Problems On Time Complexity

Assignment Solutions

PW SKILLS

Q1. Calculate the time complexity of the given function:

```
public static void fun(int n)
{
    if (n < 5)
        System.out.print("College Wallah");
    else {
        for (int i = 0; i < n; i++) {
            System.out.print(i + " ");
        }
    }
}
```

Answer: O(1) in best case and O(n) in worst case.

Explanation: This program contains if and else conditions. Hence, there are 2 possibilities of time complexity. If the value of n is less than 5, then we get only College Wallah as output and its time complexity will be O(1). But, if n>=5, then for loop will execute and time complexity becomes O(n), which is considered the worst case because it takes more time.

Q2. Calculate the time complexity of the given function:

```
public static void function(int n)
{
    int i = 1, s = 1;
    while (s < n) {
        s = s + i;
        i++;
    }
}
```

Answer: O(√n)

Explanation: We can define the 'S' terms according to the relation Si = Si-1 + i. Let k is the total number of iterations taken by the program

i	s
1	1
2	2
3	2+2
4	2 + 2 + 3
...	...
k	2 + 2 + 3 + 4 + ...+ k

When S>=n , then loop will stop at kth iterations,

$\Rightarrow$ S>=n $\Rightarrow$ S=n

$\Rightarrow$ 2 + 2 + 3 + 4 + ...+ k = n

$\Rightarrow$ 1 + (k * (k + 1))/2 = n

$\Rightarrow k^2 = n$

$k = \sqrt{n}$

Hence, the time complexity is $O(\sqrt{n})$.

Q3. Calculate the time complexity of the given function:

```
public static void fun(int a, int b){
    while (a != b) {
        if (a > b)
            a = a - b;
        else
            b = b - a;
    }
}
```

Answer: Time complexity = O(1) in best case and O(max(a, b)) worst case.

Explanation: If the values of a and b are the same, then the while loop will not be executed. Hence, time complexity will be O(1).
But if a!=b, then the while loop will be executed. Let a=16 and b=5;

No. of iterations	a	b
1	16	5
2	16-5=11	5
3	11-5=6	5
4	6-5=1	5
5	1	5-1=4
6	1	4-1=3
7	1	3-1=2
8	1	2-1=1

For this case, while loop executed 8 times (a/2⇒16/2⇒8).
If a=5 and b=16, then also the loop will be executed 8 times. So we can say that time complexity is O(max(a/2,b/2))⇒O(max(a, b)), which is considered the worst case because it takes more time.

Q4. Calculate the time complexity of the given function:

```
public static void fun(int n, int x){
   for (int i = 1; i < n; i = i * x)
        System.out.println("hello");
}
```

Answer: $O(\log_x n)$

Explanation: Let k be the no. of iteration of the loop.

No. of itr	i=i*x
1	1*x=x
2	$x*x=x^2$
3	$x^2*x=x^3$
...	...
k	$(x^k-1)*x= x^k$

⇒ The loop will stop when i>=n ⇒ $x^k = n$
⇒ $x^k = n$ (Take log both sides)
⇒ $k = \log_x n$
⇒ Hence, time complexity is $O(\log_x n)$.

Q5. Calculate the time complexity of the given function:

```
public static void fun(int n){
for (int i = 0; i < n / 2; i++)
        for (int j = 1; j + n / 2 <= n; j++)
            for (int k = 1; k <= n; k = k * 2)
                System.out.println("hello");
}
```

Answer: $O(n^2 \log_2 n)$.

Explanation:
Time complexity of 1st for loop = $O(n/2)$ ⇒ $O(n)$.
Time complexity of 2nd for loop = $O(n/2)$ ⇒ $O(n)$.
Time complexity of 3rd for loop = $O(\log_2 n)$.
Hence, the time complexity of function will become $O((n^2 \log_2 n)$.