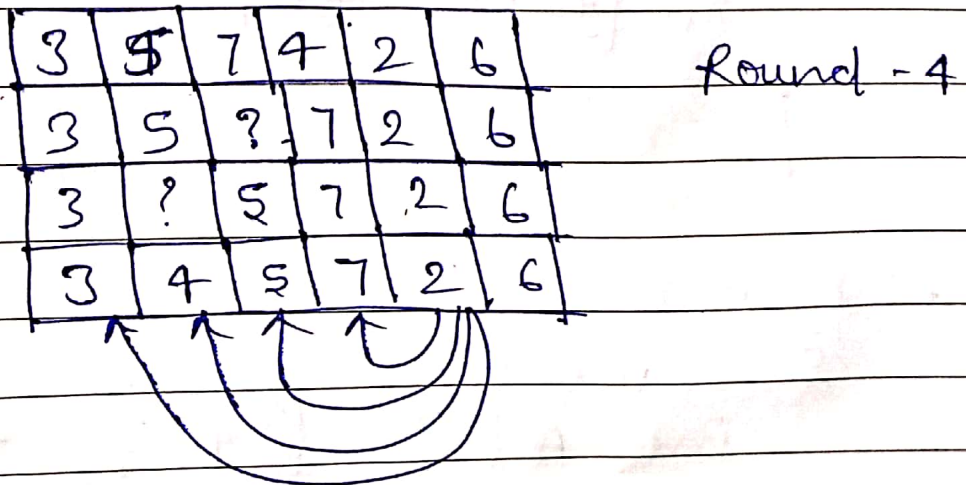
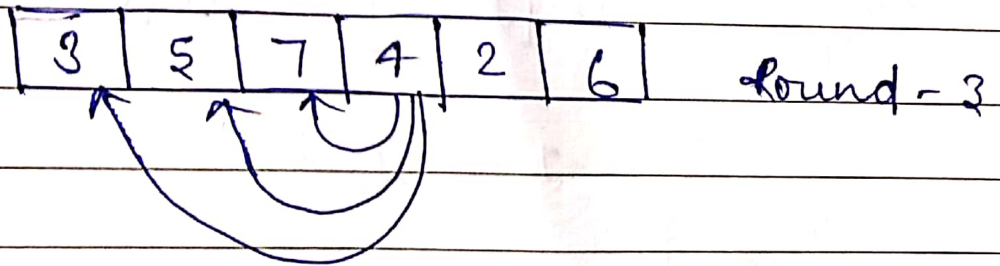
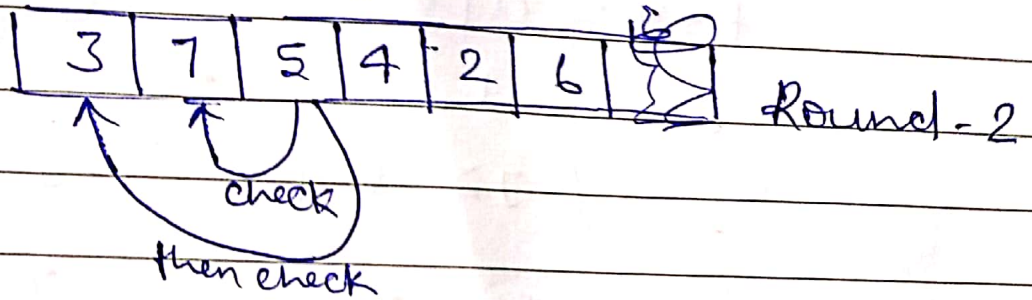
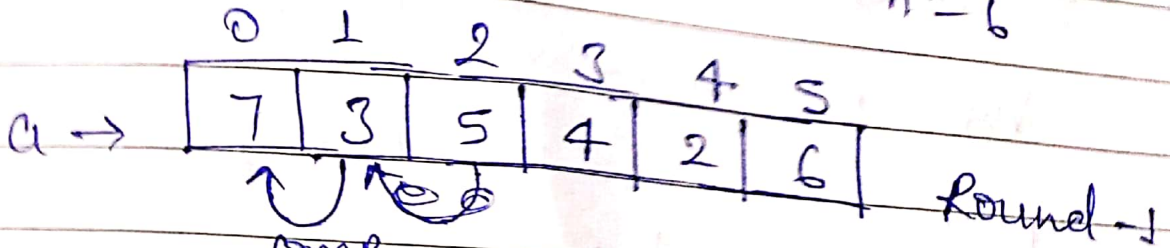


1. Insertion Sorting

Sorting Logic

$n=6$



Similar for next round

Program (Insertion Sort)

void insertion-sort [int a[], int n]

for (i = 1; i < n; i++)

↓ i = 1

key = a[i]

j = i - 1;

while (j >= 0 && a[j] > key)

{

a[j+1] = a[j]

j = j - 1;

}

a[j+1] = key;

}

→ here we write j >= 0
b/c we compare
upto '0' location
not in -ve
index.

→ 7 > 3
→ interchange

int main ()

{ int i;

int a[] = {24, 35, 4, 5, 11, 45};

insertion-sort(a, 9);

for (i = 0; i <= 8; i++)

printf("%d", a[i]);

return 0;

}

Time Complexity

in best case $\rightarrow O(n)$

in general case $\rightarrow O(n^2)$

Algorithm

INS. SORT (A, N): A is an Array with N elements.

1. $i = 1$

2. Repeat step 3 to 5 while $i < N$

3. $temp = A[i]$, $j = i - 1$

4. Repeat while $j > 0$ and $temp < A[j]$
 $A[j+1] = A[j]$ and $j = j - 1$

5. $A[j+1] = temp$, $i = i + 1$

6. Exit.

Analysis of Time Complexity for insertion sorting

for ($i = 1$; $i < n$; $i++$) $\rightarrow n$

key = $a[i]$ $\rightarrow n-1$

$j = i-1$ $\rightarrow n-1$

while ($j \geq 0$ & $a[j] > \text{key}$) $\rightarrow \sum_{i=2}^n t_i$
{

$a[j+1] = a[j]$

$j = j-1$;
}

$a[j+1] = \text{key};$
}

No. of times the while loop is executed for that value of j .

$$\sum_{i=2}^n (t_i - 1)$$

$$\sum_{i=2}^n (t_i - 1)$$

$n-1$

$$T(n) = C_1 n + C_2 (n-1) + C_3 (n-1) +$$

$$C_4 \sum_{i=2}^n t_i + C_5 \sum_{i=2}^n (t_i - 1)$$

$$+ C_6 \sum_{i=2}^n (t_i - 1) + C_7 (n-1)$$

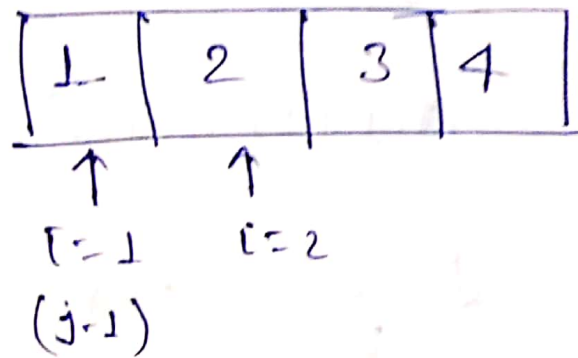
for best case :-

when $t_i = 1$ (Array is already sorted)

$$T(n) = C_1 n + C_2(n-1) + C_3(n-1) + C_4 \sum_{i=2}^n t_i + C_5 \sum_{i=2}^n (t_i - 1) + C_6 \sum_{j=2}^n (t_j - 1) + C_7(n-1)$$

Array is already sorted

let's say



$$\therefore T(n) = C_1 n + C_2(n-1) + C_3(n-1) + C_4(n-1) + C_5(0) + C_6(0) + C_7(n-1)$$

$$= (C_1 + C_2 + C_3 + C_4 + C_7)n - (C_2 + C_3 + C_4 + C_7)$$

$$T(n) = \underline{\underline{an + b}}$$