FAKULTAS ILMU KOMPUTER

CSGE602040 - Struktur Data dan Algoritma
Semester Gasal - 2019/2020
Kuis Tertulis 1
Selasa, 15 Oktober 2019

Nama Peserta Kuliah: ……. Kelas: …….

1. [14 poin] Diberikan fragmen code berikut ini, tentukan kompleksitas method dalam notasi Big Oh.

   a.
```
public static void mysterious (int N) {
    for (int i = 4; i <= N; i += 4)
        for (int j = 4; j <= N; j *= 4)
            System.out.println("i, j adalah " + i + ", " + j);
}
```

   b.
```
public static void mystery (int M) {
    for (int i = 4; i <= M; i += 4)
        System.out.println("i adalah " + i);
    for (int j = 4; j <= M; j *= 4)
        System.out.println("j adalah " + j);
}
```

2. [14 poin] Suatu problem dapat diselesaikan dengan dua buah algoritma. Algoritma pertama memiliki kompleksitas O(M) sedangkan algoritma kedua memiliki kompleksitas $O(M^3)$. Untuk input berukuran 1000, algoritma O(M) memiliki running time 1 detik, sedangkan algoritma $O(M^3)$ memiliki running time 3 detik. Jika diasumsikan bahwa running time hanya akan dipengaruhi oleh kompleksitas algoritma dan ukuran input, jawablah pertanyaan berikut ini.

   a. Berapakah running time algoritma O(M) untuk input berukuran 3000?

   b. Berapakah running time algoritma $O(M^3)$ untuk input berukuran 3000?

Kode soal: ALEXANDRITE

3. [16 poin]

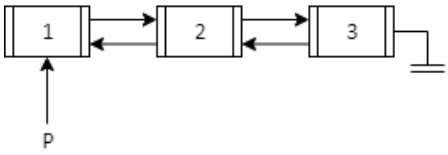
```
public class MysterySorter {

    static int[] arrayOfInt = {20, 14, 19, 22, 11, 24, 12, 15, 25, 23};

    public static void main(String[] args) {
       thatSorter(arrayOfInt);
    }

    static void thatSorter(int[] arr) {
       int interval = 5;
       while(interval > 0) {
             for(int k = 0; k < interval; k++)
                   aforementionedSorter(arr, k, interval);

             printArray(arr);
             interval -= 2;
       }
    }

    static void aforementionedSorter(int[] arr, int first, int gap) {
       for (int ii = first + gap; ii < arr.length; ii += gap) {
             int temp = arr[ii];
             int jj = ii;
             while (( jj > first) && (temp < arr[jj - gap])) {
                   arr[jj] = arr[jj - gap];
                   jj -= gap;
             }
             arr[jj] = temp;
       }
    }

    static void printArray(int[] arr) {
       for(int a = 0; a < arr.length; a++)
             System.out.print(arr[a] + " ");
       System.out.println("");
    }
}
```

a. Apa output yang dicetak oleh program di atas?

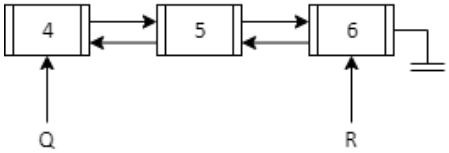
b. Method **thatSorter** adalah implementasi algoritma sorting apakah?

4. [10 poin] Dalam suatu Doubly Linked-List akan dilakukan penyisipan serangkaian *node*. Posisi penyisipan adalah setelah *node* yang ditunjuk oleh **P**. *Node* pertama pada rangkaian *node* yang akan disisipkan ditunjuk oleh **Q**. *Node* terakhir pada rangkaian *node* yang akan disisipkan ditunjuk oleh **R**.
   Diketahui P != null, P.next != null, Q != null, R != null, serta Q != R.

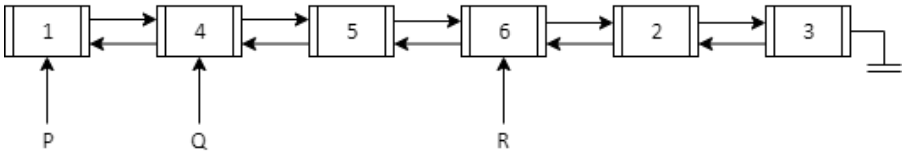
LinkedList awal



LinkedList yang akan disisipkan



Hasil penyisipan



Urutkan perintah di bawah ini agar proses penyisipan dapat dilakukan dengan benar.
Tidak semua perintah harus digunakan.

```
P.next.prev = R;        //1
R.prev = P;             //2
Q.prev =P;              //3
P.next = Q;             //4
R.next = P.next;        //5
Q.next = P.next;        //6
```

Cukup tuliskan urutan angka perintah pada jawaban Anda

5. [10 poin] Apakah output dari program di bawah ini?

```
import java.util.*;

class PrintJob implements Comparable{
    private int waktu;
    private int lembar;

    public PrintJob(int waktu, int lembar){
      this.waktu = waktu;
      this.lembar = lembar;
    }
    public int getWaktu(){
       return waktu;
    }
    public int getLembar(){
       return lembar;
    }

    public int compareTo(Object pj){
       if(waktu < ((PrintJob)pj).getWaktu()) return -1;
       else if(waktu > ((PrintJob)pj).getWaktu()) return 1;
       else{
             if(lembar < ((PrintJob)pj).getLembar()) return -1;
             else if(lembar > ((PrintJob)pj).getLembar()) return 1;
             else return 0;
       }
    }

    public void print(){
       System.out.println(waktu + "," + lembar);
    }
}

public class PrintJobQueue{
    private PriorityQueue<PrintJob> pjQueue;

    public static void main(String[] args){
       PrintJobQueue myPrintJobQueue = new PrintJobQueue();
       myPrintJobQueue.runPrinter();
    }
    public PrintJobQueue(){
       this.pjQueue = new PriorityQueue<PrintJob>();
    }
```

```
    public void runPrinter(){
       pjQueue.add(new PrintJob(1, 100));
       pjQueue.add(new PrintJob(1, 1));
       pjQueue.add(new PrintJob(2, 20));

       pjQueue.poll().print();
       pjQueue.poll().print();

       pjQueue.add(new PrintJob(2, 2));
       pjQueue.add(new PrintJob(2, 200));

       pjQueue.poll().print();
       pjQueue.poll().print();
    }
}
```

6. [16 poin]

```
public class SoalQuizku {

    public int sebuahWhile(int m, int n, int p) {
       int hasil = 0;
       while(m > n) {
             hasil += sebuahRekursif(m, p);
             m--;
       }
       return hasil;
    }

    public int sebuahRekursif(int m, int p) {
       if(p > 0)
             return sebuahRekursif(m, p-1) + sebuahRekursif(m, 0);
       if(m < 2)
             return m;
       return sebuahRekursif(m-1, p) + sebuahRekursif(m-2, p);
    }
}
```

a. Apa output yang dihasilkan jika method “sebuahWhile” menerima input parameter m = 7 , n = 5 , dan p = 10

b. Hitunglah kompleksitas dari method “sebuahWhile” dalam notasi Big Oh

7. [20 poin] Diberikan barisan bilangan sebagai berikut
   75 93 32 48 56 81 96 52 72 36 40 54

   a. Dengan menggunakan Quick Sort algoritma partition versi 2 di slide: pivot dipilih dari elemen pertama dalam array dan "dibuang sementara" (code di bawah), berapa kali terjadi swap sampai pivot pertama menempati posisi yang seharusnya?

   b. Bilangan apa saja (jawaban bisa satu atau lebih) yang terakhir kali menjadi pivot?

```
static void quickSort(int a[], int low, int high)
{
    if(high <= low) return; // base case
    pivotIdx = low; // select "best" pivot
    pivot = a[pivotIdx];
    swap (a, pivotIdx, high);     // move pivot out of the way
    int i = low, j = high-1;
    while (i <= j) {
        // find large element starting from left
        while (i<=high && a[i]<pivot) i++;
        // find small element starting from right
        while (j>=low && a[j]>=pivot) j--;
        // if the indexes have not crossed, swap
        if (i < j) swap (a, i, j);
    }
    swap(a,i,high);       // restore pivot to index i
    quickSort (a, low, i-1);   // sort small elements
    quickSort (a, i+1, high);  // sort large elements
}
```