Insertion and Selection Sort

Md. Abu Sufyan
Roll: 2003085
Department: CSE

Md. Tameem Rahman
Roll: 2003089
Department: CSE

Partha Paul
Roll: 2003078
Department: CSE

Atik Mouhtasim Rahi
Roll: 2003118
Department: CSE

Md.Barkullah
Roll: 2003071
Department: CSE

Mir Ashikur Rahman
Roll: 2003109
Department: CSE

Rajshahi University of Engineering & Technology

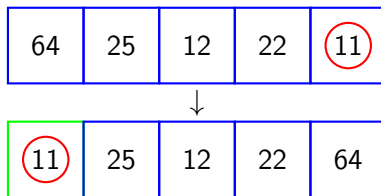January 20, 2024

Welcome

Welcome to our Presentation!

Md. Abu Sufyan
Roll: 2003085
Department: CSE

Md. Tameem Rahman
Roll: 2003089
Department: CSE

Partha Paul
Roll: 2003078
Department: CSE

Outline

Md. Abu Sufyan
Roll: 2003085
Department: CSE

Md. Tameem Rahman
Roll: 2003089
Department: CSE

Partha Paul
Roll: 2003078
Department: CSE

Selection Sort Overview

Selection Sort

Md. Abu Sufyan
Roll: 2003085
Department: CSE

Md. Tameem Rahman
Roll: 2003089
Department: CSE

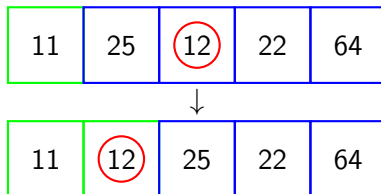Partha Paul
Roll: 2003078
Department: CSE

Selection Sort: First Pass

- For the first position in the sorted array, traverse the entire array.
- Find the minimum value (11) and swap it with the element at the first position (64).

64	25	12	22	(11)

$\downarrow$

(11)	25	12	22	64

Md. Abu Sufyan
Roll: 2003085
Department: CSE

Md. Tameem Rahman
Roll: 2003089
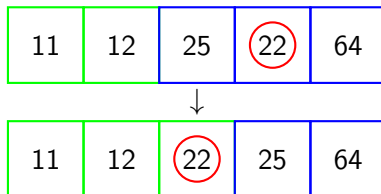Department: CSE

Partha Paul
Roll: 2003078
Department: CSE

Selection Sort: Second Pass

- For the second position, find the second minimum value (12) and swap it with the element at the second position (25).
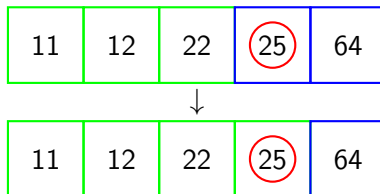
Md. Abu Sufyan   Md. Tameem Rahman   Partha Paul
Roll: 2003085   Roll: 2003089   Roll: 2003078
Department: CSE   Department: CSE   Department: CSE

Selection Sort: Third Pass

- For the third position, find the third minimum value (22) and swap it with the element at the third position (25).

Md. Abu Sufyan
Roll: 2003085
Department: CSE

Md. Tameem Rahman
Roll: 2003089
Department: CSE

Partha Paul
Roll: 2003078
Department: CSE

Selection Sort: Fourth Pass

- For the fourth position, find the fourth minimum value (25) and never swap it.

Md. Abu Sufyan   Md. Tameem Rahman   Partha Paul
Roll: 2003085   Roll: 2003089   Roll: 2003078
Department: CSE   Department: CSE   Department: CSE

Selection Sort: Fifth Pass

- The largest value (64) is automatically placed at the last position.

11	12	22	25	64

↓

11	12	22	25	64

Md. Abu Sufyan   Md. Tameem Rahman   Partha Paul
Roll: 2003085   Roll: 2003089   Roll: 2003078
Department: CSE   Department: CSE   Department: CSE
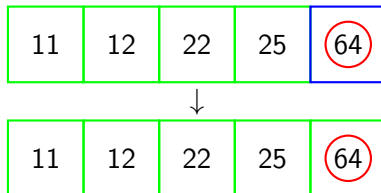
Selection Sort Algorithm: Overview

Algorithm 1: Selection Sort

Data: Array *arr* of size *n*

Result: Sorted array *arr*

1 **for** $i \leftarrow 0$ **to** $n - 1$ **do**

2      $min\_idx \leftarrow i$;

3   **for** $j \leftarrow i + 1$ **to** n **do**

4   **if** $arr[j] < arr[min\_idx]$ **then**

5             $min\_idx \leftarrow j$;

6   **if** $min\_idx \neq i$ **then**

7   Swap($arr[min\_idx]$, $arr[i]$);

Md. Abu Sufyan
Roll: 2003085
Department: CSE

Md. Tameem Rahman
Roll: 2003089
Department: CSE

Partha Paul
Roll: 2003078
Department: CSE

Selection Sort: C++ Code

```cpp
void selectionSort(int arr[], int n)
{
    int i, j, min_idx;
    for (i = 0; i < n - 1; i++) {
        min_idx = i;
        for (j = i + 1; j < n; j++) {
            if (arr[j] < arr[min_idx])
                min_idx = j;
        }
        if (min_idx != i)
            swap(arr[min_idx], arr[i]);
    }
}
```

Md. Abu Sufyan   Md. Tasneem Rahman   Partha Paul
Roll: 2003085   Roll: 2003089   Roll: 2003078
Department: CSE   Department: CSE   Department: CSE

Insertion Sort

Insertion Sort

Md. Abu Sufyan
Roll: 2003085
Department: CSE

Md. Tameem Rahman
Roll: 2003089
Department: CSE

Partha Paul
Roll: 2003078
Department: CSE
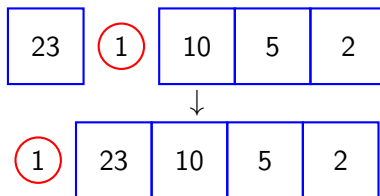
Insertion Sort: Step-by-Step

- For each element, insert it into its correct position in the sorted portion of the array.
- Shift elements greater than the key to the right.
- Repeat until the entire array is sorted.

Md. Abu Sufyan
Roll: 2003085
Department: CSE

Md. Tameem Rahman
Roll: 2003089
Department: CSE

Partha Paul
Roll: 2003078
Department: CSE

Insertion Sort: First Pass

- Initially, the first two elements of the array are compared in insertion sort.
- Here, 23 is greater than 1 hence they are not in the ascending order and 23 is not at its correct position. Thus, swap 1 and 23.

| 23 | ⬤ 1 | 10 | 5 | 2 |

↓

| ⬤ 1 | 23 | 10 | 5 | 2 |

Md. Abu Sufyan   Md. Tameem Rahman   Partha Paul
Roll: 2003085   Roll: 2003089   Roll: 2003078
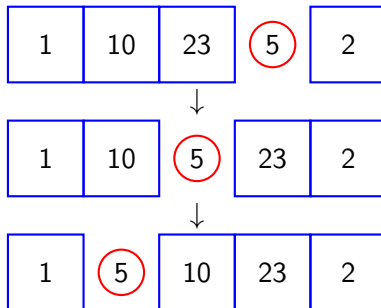Department: CSE   Department: CSE   Department: CSE

Insertion Sort: Second Pass

- Here, 23 is greater than 10 hence they are not in the ascending order and 10 is not at its correct position. Thus, swap 1 and 23. 10 also stored in a sorted sub-array along with 1
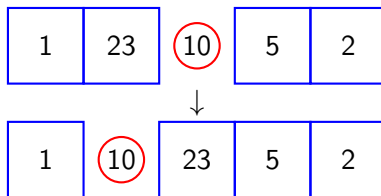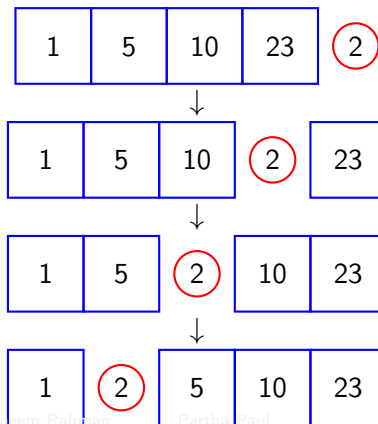
Insertion Sort: Third Pass

- Here, 5 isn't in correct position. So 5 has to be swapped with its previous position until 5 isn't greater than the previous value.

Md. Abu Sufyan
Roll: 2003085
Department: CSE   Md. Tameem Rahman
Roll: 2003089
Department: CSE   Partha Paul
Roll: 2003078
Department: CSE   Insertion and Selection Sort   January 20, 2024   16 / 27

Insertion Sort: Fourth Pass

- Here, 2 isn't in correct position. To place 2 in correct position, we have to follow the same procedure as third pass.
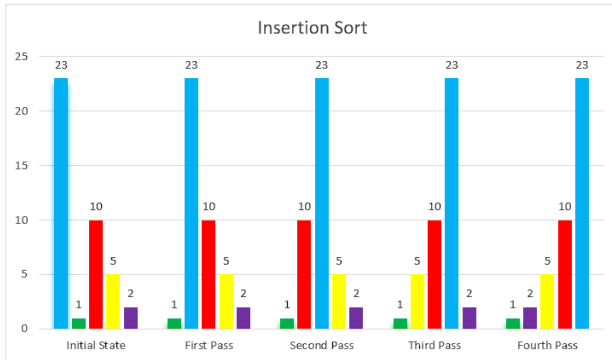
Insertion Sort: Visualization

Figure: Insertion Sort Visualization

Md. Abu Sufyan   Md. Tameem Rahman   Partha Paul
Roll: 2003085   Roll: 2003089   Roll: 2003078
Department: CSE   Department: CSE   Department: CSE

Insertion Sort Algorithm: Overview

Algorithm 2: Insertion Sort

Data: Array *arr* of size *n*
Result: Sorted array *arr*

1 **for** $i \leftarrow 1$ **to** n **do**
2 $\quad$ $key \leftarrow arr[i]$;
3 $\quad$ $j \leftarrow i - 1$;
4 $\quad$ **while** $j \geq 0$ **and** $arr[j] > key$ **do**
5 $\quad\quad$ $arr[j + 1] \leftarrow arr[j]$;
6 $\quad\quad$ $j \leftarrow j - 1$;
7 $\quad$ $arr[j + 1] \leftarrow key$;

Md. Abu Sufyan
Roll: 2003085
Department: CSE

Md. Tameem Rahman
Roll: 2003089
Department: CSE

Partha Paul
Roll: 2003078
Department: CSE

Insertion Sort: C++ Code

```cpp
void insertionSort(int arr[], int n) {
    for (int i = 1; i < n; i++) {
        int key = arr[i];
        int j = i - 1;

        while (j >= 0 && arr[j] > key) {
            arr[j + 1] = arr[j];
            j = j - 1;
        }

        arr[j + 1] = key;
    }
}
```

Md. Abu Sufian   Md. Tameem Rahman   Partha Paul
Roll: 2003085   Roll: 2003089   Roll: 2003078
Department: CSE   Department: CSE   Department: CSE

Result and Complexity Analysis

Size	Insertion (s)	Selection (s)
10	0.000011	0.000019
50	0.000176	0.000155
100	0.000412	0.000363
500	0.004555	0.004457
1000	0.020648	0.018857
5000	0.528669	0.485517
10000	2.105405	1.965638
50000	61.877976	55.541591
100000	238.821475	271.276342

Table: Performance Comparison of Insertion Sort and Selection Sort

Md. Abu Sufyan
Roll: 2003085
Department: CSE

Md. Tameem Rahman
Roll: 2003089
Department: CSE

Partha Paul
Roll: 2003078
Department: CSE

Graph with Explanation

Explanation:

- As dataset size grows, both algorithms exhibit expected quadratic time complexity, leading to increased running times.
- Graph lines depict the upward trend, showcasing the correlation between runtime and input data size.
- In smaller datasets, Insertion Sort may outperform Selection Sort, but as data size increases, Selection Sort may surpass Insertion Sort in relative efficiency.
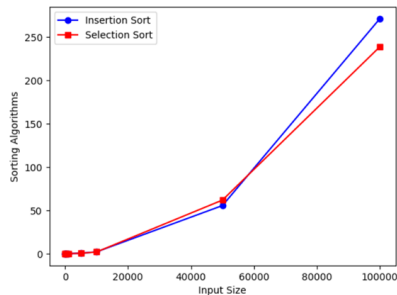

Figure: Complexity Graph.

Md. Abu Sufyan
Roll: 2003085
Department: CSE

Md. Tameem Rahman
Roll: 2003089
Department: CSE

Partha Paul
Roll: 2003078
Department: CSE

Sorting Algorithms - Time Complexity

Time Complexity:

- **Selection Sort:** $O(n^2)$ - traverses unsorted part for each element.
- **Insertion Sort:** $O(n^2)$ - iteratively compares for correct positioning.

Best-Case Scenario:

- **Selection Sort:** $O(n^2)$ - traverses entire unsorted part even with partial sorting.
- **Insertion Sort:** $O(n)$ - when already sorted, compares with adjacent elements for verification.

Md. Abu Sufyan
Roll: 2003085
Department: CSE

Md. Tameem Rahman
Roll: 2003089
Department: CSE

Partha Paul
Roll: 2003078
Department: CSE

Sorting Algorithm Performance

- **Comparison:** Evaluated Insertion Sort and Selection Sort across varying dataset sizes to understand their efficiency dynamics.
- **Findings:** Insertion Sort excels on small, well-sorted datasets, but as size increases, Selection Sort emerges as a robust option, possibly due to a decreasing constant factor in time complexity.
- **Decision Factors:** Choosing between methods should align with data characteristics and sorting task scale, acknowledging the merits of each algorithm.
- **Limitations and Future Research:** Recognized study limitations, prompting future exploration into real-world algorithm effects and optimization possibilities. Our research contributes valuable insights for informed algorithm selection in diverse applications.

Md. Abu Sufyan
Roll: 2003085
Department: CSE

Md. Tameem Rahman
Roll: 2003089
Department: CSE

Partha Paul
Roll: 2003078
Department: CSE

References

Md. Abu Sufyan
Roll: 2003085
Department: CSE

Md. Tameem Rahman
Roll: 2003089
Department: CSE

Partha Paul
Roll: 2003078
Department: CSE

Questions & Answers

Any Questions?

Md. Abu Sufyan
Roll: 2003085
Department: CSE

Md. Tameem Rahman
Roll: 2003089
Department: CSE

Partha Paul
Roll: 2003078
Department: CSE

Thanks

Thank You for Your Attention!

Md. Abu Sufyan
Roll: 2003085
Department: CSE

Md. Tameem Rahman
Roll: 2003089
Department: CSE

Partha Paul
Roll: 2003078
Department: CSE