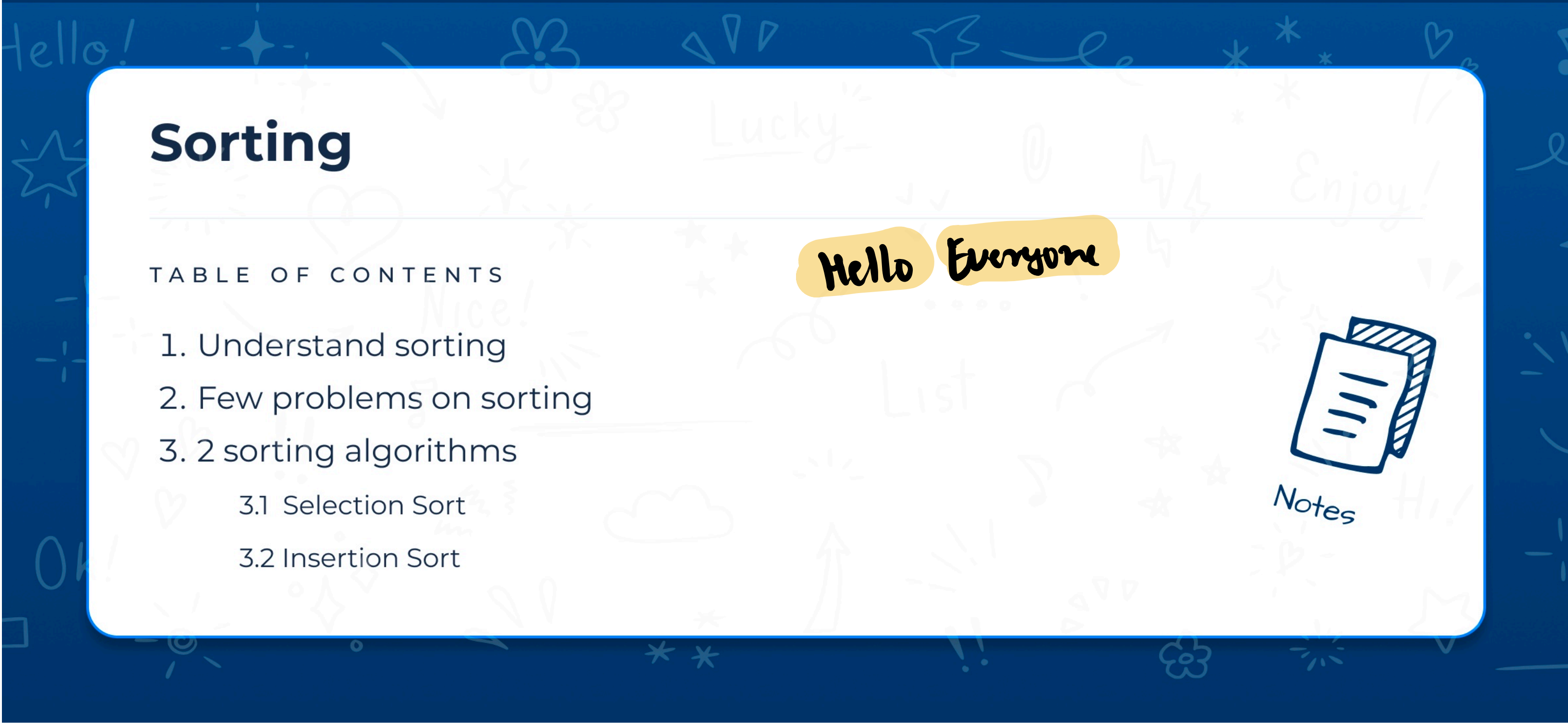


Sorting

TABLE OF CONTENTS

1. Understand sorting
2. Few problems on sorting
3. 2 sorting algorithms
   - 3.1 Selection Sort
   - 3.2 Insertion Sort

Hello Everyone

Sorting: Arrangement of data in particular order with respect to specific parameter

A[] = { 2, 3, 8, 12, 15 } ✓

A[] = { 15, 13, 10, 8, -6 } ✓

Quiz 1:

	1	13	9	6	12
#factors	1	2	3	4	6

Why Sorting:

1) Organizing
2) Analyzing
3) Searching

Question (Elements Removal)

Given N elements, at every step remove an array element.

Cost to remove an element = Sum of array of elements present in an array

Find minimum cost to remove all elements.

NOTE : First add the cost of removal and then remove it.

arr - $[\overset{0}{2}\ \overset{1}{1}\ \overset{2}{4}]$

index to remove	Cost	array
0	2+1+4=7	$\{\overset{0}{\cancel{2}}, \overset{1}{1}, \overset{2}{4}\}$
1	1+4 = 5	$\{\overset{0}{\cancel{0}}, \overset{1}{\cancel{1}}, \overset{2}{4}\}$
2	4 = 4	$\{\overset{0}{\cancel{0}}, \overset{1}{\cancel{1}}, \overset{2}{\cancel{4}}\}$

= 16 ✗

arr - $[\overset{0}{2}\ \ \overset{1}{1}\ \ \overset{2}{4}]$

index to remove	Cost	array
2	= 2+1+4 = 7	{2 1}
0	= 2+1 = 3	{1}
1	= 1	{ } ×

= 11 ✓

Quiz 2:

$$A = \left\{ \begin{array}{ccc} 0 & 1 & 2 \\ 4 & 6 & 1 \end{array} \right\}$$

index to remove	cost	Array
1	4+6+1=11	{4 1}
0	4+1=5	{1}
2	1 = 1	X
	=17	

Quiz 3:

$$A = \left\{ \begin{array}{cccc} 0 & 1 & 2 & 3 \\ 3 & 5 & 1 & -3 \end{array} \right\}$$

index to remove	Cost	Array
1	3+5+1-3=6	{3 1 -3}
0	3+1-3 = 1	{1 -3}
2	1 - 3 = -2	{-3}
3	- 3 = -3	x

= 2

Observation : Remove larger value.

$$\begin{array}{cccc} 0 & 1 & 2 & 3 \\ [a & b & c & d] \end{array}$$

Remove	Cost
a	$a + b + c + d$
b	$b + c + d$
c	$c + d$
d	d

Minimise Cost

$a + 2b + 3c + 4d$

$a > b > c > d$

Time Complexity for sorting using inbuilt fun. $O(n \log n)$

Space complexity $O(n) \mid O(1)$

// sort array in descending order.

```
cost = 0

for (i=0; i< N; i++)
{
        cost = cost + A[i] * (i+1)
}
return cost;
```

{4, 6, 2} → descending.

↳ {6, 4, 2}

   0 1 2

index	cost
0	$= 0 + 6 * (0+1) = 6$
1	$= 6 + 4 * (1+1) = 14$
2	$= 14 + 2 * (2+1) = 20$

Question (Noble Integers) { Distinct data }

Given N array elements, calculate number of noble integers.

An element ele in arr [ ] is said to be noble if { count of smaller elements = ele itself }

arr - [1 , -5 , 3 , 5 , -10 , 4]

2 1 3 5 0 4

# ele<A[i]

Ans = 3

arr - [-3 , 0 , 2 , 5]

Ans = 1

\#ele<A[i] → 0 1 2 3

Bruteforce: for each arr[i], find the no. of elements < arr[i] and check for noble integer.

TC: $O(n^2)$ SC: $O(1)$

```
ans=0
for(i=0; i<n; i++)
{       cnt=0
        for(j=0; j<n; j++)
        {        (A[j]< A[i])
            if(A[i] > A[j])
            {
                   cnt ++;
            }
        }

        if(cnt == A[i])
        {   ans++;
        }
}
    return ans;
```

	0	1	2	3	4	5
A = [	3	8	2	-1	5	-3]
#ele < A[i] →	3	5	2	1	4	0
	↑		↑			

Move all element < A[i] on 1 side

↳ Counting would be faster.

$$A = \begin{array}{cccccc} 0 & 1 & 2 & 3 & 4 & 5 \\ [\,3 & 8 & 2 & -1 & 5 & -3\,] \end{array}$$

↳ After sorting

$$\begin{array}{cccccc} 0 & 1 & 2 & 3 & 4 & 5 \\ -3 & -1 & 2 & 3 & 5 & 8 \end{array}$$

```
// sort array in ascending order

  cnt=0
 for (i=0 ; i<N ; i++)
 {
      if (A[i] == i)  cnt++;
  }
  return cnt;
```

$TC = O(N \log N + N) = O(N \log N)$

$SC = O(N) \mid O(1)$

Question (Noble Integers) : { Data can repeat }

	0	1	2	3	4
arr -	[-10 ,	1 ,	1 ,	3 ,	100]
# ele< A[i] ->	0	1	1	3	4

Ans = 3

	0	1	2	3	4	5	6	7	8
arr -	[-10 ,	1 ,	1 ,	2 ,	4 ,	4 ,	4 ,	8,	10]
#ele<A[i] ->	0	1	1	3	4	4	4	7	8

Ans = 5

	0	1	2	3	4	5	6	7	8	9	10	11	12	13
arr -	[-3 ,	0 ,	2 ,	2 ,	5 ,	5 ,	5 ,	5 ,	8 ,	8 ,	10 ,	10 ,	10 ,	14]
#e<A[i]->	0	1	2	2	4	4	4	4	8	8	10	10	10	13

Ans = 7

< / > *Code*

```
// sorting data in ascending order
    ans=0 , cnt=0

    for(i=0 ; i<n; i++)
    {
        if(i==0 || A[i] != A[i-1])
            cnt = i;
        if(cnt == A[i])
        {
            ans++;
        }
    }
    return ans;
```

0 1 2 3 4

{-10,1,1,3,10}

i	A[i]	IF cond	cnt	ans
0	-10	True	0	0
1	1	True	1	1
2	1	False	1	2
3	3	True	3	3
4	10	True	4	3

TC: O(nlogn+n) = O(nlogn)

SC: O(n) | O(1)

Break: 10-41 PM

Selection Sort

idea : Select the minimum element and send that elements to correct position by swapping.

$$A = \begin{array}{cccccc} 0 & 1 & 2 & 3 & 4 & 5 \\ [\,3 & 8 & 2 & -1 & 5 & -3\,] \end{array}$$

Find max ele in A[] → $TC = O(N)$ $SC = O(1)$

Find 2^{nd} max ele in A[] → $TC = O(2N) = O(N)$

$SC = O(1)$

Find 3^{rd} max ele in A[] → $TC = O(3N) = O(N)$

$SC = O(1)$

⋮

find K^{th} largest ele in A[] → $TC = O(K * N)$

$SC = O(K)$

→ $O(1)$

< / > Code

K=3

	0	1	2	3	4	5
A = [	3	8	2	-1	5	-3]

K elements in sorted position

if K = N-1 => sort Array.

	0	1	2	3	4	5
A = [	~~3~~	~~8~~	~~2~~	~~-1~~	~~5~~	~~-3~~]
	~~-1~~	~~-3~~	2	3	5	8
	-3	-1				

```
for (i = N-1; i >= 1; i--)
{        M=0 // index of max. element.

         for (j=1; j<=i; j++)
         {
              if (A[j] > A[M])
              {
```

i

0	1	2	3
1	~~8~~	~~2~~	~~5~~

2 5 8

} m=j;

} }

m=~~0~~1

swap(A, m, i) index

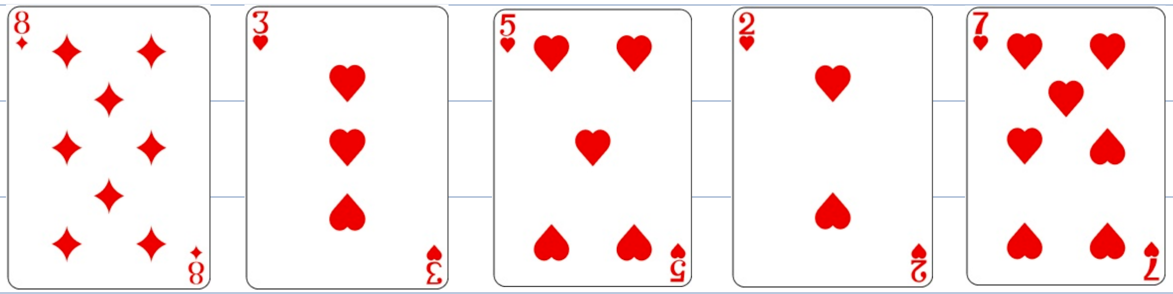
}

TC: $O(N^2)$

SC: $O(1)$

HW- Bubble sort

Insertion Sort (Arrangement of playing cards)



Why uses: It can sort running stream of data.

i/p: 7 9 12 10 8

i (i+1)

0	1	2	3	4	5	. . . .	n-1
7	8	9	10	12			

for any input → min swap = 0

Max swap = # no. of element in an Arrays.

</> Code

x=1

i

0	1	2
3	2	1

```
n=0

for all the input (x)
{
    index= n-1;
     while(index >=0)
    {
        if(A[index] > x)
        {
            A[index+1]= A[index]; shift right.
        }
        else
        {
            break; // current ele is smaller or equal to index ele.
        }
        index--;
    }
    A[index+1] =x;
    n++;
}
return A;
```

TC: $O(n^2)$

SC: $O(1)$

Doubt session

all the data → Arr

SC O(n)

n= 2

		i
0	1	2
3	2	1

index = 1

i	i+1	
1	2	3

```
if(index==-1)
{
    A[i] = inputArr[i];
}
```