SUNBEAM

Sunbeam Institute of Information Technology

Pune and Karad

Module – Data Structures and Algorithms

Trainer - Devendra Dhande

Email – devendra.dhande@sunbeaminfo.com

Selection sort

1. Select one position of the array
2. Find smallest element out of remaining elements
3. Swap selected position element and smallest element
4. Repeat above steps until array is sorted

passes = N-1

44	11	55	22	66	33
0	1	2	3	4	5

Pass 1

44	11	55	22	66	33
0	1	2	3	4	5

11	44	55	22	66	33
0	1	2	3	4	5

Pass 2

11	44	55	22	66	33
0	1	2	3	4	5

11	22	55	44	66	33
0	1	2	3	4	5

Pass 3

11	22	55	44	66	33
0	1	2	3	4	5

11	22	33	44	66	55
0	1	2	3	4	5

Pass 4

11	22	33	44	66	55
0	1	2	3	4	5

11	22	33	44	66	55
0	1	2	3	4	5

Pass 5

11	22	33	44	66	55
0	1	2	3	4	5

11	22	33	44	55	66
0	1	2	3	4	5

to select position : i : $0 \rightarrow N-2$ $i < N-1$

to find smallest element : j : $i+1 \rightarrow N-1$ $j < N$

Selection sort

i					j
44	11	55	22	66	33
0	1	2	3	4	5

i	j	minIndex
0	1	0
	2	1
	3	
	4	
	5	

```
minIndex=i
for( j=i+1 ; j<N ; j++)
    if( arr[j] < arr[minIndex])
        minIndex =j;
```

	i				j
11	44	55	22	66	33
0	1	2	3	4	5

i	j	minIndex
1	2	1
	3	3
	4	
	5	

mathematical polynomial

Degree
↳ highest power of var

- Degree term is always highest growing term

n	n^2
1	1
10	100
100	10000
1000	1000000

itrs of outer loop = n-1

itrs of inner loop = n-1
n-2
n-3
⋮
2
1

$$\text{total itrs} = 1+2+3 \ldots\ldots + \underbrace{(n-1)}_{n}$$

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

$$\text{Time} \propto \frac{1}{2}(n^2+n)$$

$$\boxed{T(n) = O(n^2)}$$

Avg
Best
worst

$$\boxed{S(n) = O(1)}$$

Bubble sort

1. Compare all pairs of consecutive elements of the array one by one
2. If left element is greater than right element , then swap both
3. Repeat above steps until array is sorted

No. of elements = n

No. of passes = n-1

pass	comps
1	n-1
2	n-2
3	⋮
4	2
5	1

$$\text{Total comps} = 1+2+3 - - - - + n = \frac{n(n+1)}{2}$$

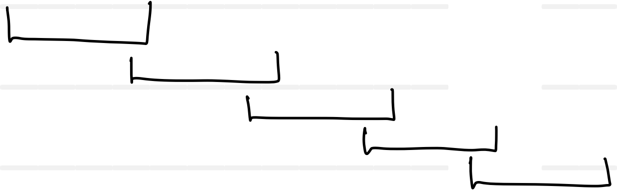
$$\text{Time} \propto \frac{1}{2}(n^2+n)$$

$$\boxed{T(n) = O(n^2)}$$ Avg worst

	j<N-1	j<N-i
i	j	j
1	0-4	0-4
2	0-4	0-3
3	0-4	0-2
4	0-4	0-1
5	0-4	0-0

Best case :

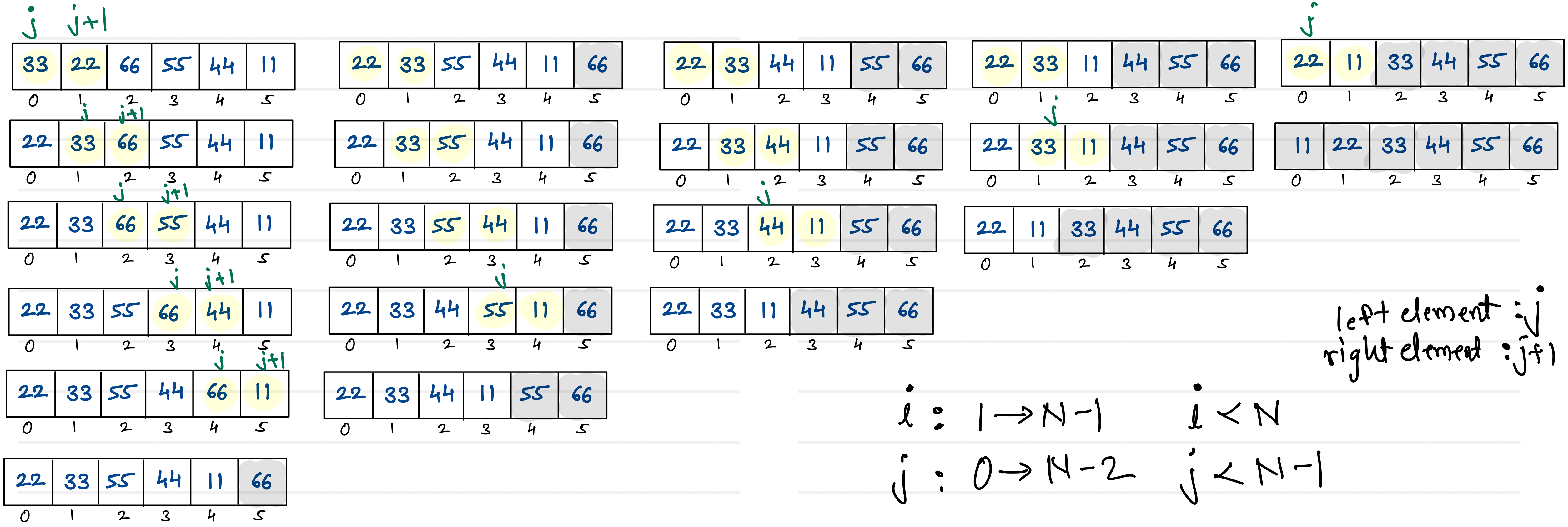
11 22 33 44 55 66



$$\boxed{T(n) = O(n)}$$ Best

Bubble sort

33	22	66	55	44	11
0	1	2	3	4	5



33	22	66	55	44	11
0	1	2	3	4	5

22	33	66	55	44	11
0	1	2	3	4	5

22	33	66	55	44	11
0	1	2	3	4	5

22	33	55	66	44	11
0	1	2	3	4	5

22	33	55	44	66	11
0	1	2	3	4	5

22	33	55	44	11	66
0	1	2	3	4	5

22	33	55	44	11	66
0	1	2	3	4	5

22	33	55	44	11	66
0	1	2	3	4	5

22	33	55	44	11	66
0	1	2	3	4	5

22	33	44	55	11	66
0	1	2	3	4	5

22	33	44	11	55	66
0	1	2	3	4	5

22	33	44	11	55	66
0	1	2	3	4	5

22	33	44	11	55	66
0	1	2	3	4	5

22	33	44	11	55	66
0	1	2	3	4	5

22	33	11	44	55	66
0	1	2	3	4	5

22	33	11	44	55	66
0	1	2	3	4	5

22	33	11	44	55	66
0	1	2	3	4	5

22	11	33	44	55	66
0	1	2	3	4	5

22	11	33	44	55	66
0	1	2	3	4	5

11	22	33	44	55	66
0	1	2	3	4	5

left element : j
right element : j+1

$i : 1 \rightarrow N-1 \quad i < N$

$j : 0 \rightarrow N-2 \quad j < N-1$

Insertion sort

1. Pick one element of the array (start from 2nd index)
2. Compare picked element with all its left neighbours one by one
3. If left neighbour is greater, move it one position ahead
4. Insert picked element at its appropriate position
5. Repeat above steps until array is sorted

No. of elements = n

No. of passes = n-1

passes	comps
1	1
2	2
3	3
⋮	
n-1	$\underset{n}{n-1}$

$$\text{Total comps} = 1+2+3+\ldots\ n$$

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

$$\text{Time} \propto \frac{1}{2}(n^2+n)$$

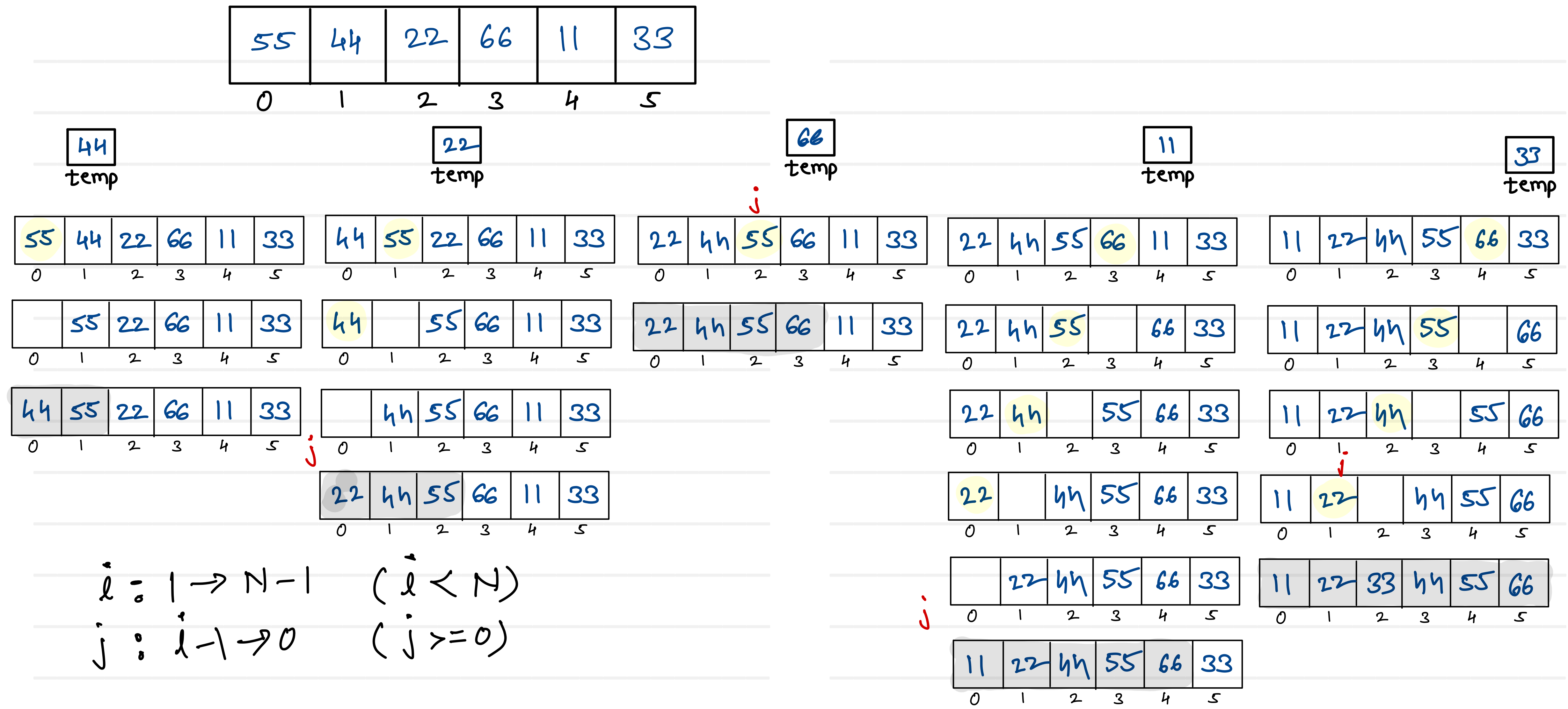
$$\boxed{T(n) = O(n^2)}$$ Worst Avg

Best case :

11 22 33 44 55

$$\boxed{T(n) = O(n)}$$ Best

Insertion sort



$i : 1 \rightarrow N-1 \quad (i < N)$

$j : i-1 \rightarrow 0 \quad (j >= 0)$

Insertion sort

0	1	2	3	4	5
11	22	33	44	55	66

```
for( i=1; i<N ; i++) {
        temp = arr[i] ;
        for( j=i-1; j>=0 ; j--) {
                if( arr[j] > temp)
                        arr[j+1] = arr[j];
                else
                        break;
        }
        arr[j+1] = temp ;
}
```

i	i<6	temp	j
1	T	44	0,-1
2	T	22	1,0,-1
3	T	66	2
4	T	11	3,2,1,0,-1
5	T	33	4,3,2,1

Remove Duplicates from sorted array

Given an integer array nums sorted in non-decreasing order, remove the duplicates in-place such that each unique element appears only once. The relative order of the elements should be kept the same. Then return the number of unique elements in nums.

Example 1:

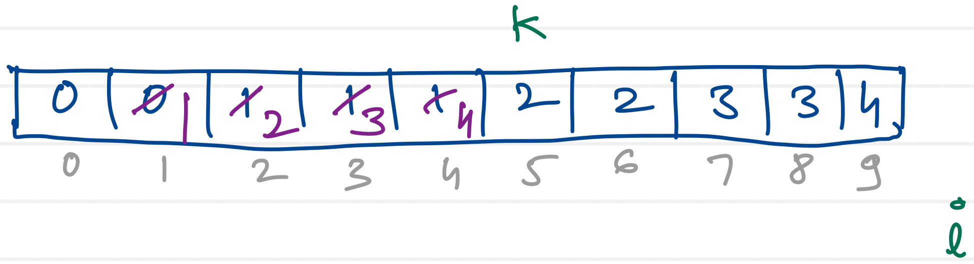
Input: nums = [1,1,2]

Output: 2, nums = [1,2,_]

Example 2:

Input: nums = [0,0,1,1,1,2,2,3,3,4]

Output: 5, nums = [0,1,2,3,4,_,_,_,_,_]



```
int removeDuplicates(int[] nums) {
        int n = nums.length;
        int k = 1;
        for( int i = 1 ; i < n ; i++) {
            if( nums[i] != nums[k-1])
                nums[k++] = nums[i];
        }
        return k;
}
```

SUNBEAM

Thank you!!!

Devendra Dhande

devendra.dhande@sunbeaminfo.com