BUBBLE SORT

(13, 46, 24, 52, 20, 9)
0 1 2 3 4 5

// push the max to the last

// compare adjacent member swap if (left > right)

// the repeat for size less 1 than previous

this can be :

first loop → 0 → (n-1)
0 → (n-2)
0 → (n-3)
⋮
0 → 1

```
for (i = n-1 ; i ≥ 0 ; i--) {
    for (j = 0 ; j <= i-1 ; j++) {
        if (arr[j] > arr[j+1]) swap;
    }
}
```

as last element won't have anyone after to compare with

COMPLEXITY → (n) + (n-1) + (n-2) + (n-3) ... + 1

$= \frac{n(n+1)}{2} = O(n^2)$ → worst
↳ average

```
    }
    int temp = a
    if (minIndex != i) {
        int temp = arr[minIndex];
        arr[minIndex] = arr[i];
        arr[i] = temp;
    }
}
}
```

COMPLEXITY -

1st 2nd

↓ ↓

inner loop → $n, n-1, n-2, n-3, \ldots 2$

runs 5, 4, 3, 2 ...

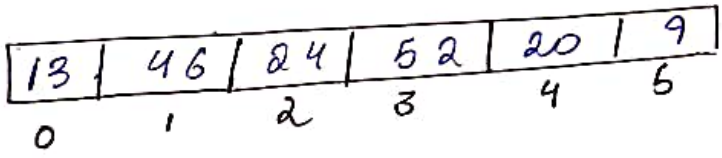
approx to sum of natural numbers

$= \frac{n(n+1)}{2} \Rightarrow \frac{n^2}{2} + \frac{n}{2}$

$O(n^2)$

Best worst average

SELECTION SORT



// pick minimum (9)
// place at first position by swapping
// pick min → swap → with second index
. . . .

step → 9 46 24 52 20 15
step 2 → 9 13 24 52 20 46
step 3 → 9 13 20 52 24 46
step 4 → 9 13 20 24 52 46
step 5 → 9 13 20 24 46 52

```
void selectionSort (int arr[]) {
    int size = arr.length;
    int minIndex = i;
for (int i=0; i<size-1; i++) {        → going upto (n-2)
        int minInden=i;
for (int j=i+1; j<size; j++){         (n-1)
        if (arr[j] < arr[minInden]) {
            minInden = j
}
```