TANISHQ
CST
34

Tanishq

Date / /

Tutorial-2

1. What is the time complexity of below code & how?

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

Time complexity - O(sqrt n)

1^{st} time $i=1$

2^{nd} time $i=3$ $(i=i+2)$

3^{rd} time $i=6$ $(i=1+2+3)$

⋮

n^{th} time $i=\frac{x(x+1)}{2} = x^2 < n$

$x = sqrt(n)$

2. Write recurrence relation for the recursive function that prints fibonacii series. Solve the recurrence relation to get complexity of the program. What will be the space complexity of this program & why.

⇒ * fib(n) = fib(n-1) + fib(n-2)

```
fib(n):
    if (n<=1)
        return 1
    return fib(n-1) + fib(n-2)
```

Tanishq

Date / /

Time complexity

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

$= 2T(n-2) + c$ (Let $T(n-1) \simeq T(n-2)$)

$T(n-2) = 2 * (2T(n-2) + c) + c$

~~= 2 * (2T~~

$= 4T(n-2) + 3c$

$T(n-4) = 2 * (4T(n-2 + 3c) + c$

$= 8T(n-3) + 7c$

$= 2^k \times T(n-3k) + (2^k - 1)c$

$n - K = 0 \Rightarrow n = K$

$T(n) = 2^n * T(0) + (2^n - 1)c$

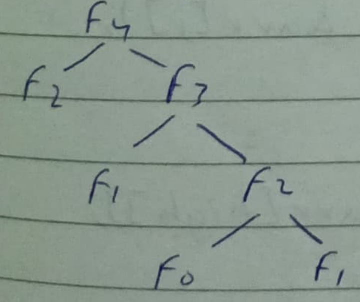
$2^n \times 1 + 2^n c - c$

$= 2^n(1 + c) - c$

$\simeq 2^n$ //constant can be ignored

Space complexity

The space is proportional to the maximum depth of the recursion tree



Hence the ~~f~~ space complexity of fibonnacii recursive is O(N)

Tanishq

Date / /

3. Write programs which have complexity - n log n), n^3, log (log n)

Sol => ~~Merge sort - nlogn~~ Quick sort - nlogn

```
void quick sort (int arr[], int low, int high)
{   if (low < high)
    { int pi = partition (arr, low, high);
      quick sort (arr, low, pi-1);
      quicksort ( ~~low~~ arr, pi+1, high);
    }
}

int partition (int arr[], int low, int high)
{
    int pivot = arr[high];
    int i = (low -1);
  for (int j = low; j <= high-1; j++)
  {
      if (arr [j] < pivot)
      {
          i++;
          swap(& arr [i], &arr[j]);
      }
  }
  swap (&arr[i+1], &arr[high]);
  return (i+1);
}
```

Tanishq

Date / /

ii) n^3

Multiplication of 2 sq. matrix

```
for (i = 0; i < r1; i++)
{
  for (j = 0; j < c2; j++)
  {
    for (k = 0; k < c1; k++)
    {
      res[i][j] = a[i][k] + b[k][j];
    }
  }
}
```

iii) log (log n)

```
for (i = 2; i < n; i = i * i)
{
    count++;
}
```

4. Solve the following recurrence relation

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

$T(n) = 2T\left(\frac{n}{2}\right) + cn^2$ $\qquad T\left(\frac{n}{2}\right) \geq T\left(\frac{n}{4}\right)$

Using master's method

$T(n) = aT(n/b) + f(n)$

$a \geq 1, b > 1, c = \log_b a$ Comparing n^c & $f(n)$

we get $c = \log_2 2 = 1$

$f(n) > n^c$

$T(n) = O(f(n)$

$= O(n^2)$

Tanishq

Date / /

5. What is the time complexity of 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 }}}
```

Sol ⇒ for $i=1 \rightarrow j=1,2,3,4 --- n$ (run for n times)
for $i=2 \rightarrow j=1,3,5, ---$ (run for $n/2$ times)
for $i=3 \rightarrow j=1,4,7 ---$ (run for $n/3$ times)

$$T(n) = n + n/2 + n/3 + n/4 + ---$$
$$= n\left(1 + \tfrac{1}{2} + \tfrac{1}{3} + \tfrac{1}{4} + ---\right)$$
$$= n \int_1^n \frac{1}{x} \Rightarrow n \int_1^n \frac{dx}{x} \Rightarrow \log x \Big]_1^n$$
$$= \underline{n \log n} \text{ (Time complexity)}$$

6. What should be the time complexity of following function

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

where K is a constant

Sol ⇒ for first iteration $i=2$
2^{nd} iteration $i=2^{\wedge k}$
3^{rd} iteration $= i = (2^k)^K = 2^{K^2}$
⋮
n^{th} iteration $i = 2^{K^i}$ loop ends at $2^{K^i} = n$

Tanishq

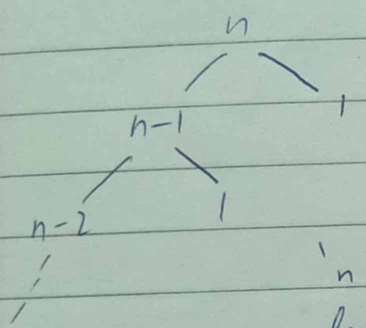
Date / /

Apply log $\log n = \log 2^{k^i} \Rightarrow k^i = \log n$

Again apply log $\log(k^i) = \log n \Rightarrow i = \log_k(\log n)$

7. Write a recurrence relation when Quick Sort repeatedly divides the array in two parts of 99% ~~...~~ & 1%. Derive the time complexity in this case. Show the recursion tree while deriving time complexity & find the difference in heights of both the extreme parts. What do you understand by this analysis?

Sol. $\therefore T(n) = T(n-1) + O(1)$



'n' work is done at each level for merging

$T(n) = (T(n-1) + T(n-2) + \dots T(1) + O(1)$

$= n \times n$

$\therefore \underline{T(n) = O(n^2)}$

Lowest height = 2

Height " = n

$\therefore \underline{\text{diff} = n - 2}$ $n > 1$

The given algorithm produces linear result.

Tanishq

Date / /

8-

a) $100 < \log\log n < \log n < (\log n)^2 < \sqrt{n} < n$
$< n\log n < \log(n!) < n^2 < 2^n < 4^n < 2^{2^n}$

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

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