TIME AND SPACE COMPLEXITY ASSIGNMENT

1) **Analyse 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++;
        }
}
```

Ans: The complexity of this code is O(n^2) as it uses nested loops, where the outer loop runs n times and the inner loop runs i times where i varies from 1 to n.

The total number of operations performed can be calculated by summing the total number of operations in each iteration of the outer loop. The inner loop will run i times on the i-th iteration of the outer loop. So the number of operations is (1+2+3+....+n) which is n(n+1)/2, which is O(n^2).

One way to improve the time complexity of this code is to use a mathematical formula to find the sum instead of using nested loops.

2) **Find the value odT(2) for the recurrence relation t(n) =3T(n-1)+12n,given that t(0)=5.**

Ans: Given T(n) = 3T(n-1)+12n

Substituting the values in the relation :
T(1) =3T(0)+12
=>T(1)=15+12=27
T(2)=3T(1)+12*2
=>T(2)=3*27+24=81+24
Hence T(2)=105.

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

Ans: Let the solution be T(n) =O(n), now let's prove this using the induction method.

For that to happen T(n) <=cn where c is some constant.
T(n)=T(n-1)+c
T(n-1) =T(n-2)+c
T(n-2) =T(n-3)+c
|

|
T(2) =T(1)+c
—---------------------------Adding all above equations
T(n) = k+cn
Let us assume T(1) to be a constant value.
T(n)=k+cn
Therefore, T(n)<=cn
Hence we can conclude T(n) =O(n).

4) **Given a recurrence relation:**
   T(n) =16T(n/4)+n2log_n
   Find the time complexity of this relation using the master theorem.

Ans: From the given recurrence relation we can obtain the value of different parameters such as a,b,p and k.

The relation : T(n) =16T($\frac{n}{4}$)+n2 logn

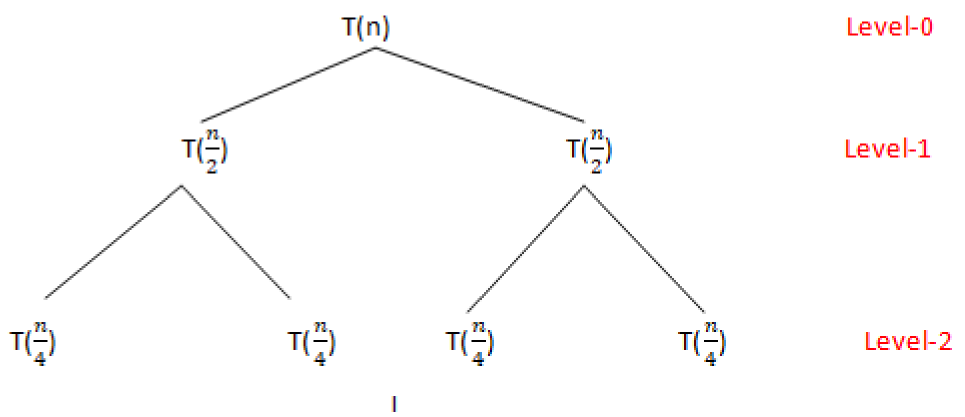
Here,a=16 b=4 k=2 p=1
b^k=4^2=16
Here a=b^k
Also p>1
Hence T(n) =θ(n$log_4$16*log1+1n)=θ(n$\frac{1}{2}log_2$n)

5) **Solve the following recurrence relation using recursion tree method**
   T(n) =2T($\frac{n}{2}$)+n

Ans: The given recurrence relation shows-
A problem of size n will get divided into 2 subproblems of size n/2.

- Then, each sub-problem of size $\frac{n}{2}$ will be divided into 2 subproblems of size $\frac{n}{4}$ and so on.
- At the bottom most layer, the size of sub-problems will be reduced to 1.

This is illustrated through following recursion tree-

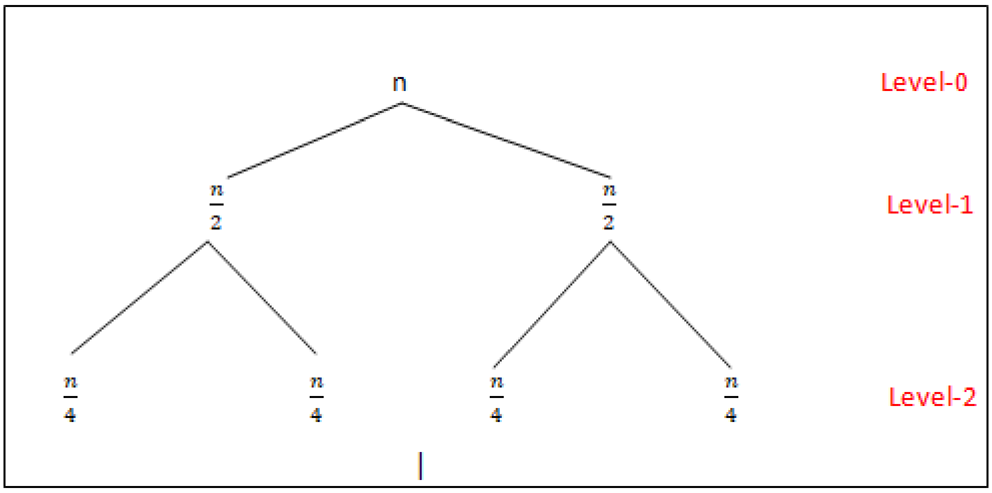


The given recurrence relation shows-

- The cost of dividing a problem of size n into its 2 sub-problems and then combining its solution is n.

- The cost of dividing a problem of size n/2 into its 2 sub-problems and then combining its solution is n/2 and so on.

This is illustrated through following recursion tree where each node represents the cost of the corresponding sub- problem



Step-02:
Determine cost of each level-

- Cost of level-0 = n
- Cost of level-1 = n/2+ n/2 =n
- Cost of level-2 = n/4 + n/4 + n/4 + n/4 = n and so on.

Step-03:
Determine total number of levels in the recursion tree-

- Size of sub-problem at level-0 =n/20
- Size of sub-problem at level-1 =n/21
- Size of sub-problem at level-2 =n/22

Continuing in similar manner, we have
Size of sub-problem at level-i = n/2i
Suppose at level -x (last level), size of the sub-problem becomes 1. Then n/2x=1
2x = n
Taking log on both sides(with base 2),we get
xlog2= logn
x= log2n
∴ Total number of levels in the recursion tree = log2n +1

Step-04 :
Determine number of nodes in the last level-

- Level-0 has 20 nodes i.e 1 node
- Level-1 has 21 nodes i.e 2 nodes
- Level-2 has 22 nodes i.e 4 nodes

Continuing in similar manner,we have Level-log2n has 2log2n nodes i.e n nodes

Step-05:
Determine cost of last level
Cost of last level = n * T(1) = θ(n)

Step-06:
Add costs of all the levels of the recursion tree and simplify the expression so obtained in terms of asymptotic notation-

$$T(n) = \underbrace{\{ n+n+n+......\}}_{\text{For } log_2 n \text{ levels}}+\theta(n)$$

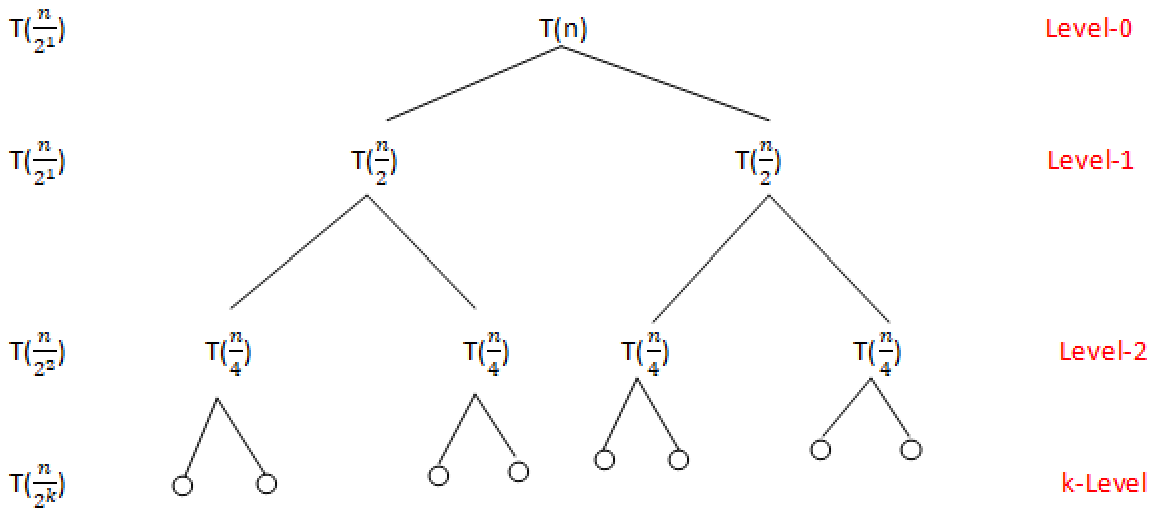
$= n * log_2 n + \theta(n)$

$= n log_2 n + \theta(n)$

$= \theta(n log_2 n)$

6) T(n) =2T(n/2) +k,solve using recurrence tree method.

Ans: Step1. Drawing Recursion Tree



T(n)= 2T(n/2)+k

Step-2. Calculating height of tree
As we know that (n/2^k) =1
n=2^k
Taking log both sides
log(n) = log(2^k)
log(n) = klog(2)
k= log(n)/log(2)
k= log2(n)
Height of tree is log(n) base 2

Step3: Calculate cost at each level

Level 0 = K
Level 1 = K + K = 2K
Level 2 = K + K + K + K = 4K and so on..

Step 4: Calculate number of nodes at each level
Level 0 = 2^0 = 1
Level 1 = 2^1 = 2
Level 2 = 2^2 = 4 and so on..

Step 5: Calculating final cost:
The total cost can be written as,

Total cost =cost of all levels except last level + Cost of last level
Total cost =cost of level-0 + cost of level-1 + cost of level-2 + …… +cost of level-log(n) + Cost of last level

The cost of the last level is calculated separately because it is the base case and no merging is done at the last level so, the cost to solve a single problem at this level is some constant value. Let's take it as O(1)

Let's put the values into the formulae,
T(n) = K+ 2*k + 4*k + …+ long(n) times + O(1) *n
T(n) = K (1+ 2 + 4 + …+ long(n) times) + O(n)
T(n) = K (2^0+ 2^1+ 2^2 + …+ long(n) times) + O(n)

In the GP formed above, a=1 and r=2,after solving this we get,
T(n) = k*(1/(2-1)) +O(n)
T(n) = k + O(n)
T(n) = O(n)

1D ARRAY ASSIGNMENT

1) **Write a program to print the sum of all the elements present on even indices in the given array.**

Input 1: arr[] = {3,20,4,6,9}
output 1 : 16
Input 2: arr[] = {4,3,6,7,1}
output 2 : 11
Ans: ass_1Dcode1

2) **Write a program to traverse over the elements of the array using for each loop and print all even elements.**

Input 1: arr[] = {34,21,54,65,43}
output 1 : 34 54
Input 2: arr[] = {4,3,6,7,1}
output 2 : 4 6
Ans: ass_1Dcode2

3) **Write a program to calculate the maximum element in the array.**

Input 1: arr[] = {34,21,54,65,43}
output 1 : 65
Input 2: arr[] = {4,3,6,7,1}
output 2 : 7
Ans: ass_1Dcode3

4) **Write a program to find out the second largest element in a given array.**

Input 1: arr[] = {34,21,54,65,43}
output 1 : 54
Input 2: arr[] = {4,3,6,7,1}
output 2 : 6
Ans: ass_1Dcode4

5) **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 neighbour.**

Input 1: arr[] = {1,3,2,6,5}
Output 1: 6
Input 2: arr[] = {1 4,7,3,2,6,5}
Output 2: 7
Ans: ass_1Dcode5

2D ARRAY ASSIGNMENT

1) 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.
   Input 1:
   1 2 -3 4
   0 0 -4 2
   -4 -5 -7 0

   Output :
   number of positive numbers = 7
   number of negative numbers = 6
   number of odd numbers = 7
   number of even numbers = 9
   number of 0 = 3
   Ans: ass_2Dcode1

2) write a program to print the elements above the secondary diagonal in a user inputted square matrix.
   Input 1:
   1 2 3
   4 5 6
   7 8 9

   Secondary diagonal elements are 3 5 7
   Primary diagonal elements are 1 5 9

   Output : 1 2 4
   Ans : ass_2Dcode2

3) write a program to print the elements of both the diagonals in a user inputted square matrix in any order.
input 1:
```
1   2   3
4   5   6
 7  8   9
```
Secondary diagonal elements are 3 5 7
Primary diagonal elements are 1 5 9

Output : 1 3 5 7 9
Ans: ass_2Dcode3

4) Write a program to find the largest element of a given 2D array of integers.
input 1:
```
1 2 4 0
2 5 7 -1
4 2 6 9
```
Output : 9

Ans: ass_2Dcode4

5) 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...]
input 1:
```
1 2 3 4 5
3 4 5 6 7
7 6 5 4 3
8 7 6 5 4
1 2 37 8 0
```
Output : 3 5 5 6 37 7 6 4 3

Ans: ass_2Dcode5