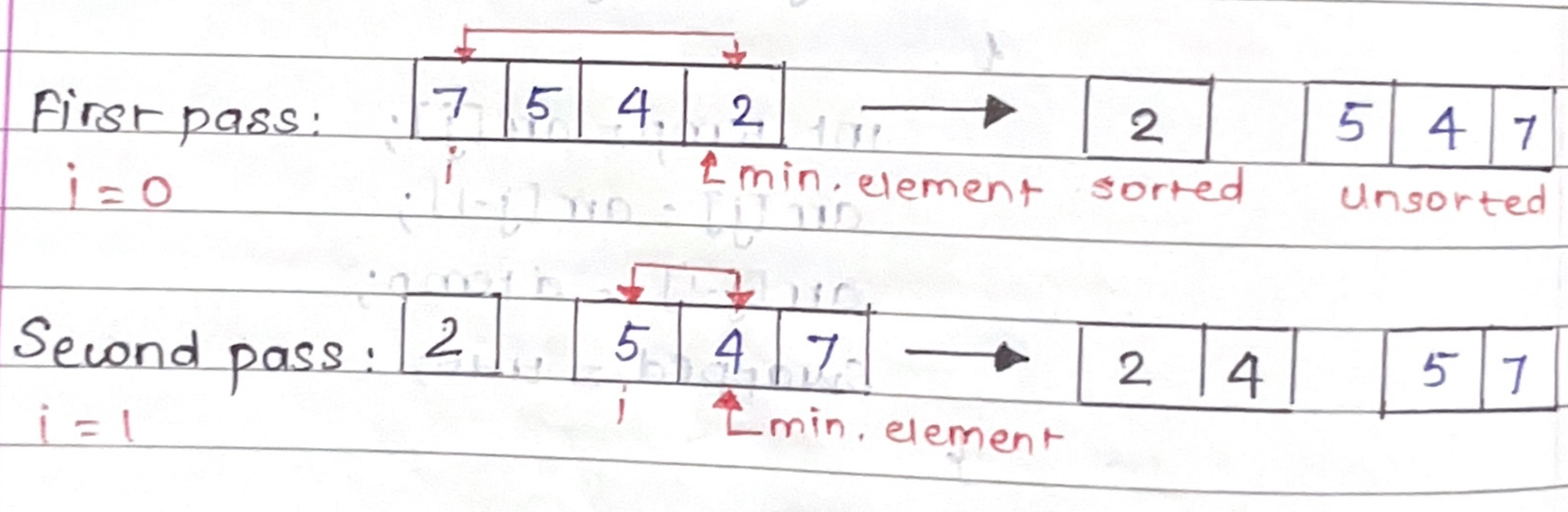
SELECTION SORT

Selection sort is a simple and efficent sorting algorith that works by repeatedly selecting the smallest (or largest) element from the unsorted portion of the list and moving it to sorted portion of the list.

Working -

1. Start:
2. Repeat: For each position in the list:
   - Find the smallest/largest number in remaining unsorted part
   - Swap it with the number in the wrrent position
3. End: After going through all positions, the list will be sorted.



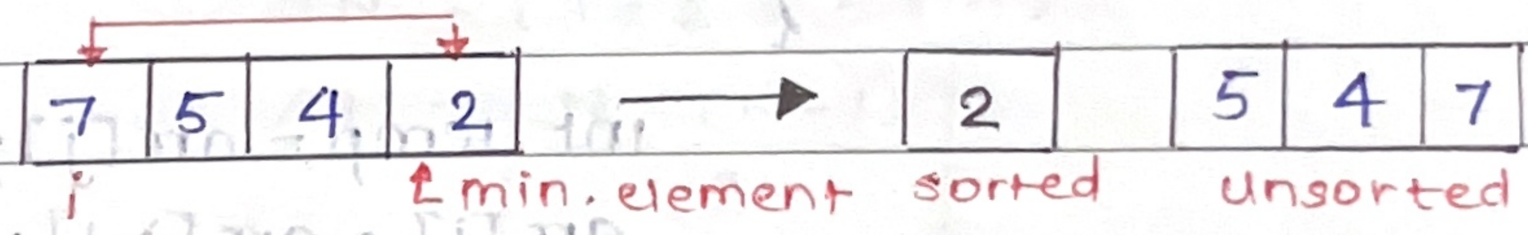
SELECTION SORT

Selection sort is a simple and efficent sorting algorithm that works by repeatedly selecting the smallest (or largest) element from the unsorted portion of the list and moving it to sorted portion of the list.

Working -

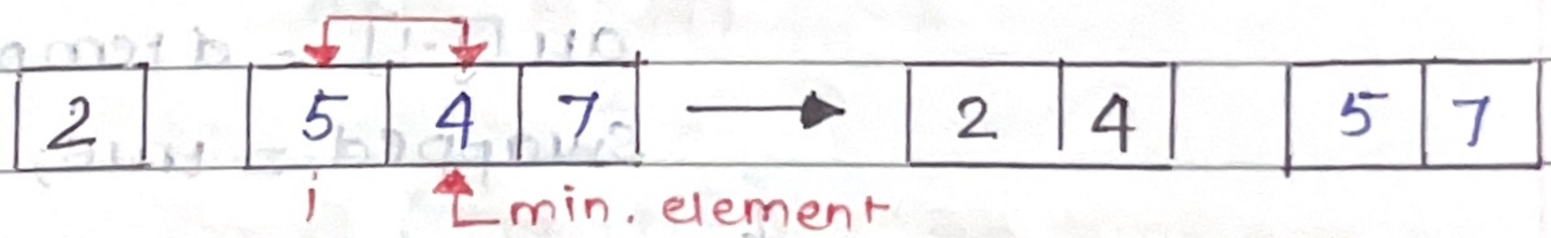
1. Start:
2. Repeat: For each position in the list:
   - Find the smallest/largest number in remaining unsorted part
   - Swap it with the number in the wrrent position
3. End: After going through all positions, the list will be sorted.

First pass:



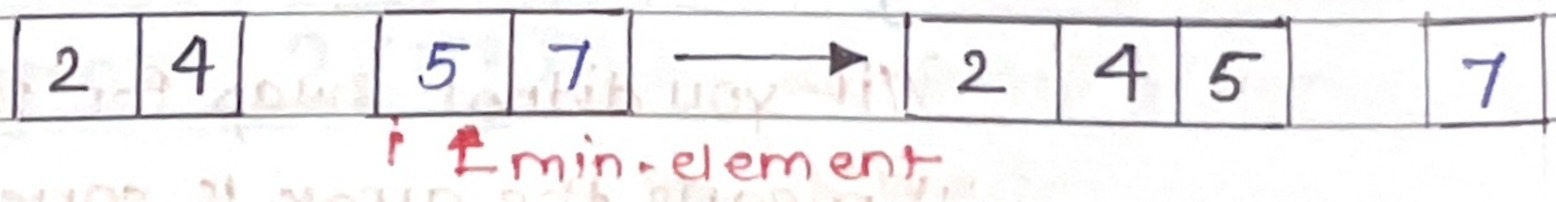
i = 0

Second pass:



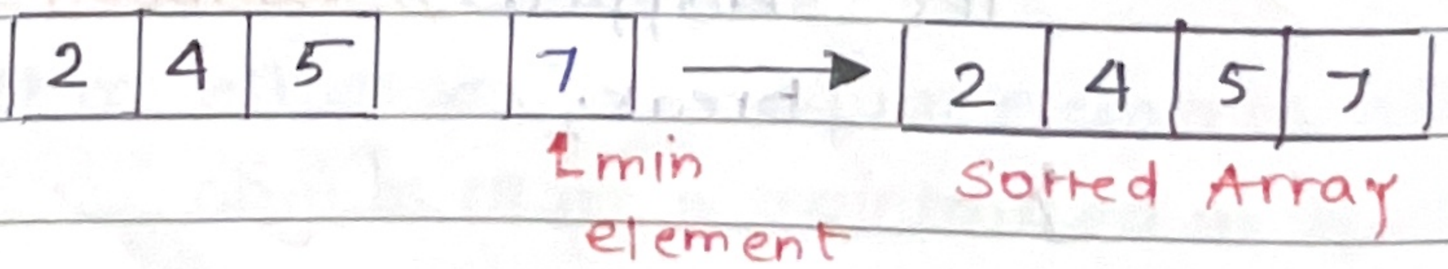
i = 1

Third pass:



i = 2

Fourth pass:



i = 3

- Time complexity

4	5	1	2	3	(n-1)
4	3	1	2	5	(n-2)
2	3	1	4	5	(n-3)
2	1	3	4	5	:
1	2	3	4	5	1
					0

Total comparisons: $0 + 1 + 2 + 3 \cdots (n-1)$

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

$$\frac{n^2 - n}{2}$$

$$O(n^2)$$

- Worst Case ⟶ $O(N^2)$
- Best Case ⟶ $O(N^2)$

Selection Sort is an unstable sorting algorithm

Example:

```
class Main {
    public static void main (String [] args) {
        int [] arr = { 64, 25, 12, 22, 11};
        Selection (arr);
        System.out.println (Arrays.toString (arr));
    }
    Static void swap (int[] arr, int first, int second) {
        int temp = arr [first];
        arr [first] = arr [second];
        arr [second] = temp;
    }
    static int minIndex (int[] prr, int start, int end) {
        int min = start;
        for (int i = start ; i < end; i++) {
            if (arr[i] < arr [start])
                min = i;
        }
        return min;
    }
    Static void selection (int[] arr) {
        for (int i=0; i < arr.length ; i++) {
            int min = minIndex (arr, i, arr.length);
            swap (arr, i, min);
        }
    }
}
```

O/P- [11, 12, 22, 25, 64]

Working -

	0	1	2	3	4
Orignal Array	64	25	12	22	11
	i				

First Pass: The smallest element is 11 (at index 4)
Swap 11 and the element at the begning (index 0) i.e i
Array after pass:

11	25	12	22	64
	i			

Second Pass: The smallest element in remaing part (from index 0 to end) is 12 (at index 2)
Swap 12 and the element at index 1 i.e i
Array after pass:

11	12	25	22	64
		j		

Third Pass: The smallest elemedht in remaing part (from index 2 to end) is 22 (at index 3)
Swap 22 and the element at index 3 i.e i
Array after pass:

11	12	22	25	64
			i	

fourth Pass: The smallest element in remaining part (from index 3 to end) is 25 (at index 3)
Swap 25 and the element at index 3
Array after pass

11	12	22	25	64

fifth pass: All elements are sorted so no swaps needed
The array remains -

11	12	22	25	64