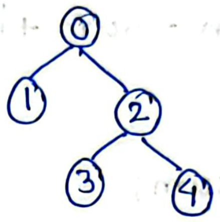
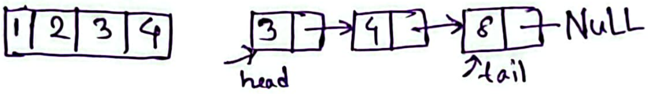
Data Structure and Algorithm II

Data Structure

- Linear
  - First element define
  - Last element define
  - every element (exept the first one) must have previous element
  - every element (exept the last one) must have next element
- Non linear →





Algorithms → set of steps to figer out result from given input.

Algorithm analysis steps:

① Correctness
- → Halt
- → Correct output

⑪ Time complexity

⑪ Space/Memory Complexity

Ⓘⓥ Readability (//, variable name)

ⓥ Optimality.

Time Complexity

Example 1 :

```
int sumN(int N) {
    int sum = 0;                       ———————— 1
  for(int i = 1; i<=n ;i++) {          ————→ n+1
      sum = sum + i;                   ————→ n
  }
  return sum;                          ————→ 1
}
```

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

$= 2n + 3$

$= O(n).$

Three types of time complexity

① Worst case → O

② best case → Ω

③ average case → Θ

$T(n) = 5n^4 + \underbrace{2n+3}$ → asymptotic notation এর জন্য Negligible

$n=1 = T(1) = 5+5 = 10$

$n=2 = T(2) = \boxed{80} + \boxed{7} = 87$

"প্রথম Part থেকে 2nd Part এর change টা অনেক ছোট তাই গন্য করা হয়না"

$n=3 =$

$n=5 = T(5) = 3125 + 13 = 3138$

$n = 10 ; T(10) = 50000 + 23 = 50023$

$n = 100 ; T(100) = 500000000 + 203 = 500000203$

Series of Time Complexity :

$$1 < \log_2(n) < \sqrt{n} < n < n\log_2(n) < n\sqrt{n} < n^2 \dots < 2^n < n!$$

Example - 3

```
int search(int arr[], int n, int key){              Worst      best
    for(int i=0; i<n ; i++){  ————————————          n+1         1
        if(arr[i] = key){  ————————————              n          1
            return i;  ————————————                  0          1
        }
        return -1;  ————————————                     1          0
}
```

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

$= 2n+2$

$= O(n)$

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

$= 3$

$= \Omega(c)$

$= \Omega(1)$

⑨
L-2
26-09-23

Example - 2

	Worst	best
`int func( int n) {`		
`int weirdsum = 0`	1	1
`for(int i = 1; i ≤ n; i++) {`	n+1	n+1
`weirsum = weirdsum + 1;`	n	n
`}`		
`if(weirdsum %2 == 0) {`	1	1
`return weirdsum;`	0	1
`}`		
`for(int i=1; i<=n; i++) {`	n+1	0
`weirsum *= i;`	n	0
`}`		
`return weirdsum.`	1	0
`}`		

Worst $T(n) = 1 + (n+1) + n + 1 + (n+1) + n + 1$

$= 4n + 5$

$= O(n)$

best $T(n) = 1 + (n+1) + n + 1 + 1$

$= 2n + 4$

$= \Omega(n)$

Find time complexity from T(n)

$1 < \log_2 n < \sqrt{n} < n < n\log_2 n < n\sqrt{n} < n^2 \dots 2^n < n!$

upper bound

tight bound

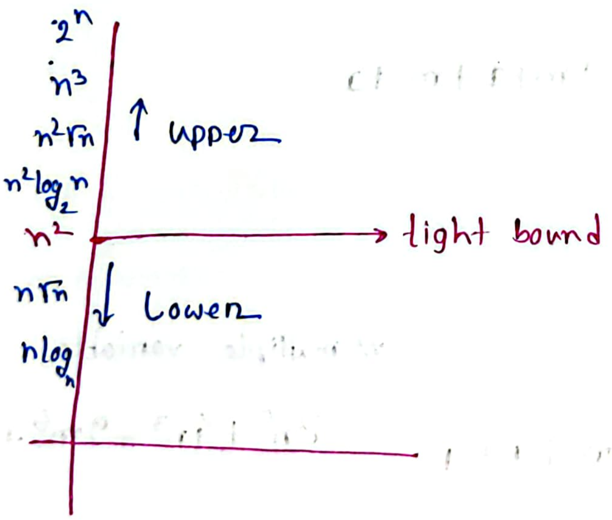
Lower bound

$T(n) = 2n^5 + 3n^2 + 9n + 2$

for all n, / $\forall n$, $2n^5 + 3n^2 + 9n + 2 \leq \boxed{16n^5}$

$O(n^5)$ ['+','-' doesn't mater always যোগ বিয়োগ]

$LB \leq TB \leq UP$



Example-4

```
                                                   worst        best
Int func(int n, int m) {                                          1
    int sum=0;                                       1
                                                                 n+1
    for(int i=1; i<=n; i++){                        n+1
                                                                  n
        sum = sum+i;                                  n
    }
                                                                  1
    if(sum % 2)                                       1
        return sum;                                   0           1

    else {
        for(int i=1; i<=m; i++){                     m+1          0
            sum =sum+i;                               m           0
        }
    }
    return sum;                                       1           0
}
```

$W \rightarrow T(n,m) = 1 + n+1 + n + 1 + m+1 + m + 1$

$= 2n + 2m + 5$

$= O(n+m)$

multiple variable

$3n^5 + 4n^3 + 9m^6 + 3n^7 + 8m + 5o^3 + 8o^2 + 9$

$\Rightarrow O(n^7 + m^6 + o^3)$

$B \rightarrow T(n,m) = 1 + n+1 + n + 1 + 1$

$= 2n + 4$

$= \Omega(n)$

Example -5 $\left(\frac{n}{2}\right)$

```
int func(int n){
    int prod = 1;
    for(i=1; i<=n; i=i+2){
        prod = prod+i;
    }
    return prod |
}
```

worst	best
1	1
$\frac{n}{2}+1$	$\frac{n}{2}+1$
$\frac{n}{2}$	$\frac{n}{2}$
1	1

কিভাবে $\frac{n}{2}$ হয়?

n=10

i=1 ; 1<=10 : T

i=3 ; 3<=10 : T

i=5 ; 5<=10 : T

i=7 ; 7<=10 : T

i=9 ; 9<=10 : T

i=11 ; 11<=10 : F

Formula/Rules :

```
for(int i=1 ; i<=n ; i=i+k){
    // statment
}
```

for line —— $\frac{n}{k}+1$

statment —— $\frac{n}{k}$

```
for(i=n; i<=n ; i=i-k){
    // statment
}
```

for line —— $\frac{n}{k}+1$

statment —— $\frac{n}{k}$

Example - 6

```
int func(int n) {
    int prod = 1;                        ———— 1
    for(int i = n; i>=1 ; i= i-4) {     ———— n/4 +1
        prod = prod +i;                  ———— n/4
    }
    return prod;                         ———— 1
```

$$T(n) = 1 + \frac{n}{4} + 1 + \frac{n}{4} + 1$$

$$T(n) = 2\frac{n}{4} + 3$$

$$= \frac{n}{2} + 3$$

$$= \frac{1}{2}n + 3$$

$$= O(n)$$

Binary Search এর Time Complexity কেন $\log_2 n$ হয়?

$$n \quad \frac{n}{2} \quad \frac{n}{4} \quad \frac{n}{8} \quad \frac{n}{16} \quad \cdots\cdots \quad 1$$

$$\frac{n}{2^0} \quad \frac{n}{2^1} \quad \frac{n}{2^2} \quad \frac{n}{2^3} \quad \frac{n}{2^4} \quad \cdots\cdots \quad \frac{n}{2^i}$$

$$\frac{n}{2^i} = 1$$

$$n = 2^i$$

$$\log_2 n = \log_2 2^i$$

$$\log_2 n = i \log_2 2$$

$$\therefore \boxed{i = \log_2 n}$$

$\log_2 n$

Example 7 :

```
int func ( int n ){
    int sum = 0;                        ———— 1
   for ( i = 1; i <= n ; i = i * 2) {   ——  $\log_2 n + 1$
     sum = sum + i ;                    ———  $\log_2 n$
   }
   return sum;                          ———— 1
}
```

$[*, / \rightarrow \log_2 n]$

$T(n) = 3 + 2\log_2 n$

$= O(\log_2 n)$

Formula/Rules :

```
for (int i = 1 ; i <= n ; i = i*k) {   ——  $\log_k n + 1$
   // statment                         ——  $\log_k n$
}

for (int i = n ; i >= 1 ; i = i/k) {   ——  $\log_k n + 1$
   // statment                         ——  $\log_k n$
}
```

. lets see how it work →

$n = 100$

$$\left.\begin{array}{l} i=1 ;\ i<=100 : T \\ i=2 ;\ 2<=100 : T \\ i=4 ;\ 4<=100 : T \\ i=8 ;\ 8<=100 : T \\ i=16 ;\ 16<=100 : T \\ i=32 ;\ 32<=100 : T \\ i=64 ;\ 64<=100 : T \end{array}\right] ⑦ \qquad \log_2(100) = \lceil 6.64 \rceil = ⑦$$

$i = 128 ;\ 128 <= 100 : F]$ 1

Example 8:

```
intF(int n){
    int sum=0;          ——————  1
    int k=n;            ——————  1
  while (k>=1){         ——————  log5 n +1
    sum = sum+k;        ——————  log5 n
    k=k/5               ——————  log5 n
  }.
  return sum;           ——————  1
}
```

$$\therefore T(n) = 1+1+\log_5 n+1+\log_5 n+\log_5 n+1$$

$$= O(\log_5 n)$$

n=100

$$\left.\begin{array}{l} k=100\ ;\ 100>=1\ :T \\ k=20\ ;\ 20>=1\ :T \\ k=4\ ;\ 4>=1\ :T \end{array}\right] ③ \rightarrow \log_5(100) = \lceil 2.86 \rceil$$

≈ ③

k=0 ; 0>=1 : F]1

Example - 9 (root, $\sqrt{n}$)

```
                                                   W          B
bool Isprime (int n)
    for (int i=2 ; i*i <=n ; i++){       — √n               1
       if ( n%i ==0) {                    — √n-1             1
            return false;                 — 0                1
       }
    }
    return true;                          — 1                0
}
```

$W \rightarrow T(n) = \sqrt{n} + \sqrt{n} - 1 + 1$

$= 2\sqrt{n}$

$= O(\sqrt{n})$

$B \Rightarrow T(n) = 1 + 1 + 1$

$= 3$

$= \Omega(c)$

$= \Omega(1)$

\# for (i=1; i<=n ; i++) — n+1

for (i=2; i<=n; i++) — n

for (i=2; i <=√n ; i++) — $\sqrt{n}$

$\boxed{i * i < n}$

$i^2 < n$

$\sqrt{i^2} < \sqrt{n}$

$\boxed{i < \sqrt{n}}$

what is the time complexity for both for (i=2 ; i<=n ; i = i*2)
& for (i=2 ; i<=√n ; i++) ?

for (i=2; i<=n; i=i*2) — $\boxed{\log_2 n}$

for (i=2; i<=√n; i++) — $\boxed{\sqrt{n}}$

now, for (i=2; i<=√n ; i=i*2) — $\boxed{\log_2 n \quad \sqrt{n}}$

Example -10:

```
                                                    worst            best
void func (int n) {
    for (int i = 1; i <= n; i++) {      —— n+1                    n+1
        func2();                        —— (log₂ n)×n             1×n
    }
}
```

hint for func 2() { Best case : $\Omega(1)$
worst case $O(\log_2 n)$

$W \Rightarrow T(n) = n+1 + n\log_2 n$

$= O(n\log_2 n)$

$B \Rightarrow T(n) = n+1 + n$

$= 2n+1$

$= \Omega(n)$

Example — 11:

```
void func (int n, int m) {
    int sum = 0                          —— 1
    for (int i = 1 ; i <= n ; i++) {     —— n+1
        for (j = 1 ; j <= m ; j++) {     —— n(m+1)
            sum = sum + i + j ;          —— m×n
        }
        sum = sum + i ;                  —— n
    }
}
```

$T(n) = 1 + n+1 + n(m+1) + mn + n$

$= 2nm + 3n + 2$

$= O(nm)$.

Example -12

```
void fune (int n) {
    for (int i = n ; i>=1 ; i--) {                 — n+1
        for (int j = 2 ; j<=n ; j = j*3) {         — n * (log_3 n)
            printf ("%d %d ·", i, j)               — (log_3 n -1) n
        }
        printf ( "%d %d ", i, n);                  — n
    }
}
```

$T(n) = n+1 + n\log_3 n + n\log_3 n - n + n$

$= 2n\log_3 n + n + 1$

$= O(n\log_3 n)$

· [illegible]

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

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

here, $f(n) = \frac{n(n+1)}{2}$

$\therefore f(n+2) = \frac{(n+2)(n+2+1)}{2}$

$= \frac{(n+2)(n+3)}{2}$

also, $5+6+7 \cdots +(n+2) - ?$

$$\frac{(n+2)(n+3)}{2} - \textcircled{10} \longrightarrow \frac{4(4+1)}{2} = 10$$

Example -13:

```
int func(int n) {

    int sum = 0;                           ———— 1

    for (int i = 1; i <= n; i++) {         ———— (n+1)
        for (int j = i; j <= n; j++) {     ———— (n+1)(n+2)/2 - 1
            sum = sum + i + j;             ———— n(n+1)/2
        }
        sum = sum + i;                     ———— n
    }

    return sum;                            ———— 1

}
```

" here, 2nd loop depends on 1st for loop ; for this dependeneis the ঘাত will come "

$$T(n) = 1 + n + \cancel{1} + \frac{(n+1)(n+2)}{2} - \cancel{1} + \frac{n(n+1)}{2} + n + 1$$

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

$$= O(n^2)$$

i = 1	2	3	4	5	....	n	
in → head : n+1	n-	n-1	n-2	n-3	....	+2 =	$\frac{(n+1)(n+2)}{2} - 1$
in → body : n	n-1	n-2	n-3	n-4	....	+1 =	$\frac{n(n+1)}{2}$

```
for (int j = 100 ; j <= 100 ; j++) — 2
    sum = sum + i + j ;            ———— 1
}
```

j = 100 ; 100 <= 100 : T

j = 101 ; 101 <= 100 : F

Example – 14 :

```
int func(int n, int m){                                            worst                best
    int sum = 1;                           ————                      1                    1
    for(i=0; i<n; i++){                    ————                     n+1                  n+1
        for(j=0 ; j<=i ; j++){             ————        (n+1)(n+2)/2 -1      (n+1)(n+2)/2 -1
            printf("Inner Part");          ————                n(n+1)/2             n(n+1)/2
            sum = sum+i+j;                 ————                n(n+1)/2             n(n+1)/2
        }
        if(sum%2==0){                      ————                       n                    n
            for(j=1; j*j <=m ; j++){       ————               n(√m +1)                    0
                sum = sum +j;              ————                  (√m) n                    0
            }
        }
    }
    return sum;                            ————          1        1
}
```

inner 1: dependent on 1st loop or variable "i"

inner 2: Independent

```
# for(j=0 ; j<=i; j++ ){
     // body
  }
```

$$\overset{w}{T}(n,m) = 1+n+1+\frac{(n+1)(n+2)}{2}-1+\frac{n(n+1)}{2}+\frac{n(n+1)}{2}+n+n(\sqrt{m}+1)+(\sqrt{m})n+1$$

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

$$= O(n^2+n\sqrt{m}),$$

best

$$T(n,m) = 1+\frac{(n+1)(n+2)}{2}-1+\frac{n(n+1)}{2}+\frac{n(n+1)}{2}+n+1$$

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

i	0	1	2	3	. . .	n-1	
inn1:head	2	3	4	5	. . .	n+1	$\frac{(n+1)(n+1+1)}{2}-1$
inn1:body	1	2	3	4	. . .	n	$\frac{n(n+1)}{2}$

Time complexity	Algorithm A	Algorithm B
Best	$\Omega(n)$	$\Omega(n\log_2 n)$
Worst	$O(n^2)$	$O(n\log_2 n)$

Real life Avg Case.