Data structure & Algorithm.

① **Time complexity**: একটি Program চলতে কী রকম সময় লাগে।

Tool to calculate time complexity:

O(1) ⓐ Assignment operation: a=6, a=20

O(1) ⓑ comparison " : a>6, a>=c

O(1) ⓒ Mathematical operation: +, -, *, : 2+6.

ধরা হয় না / কিন্তু O(1) ⓓ ফাংশন কল: asd(n).

O(...) ⓔ ফাংশনের ভিতরের কাজ:

O(...) = Big "O" notation / Order of (...)

(worst case)

```
#include
int main()
{
    int n1, n2, n3;
    n1 = 10;  ⟹ O(1)
    n2 = 20;  ⟹ O(1)
    n3 = n1 + n2;  ⟹ O(1), O(1)
    Return 0;
}
```

এটি linear operation. মোট operation ~~O(1)~~ O(1) এর সংখ্যা সর্বদা নির্দিষ্ট এবং input এর উপর নির্ভর করে না.

তাই, T.C. = O(1).

```
#include <stdio.h>
int main()
{
    int n, returnnomoti's;
    scanf("%od", &n);
    returnnomoti's = n*(n+1)/2;
    Printf( "result = %d\n", returnnomoti's);
}
```

O(1), O(1), O(1), O(1)

∴ মোট 4 টি operation ; so, T.C = O(1)

```
result = 0;
for (i=1; i<=n; i++)
{
    result = result + 1;
}
```

O(1), O(1)

যখন n এর মান যত, loop তত বার চলে।

n=1 এর জন্য O(2), n=2 ⇒ O(4), ...

∴ T.C. O(2n) ⇒ 2 × O(n) ⇒ O(n).

```
count = 0;
for (i=0; i<n; i++){
    for(j=0; j<n ; j++){
        count = count +1;
    }
}
```

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

$O(n)$ $\Rightarrow O(1), O(1)$

n	count
1	1
2	4
3	9
10	100
100	10,000

Nested loops গুলোতে সাধারণত T.C. multiplication এ আসে

```
for (i=0; i<n; i++){
    for(j=0; j<3 ; j++){
        count = count+1;
    }
}
```

$O(3*n) = O(3n) = O(n)$

$O(3)$

$O(n!) > O(n^n) > O(n^3) > O(n^2) > O(n) > O(\sqrt{n}) > O(\log n) > O(1)$

$O(n^2) + O(n)$

$= O(n^2+n)$

$= O(n^2)$ ∴ ছোট গুলা কে গ্রাহ্য না করে।

$O(n! + n^3 + n)$

$= O(n!)$

4 (ii) Space complexity. (worst case)

```
#include <stdio.h>
int main(){
    int i, n, even[100];
    for(i=0; i<100; i++){
        even[i]=0;}
    for(i=0; i<100; i+=2){
        even[i]=1;}
    scanf("%d", &n);
    if(even[n]){
        printf("%d is even\n");}
    else { printf("%d is odd\n");}
    return 0;
}
```

O(1) O(1) O(100)

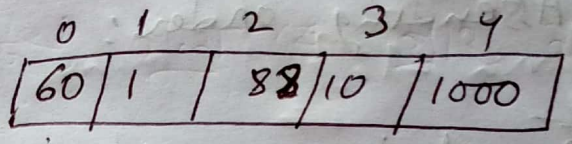
So, S.C = O(100).

অথবা array size = O(array size)

s[i][j] ⇒ S.C. O(n*n) ∴ O(n²)

Recursion ⇒

(5) Linear search :



(i) [0] == 10? ✗

(ii) [1] == 10? ✗

(iii) [2] == 10? ✗

(iv) [3] == 10? ✓

```
int linear_search (int A[], int n, int x){
    int i;                          O(1) => S.C.
    for(i=0; i<n; i++){
        if( A[i] == x){
            return i;
        }
    }
    return -1;
}
```

input is not S.C for the algorithm.

6 বাইনারী সার্চ :

0	1	2	3	4	5	6	7	8	9
8	7	1	2	4	5	19	20	3	6

sorting :

0	1	2	3	4	5	6	7	8	9
1	2	3	4	5	6	7	8	19	20

Assending order:

Algorithm: নিয়ম

① sort in a order (Assending)

② Get the middle number 5 (4)

③ compare target 8 with the middle number 5

ⓐ If target > middle number;
then, set the right part of the middle number as point ① and do ① to ③

Ex: 5 | 6 | 7 | 8 | 19 | 20

ⓑ If target < middle number;
then, set the left part of the middle number as point ① and do ① to ③.

Ex: 1 | 2 | 3 | 4 |

ⓒ If target == middle number; return index of middle number.

Binary search এ প্রতিটি array তে কত বার সর্বোচ্চ search loop চালাতে হবে, তা বের করা যায় $\log_2 x$ থেকে, x = array length

∴ 20 array length এ সর্বোচ্চ search loop হবে

$$\log_2 20 = ?$$

2) 20 (10
20
0

2) 10 (5
10
0

2) 5 (2
4
1

2) 2 (1
2
0

4

2) 10 (· 5
10
0

2) 50 (· 025
50
0

∴ Time complexity of Binary search in:

$\log_2 n$