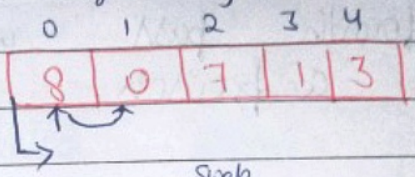
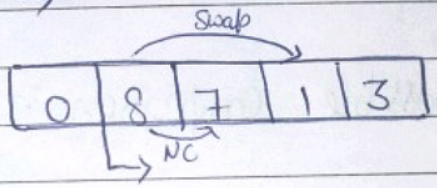
Date: / / Page no:

Selection - Sort :-

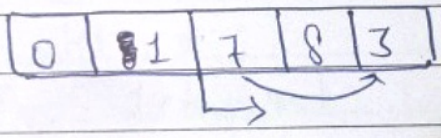
The selection sort algorithm sorts an array by finding the minimum element repeatedly from unsorted part and putting it at the beginning.



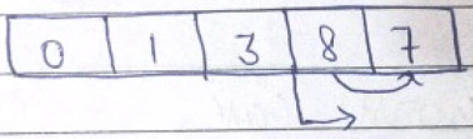
(i=0) 1st pass :-



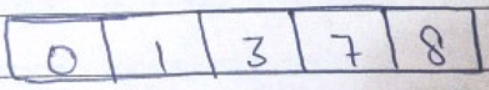
(i=1) 2nd pass :-



(i=2) 3rd pass :-



(i=3) 4th pass :-



{ array is sorted }

no of passes => (n-1)

n = length of array.

Date: / / Page no:

i) In first pass we had 4 possible comparison.

ii) In second pass we had 3 possible comparison.

iii) In third pass we had 2 possible comparison.

iv) In fourth pass we had only 1 possible comparison.

Total possible comparison :- $1+2+3+\dots+(n-1)$

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

$= \frac{n^2-n}{2}$ (Constant and less dominating term are removed)

Worst case :- $O(n^2)$

* It is not a stable sort.

* Selection sort is not adaptive.

means in sorted array we also need the same no of comparison.

Date: / / Page no:

- **Algorithm :-**

i) Start with 0th index.
ii) Iterate through the unsorted portion of the array, starting from the 1st index.
iii) Swap the min with current element.
iv) Increment the position of sorted portion by one.
v) Repeat the step 2-4.

* **Code implementation :-**

```
import java.util.*;
public class selectionsort
{
    int[] arr = {9, 26, 13, 5, 28, 16};
    Selection (arr);
    Sys o (Arrays.toString(arr));
}
Static void Selection (int[] arr)
{
  for (int i=0 ; i<arr.length-1 ; i++)
  {
    int min = i;
    for (int j=i+1 ; j<arr.length ; j++)
    {
      if (arr[j] < arr[min])
      {
        min = j;
      }
```

Date: / / Page no:

```
        }
            int temp = arr[min];
            arr[min] = arr[i];
            arr[i] = temp;
        }
    }
}
```