SANA GARG
CST SPL-2
23
Date ___/___/_____

Saathi

Assignment
DAA
Tutorial - 2

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++;
    }
}
```

i = 0	j =
0	1
1	2
3	3
6	4
10	5
⋮	⋮

No of times loop is running be k.

$S_k = 1 + 3 + 6 + 10 + \ldots T_k$

$S_{k-1} = 1 + 3 + 6 + \ldots + T_{k-1}$

Subtracting both

$S_k - S_{k-1} = 1 + 2 + 3 + 4 + \ldots (k-1)$

$T_k = \frac{(k-1)k}{2}$

Given that k^{th} term is n.

$TK = n$

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

$$\Rightarrow K^2 = n$$
$$\Rightarrow K = \sqrt{n}$$

$$\boxed{T(n) = O(\sqrt{n})}$$

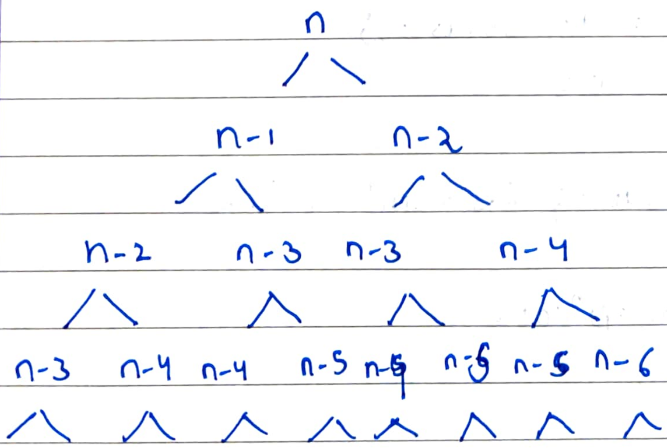
2. $T(n) / = \pi l$

Write reccurance relation for the recursive function that prints fibonacci series. Solve the reccurence relation to get time complexity of the program. What will be the space complexity of this program and why.

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

for reccursive fibonacci solution.

Recursion Tree :



Date __ / __ / ____

No of times function is running will be sum of the series :

$S = 1 + 2 + 4 + \dots + 2^n$

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

Time complexity

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

After removing constant

Q3 → Write programs which have complexity - n(logn), n^3, log(logn)

O(n logn) -

```
# include <iostream>
using namespace std;
int partition (int arr[] int start, int end)
{
    int pivot = ar [start];
    int count = 0;

  for (int i = start; i<=end ; i++)
  {
      if (ar[i] < = pivot)
      count ++ ;
  }
  int pivot_int = start + count;

  swap (arr [pivot_ind], ar (start]);
```

Page No.

Date __/__/__

```
int i= start , j = end;

while (i< pivot_ind && j> pivot_ind )
{
   while (arr [i] <=pivot)
   {
      i++;
   }
   while (arr [j] >pivot )
   {
      j--;
   }

  if (i< pivot_ind && j> pivot_ind)
  {
     swap [ arr [i++] , arr [j--] );
   }
 }
 return pivot_ind;
}

void quick (int ar[], int start, int end )
{
   if (start > = end )
     return;

   int p= Partition (ar, start, end );
   quickSort (ar , start, p -1 );
    quick sort (ar, p+1 , end );
 }
```

Date ___ / ___ / ______

```
int main()
{
  int arr[] = {6, 8, 5, 2, 1}
  int n = 5;
   quick sort (ar, 0, n-1);
   return 0;
 }
```

ii) $O(N^3)$-

```
int main ()
{
   int n = 10;
   for (int i=0; i<n; i++)
    for (int j=0; j<n; j++)
     for (int k=0; k<n; k++)
      {
          printf ("*");
      }
     }
    }
   return 0;
 }
```

iii) O(log (logn))-

```
int count Primes (int n)
{
   if (n< 2)
     return 0;

   boolean non prime - new boolean [n];
```

Date ___/___/___

```
non prime [1] = true;

int numNonprime = 1;

for (int i=2; i<n; i++)
{
    if (non Prime [i])
      continue;

    int j= i * 2;
    while (j<n)
    {
        if (!nonprime [j])
        {
          nonprime [j] = true;
          numNon prime ++;
        }
        j += i;
    }
  }
  return (n-1) - num Non Prim1 ;
}
```

Date ___/___/___

Q4 → Solve the following reccurance relation
$T(n) = T(n/4) + T(n/2) + cn^2$

using Masters theorem

we can assume $T(n/2) >= T(n/4)$

equation can be rewritten as

$$T(n) <= 2T(n/2) + n^2$$

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

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

$$T(n) = \Omega(n^2)$$

$T(n) = O(n^2)$ and $T(n) = \Omega(n^2)$

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

Date ___/___/___

Q5 → 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 += i)
  {
      Some O(1) task
  }}]
```

Ans 5 → For $i = 1$, inner loop is executed n times.
For $i = 2$, inner loop is executed $n/2$ times.
For $i = 3$, inner loop is executed $n/3$ times.

It is forming a series –

$$\Rightarrow n + \frac{n}{2} + \frac{n}{3} + \cdots + \frac{n}{n}$$

$$\Rightarrow n\left(1 + \frac{1}{2} + \frac{1}{3} + \cdots \frac{1}{n}\right)$$

$$= n \times \sum_{K=2}^{n} \frac{1}{K}$$

$$= n \times \log n$$

= Time complexity $= O(n \log n)$ Ans.

Date __ / __ / ____

Saathi

Q6-) What should be the time complexity of

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

where k is a constant.

Ans6-)

```
for (int i = 2; i <= n ; pow(i, k))
  {
     //
  }
```

with iterations

i take values

for 1st iteration → 2

for 2nd iteration → 2^k

for 3rd iteration → $(2^k)^k$

for n iteration → $2^{k \log_k (\log(n))}$

∴ last term must be less than or equal to n.

$2^{k \log(\log(n))} = 2^{\log n} = n$

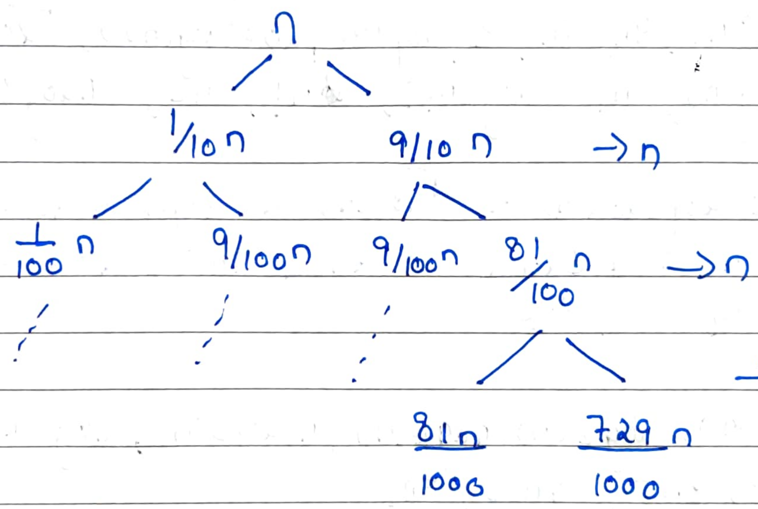
Each iteration takes constant time.

∴ Total iteration = $\log_k (\log(n))$

$\boxed{\text{Time complexity} = O(\log(\log(n)))}$ Ans.

Date __/__/__

Q7) Write a recurrance relation when quick sort repeatedly divides the array into two parts of 99% and 1%. Derive the time complexity in this case. Show the recursion tree while deriving time complexity and find the dif in heights of both the extreme parts. what do you understand by this analysis.



If we split in this manner

Recurrance Relation - $T(n) = T(9n/10) + T(n/10) + O(n)$

Date __/__/__

Q8 → Arrange the following in incresing 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^n), 4^n, n^2, 100

(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) 2(2^n), 4n, 2n, 1, log(n), log(log(n)), $\sqrt{\log(n)}$, log 2n, 2 log(n), n, log(n!), n!, n2, n log(n).

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

c) $8^n(2n)$, $\log_2(n)$, $n\log_e(n)$, $n\log_2(n)$, log(n!), n!, $\log_e(n)$, 96, $8n^2$, $7n^3$, 5n.

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