Tutorial - 2

****

In this tutorial, we will go through the concepts of recursion, time and space complexities, and how to solve recurrence relations

****

Recursion:

Question 1)

```c
#include<stdio.h>



void func(int n){

    if(n>0){
        func(n-1);
        printf("%d ",n);
        func(n-1);
    }

}

void main(){
    func(4);


}
```

```
1 2 1 3 1 2 1 4 1 2 1 3 1 2 1
```

Explanation:
Draw recursion tree and traverse in top to bottom, left to right fashion

Question 2)

```c
#include<stdio.h>
int c=0;

void func(int n){

    if(n==0){
        return;
    }
    c++;
    func(n/10);

}

void main(){

    func(123456789);
    printf("Value of c is %d",c);

}


```

Output:

Value of c is 9

Explanation:

Draw a recursion tree and traverse in top to bottom fashion. Final value of c=9, after that we reach the base case n=0 and so return.

Efficiency of Algorithms:

Question 1)

Find the Time Complexity and Space Complexity of the following snippet:

int a = 0, b = 0;

```
for (i = 0; i < N; i++) {
    a = a + rand(); //rand() returns a random number
}
for (j = 0; j < M; j++) {
    b = b + rand();
}
```

Answer:

Time Complexity = O(m+n)
Space Complexity = O(1) (constant number of variables)

Question 2)

Find the Time Complexity of the following snippet:

```
int a = 0;
for (i = 0; i < N; i++) {
    for (j = N; j > i; j--) {
        a = a + i + j;
    }
}
```

Answer:

code runs total no of times
= N + (N – 1) + (N – 2) + ... 1 + 0
= N * (N + 1) / 2
= 1/2 * N^2 + 1/2 * N
O(N^2) times.

Question 3)

Find the Time Complexity of the following snippet:

```
int a = 0, i = N;
while (i > 0) {
```

```
    a += i;
    i /= 2;
}
```

Answer:

O(logn)

Question 4)

Find the Time Complexity of the following function:

```
int fun(int n) {
    for (int i = 1; i <= n; i++) {
        for (int j = 1; j < n; j += i) {
            // Some O(1) task
        }
}
}
```

Answer:

O(n*log n)

Explanation:

Harmonic Series:

$$\sum_{k=1}^{n} \frac{1}{k} = 1 + \frac{1}{2} + \ldots + \frac{1}{n} \approx \ln n$$

For i = 1, the inner loop is executed n times.
For i = 2, the inner loop is executed approximately n/2 times. For i = 3, the inner loop is executed approximately n/3 times. For i = 4, the inner loop is executed approximately n/4 times.
……………………………………………………
……………………………………………………
For i = n, the inner loop is executed approximately n/n times.

The total time complexity of the above algorithm is (n + n/2 + n/3 + … + n/n)
= n * (1/1 + 1/2 + 1/3 + … + 1/n)
[Recall the complexity of harmonic series]

Hence, the time complexity of fun is O(nLogn)

Question 5)

Find the Time Complexity of the following function:

```
void demo()
{
   int i, j;
   for (i=1; i<=n; i++)
      for (j=1; j<=log(i); j++)
         System.out.println("Hello World");
}
```

Answer:

$\Theta(\log 1) + \Theta(\log 2) + \Theta(\log 3) + \ldots + \Theta(\log n)$
$= \Theta(\log n!)$
$= \Theta(n \log n)$

Solving Recurrence Relations:

Question 1)

Solve the following recurrence:

T(n) = 2T(n/2) + n
T(1) = 0

Answer:

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

= 2[2 T(n/2^2) + n/2] +n = 2^2*T(n/2^2) + 2n =
2^2 *[2T(n/2^3)+n/2^2]+2n = 2^3 T(n/2^3) + 3n
……..
= 2^k T(n/2^k) + kn
If n/2^k = 1 => n = 2^k => k=log n (with base 2)
Complexity = O(nlogn)

Question 2)

Find time complexity of the recursive fibonacci method

```
public static int rFib( int n) {
if (n == 0 || n == 1)
return 1;
else
return rFib (n-1) + rFib (n-2);
}
```

Answer:
O(2^n)

Few Basic Questions:

Find the time complexity of the given expressions?

Q1) 3 n log n + 2 n^1.8 **Ans:** O(n^1.8)

Q2) 8 log n + 4 log log n **Ans:** O(logn)

Q3) 3n + 2 (log n)^2 + 4 log n **Ans:** O(n)

Q4) An algorithm takes 0.5 ms for input size 100. How large a problem can be solved in 1 min if the running time is Linear (assume low-order terms are negligible)

Answer: x/100 = 60,000/0.5

solving for x gives an input size of 12,000,000

Q5) An algorithm takes 0.5 ms for input size 100. How large a problem can be solved in 1 min if the running time is O(nlogn) (assume low-order terms are negligible)

Answer: xlogx /100log100 = 60,000/0.5
solving for x gives an input size of 3,656,807

Q7)

Apply Master's Theorem

$T(n) = 64\ T(n/8) - n^2 \log n$

Ans. Master's Theorem cannot be applied here because the recurrence has a -ve sign which denotes that if you don't do extra work $(n^2 \log n)$ the problem gets divided into subproblems. Master's Theorem says this is an invalid equation form.

Q8)

Apply Master's Theorem

$T(2^k) = 3T(2^{k-1}) + 1;\ T(1) = 1$

Ans. Let $2^k = n$

$\Rightarrow 2^{k-1} = 2^k/2 = n/2$

$\Rightarrow T(n) = 3T(n/2) + 1$

$\Rightarrow T(n) = Ø(n^{\log 3})$ [theta]

$\Rightarrow T(n) = Ø(2^{k\log 3})$

$\Rightarrow T(n) = Ø(3^k)$

Homework Questions:

Q1) An algorithm takes 0.5 ms for input size 100. How large a problem can be solved in 1 min if the running time is Quadratic (assume low-order terms are negligible)

Q2) An algorithm takes 0.5 ms for input size 100. How large a problem can be solved in 1 min if the running time is Cubic (assume low-order terms are negligible)

Q3) Use the definition of Big-Oh to prove that $n^{1+0.001}$ is not O(n).

Q4) When $n = 2^{2k}$ for some k ≥ 0, the recurrence relation

$T(n) = \sqrt{(2)}\ T(n/2) + \sqrt{n},\ T(1) = 1$

evaluates to?