09-09 Algorithmic Analysis - 2

09 September 2022 19:39

1. Algorithmic analysis
2. Goal of algorithmic analysis
3. types of algo. anal.
   1. Experimental anal.
   2. Asymptotic anal.
4. Types of measurements
   (i) Best case (Ω)
   (ii) Avg case (Θ)
   (iii) Worst case (O)
5. Big-oh notation.

---

(1) Big-oh notation.

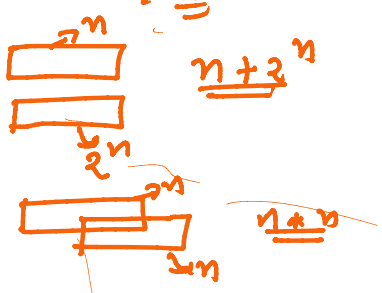
→ Multiple examples.

General rules:-

1. Ignore constants.
2. $1 < \log n < n < n\log n < n^2 < 2^n < n!$

TC => $n\log n + n^2$

$\Rightarrow n^2$



* Big-oh

Input size	Computer Steps
$n=5$	10
$n=10$	40

$2*n$

Q You have been given an array. You need to find an element in the array. Return true / false

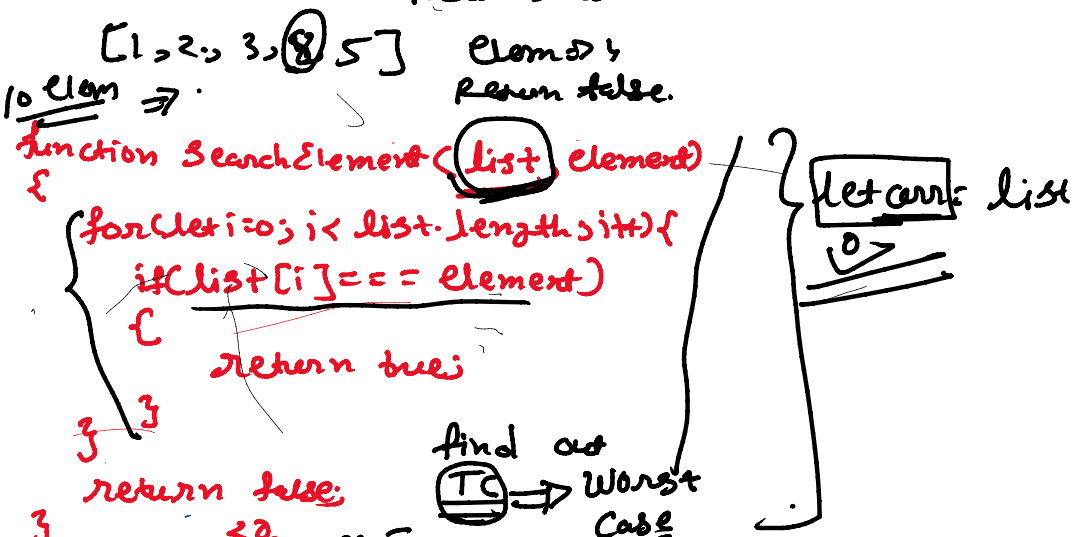
[1, 2, 3, 4, 5] elem = 4

Return true.

[1, 2, 3, 8, 5] elem = 4

Return false.

10 elem ⇒



12 = O(n)

1 2 · 3 ----- 20

$\frac{20}{2}$

Avg case

10th / 11th

$\frac{n}{2} \approx \frac{n+1}{2} \simeq n$

Q
```
function foo(n)
{
    let ans = 0;
    for(i=1; i<n; i++) {
      for(j=1; j< math.log(n); j++)
      {
          ans += 1;
} }
}
 consale.log(ans);
```

log(n)

$\ln(n)$

```
for(i=0; i<n; i++)        // n  O(n) Comp. steps
{
   for(j=0; j< math.log(i); j++){
        ans += 1;
   }                                  O(logn)
}                          O(n logn)
```

1 cs → log i

2 → log i

3 → log i

⋮

n → log n

n + log n }}

O(n)

Q
```
function foo(n,m) {
    let a=0;
    for(i=1; i<n; i++) {
        a += 1;          // O(1)
    }                                  O(n)
    let b=0;                           m=2n
    for(i=1; i<m; i++) {               n=2m
        b += 1;                        m=2n
    }                                  O(m)
}
```

O(n+m)

O(n+2n)

O(3n)

O(n)

$O(\frac{n}{2}+n)$ $O(n)$

$O(\frac{3n}{2})$ $O(n)$

```
for(i=0; i<n; i++){
    console.log(i);
}
```

n n

```
function foo(n){
    let ans = 0;
    for(let i=1; i<n; i++){
        for(let j=n; j>i; j--)
        {
            ans += 1;
        }
    }
    console.log(ans)
}
```

$i=1$: $j=n$ ..... $j=1$ } $n-1$ steps

$i=2$: $j=n$ ..... $j=2$ } $n-2$ steps

$i=3$: $j=n$ ..... $j=3$ } $n-3$ steps

$i=n$

$(n-1)+(n-2)+(n-3)+\cdots$

n → n times

$n*n$ $O(n^2)$

$\frac{n^2+n}{2}$

$1+2+3+4+\cdots+n \Rightarrow \frac{n(n+1)}{2}$

Q

80% Explaination.

7-8% Code.

13% TC & SC

extra ds $O(1)$

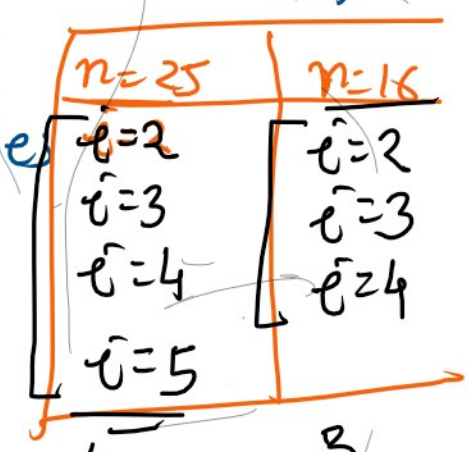
Q TC ( )

```
function foo ){
    for(let i=2; i<=Math.sqrt(n); i=i*i){
```

$5<=4$

$i=5*5$

```
for(let i=2; i<=Math.sqrt(n); i++)
{
    if(n%i==0){
        return false;
    }
}
return true;
}
```



OJ log - Al. - 2

log - Revision

log - Doubts

logarithmic rules!-

(1) $\log(x \cdot y) = \log x + \log y$

(2) $\log(x/y) = \log x - \log y$

(3) $\log(x^y) = y \log x$

(4) $\log_e(e) = 1$

(5) $\log 1 = 0$

(6) $\log\left(\frac{1}{x}\right) = -\log(x)$

⊛ Time Complexity for recursion!-

1. Substitution method
2. Master theorem
3. Recursion tree.

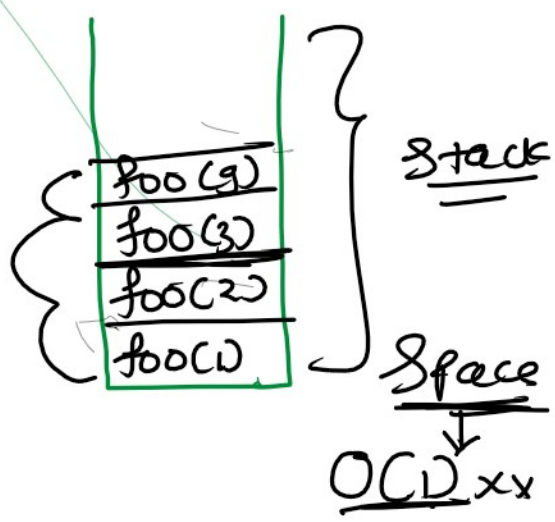
} future classes

⇒ iterative Soln ⇒ DSXX ⇒ Space Comp O(1)

fn call itself

1. Recurrence rel^n

1. Recurrence rel^n



Q Print first n natural nos

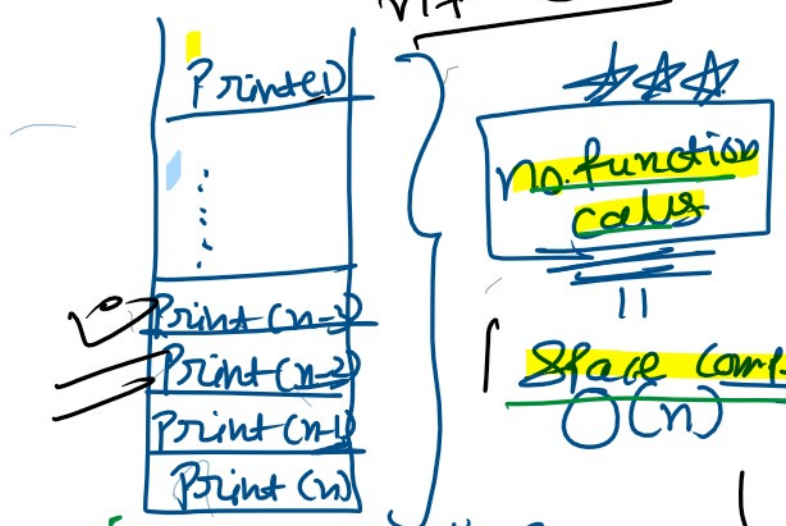
$n=5 \rightarrow 1, 2, 3, 4, 5$

$n=2 \rightarrow 1, 2$

⇒ 3 things [1. Base case
[2. Recursive call
[3. Self work.

$n=5$ [1 2 3 4] 5

```
function Print(n) {
    if(n==0 || n==1)
        Consol.log(n)
    Print(n-1) // 1 ... 4
    Console.log(n) // 5
}
```



R.S.

```
function Print(n) {
    if(n==0 || n==1)
        Consol.log(n)
    Print(n-1) // 1 ... 4
    Console.log(n) // 5
}
```

T.S.

```
Print(n) {
    for(i=1; i<=n; i++)
    {
        Console.log(n)
    }
}
```

TC ⇒ O(n)

S.C ⇒ O(n)

TC ⇒ O(n)

SC ⇒ O(1)

[1 2 3 4 5]

Q) Sum of all elements

Recursion

```
function Sum(list, i){
{ if(i == list.length) {
      return list[i];
  }
  res = Sum(list, i+1)
  return res + list[i];
}
```

R.R ⇒ $T(n) = T(n-1) + C$ — (1)

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

$T(n-1) = T(n-2) + C$ — (2)

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

$T(n) = T(n-2) + 2C$ — (3)

Putting (3) → (1)

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

$T(n-2) = T(n-3) + C$ — (4)

$T(n) = T(n-3) + C + 2C$

$T(n) = T(n-3) + 3C$

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

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

$T(n) = T(n-3) + 3C$

$\vdots$

$T(n) = T(1) + nC$ — (5)

$T(n) = T(n-i) + iC$

$T(1) + nC$

$O(n)$

$n - (n-1)$

$n - n + 1$

$n - (n-i)$

$n - n + i$

$(n-1)C$

$T(n-1)$

$T(n-2)$

$\downarrow$

$n-3$

$\hookrightarrow n-8$

$n-4$

$n=16$

$T(n) = O + nC$

$= nC$

$T(n) = n$

$O(n)$

nC

$T(1) + (n-1)C$

$O + nC - C$

$O(n)$

$n-1, n-2, n-3, \ldots\ldots 5, 4, 3, 2, 1, 0, -1, -3$

5

$5, 4, 3, 2, 1, 0, -1$

$n \quad n-1 \quad n-2 \quad n-3 \ldots$

$(n-i)$

$n-(n-i)$

2

3