Design and Analysis of Algorithms

Tutorial - 2

Sharad Gumber
CST, 12

Q.1 what is the time complexity of below code and how?

```
void fun (int n) {
    int j = 1, i = 0;
    while (i < n) {
        i = i + j;
        j++; } }
```

Sol On the execution of while loop :-

1^{st} iteration, $i = 1$

2^{nd} iteration, $i = 1 + 2$

3^{rd} iteration, $i = 1 + 2 + 3$

4^{th} iteration, $i = 1 + 2 + 3 + 4$

$\therefore$ For i times, $i = (1 + 2 + 3 + 4 - - - + i)$

this makes the series where sum $\Rightarrow i = \frac{i(i+1)}{2}$

Now, $i < n$ (for complexity to exist upper bond)

$\Rightarrow \frac{i^2+1}{2} < n$

$\Rightarrow i^2 < n$ (removing lower order)

$\Rightarrow i = \sqrt{n}$

$\Rightarrow$ Complexity $= O(\sqrt{n})$.

Sharad

Q-2 Write recurrence relation for the recursive function that prints Fibonacci series. Solve the recurrence relation to get time complexity of the program. What will be the space complexity of this program and why?

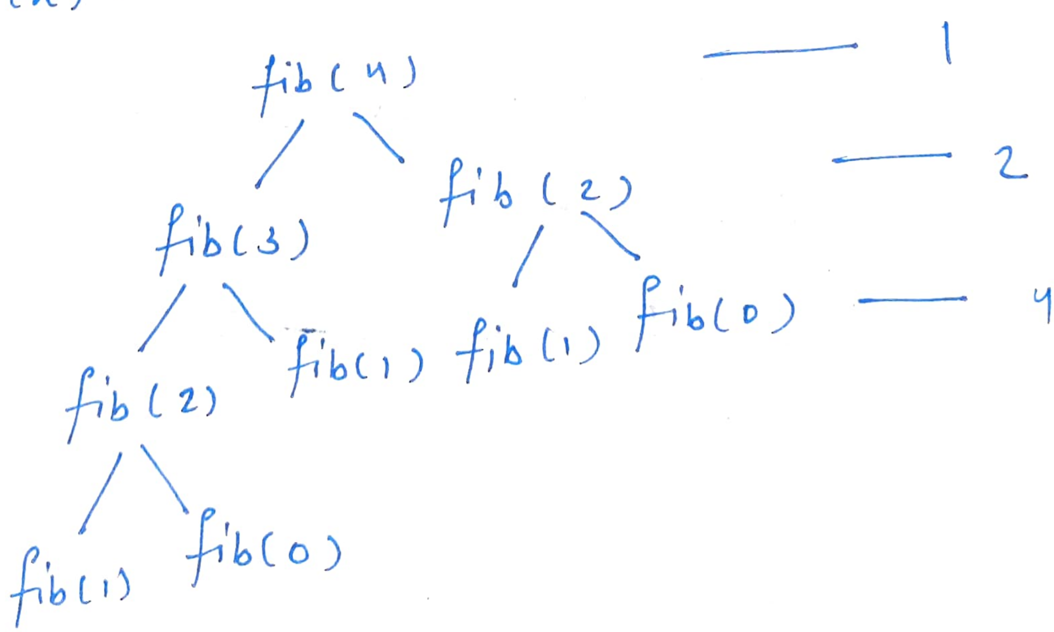
Sol

```
int fib( int n )
{
  if (n <= 1)
      return n;

  else
  return fib (n-1) + fib (n-2);

}
```

$T(n) = T(n-1) + T(n-2) + 1$



Sharad

$$T(n) = 1 + 2 + 4 + 8 + \cdots + n$$

$$T(n) = \frac{a(r^n - 1)}{r - 1}$$

$$= \frac{1(2^{n+1} - 1)}{2 - 1}$$

$$T(n) = 2\,2^n - 1$$

$$T(n) = O(2^n)$$

Space Complexity = $O(1)$

As recursive implementation doesn't store any values from and calculate every value from scratch.

So space complexity is $O(1)$.

Sharad

Q.3 Write programs which have complexity
$\Rightarrow$ $n\log n$, n^3, $\log(\log n)$

Sol Quick Sort :-

```
# define MAX 100
# include <stdio.h>
void quicksort (int [], int , int );
int main ()
  {
    int n, t=0;
    Scanf ("%d", &n);
    int A[n];
    for (int i=0; i<n; i++)
      {
        Scanf ("%d", &A[i]);
      }
    quicksort (A, t, n-1);
    for (int i=0; i<n; i++)
      {
        printf ("%d", A[i]);
      }
  }

void quicksort (int A[], int lb, int ub)
  {
    int i=lb, j=ub, Key = A[lb], t=0;
    if (lb >= ub)
      {
```

Sharad

```
return;
}

while (i<=j)
{
 while (Key >= A[i] && i<j)
   {
    i++;
   }
 while (Key < A[j])
  {  j--;
  }
 if (i<j)
  {
   t = A[i];
   A[i] = A[j];
   A[j] = t;
   }
}
 A[lb] = A[j];
 A[j] = Key;

quickSort (A, lb, j-1);

quick sort (A, j+1, ub);

}
```

Here time complexity of quick sort is n(logn).

Sharad

3 - variable equation solution:-

```
#include <stdio.h>
int main()
{
int A [15];
int p= 0;
for (int x=0; x<n; x++)
{
for (int j=0; j<n; j++)
{
for (int K=0; K<n; K++)
{
if (3*x + 9*j + 8*K
{
A[p] = x;
A[p+1] = j;
A[p+2] = K;
}
}
}
}
}
```

Here, Time complexity is $O(n^3)$.

Sharan

when loop variable expands or shrinks:-

```
# include <stdio.h>
int main ()
{
int n;
Scanf("%d", &n);
int K = 1;
for (int i = 2; i <= n; i = pow(i, K))
{
K ++;
}
}
```

Hence, Time complexity is $O(\log(\log n))$.

Prasad

Q.4 Solve the following recurrence relation

$$T(n) = T(n/4) + T(n/2) + Cn^2$$

Sol Here, we assume

$$T(n/4) \le T(n/2)$$

~~T(n/4)~~ $T(n) = 2T(n/2) + Cn^2$

$\Rightarrow$ Applying master's method

$a = 2, \; b = 2$

$c = \log_2^2 = 1$

$n^c = n^1 = n$

Comparing n with $f(n)$

$\therefore \; n < n^2$

Time Complexity $= O(n^2)$.

Sharad

Q.5 what is the time complexity of following function fun()?

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

Sol 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.

So, the total time complexity of the above algorithm is $(n + n/2 + n/3 + \cdots + n/n)$.

which becomes $n * (1/1 + 1/2 + 1/3 + \cdots + 1/n)$.

The important thing about series $(1/1 + 1/2 + 1/3 + \cdots + 1/n)$ is, it equal to $O(\log n)$.

So, the time complexity of the above code is $O(n \log n)$.

Q.6 what should be the time complexity of

```
for (int i=2; i<=n; i = pow(i, k))
{
  // Some O(1) expressions or statements
}
```

Where, K is a constant.

Sol Here,

i takes value, $2, 2^K, (2^K)^K, ((2^K)^K)^K \ldots$

$2^{K^{\log_K(\log(n))}}$

Here last term must be equal or less than n and we have $2^{K^{\log_K(\log(n))}} = 2^{\log(n)} = n$ which is equal to the last term.

Total iteration $= \log_K(\log(n))$ which take constant amount of time to run.

So, time complexity is $O(\log(\log(n)))$.

Q.8 Arrange the following in increasing order of rate of growth :-

(a) n, $n!$, $\log n$, $\log\log n$, root (n), $\log(n!)$, $n\log n$, $\log^2 n$, 2^n, 2^{2^n}, 4^n, n^2, 100.

Sol $100 < \log\log n < \log n < (\log n)^2 < \text{root}(n) < n < n\log n < \log(n!) < n^2 < 2^n < 4^n < 2^{2^n}$

(b) $2(2^n)$, $4n$, $2n$, 1, $\log(n)$, $\log(\log(n))$, $\sqrt{\log(n)}$, $\log 2n$, $2\log(n)$, n, $\log(n!)$, $n!$, n^2, $n\log(n)$.

Sol $1 < \log\log n < \sqrt{\log(n)} < \log(n) < \log 2n < 2\log(n) < n < n\log n < 2n < 4n < \log(n!) < n^2 < n! < 2^{2^n}$

(c) 8^{2n}, $\log_2(n)$, $n\log_6(n)$, $n\log_2(n)$, $\log(n!)$, $n!$, $\log_8(n)$, 96, $8n^2$, $7n^3$, $5n$

Sol $96 < \log_8(n) < \log_2(n) < 5n < n\log_6(n) < n\log_2(n) < \log(n!) < 8n^2 < 7n^3 < n! < 8^{2n}$

Sharad