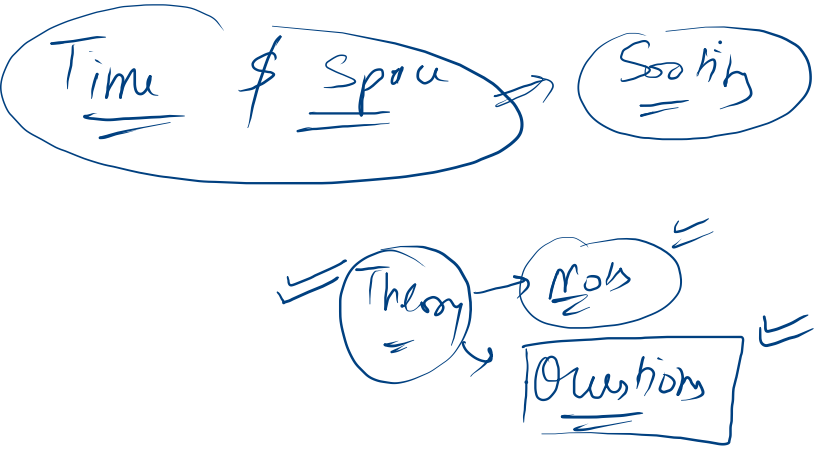
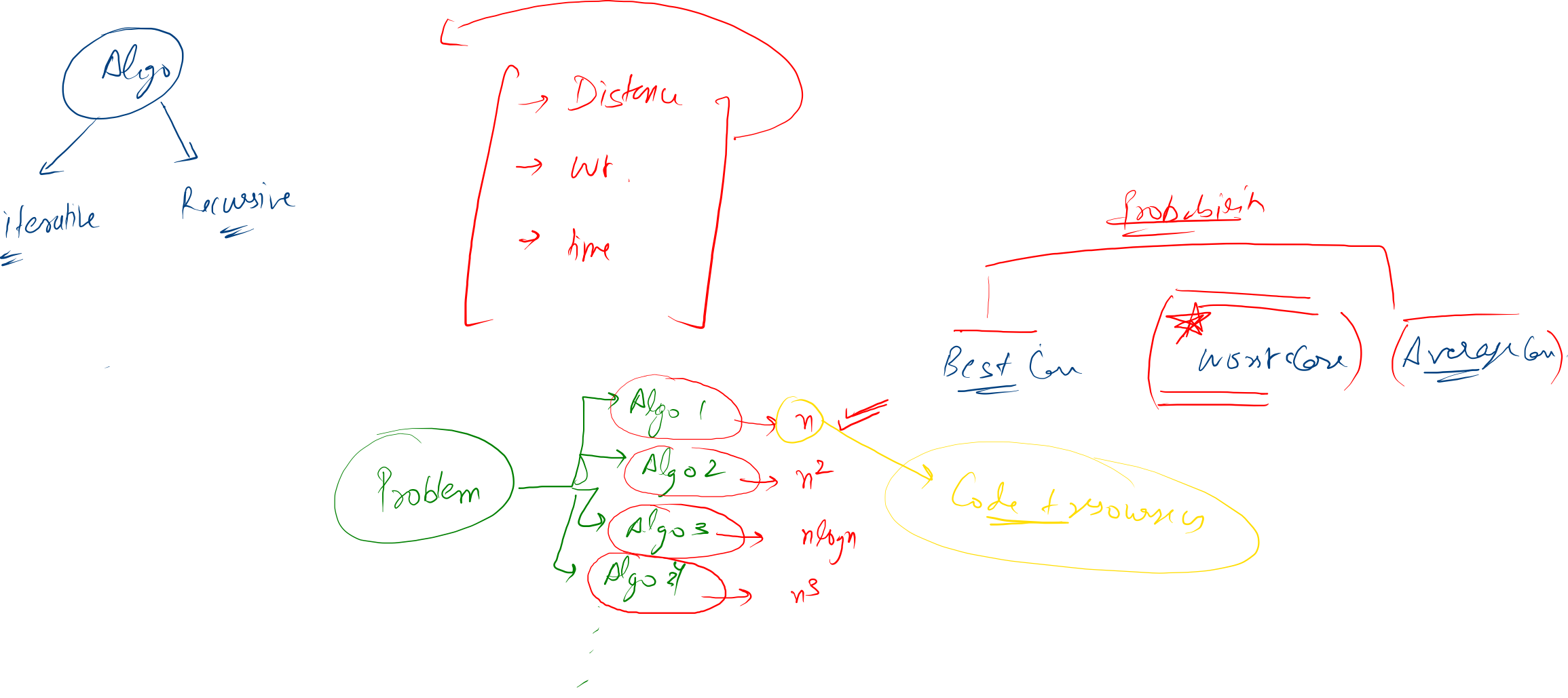
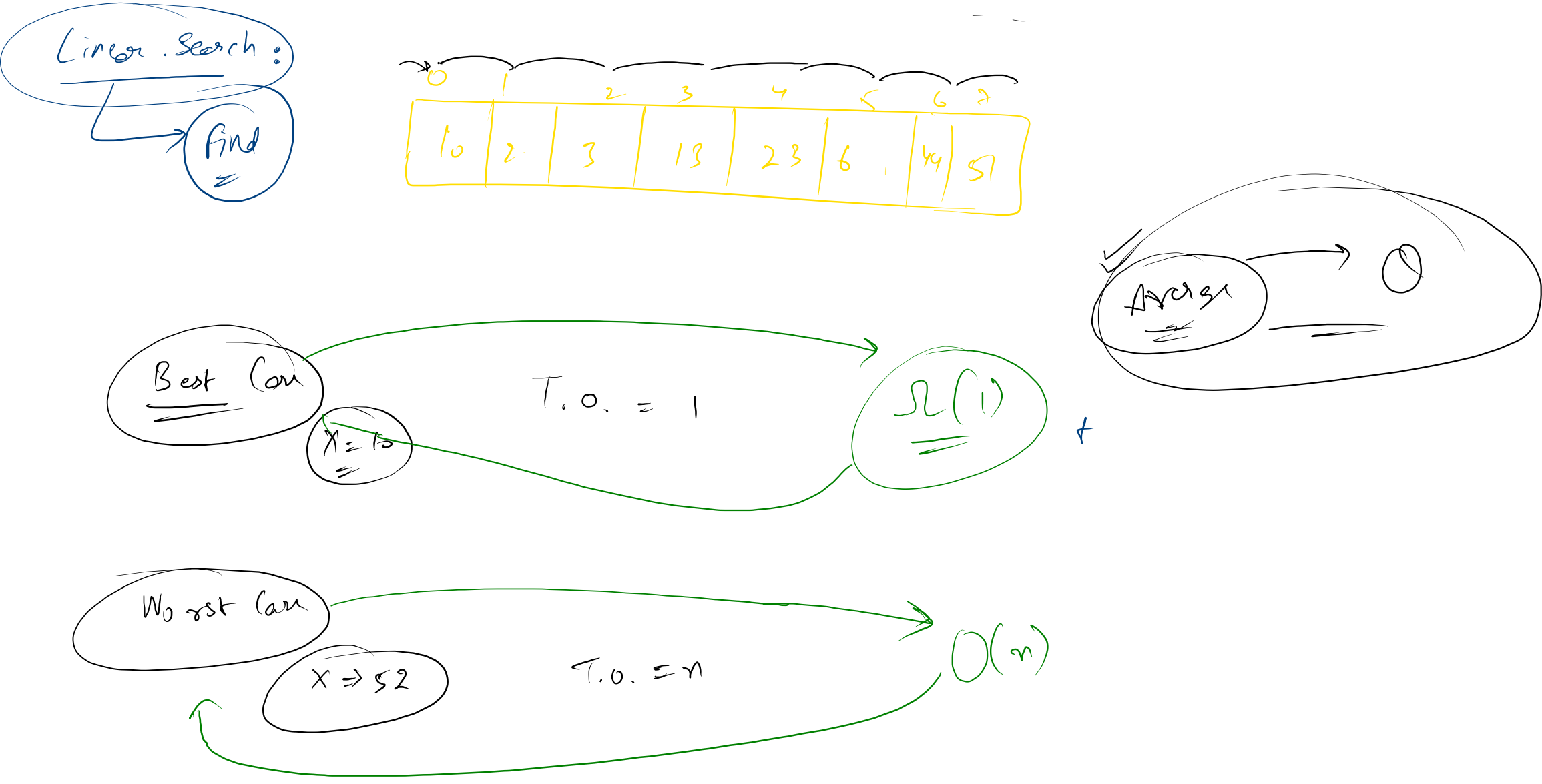
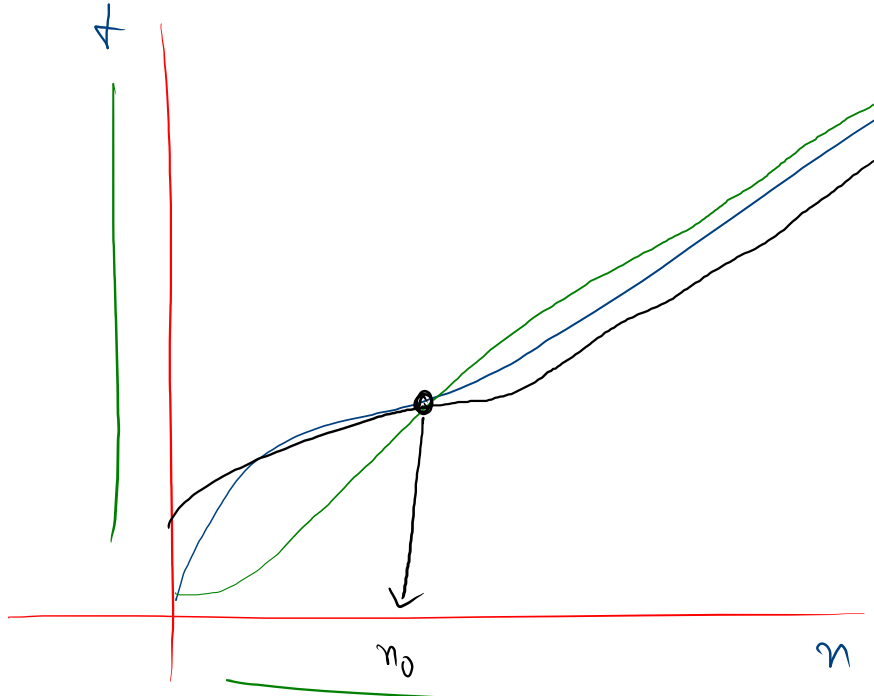

Time & Space
Sorting
Theory
Algos
Questions


Algo
Iterative
Recursive
Distance
Wt
time
Probabilistic
Best Case
Worst Case
Average Case
Problem
Algo 1
Algo 2
Algo 3
Algo 4
n
n^2
$n \log n$
n^3
Code + resources


Linear Search :
Find
0 1 2 3 4 5 6 7
10 2 3 13 23 6 44 51
Best Case
X = 10
T.C. = 1
Ω(1)
Worst Case
X = 52
T.C. = n
O(n)

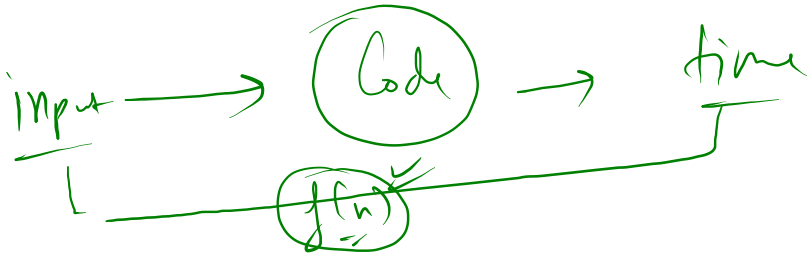


$O(g(n))$

$g(n) >= c \cdot f(n)$, $c > 0$, $n > n_0$

$h(n) <= c \cdot f(n)$, $c > 0$, $n > n_0$

$\Omega(h(n))$



$n \to \infty$ $\left[f(n) \right] \to$ cos

$n \to 0$ $\left[f(n) \right] \Rightarrow$ 1 s s

$O(n^2)$

$(1 < \log n < \sqrt{n} < n \log n < n^2 < n^2 \log n < n^3 < \dots)$

$n!$, a^n

n^2 , $n^2 + 3n + 2$, $\frac{n^2 + n}{5}$, $n^2 + n - 25$ . . . .

$O(n^2) \Rightarrow (Kn^2) \Rightarrow n^2$

T.C. $\Rightarrow$ $n^3 + n^2 + n \log n$

T.C. $\Rightarrow O(n^3)$

T.C. $\Rightarrow$ $n \log n + 2^n + n^2$

T.C. $\Rightarrow 2^n$

$100 \log 100 + 2^{100}$, $(100)^2$

$(2^{10})^{10}$

$(1024)^{10}$

Iterable

n times

```
for (int i=0 , i<n; i++) {
    for (int j=0; j<n; j++){
        System.out.println("hello");
    }
}
```

→ O(1)

n^2 times

T.O. → n^2

T.C. → $O(n^2)$

$\Omega(n^2)$

k is some input by user?

Q.

Best, worst

```
for (int i = 0; i < n; i++) {          // n times
    for (int j = 0; j <= n; j++) {     // i times
        if (k % 2 == 0) {
            break;
        } else {
            System.out.println("hello");
        }
    }
}
```

W.C.
T.O. ⇒ n^2
T.C. ⇒ $O(n^2)$

B.C.
T.O. ⇒ n times
T.C. ⇒ $\Omega(n)$

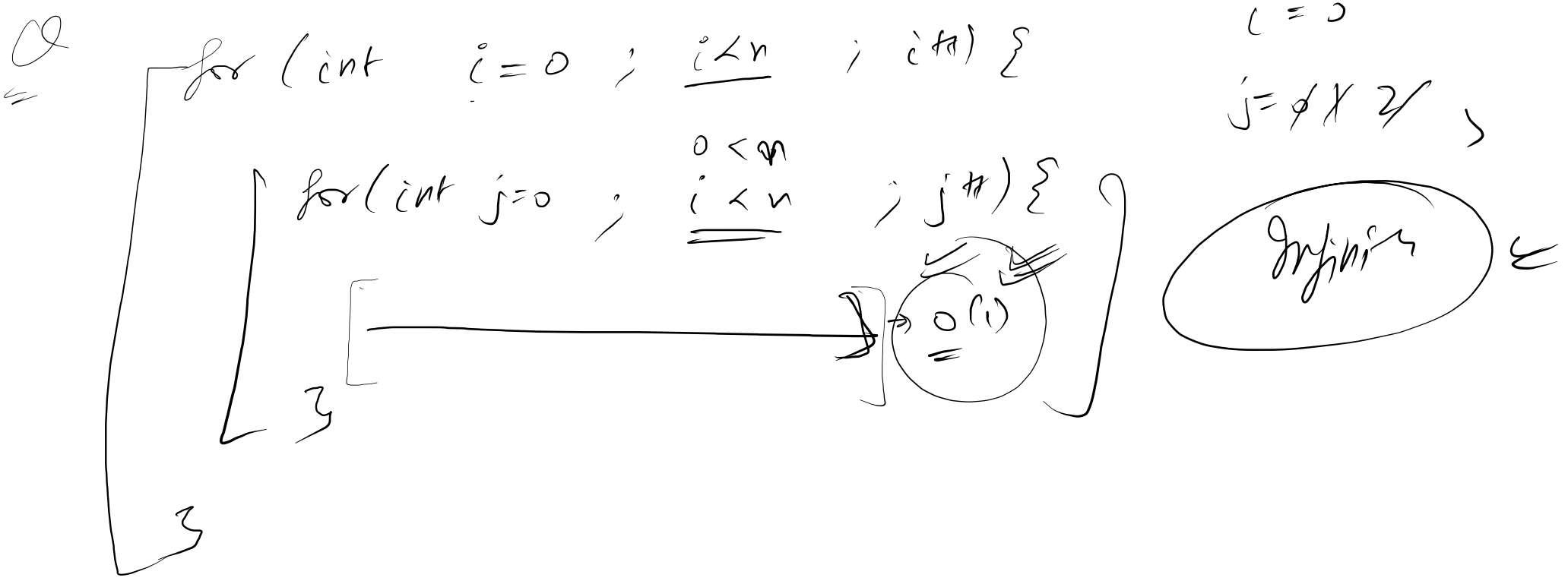
Q Worst Case

```
for( int i=0 ; i<n ; i=i*c){
    [ ------------ ] O(1)
}
```

[n, nlogn, √n]

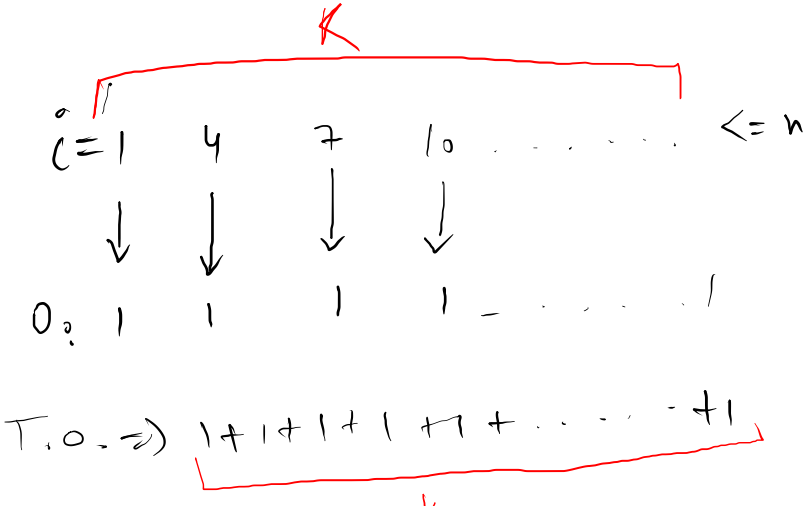
⇒ i=0

- Logical Error
- TLE
- Infinite Loop


for (int i=0 ; i<n ; i++) {
for(int j=0 ; i<n ; j++) {
O(1)
i=0
j=0 X 2
Infinite

Q

```
for(int i=1 ; i<=n ; i = i+3) {
    print("Hello");
}
```



$a_n = a + (n-1)\,d$

$a_k = a + (k-1)\,d$

$= 1 + (k-1)\,3$

$a_k = 3k - 2$

$a_k <= n$

$3k - 2 <= n$

$3k <= n+2$

$k <= \left(\frac{n+2}{3}\right)$

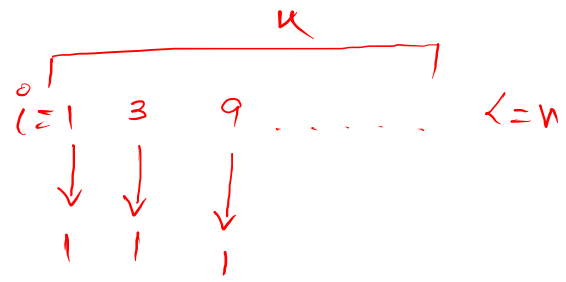
T.O. => $k <= \left(\frac{n+2}{3}\right)$

T.O. <= $\frac{n+2}{3}$

T.C. => $O(n)$

Ques

```
for(int i=1 ; i<=n ; i = i*3) {
    print("Hello");
}
```



k times

T.C. => 1+1+1+1+1+1+1....

T.C. => k

T.C. <= $\log_3 n + 1$

T.C. => $O(\log_3 n)$

$a_n = a(r)^{n-1}$ ✓

$a_k = (3)^{k-1}$

$a_k <= n$

$3^{k-1} <= n$

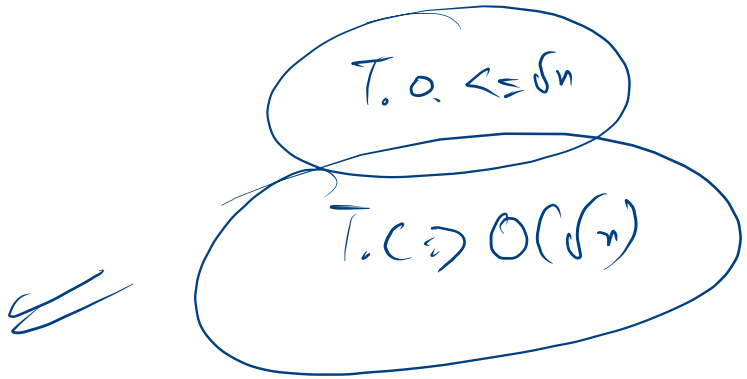
taking log both sides

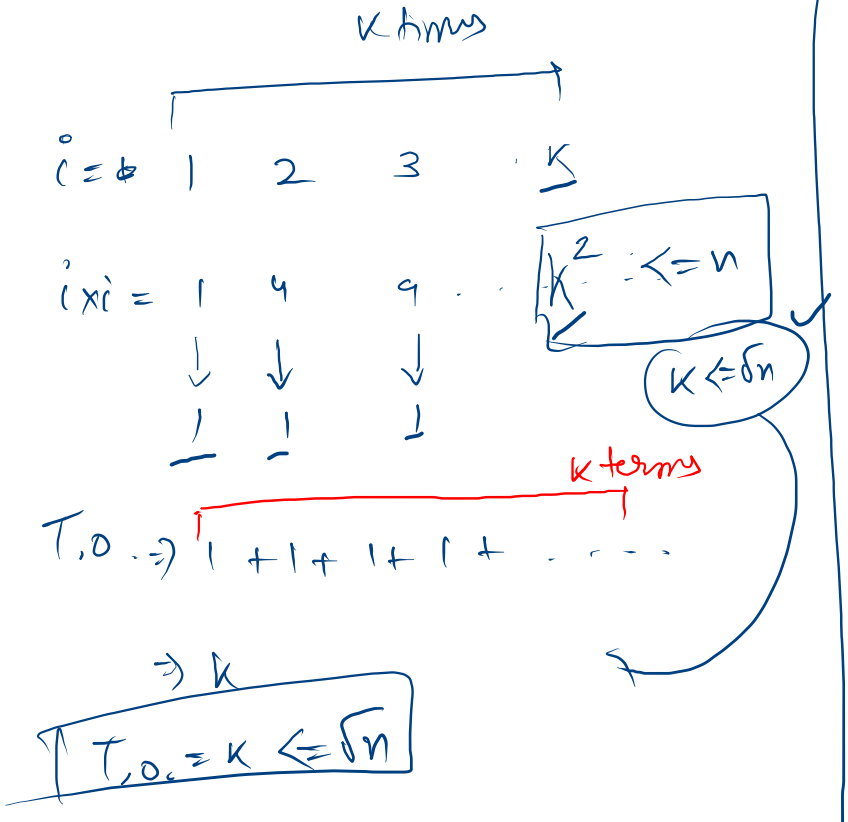
$k-1 <= \log_3 n$

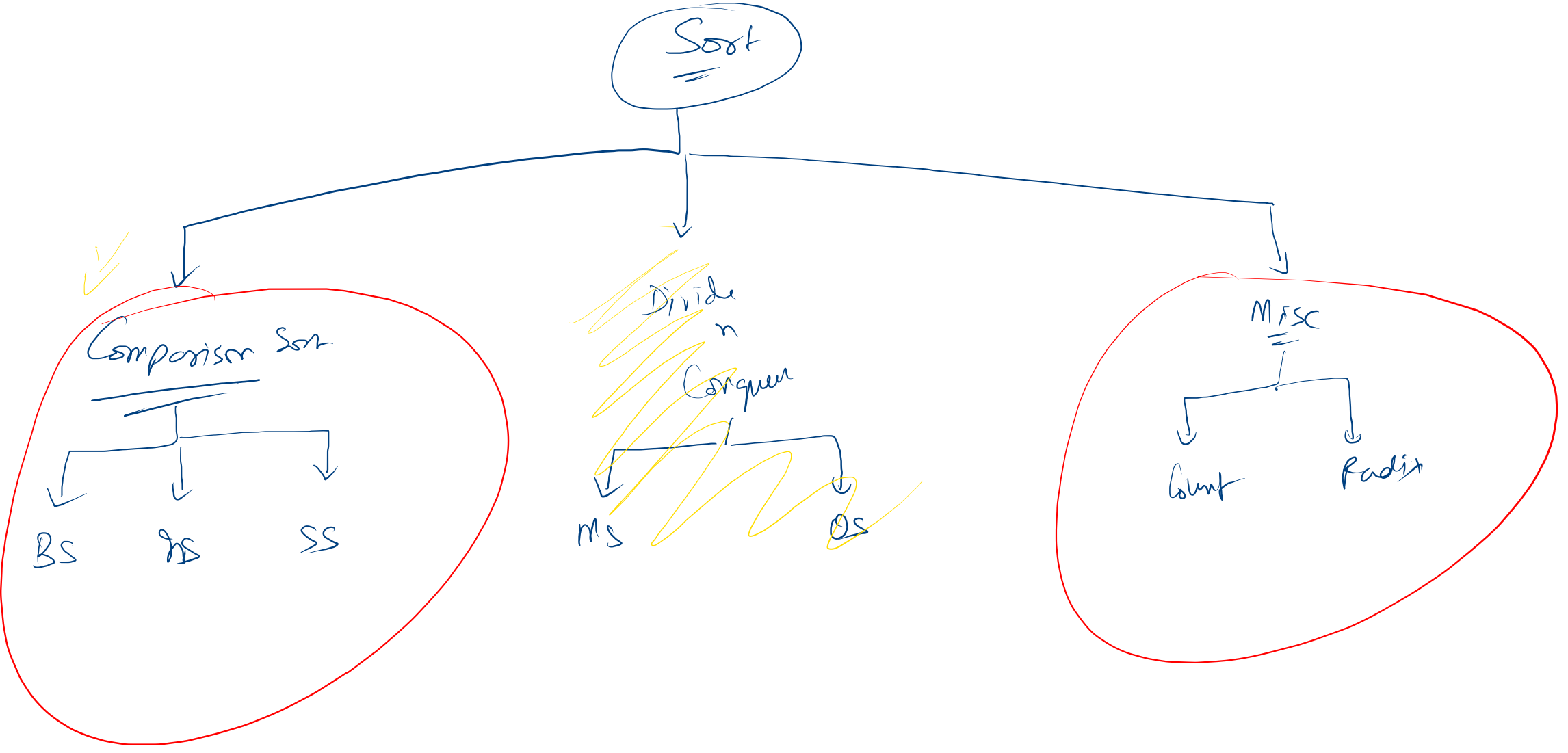
$k <= \log_3 n + 1$

9

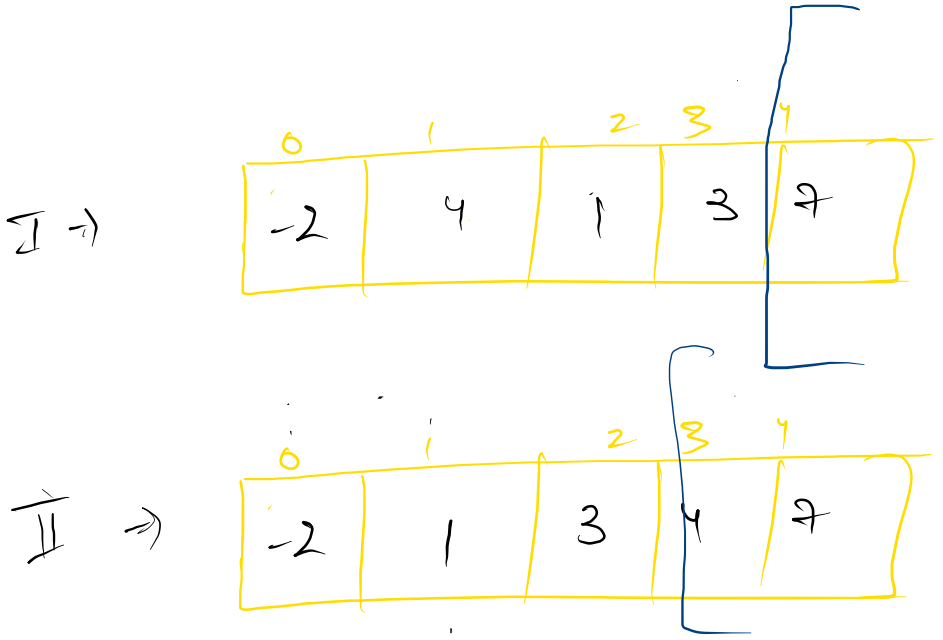
```
for(int i=1; i*i<=n ; i++){
    ] O(1)
}
```

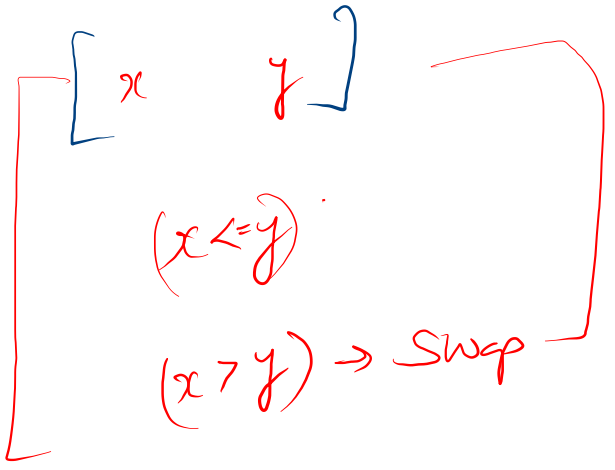


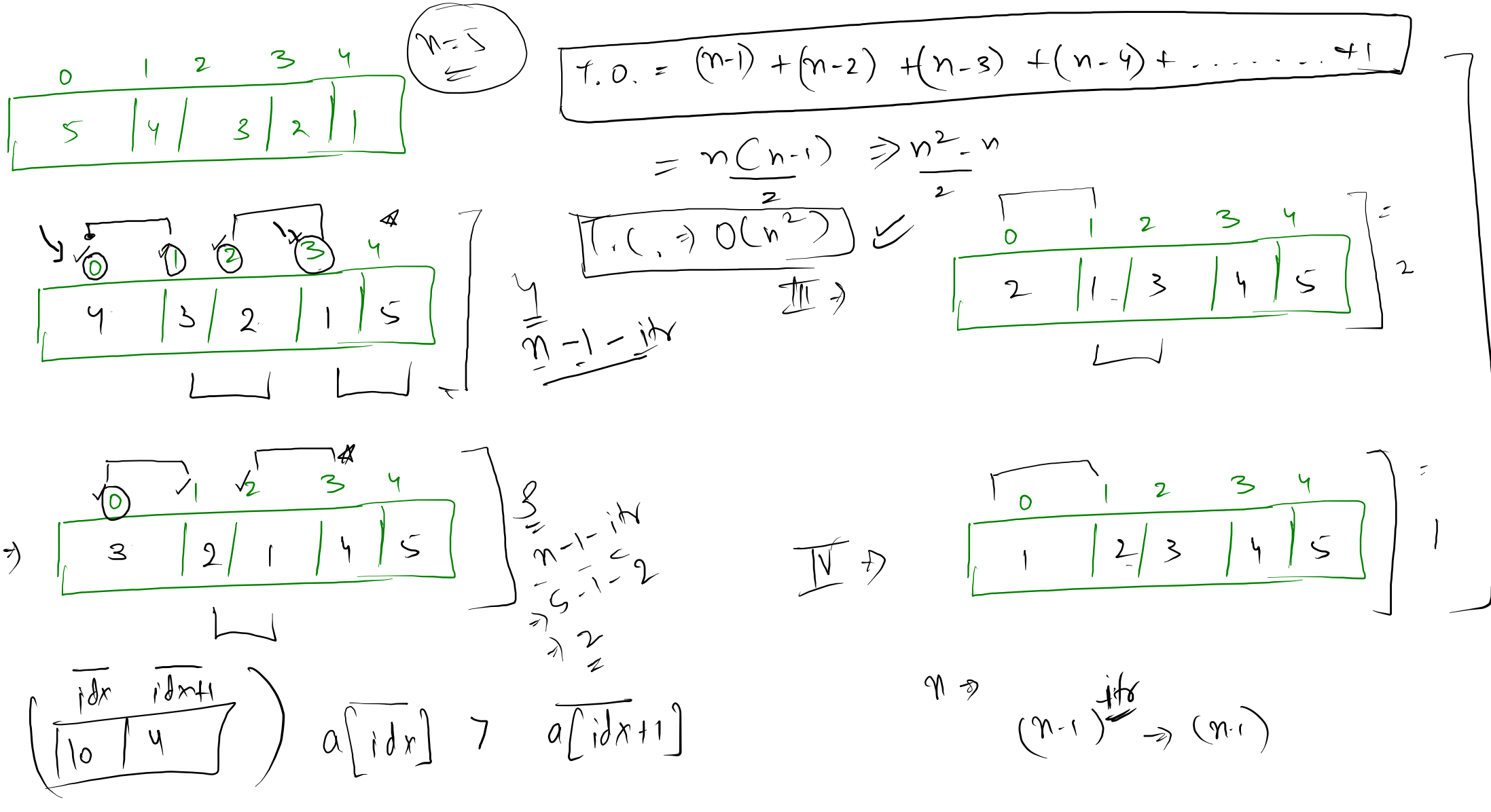



Sort
Comparison Sort
BS
IS
SS
Divide
n
Conquer
MS
QS
Misc
Count
Radix

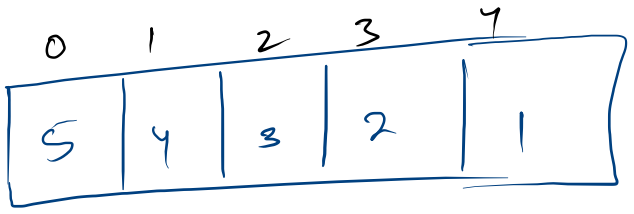
Sort → inc



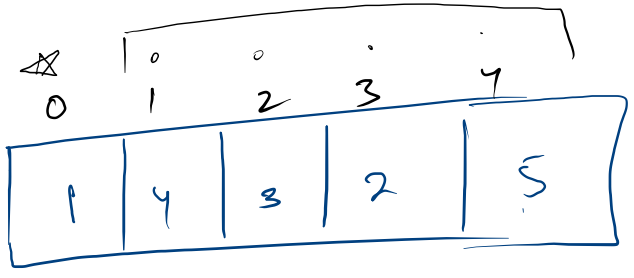



n = 5
0 1 2 3 4
5 4 3 2 1
T.C. = (n-1) + (n-2) + (n-3) + (n-4) + ... + 1
= n(n-1)/2 ⇒ n²/2 - n/2
⇒ O(n²)
I →
0 1 2 3 4
4 3 2 1 5
n - 1 - itr
II →
0 1 2 3 4
3 2 1 4 5
n - 1 - itr
5 - 1 - 2
⇒ 2
III →
0 1 2 3 4
2 1 3 4 5
2
IV →
0 1 2 3 4
1 2 3 4 5
1
idx idx+1
10 4
a[idx] > a[idx+1]
n ⇒ (n-1) itr ⇒ (n-1)

Selection

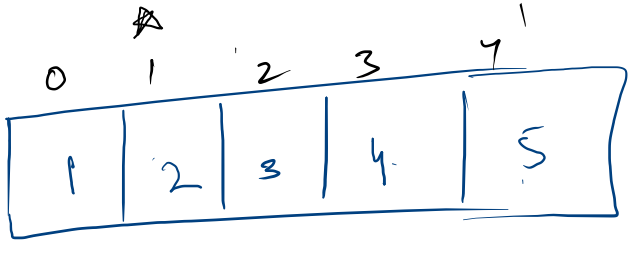


I ⇒



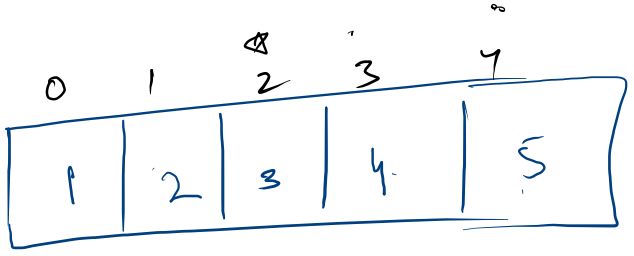
minIdx = 0 ~~1 2 3~~ 4

II ⇒



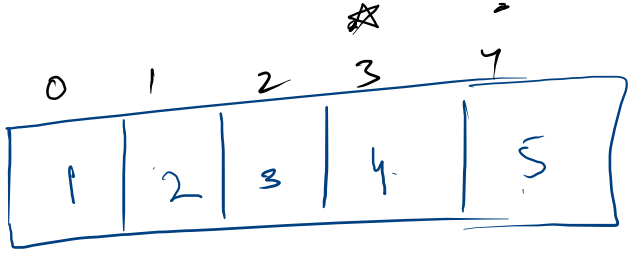
minIdx = ~~1 2~~ 3

III ⇒

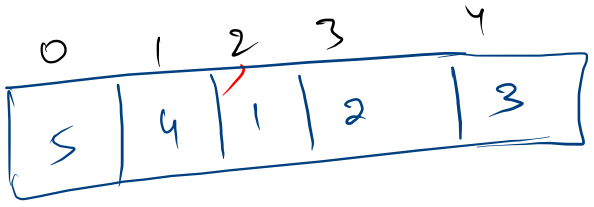


minIdx = 2

IV ⇒

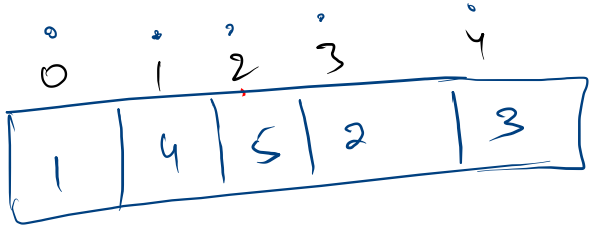


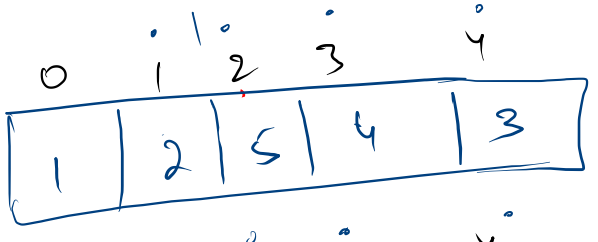
minIdx = 3

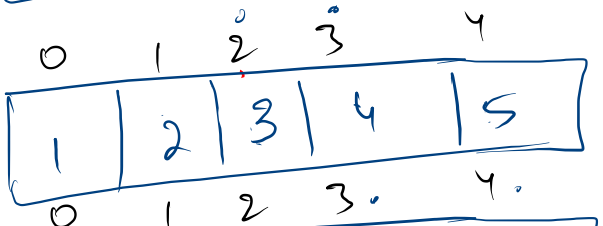


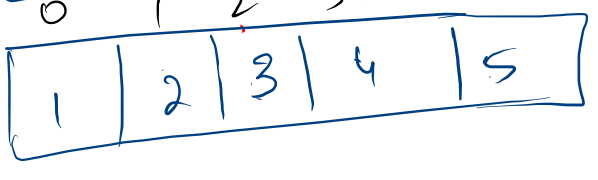
n=5

```
int n =  arr.length;
for(int itr = 1; itr <= n-1; itr++){
    int minIdx = itr-1;

    for(int idx = itr ; idx < n ; idx++){
        if(isSmaller(arr,idx,minIdx)){
            minIdx = idx;
        }
    }

    swap(arr,itr-1,minIdx);
}
```

I →


II →


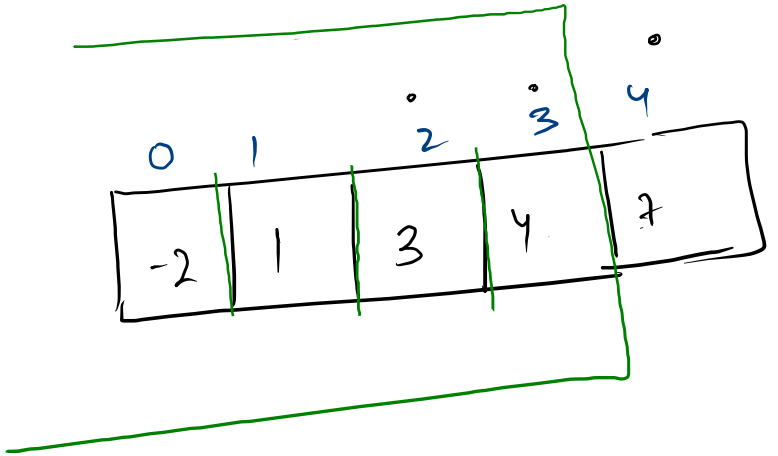
III →


IV →


itr	minIdx	idx	T.C.
1	~~0~~ / 2	1,2,3,4	4
2	~~1~~ / 3	2,3,4	3
3	~~2~~ / ~~3~~ / 4	3,4	2
4	3	4	1

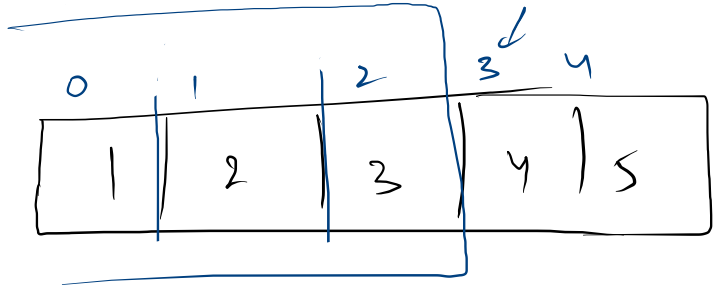
T.C. => $\frac{n(n-1)}{2}$

T.C. => $O(n^2)$


0
1
2
3
4
-2
1
3
4
7

Syntax → Practice → Algo → Code

Logic




```
int n = arr.length;
for(int itr = 1; itr <= n-1; itr++){
    for(int idx = itr ; idx > 0 ; idx--){
        if(isGreater(arr,idx-1,idx)){
            swap(arr,idx-1,idx);
        }else{
            break;
        }
    }
}
```

n-1

Itr	idx
1	1
2	2
3	3
4	4

= 4

T.C. → (n-1)

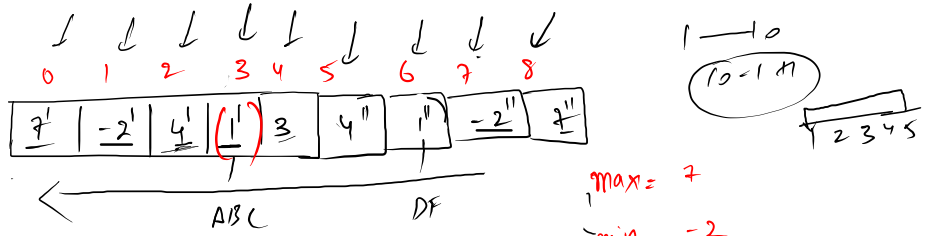
T.C. → O(n)

Bestcase

S.C. → O(1)

B.C. → O(n²)

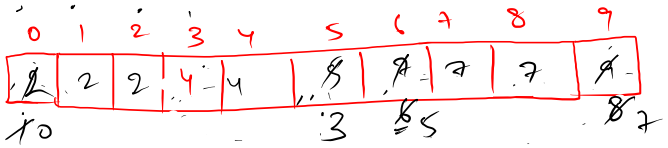
Count Sort



stable

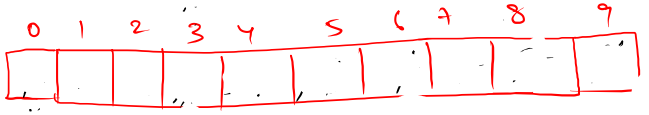
max = 7
min = -2

① freq =



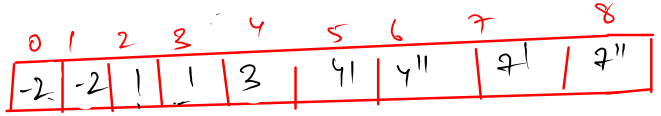

```
for(int val : arr){
    freq[val-min]++;
}

for(int idx = 1 ; idx < numOfUnique ; idx++){
    freq[idx] = freq[idx] + freq[idx-1];
}
```

② PSA =




```
for(int idx = arr.length-1 ; idx >= 0 ; idx--){
    int val = arr[idx];
    int pos = val - min;
    int place = freq[pos];
    res[place-1] = val;
    freq[pos]--;
}
```

③ res =



no. of unique elements = max - min + 1
= 10

-2 -1 0 1 2 3 4 5 6 7
↓ ↓ ↓ ↓ ↓ ↓ ↓ ↓ ↓ ↓
0 1 2 3 4 5 6 7 8 9

idx = val - min
4 - (-2)
= 6

[max, min] n times

```
int numOfUnique = max-min+1;
int freq[] = new int[numOfUnique];

for(int val : arr){
    freq[val-min]++;
}

for(int idx = 1 ; idx < numOfUnique ; idx++){
    freq[idx] = freq[idx] + freq[idx-1];
}

int res[] = new int[arr.length];

for(int idx = arr.length-1 ; idx >= 0 ; idx--){
    int val = arr[idx];
    int pos = val - min;
    int place = freq[pos];
    res[place-1] = val;
    freq[pos]--;
}

for(int i = 0 ; i < arr.length ; i++){
    arr[i] = res[i];
}
```

n times (first loop)

nRange (second loop)

n times (fourth loop)

n times (last loop)

n → elements

1 | 1000 → 10^5

Analysis

no. of unique → nRange ⇒ max − min+1

Time

T.O. ⇒ n + n + nRange + n + n

T.O. → 4n + nRange

if (n >> nRange)

↳ T.C. ⇒ O(n)

if (nRange >> n)

↳ T.C. → O(nRange)

T.C. → O.Max(n, nRange)

S.C. → O.Max(n, nRange)

Radix Sort

least Sig

Most Sig

9214/10 → 921/10 → 92/10 → 9/10

(9214/100) % 10 → 2

(num / exp) % 10

((2196/100) → 21) % 10 → 1

exp = 1

9999
0008
1005
2196
1998
1012
0007
0013
0224
9214
1111
5555

exp = 10

1011
1012
0013
0224
9214
1005
5555
2196
0007
0008
1998
9999

exp = 100

1005
0007
0008
1111
1012
0013
9214
0224
5555
2196
1998
9999

exp = 1000

1005
0007
0008
1012
0013
1111
2196
9214
0224
5555
1998
9999

0007
0008
0013
0224
1005
1012
1111
1998
2196
5555
9214
9999

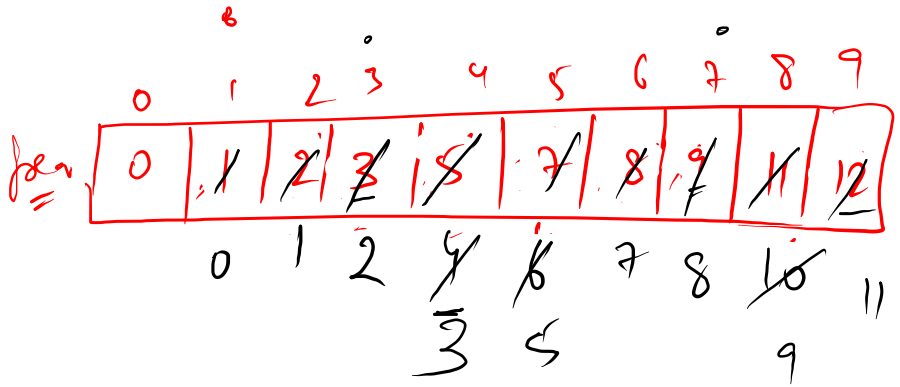
```
for(int i = 1 ; i < 10 ; i++){
    freq[i] += freq[i-1];
}

int res[] = new int[arr.length];
for(int i = arr.length-1 ; i >= 0 ; i--){
    int vl = arr[i];
    int pos = (vl/exp)%10;
    int place = freq[pos];
    res[place-1] = vl;
    freq[pos]--;
}

for(int i = 0 ; i < arr.length ; i++){
    arr[i] = res[i];
}
```

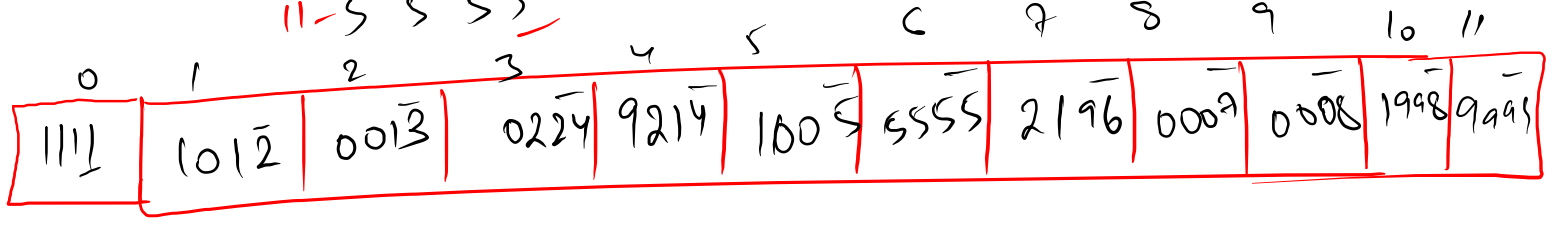
i = 1

Exp = 1

Exp = 1

0- 9999
1- 0008
2- 1005
3- 2196
4- 1998
5- 1012
6- 0007
7- 0013
8- 0224
9- 9214
10- 1111
11- 5555





DD MM YYYY

Day

Month

Year

H.W.

12 04 1996

20 10 1996

05 06 1997

12 04 1989

11 08 1982

32

13

1 to 31

2

3

$day = (Date / 1000000)$

$Month = (Date / 10000) \% 100$

$Year = (Date \% 10000)$