DSA-ASSIGNMENT-1

Time and Space Complexity

Question 1: Analyze the time complexity of the following Java code and suggest a way to improve it:

```
int sum = 0;
for(int i = 1; i <= n; i++) {
for(int j = 1; j <= i; j++) {
sum++;
}
}
```

Solution: The Time Complexity of the given code is O(n^2) because it uses nested loops where outer loop runs n times and inner loop runs i times value of i is 1 to n.
To improve this remove the loops and instead of loops use mathematical formula.

Question 2: Find the value of T(2) for the recurrence relation T(n) = 3T(n-1) + 12n, given that T(0) = 5.
Solution:
T(1)=3T(1-1)+12(1)
=3T(0)+12
=3*5+12
T(1)=27

T(2)=3T(2-1)+12(2)
=3T(1)+24
=3*27+24
T(2)=105

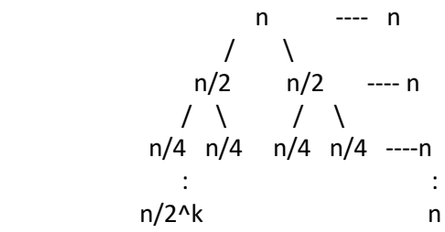
Question 3: Given a recurrence relation, solve it using a substitution method. Relation : T(n) = T(n - 1) + c.
Solution:
T(n)=T(n-1)+c

T(n-1)=T(n-1-1)+c
T(n-1)=T(n-2)+c
Therefore T(n)=T(n-2)+c+c

T(n-2)=T(n-3)+c
Therefore T(n)=T(n-3)+c+c+c
: : :
T(n)=T(n-k)+kc
Let us assume T(1)=constant
Therefore n-k=1
k=n-1
T(n)=1+kc
T(n)=1+(n-1)c
T(n)=1+nc-c
Therefore T(n)=O(n)

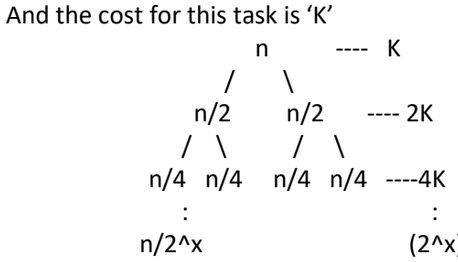
Question 4: Given a recurrence relation:
T(n) = 16T(n/4) + n2logn
Find the time complexity of this relation using the master theorem.
Solution:
T(n) = 16T(n/4) + n^2 logn
a=16
b=4
k=2
p=1
b^k = 4^2 = 16
here, a=b^k
and p>-1
hence T (n) = O(n^logb^a log^(p+1)n)
Therefore T(n)=O(n^log4^16 log^(1+1)n)
.: T(n)=O(n^log4^4^2 log^2 n)
.: T(n)=O(n^2log^2n)

Question 5: Solve the following recurrence relation using recursion tree method
T(n) = 2T(n/2) + n
Solution:
In the given equation T(n) = 2T(n/2) + n
T(n) divides into two n/2 subproblems
Hence T(n) = T(n/2)+T(n/2) + n
And the cost for this task is 'n'



the tree will be continued till we get 1
and the cost for each level is 'n'
let us assume we get the value 1 at the kth level hence the cost will be n+n+n+n+……n
hence cost will be= kn ……(1)
hence n/2^k=1 therefore n=2^k
taking log on both sides we get, k=logn
now substituting value of k in equation (1) we get,
n(logn)
Hence the solution will be **O(nlogn)**.

Question 6. T(n) = 2T(n/2) + K, Solve using Recurrence tree method.
Solution:
In the given equation T(n) = 2T(n/2) + K
T(n) divides into two 'n/2' sub-problems
Hence T(n) = T(n/2)+T(n/2) + K
And the cost for this task is 'K'



the tree will be continued till we get 1
and the cost for each level is increasing by multiple of '2' i.e K, 2K, 4K, 8K…
let us assume we get the value 1 at the 'x'th level hence the cost will be(K+2K+4K+8K…..2^xK)
which same as the number of nodes i.e 2^x ….(1)
Now, n/2^x=1 therefore n=2^x
taking log on both sides we get, x=logn
now substituting value of x in equation (1) we get,
2^logn base2
Hence the solution will be **O(n)**.

1D Array

Q1: Write a program to print the sum of all the elements present on even indices in the given array.
Solution:

```
public class SumEven {
    public static void main(String[] args) {
        int arr[]={3,20,4,6,9};
        int sum=0;
        for(int i=0;i< arr.length;i+=2)
        {
            sum +=arr[i];
        }
        System.out.println("sum of all the elements present on even indices is: "+sum);
    }
}
```

Q2: Write a program to traverse over the elements of the array using for each loop and print all even elements.
Solution:

```
public class PrintEven {
    public static void main(String[] args) {
        int arr[]={34,21,54,65,43};
        for (int i:arr)
        {
            if(i%2==0)
                System.out.println(i);
        }
    }
}
```

Q3: Write a program to calculate the maximum element in the array.
Solution:

```
public class LargestNum {
    public static void main(String[] args) {
        int arr[]={34,21,54,65,43};
        int max=0;
        for(int i=0;i<arr.length;i++)
        {
            if (arr[i]>max)
                max=arr[i];
        }
        System.out.println("Largest element is: "+max);
    }
}
```

Q4: Write a program to find out the second largest element in a given array.
Solution:

```
public class SecondLarge {
    public static void main(String[] args) {
        int arr[] = {34, 21, 54, 65, 43};
        int max = Integer.MIN_VALUE;
        int large = Integer.MIN_VALUE;
        for (int i = 0; i < arr.length; i++) {
            if (arr[i] > max)
                max = arr[i];
        }
        for (int i = 0; i < arr.length; i++) {
            if (arr[i] > large && arr[i]!=max) {
                large = arr[i];
            }
        }
        System.out.println("second largest element is: " + large);
    }
}
```

Q5: Given an array. Find the first peak element in the array. A peak element is an element that is greater than its just left and just right neighbor.
Solution:

```
public class PeakFinder {
    public static void main(String[] args) {
        int arr[] = {1, 3, 2, 6, 5};
        for (int i = 1; i < arr.length - 1; i++) {
            if (arr[i] > arr[i - 1] && arr[i] > arr[i + 1]) {
                System.out.println("First Peak Element: "+arr[i]);
                break;
            }
        }
    }
}
```

2D-Array

Q1: Take m and n input from the user and m * n integer inputs from user and print the following:
number of positive numbers

number of negative numbers
number of odd numbers
number of even numbers
number of 0.
Solution:

```
import java.util.Scanner;

public class PrintNum {
    public static void main(String[] args) {
        Scanner sc=new Scanner(System.in);
        int pos=0,neg=0,odd=0,even=0,zero=0;
        System.out.println("Enter no. of rows");
        int m=sc.nextInt();
        System.out.println("Enter no. of columns");
        int n=sc.nextInt();
        int arr[][]=new int[m][n];
        System.out.println("Enter elements");
        for (int i=0;i<m;i++){
            for (int j=0;j<n;j++){
                arr[i][j]=sc.nextInt();
            }
        }
        for (int i=0;i<m;i++) {
            for (int j = 0; j < n; j++) {
                if (arr[i][j] > 0) pos++;
                if (arr[i][j] < 0) neg++;
                if (arr[i][j] % 2 != 0) odd++;
                if (arr[i][j] % 2 == 0) even++;
                if (arr[i][j] == 0) zero++;
            }
        }
        System.out.println("number of positive numbers: "+pos);
        System.out.println("number of negative numbers: "+neg);
        System.out.println("number of odd numbers: "+odd);
        System.out.println("number of even numbers: "+even);
        System.out.println("number of zeroes: "+zero);

    }
}
```

Q2: write a program to print the elements above the secondary diagonal in a user Inputted square matrix.
Solution:

```
import java.util.Scanner;

public class AboveSecDiag {
    public static void main(String[] args) {
        Scanner sc = new Scanner(System.in);
        System.out.println("Enter number of Rows:");
        int m = sc.nextInt();
        System.out.println("Enter number of coulmns:");
        int n = sc.nextInt();
        // Square Matrix
        int arr[][] = new int[m][n];
        System.out.println("Enter elements");
        for (int i = 0; i < m; i++) {
            for (int j = 0; j < n; j++) {
                arr[i][j] = sc.nextInt();
            }
        }
        System.out.print("The elements above secondary diagonal are: ");
        for (int i = 0; i < m; i++) {
            for (int j = 0; j < n; j++) {
                if (i + j < m - 1) System.out.print(arr[i][j]+" ");
            }
        }
    }
}
```

Q3: write a program to print the elements of both the diagonals in a user inputted Square matrix in any order.
Solution:

```
import java.util.Scanner;

public class PrintDiag {
    public static void main(String[] args) {
        Scanner sc = new Scanner(System.in);
```

```
        System.out.println("Enter no. of rows");
        int m = sc.nextInt();
        System.out.println("Enter no. of columns");
        int n = sc.nextInt();
        int arr[][] = new int[m][n];
        System.out.println("Enter elements");
        for (int i = 0; i < m; i++) {
            for (int j = 0; j < n; j++) {
                arr[i][j] = sc.nextInt();
            }
        }
        System.out.print("The elemnts of both diagonal: ");
        for (int i = 0; i < m; i++) {
            for (int j = 0; j < n; j++) {
                if (i == j) System.out.print(arr[i][j] + " ");
               else if (i + j == m - 1) System.out.print(arr[i][j] + " ");
            }
        }
    }
}
```

Q4: Write a java program to find the largest element of a given 2D array of integers.
Solution:

```
import java.util.Scanner;
public class FindLargest {
    public static void main(String[] args) {
        Scanner sc = new Scanner(System.in);
        System.out.println("Enter no. of rows");
        int m = sc.nextInt();
        System.out.println("Enter no. of columns");
        int n = sc.nextInt();
        int arr[][] = new int[m][n];
        System.out.println("Enter elements");
        for (int i = 0; i < m; i++) {
            for (int j = 0; j < n; j++) {
                arr[i][j] = sc.nextInt();
            }
        }
        int max = Integer.MIN_VALUE;
        for (int i = 0; i < m; i++) {
            for (int j = 0; j < n; j++) {
                if (arr[i][j] > max) max = arr[i][j];
            }
        }
        System.out.println("The largest element in 2D array is: " + max);
    }
}
```

Q5: Write a function which accepts a 2D array of integers and its size as arguments And displays the elements of middle row and the elements of middle column. Printing can be done in any order. [Assuming the 2D Array to be a square matrix with odd dimensions i.e. 3x3, 5x5,7x7 etc...]
Solution:

```
import java.util.Scanner;
public class PrintMiddle {
    public static void main(String[] args) {
        Scanner sc = new Scanner(System.in);
        //Square Matrix
        System.out.println("Enter no. of rows(odd): ");
        int m = sc.nextInt();
        System.out.println("Enter no. of columns(odd):");
        int n = sc.nextInt();
        int arr[][] = new int[m][n];
        System.out.println("Enter elements");
        for (int i = 0; i < m; i++) {
            for (int j = 0; j < n; j++) {
                arr[i][j] = sc.nextInt();
            }
        }
        System.out.print("The elements of middle row and columns are: ");
        for (int i = 0; i < m; i++) System.out.print(arr[i][m / 2]+" ");
        for (int j = 0; j < n; j++) {
            if (j == m / 2) {
                continue;
            }
            System.out.print(arr[m / 2][j]+" ");
        }
    }
}
```