Name - Abhinav Narain Srivastav

Section - CST SP1-2

Roll no. - 41

DAA Tutorial - 2

(Q1.) What is the time complexity of the below-code.

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

Ans Time complexity = $\Theta(\sqrt{n})$

$\therefore$ 1st time i=1

2nd time i=3 (i=1+2)

3rd time i=6 (i=1+2+3)

⋮

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

$\therefore$ $n \in \Theta(\sqrt{n})$

(Q2.) Write recurrence relation for the recursive function that prints fibonacci series. Solve the recurrence relation to get complexity of the program. What will be the space complexity of this program & why?

Ans * $fib(n) = fib(n-1) + fib(n-2)$

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

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

Time complexity:

Let $T(0)=1$

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

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

$T(n-2) = 2^*(2T(n-2-2))$

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

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

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

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

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

$\Rightarrow n-k=0 \Rightarrow n=k$

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

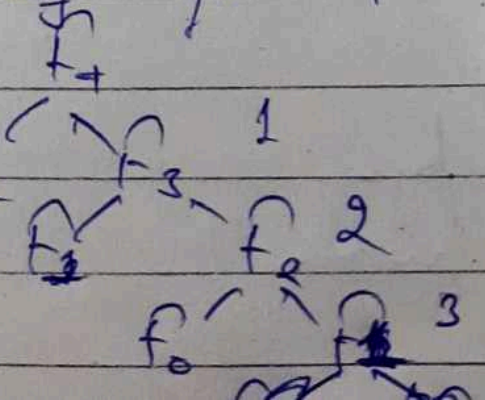
$= 2^n*1+2^nc-c$

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

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

Space complexity: The space is proportional to the maximum depth of the recursion tree.

For e.g.: for f_4



∴ space complexity of fibonacci series = $O(N)$.

(Q3.) Write programs which have complexity - $n(\log n)$, n^3, $\log(\log n)$

Ans Merge sort - $n \log n$

- For time complexity = n^3.

We can use three nested loops - $O(n^3)$

```
for (int i=0; i<n; i++)
{
  for (int j=0; j<n; j++)
  {
    for (int k=0; k<n; k++)
    {
      // Some O(1) expressions
    }
  }
}
```

» For time complexity = $\log(\log n)$

We can use the following function

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

where k is constant

» For time complexity = $n \log n$

We can use the following function

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

(Q4)

Ans

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

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 $c = \log_2 2 = 1$

$\therefore f(n) > n^c \ \because\ n^c > 1$

$\therefore T(n) = \Theta(f(n))$

$\Rightarrow T(n) = \Theta(n^2)$

Q5.) What is 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) expression
    }
```

Ans

for $i=1 \to j = 1, 2, 3, 4, \ldots n$ (Sum for n terms)

for $i=2 \to j = 1, 3, 5, 7 \ldots$ (Sum for $n/2$ terms)

for $i=3 \to j = 1, 4, 7, \ldots$ (Sum for $n/3$ terms)

$T(n) = n + n/2 + n/3 + n/4 + \ldots$

$= n(1 + 1/2 + 1/3 + 1/4 + \ldots)$

$= n \int_1^n 1/n = n \int_1^n dn/n = \log n \int_1^n dn = n \log n$

$\therefore$ The time complexity is $n \log n$

Q6) What should be the time complexity of following function

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

where k is a constant.

Ans for first iteration i=2

2nd iteration $i = 2^k$

3rd iteration $i = (2^k)^k = 2^{k^2}$

:

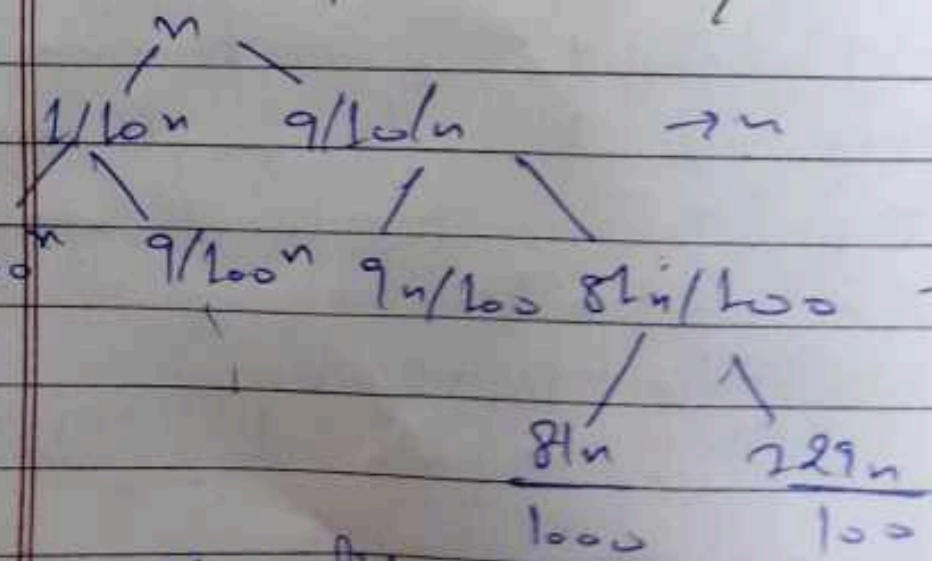
n^{th} iteration $i = (2^{k^{n-1}})$. Let n^{th} term $= 2^{k^{n-1}}$

$\Rightarrow$ applying log, $\log_k n = \log_k 2^{k^{n-1}} = \log_k k^{n-1}$

Applying log again, $\log(\log_k n) = \log(n-1) \approx \log(\log_k n)$

Q7) Write a recurrence relation when quick sort repeatedly divides the array into two parts of 99% & 1%. Derive the time complexity & find the difference in heights of both the extreme parts. What do you understand by worst?

Ans



∴ If we split in the minimum

Recurrence relation = $T(n) = T(99n/100) + T(n/100) + O(n)$

where first branch is of $9n/10$ & second is of $n/10$

Solving the above using recursion-tree approach calculating value

At 1st level, value $= n$

At 2nd level, value $= \frac{9n}{10} + \frac{n}{10}$ = n

Values remains same at all levels of n.

Time complexity = Summation of values

$= O(n \times \log_{10/9} n)$ (upper bound)

$= \Omega(n \log_{10} n)$ (lower bound)

$= O(n \log n)$ ∎

(Q8.) Arrange the following in increasing order of rate of growth:

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

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

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

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

c.) $8^{2n}, \log^k n, n \log^k n, \log(n!), n!, \log^8(n), 96, 8n^2, 7n^3, 5n$.

Ans $96 < \log^8(n) < \log^2(n) < \log 5n < n \log^c n < n \log^2 n < 7n \log^2 n < \log n! < 2^{2n}$