Tutorial 05 Big-O & Sorting

Question One: Big-O

What is the Big-O complexity of each of the following methods?

`public static void q1A(int n) {`
`  int count = 0;`
`  for(int i=0; i<20000; i++)`
`  count++;`
`}`
	O( )

`public static void q1B(int n) {`
`  int count = 0;`
`  for(int i=0; i<n; i++)`
`  for (int j=0; j<100; j++)`
`  count++;`
`}`
	O( )

`public static void q1C(int[] a) {`
`  int result = 0;`
`  int n = a.length;`
`  for (int i=0; i < n; i++)`
`  result = result + a[i];`
`}`
	O( )

`public static void q1D(int[][] a) {`
`  int result = 0;`
`  int n = a.length;`
`  for (int i=0; i < n; i++) {`
`  for (int j=0; j < i; j++)`
`  result = result + a[i][j];`
`  }`
`}`
	O( )

`public static void q1E(int n) {`
`  int count = 0;`
`  for(int i=0; i<n; i++)`
`  count++;`
`  for (int j=0; j<n; j++)`
`  count++;`
`}`
	O( )

`public static void q1F(int[] a) {`
`  int result = 0;`
`  int n = a.length;`
`  for (int i=0; i < n; i++)`
`  result = doG( a[i]);`
`  }`
`}`
`private static int doG(int a) {`
`  int result = 0;`
`  for (int i=0; i < a; i++) {`
`  result += i;`
`  }`
`  return result;`
`}`
	O( )

Question Two:

a) Trace the **selection sort** as it sorts the following array into ascending order:

2 3 1 6 4 5

b) Trace the **insertion sort** as it sorts the following array into ascending order:

2 3 1 6 4 5

c) Trace the **bubble sort** as it sorts the following array into ascending order:

2 3 1 6 4 5

Question Three

The following is a list of birthdays in DD/MM format. Trace the **bubble sort** into ascending order by month, for the cases of more than one birthday within a month, sort them by day.

20/09
12/01
12/03
25/12
01/01

Question Four

Given the source code for the method merge() – see also file: `T05Q3.java`. Write the recursive method of Merge sort (in `T05Q3.java`).

```
private void merge(int[] a, int start, int mid, int end) {
      int temp[] = new int[end-start];
      int i = start;
      int j = mid;
      for(int k = 0; k < temp.length; k++) {
            if (j >= end || i < mid && a[i] <= a[j])
                  temp[k] = a[i++];
            else
                  temp[k] = a[j++];
      }
      for(int k = start; k < end; k++)
            a[k] = temp[k-start];
}

public static void mergeSort (int[] a, int start, int end) {
/* please complete this recursive merge sort method */
```

Answer:

Q1

`public static void q1A(int n) {`
`  int count = 0;`
`  for(int i=0; i<20000; i++)`
`  count++;`
`}`
	O(1)

`public static void q1B(int n) {`
`  int count = 0;`
`  n = n/10;`
`  for(int i=0; i<n; i++)`
`  for (int j=0; j<100; j++)`
`  count++;`
`}`
	O(n)

`public static void q1C(int[] a) {`
`  int result = 0;`
`  int n = a.length;`
`  for (int i=0; i < n; i++)`
`  result = result + a[i];`
`}`
	O(n)

`public static void q1D(int[][] a) {`
`  int result = 0;`
`  int n = a.length;`
`  for (int i=0; i < n; i++) {`
`  for (int j=0; j < i; j++)`
`  result = result + a[i][j];`
`  }`
`}`
	O(n^2)

`public static void q1E(int n) {`
`  int count = 0;`
`  for(int i=0; i<n; i++)`
`  count++;`
`  for (int j=0; j<n; j++)`
`  count++;`
`}`
	O(n)

`public static void q1F(int[] a) {`
`  int result = 0;`
`  int n = a.length;`
`  for (int i=0; i < n; i++)`
`  result = doG( a[i]);`
`  }`
`}`
`private static int doG(int a) {`
`  int result = 0;`
`  for (int i=0; i < a; i++) {`
`  result += i;`
`  }`
`  return result;`
`}`
	O(n^2)
	The outer loop executes n times and the number of iterations of the inner loop ranges from 1 to n. So this is the sum of a series
	= 1 + 2 + ... + n-1 + n
	= n(n+1)/2

Q2.a

At each pass, the selected element is underlined.

2 3 1 6 4 5
2 3 1 5 4 **6**
2 3 1 4 **5 6**
2 3 1 **4 5 6**
2 1 **3 4 5 6**
1 2 3 4 5 6

Q2.b

2 3 1 6 4 5
2 3 1 6 4 5
1 2 3 6 4 5
1 2 3 6 4 5
1 2 3 4 6 5
1 2 3 4 5 6

Q2.c

Pass 1
2 3 1 6 4 5
2 3 1 6 4 5
2 1 3 6 4 5
2 1 3 6 4 5
2 1 3 4 6 5
2 1 3 4 5 6
Pass 2
2 1 3 4 5 6
1 2 3 4 5 6
1 2 3 4 5 6
1 2 3 4 5 6
1 2 3 4 5 6
There are no exchanges during Pass 3, so the algorithm will terminate.

Q3

By month:

First pass:	Second pass:	Third pass:	Fourth pass:
20/09 12/01 12/01 12/01 12/01	12/01 12/01 12/01 12/01	12/01 12/01 12/01	12/01 01/01
12/01 20/09 12/03 12/03 12/03	12/03 12/03 12/03 12/03	12/03 12/03 01/01	01/01 12/01
12/03 12/03 20/09 20/09 20/09	20/09 20/09 20/09 01/01	01/01 01/01 12/03	12/03 12/03
25/12 25/12 25/12 25/12 01/01	01/01 01/01 01/01 20/09	20/09 20/09 20/09	20/09 20/09
01/01 01/01 01/01 01/01 25/12	25/12 25/12 25/12 25/12	25/12 25/12 25/12	25/12 25/12

Q4

```
public static void mergeSort (int[] a, int start, int end) {
        if (start < end - 1) {
                int mid = (start + end)/2;
                mergeSort(a, start, mid);
                mergeSort(a, mid, end);
                merge(a, start, mid, end);
        }
}
```