What is the time complexity of below code and why?

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

i	j	time
0	1	1
1	2	1

Scanned with CamScanner

3	3	1
6	4	1
10	5	1

Thus, Total time : $1+3+6+10+\cdots+k = n$

$\Rightarrow \frac{k}{2}[2+(k-1)] = n$

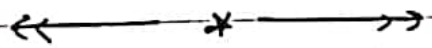
$\Rightarrow k^2+k = 2n$

$\Rightarrow k^2 = n$

$\Rightarrow k = \sqrt{n}$

Total no. of steps = k
T.C. at each step = O(1)

∴ Total time complexity = $O(k*1)$
$= O(\sqrt{n})$

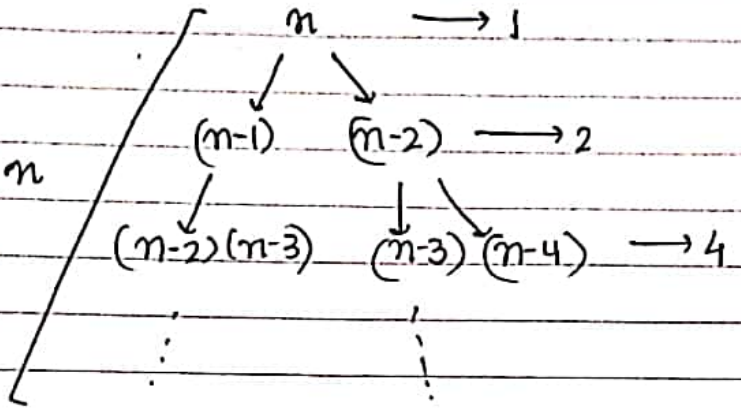


Write recurrence relation for the recursive function that prints fibonacci series. Solve the recurrence relation to get time complexity of the program. What will be the space complexity and why?

Recurrence relation : $T(n) = T(n-1)+T(n-2)+1$

Solving by recurrence tree,

Scanned with CamScanner



depth of the tree is n.

Thus recursive stack will be of size n.

Thus, the space complexity = $O(n)$.

Total no. of steps = 2^n

$\Rightarrow 1+2+4+8+\cdots+2^n = sum$

$\therefore sum = \frac{a(r^{term}-1)}{r-1}$

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

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

Time complexity = $O(2^{n+1}-1) = O(2^n)$.

Scanned with CamScanner

Write a code which has time complexity :

$n \log n$.

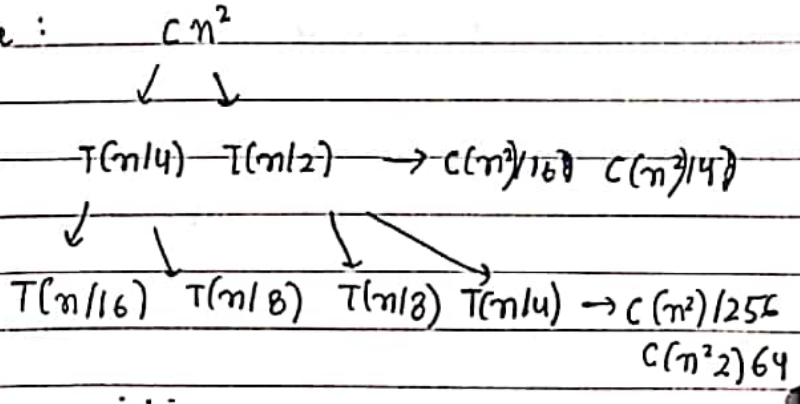
```
int main()
{   int count=0;
    for(int i=0; i<n; i++)
        for(int j=0; j<n; j*=2)
            count++;
}
```

n^3

```
for(int i=0; i<n; i++)
    for(int j=0; j<n; j++)
        for(int k=0; k<n; k++)
            count++;
```

Solve the following recurrence relation $T(n) = T(n/4) + T(n/2) + cn^2$.

Recurrence Tree :



$T(n) = c[n^2 + 5n^2/16 + 25n^2/256 + \cdots +$

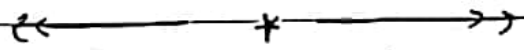
$\therefore$ T.C. $= O(n^2/(1-5/16)) = O(n^2)$ Ans

Scanned with CamScanner

What is the T.C. of the following :

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

∴ Total time complexity = $O(n\sqrt{n})$.



T.C. of :—

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

where, k is a constant.

$i = 2^k, 2^{kk}, 2^{k^{kk}}, \ldots, (2^k)^f = n$

$= 2^k, 2^{k^2}, 2^{k^3}, \ldots, 2^{k^f} = n$

$\Rightarrow 2^{k(\log k \log n)} = 2^{\log_2 n}$

$\Rightarrow$ Total no. of iterations = $\log k \log n$.

∴ T.C. = $O(\log \log(n))$.

Scanned with CamScanner

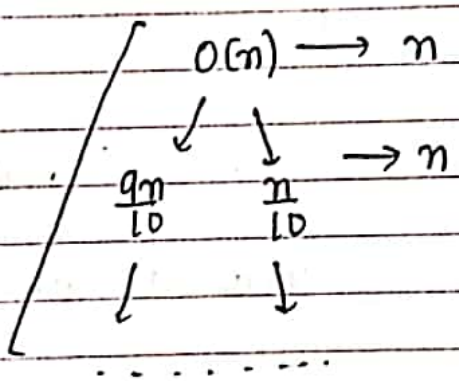
Given, quick sort divides array into two parts of 9% and 1%.

$\Rightarrow \frac{9n}{10} + \frac{n}{10}$ (Supposing size of array = n)

∴ Recurrence relation = $T(n) = T(9n/10) + T(n/10) + O(n)$

↓ complexity of partitioning algorithm.

Solving by Tree Method.



∴ No. of iterations = $\log_{\frac{10}{9}} n$.

T.C. = $O(n * \log_{\frac{10}{9}} n)$ taking longer branch.

$= O(n \log n)$.

Scanned with CamScanner

Arrange the following in increasing order of rate of growth :

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

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

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

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

8^{24}, $\log_2 n$, $n\log_6 n$, $n\log_2 n$, $\log(n!)$, $n!$, $\log_8(n)$, 96, $8n^2$, $7n^3$, $5n$.

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