 T(\frac{n}{8}) + \frac{3^2}{4}n + \frac{3}{2}n + n$$

$$\vdots$$

$$= 3^i T(\frac{n}{2^i}) + n\cdot\left[1 + \frac{3}{2} + (\frac{3}{2})^2 + \dots + (\frac{3}{2})^{i-1}\right]$$

Since $i = \log_2 n$, $T(1) = 1$ we have: $T(n) = 3^{\log_2 n} T(\frac{n}{2^{\log_2 n}}) + n\cdot\left[1 + \frac{3}{2} + (\frac{3}{2})^2 + \dots + (\frac{3}{2})^{\log_2 n - 1}\right]$

$$= n^{\log_2 3} + n\cdot\left[2\cdot n^{\log_2 \frac{3}{2}} - 2\right]$$

$$= n^{\log_2 3} + 2\cdot n^{\log_2 3 - \log_2 2 + 1} - 2n$$

$$= n^{\log_2 3} + 2\cdot n^{\log_2 3} - 2n$$

$$= 3\cdot n^{\log_2 3} - 2n = O(n^{\log_2 3})$$

(d) $T(n) = 2T(\frac{n}{4}) + \sqrt{n}$

$$= 2\left[2T(\frac{n}{16}) + \sqrt{\frac{n}{4}}\right] + \sqrt{n} = 2^2 T(\frac{n}{4^2}) + \sqrt{n} + \sqrt{n}$$

$$= 2^2\left[2T(\frac{n}{4^3}) + \sqrt{\frac{n}{4^2}}\right] + \sqrt{n} + \sqrt{n} = 2^3 T(\frac{n}{4^3}) + \sqrt{n} + \sqrt{n} + \sqrt{n}$$

$$\vdots$$

$$= 2^i T(\frac{n}{4^i}) + i\cdot\sqrt{n}$$

Since: $i = \log_4 n$, $T(1) = 1$ we have: $T(n) = 2^{\log_4 n} T(\frac{n}{4^{\log_4 n}}) + \sqrt{n}\cdot\log_4 n$

$$= \sqrt{n} + \sqrt{n}\cdot\log_4 n = O(\sqrt{n}\log n)$$

(e) Guess $T(n) = O(n\log n)$

Prove by induction:

Base case: $n=1$. $T(1) = 1$

Suppose for the smaller n: $T(k) \le ck\log k$

$$T(n) = T(\frac{n}{5}) + T(\frac{3}{5}n) + n \le c\frac{n}{5}\log\frac{n}{5} + c\frac{3n}{5}\log\frac{3n}{5} + n$$

$$= \frac{4}{5}cn\log n - cn(\frac{4}{5}\log 5 - \frac{3}{5}\log 3) + n$$

$$\le cn\log n - cn(\frac{4}{5}\log 5 - \frac{3}{5}\log 3) + n$$

when n is bigger enough. $cn\log n$ will dominate so

$$cn\log n - cn(\frac{4}{5}\log 5 - \frac{3}{5}\log 3) + n = O(n\log n)$$

So, above all. $T(n) = O(n\log n)$

Q3.

(a) Initialize $A[0\ldots n-1][0\ldots n-1]$ and $B[0\ldots n-1]$. each element is zero.

A is the $n\times n$ matrix B is an array which is a flag of whether the column has already been put "1"

```
Generate ( A, B, row, n)
    if row ← n then output the whole matrix A
        return
    for col ← 0 to n-1
        if B[col] ≠ 1 then
            A[row][col] ← 1
            B[col] ← 1
            Generate ( A, B, row+1, n)
            B[col] ← 0
            A[row][col] ← 0
        return
```

First call: Generate (A, B, 0 , n)

(b). Let $T(n)$ be the running time

Base case: if $n=1$. obviously. $T(1)=1$

Recursive $(n>1)$: For each row, we need to place "1" in the column which is unused. for the first row, there are n choice. for the second row, there are $(n-1)$ choice.

So: $T(n) = n \cdot T(n-1)$

$$\Rightarrow \quad T(n) = n(n-1)T(n-2)$$

$$= n(n-1)(n-2)T(n-3)$$

$$\vdots$$

$$= n(n-1)(n-2)\cdots T(1)$$

$$= n! = O(n!)$$

So: the running time is $O(n!)$

Q4.

(a) The maximum number is 9

Exlanation: Suppose the number is x. the minimum number of occurrences is $(\frac{n}{10}+1)$ for each. we have:

$$x \cdot \left(\frac{n}{10}+1\right) \le n$$

$$\Rightarrow \quad x \le \frac{n}{\frac{n}{10}+1} = 10\left(\frac{\frac{n}{10}}{\frac{n}{10}+1}\right) = 10\left(1-\frac{1}{\frac{n}{10}+1}\right) = 10-\frac{10}{\frac{n}{10}+1} < 10$$

so. the maximum number of x is 9

(b)

```
find (A, p, r, n)
    if p = r then return A[p]      # divide into to minimum — one element
    q ← [ (p+r)/2 ]
    L_major ← find (A, p, q, n)    # first handle the left part
    R_major ← find (A, q+1, r, n)  # then heudle the right part
    creat one new empty array: temp[]     # it is a candidate array
```

```
        append L_major at the end of temp
        for each element in R_major
             if element not in temp then
                  append this element at the end of temp

    creat one new empty array: valid [1.2, .... 9]   # this is the array which
                                                       stores the valid 10-major number
    for each element in temp
         count ← 0                    # Determine whether any number in the candidate
                                      array: temp , appears more than n/10 times in each stage of array
         for i ← p to r
              if A[i] = the element right now in temp then
                   count ← count+1
         if count > (r-p+1)/10 then
              append this element at the end of valid

    return valid
```

First call : find (A, 1, n, n)

(c) Because it dividing the problem into two sub problems, each is the half size then for each element, you should compare it whether in the temp. whats more, it also need to count the each element in temp whether satisfied 10-majors in this small period of A[p,r]. if yes, then put it in the valid. For this counting and checking, they have linear complexity concerning the subarray's size

so we have: $T(n) = 2T(\frac{n}{2}) + cn \quad (n>1)$

(d): $T(n) = 2T(\frac{n}{2}) + cn$

$= 2\left[2T(\frac{n}{2^2}) + \frac{cn}{2}\right] + cn = 2^2T(\frac{n}{2^2}) + cn + cn$

$\vdots$

$$= 2^{i} T\left(\frac{n}{2^{i}}\right) + i \cdot cn$$

We have: $i = \log_2 n$. $T(1) = 1$

So: $T(n) = 2^{\log_2 n} T\left(\frac{n}{2^{\log_2 n}}\right) + (\log_2 n) \cdot cn$

$$= n + cn \cdot \log_2 n$$

$$= O(n \log n)$$