Sorting →

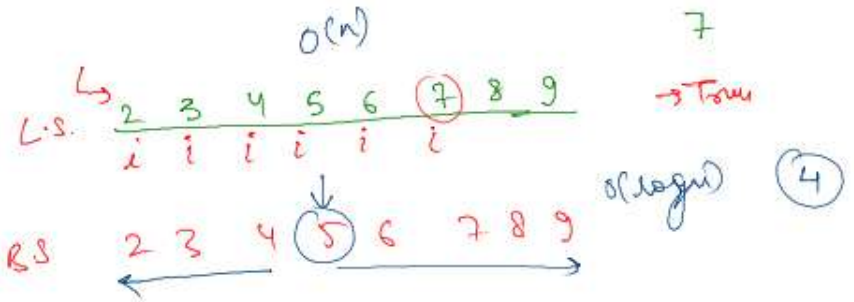
Arrangement of elements in a particular order either increasing or decreasing.

O(n) → O(logn)

Uses

✓ → Data stored in structured format
So, max & min

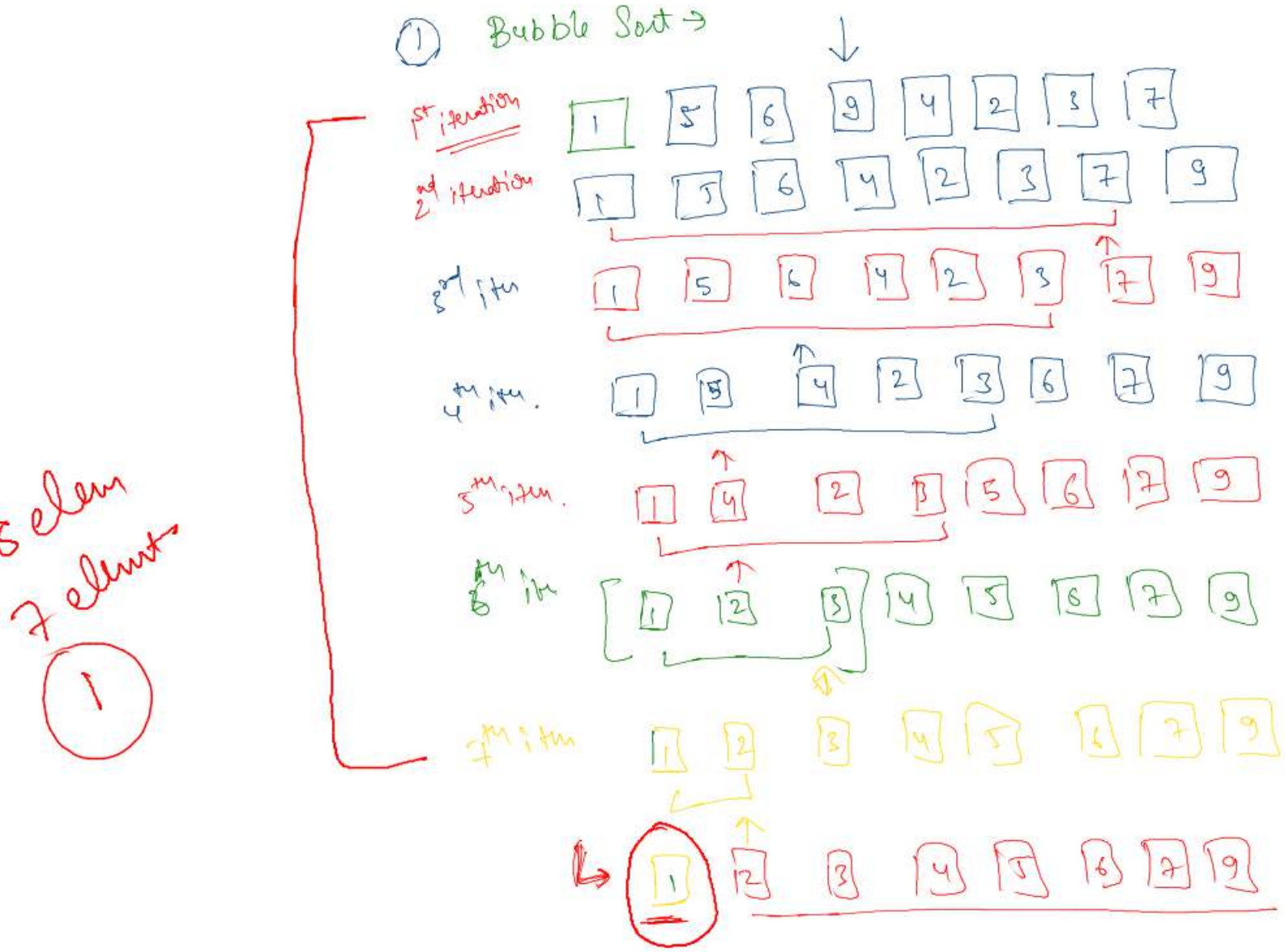
→ Searching of element will be easy

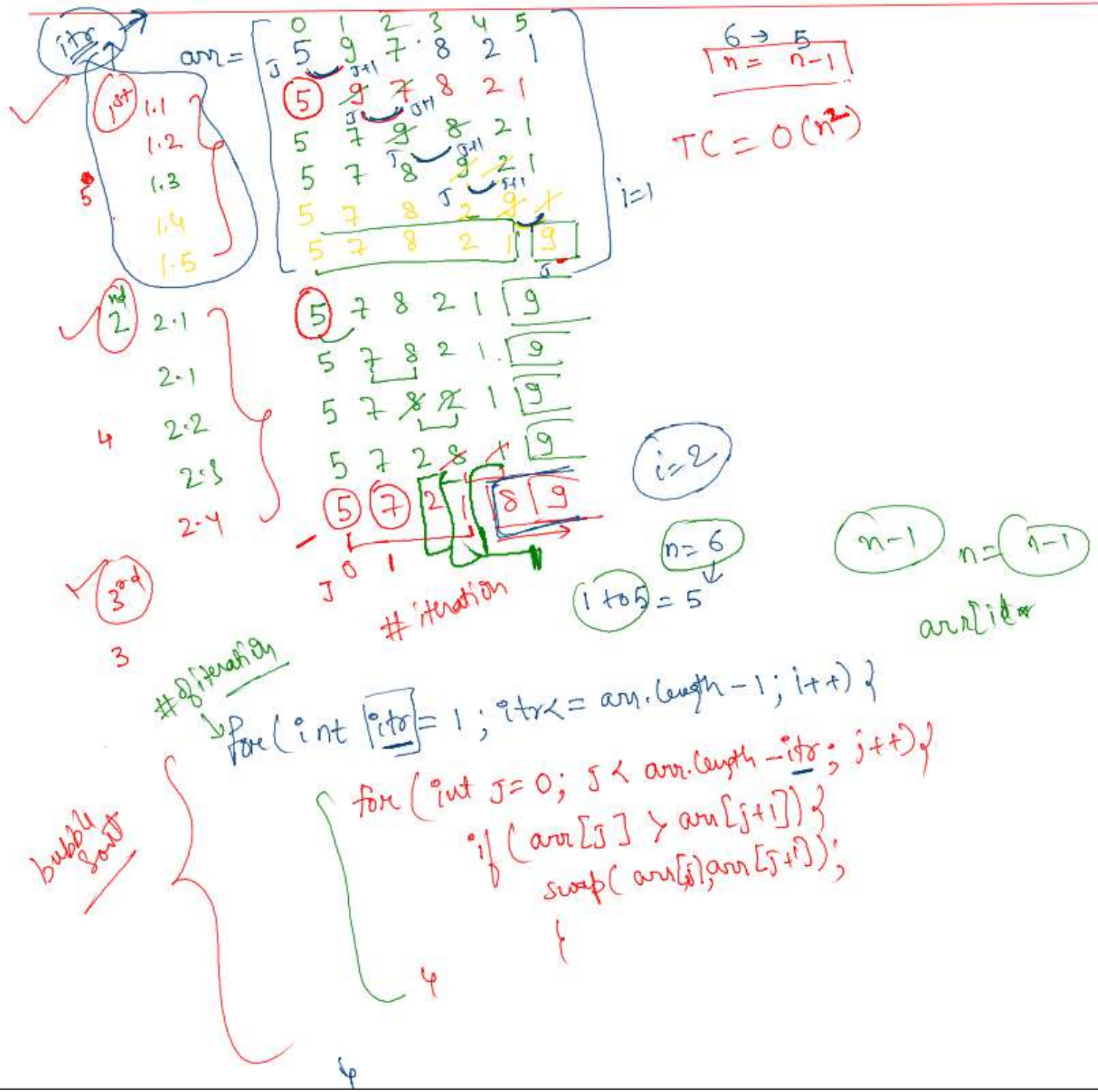


- Bubble Sort
- Selection Sort
- Insertion Sort
- Quick Sort
- Merge Sort

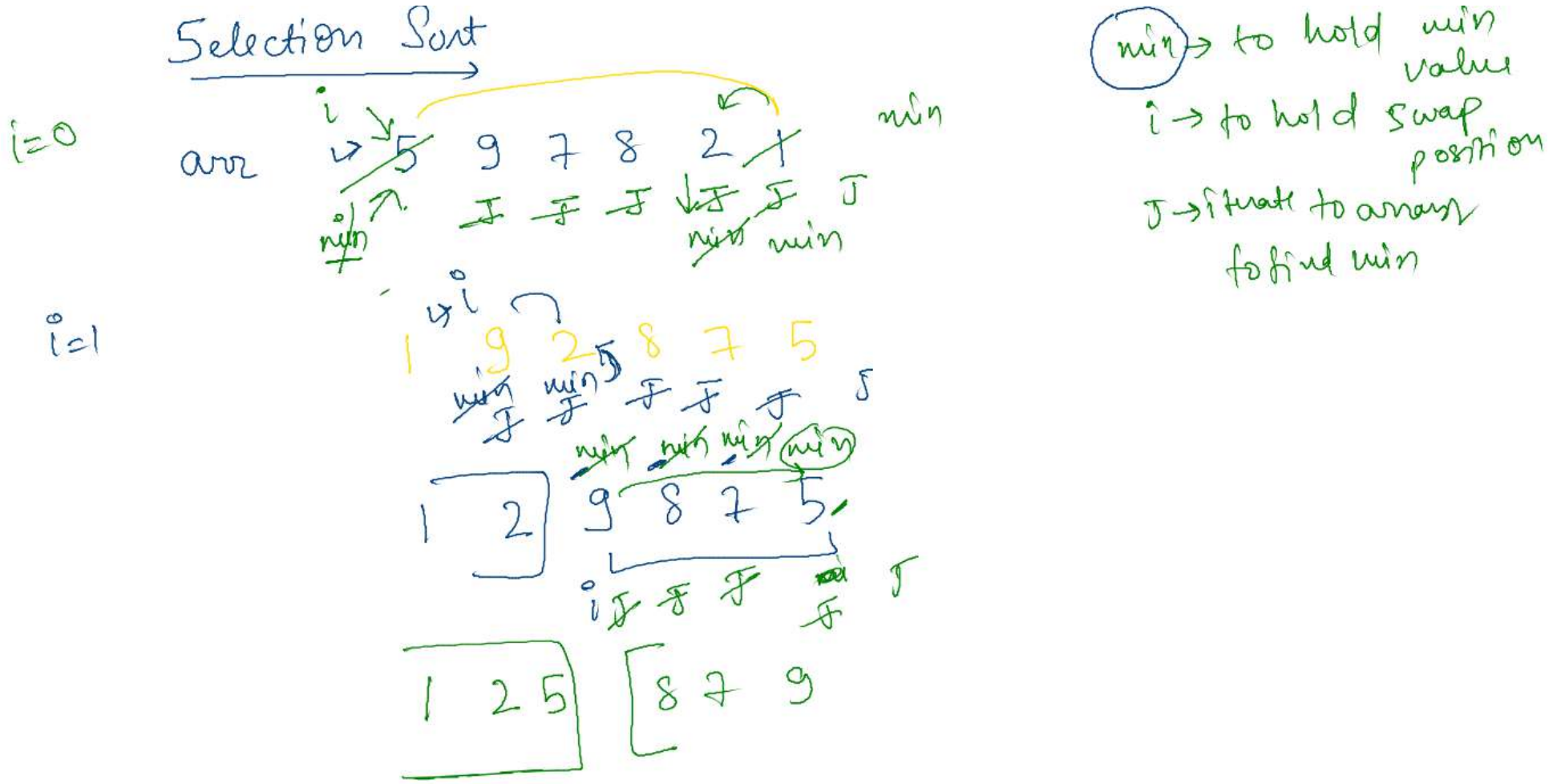
} important

- Date Sort
- Parity Sort
- Heap Sort (max min) ✓
- Count Sort
- Bucket Sort
- Radix Sort

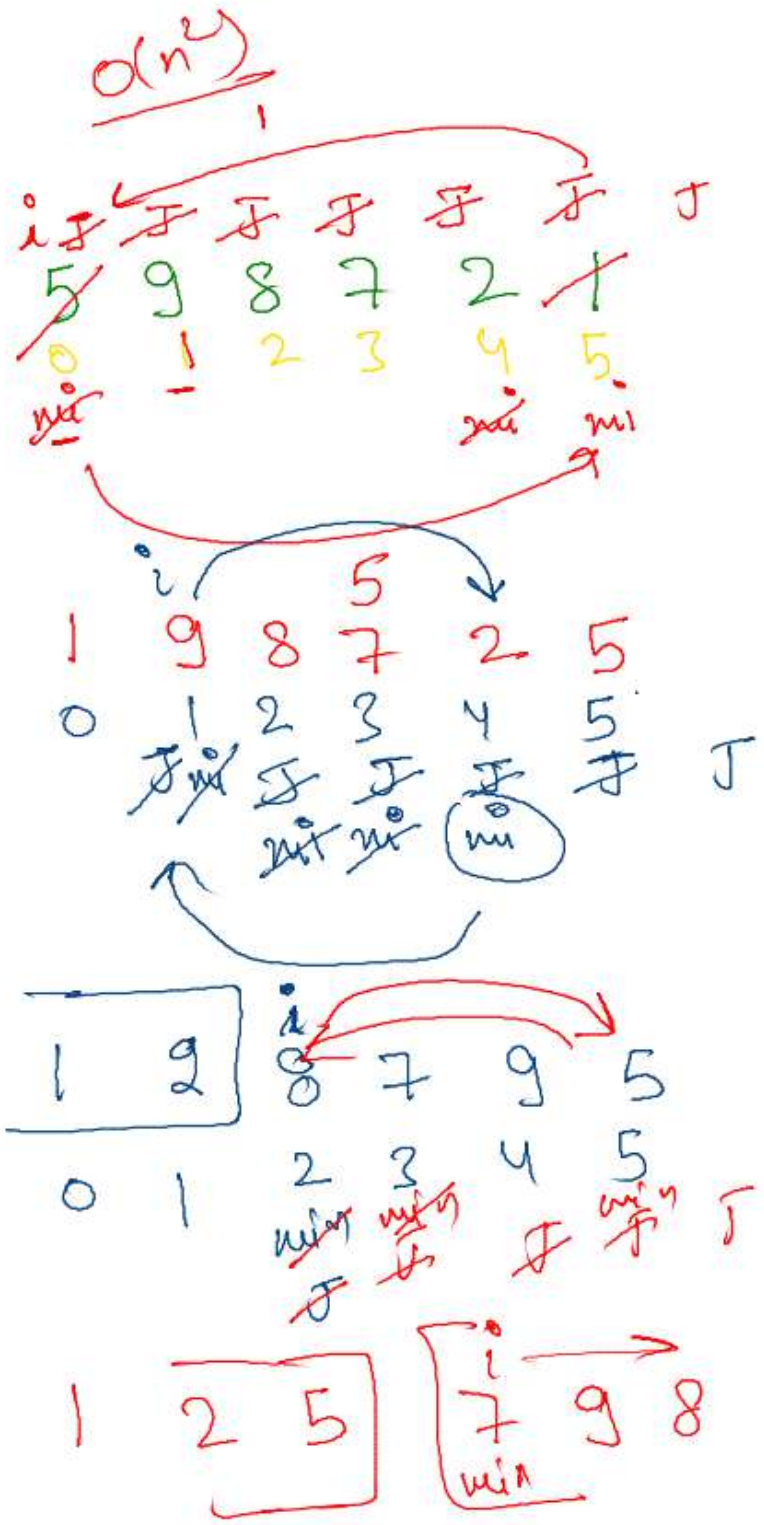

1 Bubble Sort →
1st iteration
2nd iteration
3rd itn
4th itn.
5th itn.
6th itn
7th itn
1 5 6 9 4 2 3 7
1 5 6 4 2 3 7 9
1 5 6 4 2 3 7 9
1 5 4 2 3 6 7 9
1 4 2 3 5 6 7 9
1 2 3 4 5 6 7 9
1 2 3 4 5 6 7 9
1 2 3 4 5 6 7 9
1
7 elem
7 elem
1

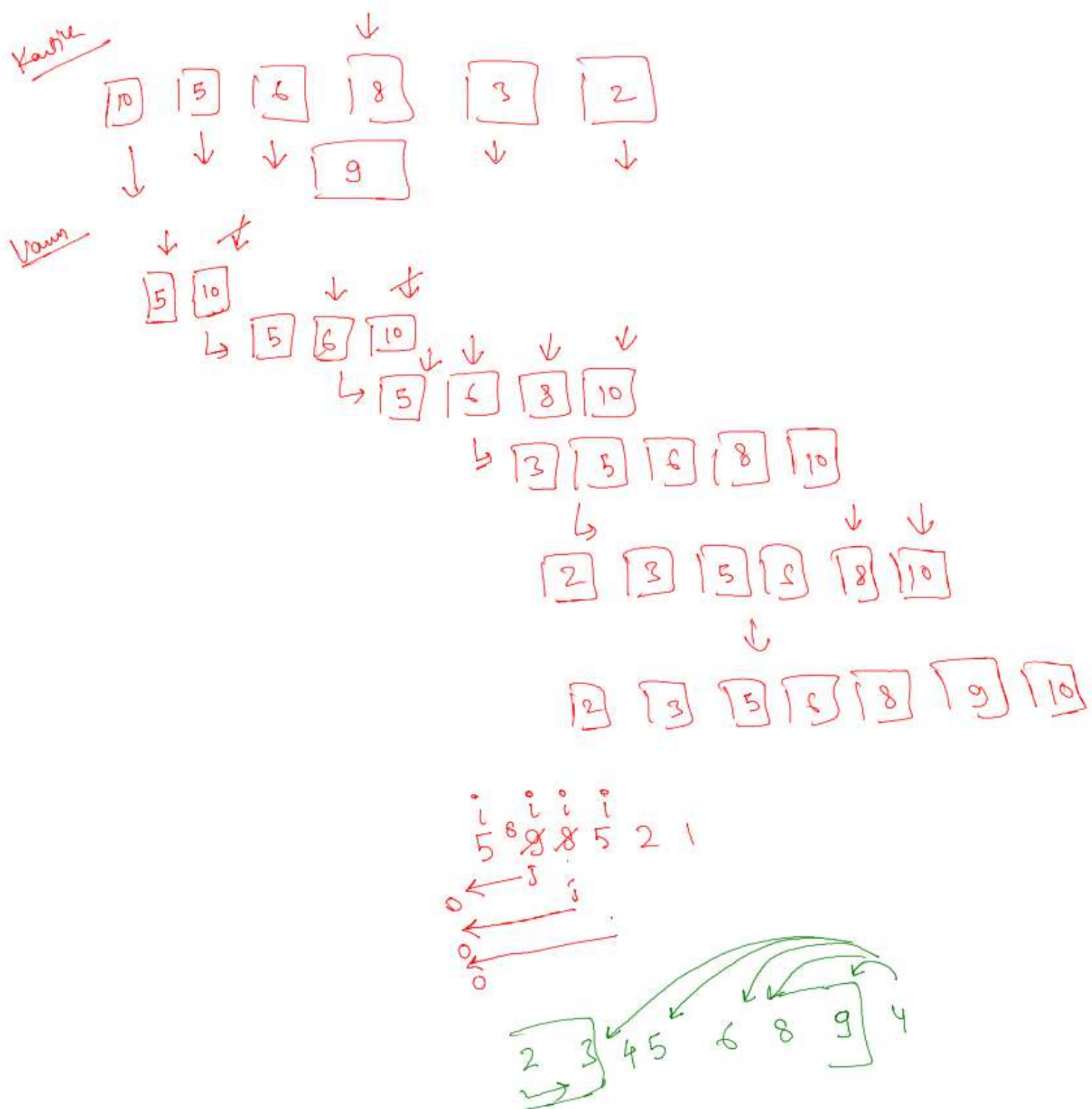

itr
arr =
0 1 2 3 4 5
5 9 7 8 2 1
1st
1.1
1.2
1.3
1.4
1.5
5
i=1
2nd
2.1
2.1
2.2
2.3
2.4
4
3rd
3
i=2
n=6
1 to 5 = 5
iteration
6 → 5
n = n-1
TC = O(n²)
n-1
n = n-1
arr[itr
# of iteration
bubble Sort
for(int itr = 1; itr <= arr.length - 1; i++) {
for(int j = 0; j < arr.length - itr; j++){
if(arr[j] > arr[j+1]){
swap(arr[j], arr[j+1]);
}
}
}

Selection Sort



```java
/* Enter your code here. Read input from STDIN. Print output to S
Scanner scn = new Scanner(System.in);
int n = scn.nextInt();
int[] arr = new int[n];
for(int i=0;i<n;i++){
    arr[i]= scn.nextInt();
}
//algorithm

for(int i=0;i<arr.length;i++){
    int mi = i;
    for(int j=i;j<arr.length;j++){
        if(arr[mi]>arr[j]){
            mi=j;
        }
    }

    int temp = arr[mi];
    arr[mi]=arr[i];
    arr[i]=temp;
}



for(int i =0;i<arr.length;i++){
    System.out.print(arr[i]+" ");
}
}
```




Kaise
10 5 6 8 3 2
9
Vaun
5 10
5 6 10
5 6 8 10
3 5 6 8 10
2 3 5 5 8 10
2 3 5 5 8 9 10
i i i i
5 6 8 8 5 2 1
j i
0
0
0
2 3 4 5 6 8 9