Insertion and Selection Sort

Md. Abu Sufyan
Roll: 2003085
Department: CSE

Tameem Rahman
Roll: 2003089
Department: CSE

Partha Paul
Roll: 2003078
Department: CSE

Atik Mohtasin Rahi
Roll: 2003118
Department: CSE

Barkatulla Barik
Roll: 2003071
Department: CSE

Mir Ashikur Rahman
Roll: 2003109
Department: CSE

Rajshahi University of Engineering & Technology

January 20, 2024

Welcome

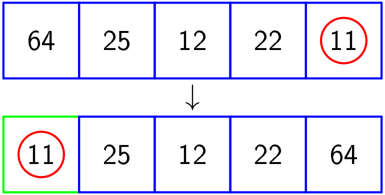
Welcome to our Presentation!

Outline

1. Welcome
2. Selection sort First Pass
   - Selection Sort: Second Pass
   - Selection Sort: Third Pass
   - Selection Sort: Fourth Pass
   - Selection Sort: Fifth Pass
   - Selection Sort Algorithm: Overview
3. Insertion Sort Steps
   - Insertion sort First Pass
   - Insertion sort Second Pass
   - Insertion sort Third Pass
   - Insertion sort Forth Pass
   - Insertion Sort Demonstration
   - Insertion Sort Algorithm
   - Insertion Sort C++ Code
4. Conclusion
5. References
   - References for Insertion Sort
   - References for Selection Sort
6. Questions & Answers
7. Thanks

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).



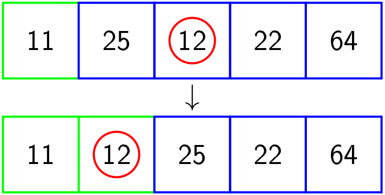
64	25	12	22	11

↓

11	25	12	22	64

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).



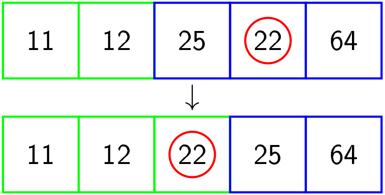
11	25	12	22	64

↓

11	12	25	22	64

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).



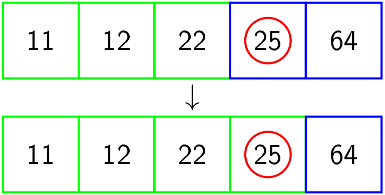
11	12	25	22	64

↓

11	12	22	25	64

Selection Sort: Fourth Pass

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



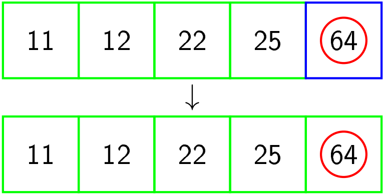
11 12 22 25 64

↓

11 12 22 25 64

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

Selection Sort Algorithm: Overview

Algorithm 1: Selection Sort

```
Data: Array $arr$ of size $n$
Result: Sorted array $arr$
for $i \leftarrow 0$ to $n - 1$ do
    $min\_idx \leftarrow i$;
    for $j \leftarrow i + 1$ to $n$ do
        if $arr[j] < arr[min\_idx]$ then
            $min\_idx \leftarrow j$;
    if $min\_idx \neq i$ then
        Swap($arr[min\_idx]$, $arr[i]$);
```

Selection Sort: C++ Code

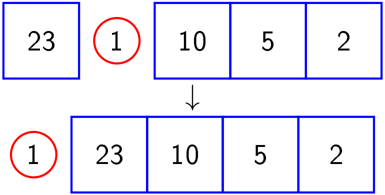
```
void selectionSort(int arr[], int n)
{
    int i, j, min_idx;
    for (i = 0; i < n - 1; i++) {
        min_idx = i;
        for (j = i + 1; j < n; j++) {
            if (arr[j] < arr[min_idx])
                min_idx = j;
        }
        if (min_idx != i)
            swap(arr[min_idx], arr[i]);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

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.

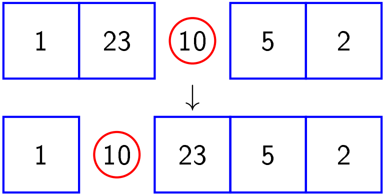
Insertion Sort: First Pass

- 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.



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



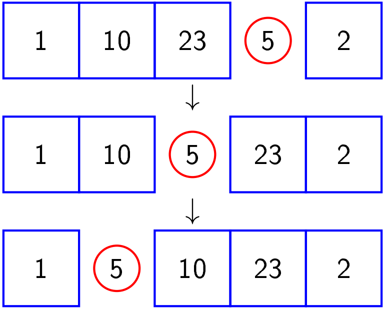
1 23 10 5 2

↓

1 10 23 5 2

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.



1 10 23 5 2

↓

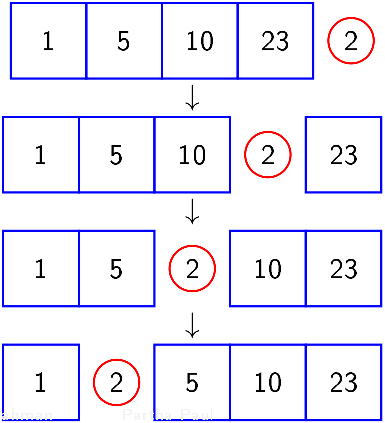
1 10 5 23 2

↓

1 5 10 23 2

Insertion Sort: Forth Pass

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



1 5 10 23 2

↓

1 5 10 2 23

↓

1 5 2 10 23

↓

1 2 5 10 23

Insertion Sort: Visualization

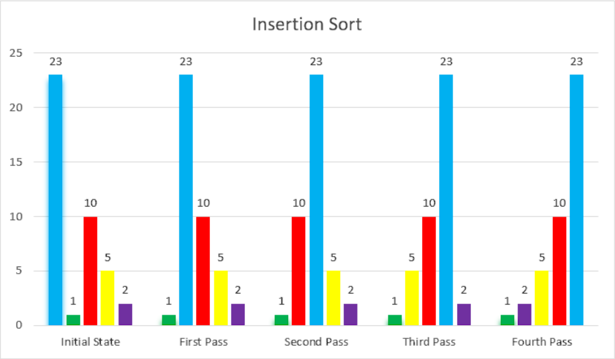


Figure: Insertion Sort Visualization

Insertion Sort Algorithm: Overview

Algorithm 2: Insertion Sort

```
**Data:** Array $arr$ of size $n$
**Result:** Sorted array $arr$
for $i \leftarrow 1$ to $n$ do
    $key \leftarrow arr[i]$;
    $j \leftarrow i - 1$;
    while $j \geq 0$ and $arr[j] > key$ do
        $arr[j + 1] \leftarrow arr[j]$;
        $j \leftarrow j - 1$;
    $arr[j + 1] \leftarrow key$;
```

Insertion Sort: C++ Code

```
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;
    }
}
```

Conclusion

- Insertion Sort is a simple and intuitive sorting algorithm.
- It efficiently builds the final sorted array one element at a time.
- While not as efficient on large datasets as more advanced algorithms, it performs well on small datasets or nearly sorted datasets.

References

- Author et al., *Introduction to Algorithms*, 3rd Edition.
- Author and Collaborator, *Journal of Sorting Algorithms*, 20XX.

References for Insertion Sort

- Author et al., *Journal of Sorting Algorithms*, 20XX. https://example.com/paper1
- Author and Collaborator, *Conference on Algorithms*, 20XX. https://example.com/paper2

References for Selection Sort

- Author et al., *Journal of Sorting Algorithms*, 20XX. https://example.com/paper1
- Author and Collaborator, *Conference on Algorithms*, 20XX. https://example.com/paper2

Questions & Answers

Any Questions?

Thanks

Thank You for Your Attention!