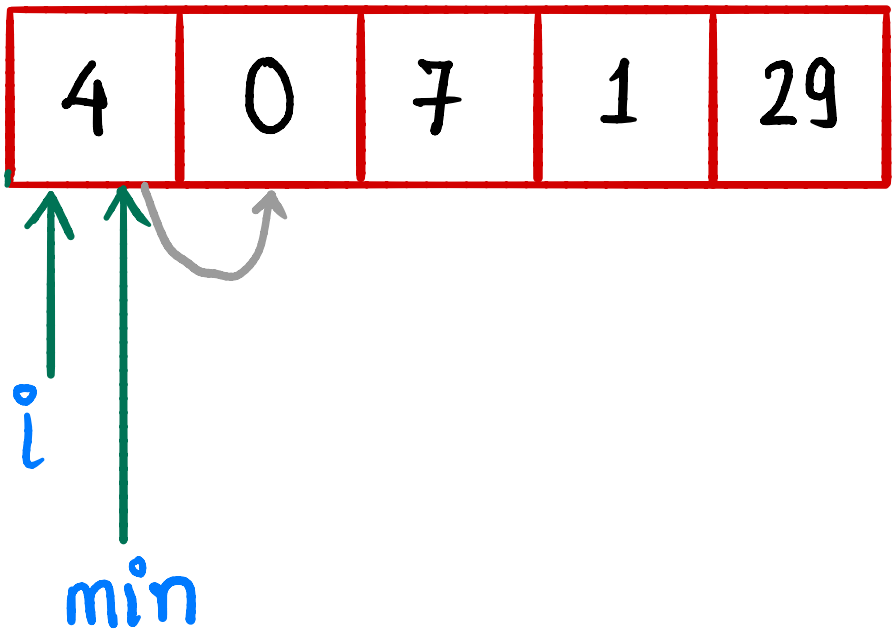
Example 7 :- (Selection Sorting)

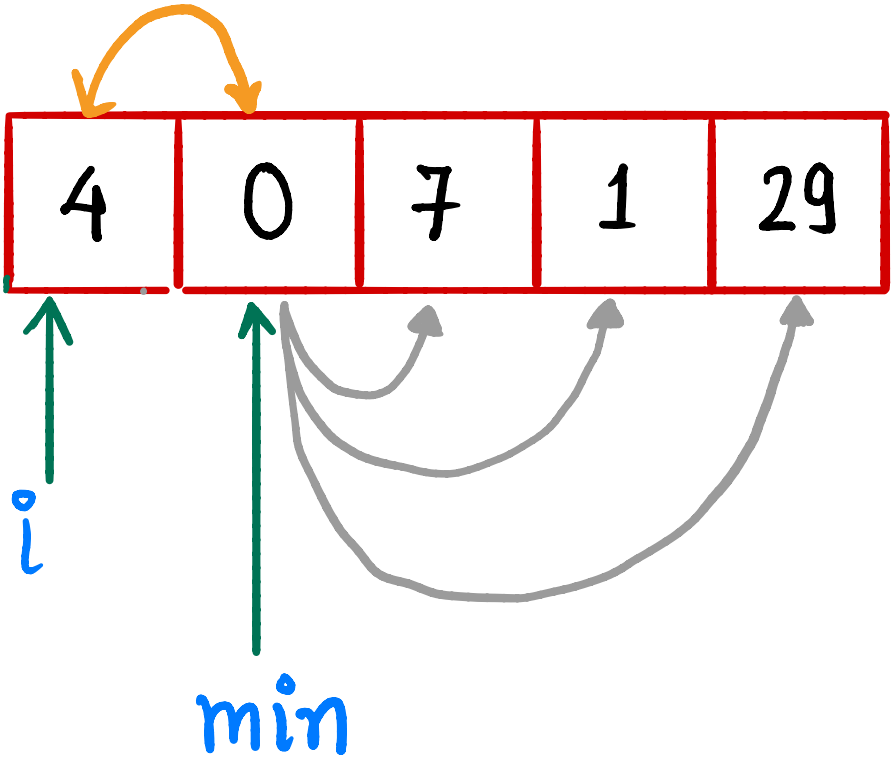
(List 1)	(List 2)
$\langle 4, 0, 7, 1, 29 \rangle$	$\langle \ \rangle$
$\langle 4, 7, 1, 29 \rangle$	$\langle 0 \rangle$
$\langle 4, 7, 29 \rangle$	$\langle 0, 1 \rangle$
$\langle 7, 29 \rangle$	$\langle 0, 1, 4 \rangle$
$\langle 29 \rangle$	$\langle 0, 1, 4, 7 \rangle$
$\langle \ \rangle$	$\langle 0, 1, 4, 7, 29 \rangle$

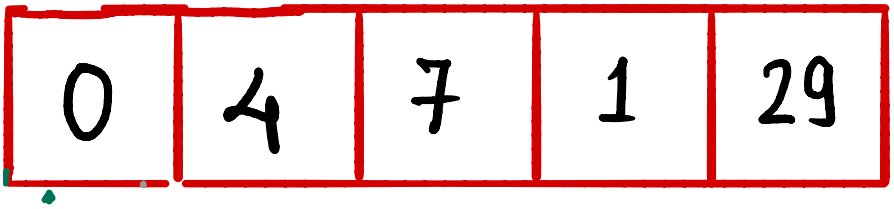
Only problem is the above algorithm is, it is not inplace sorting, You require list 2. which is extra-overhead. It is easy to modify the algorithm for in-place sorting.

i=0

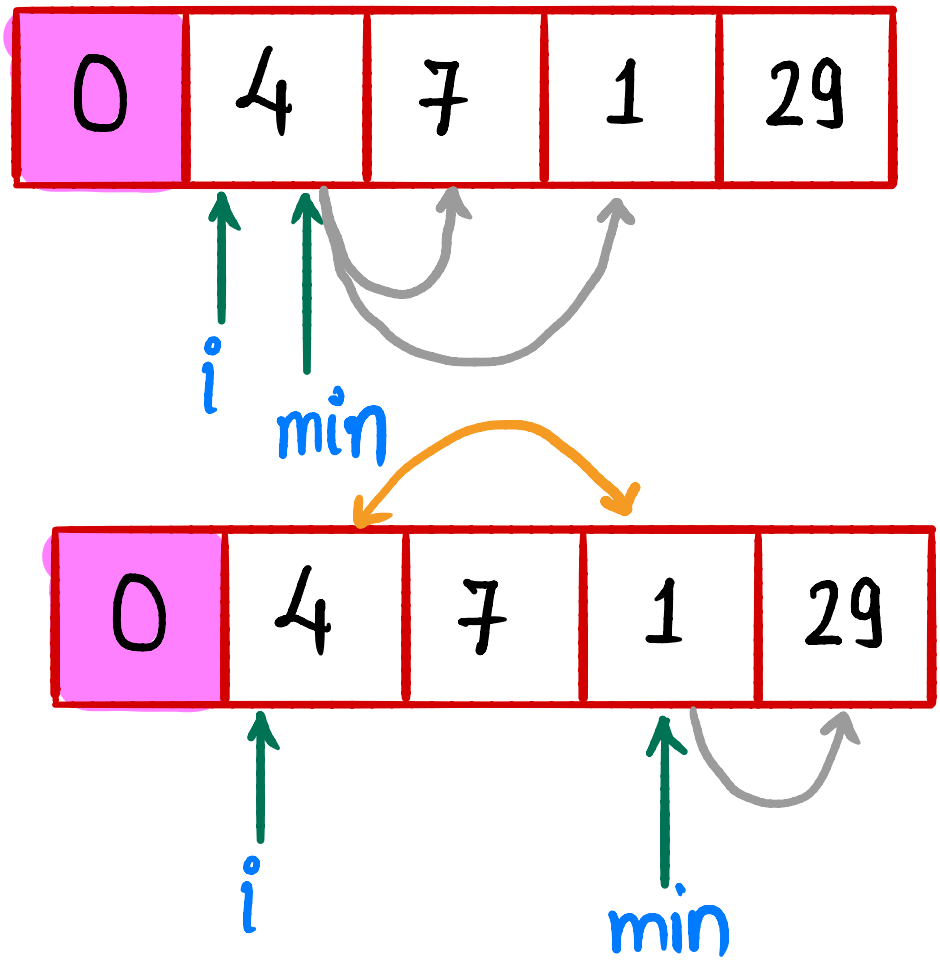
Inplace Selection Sorting

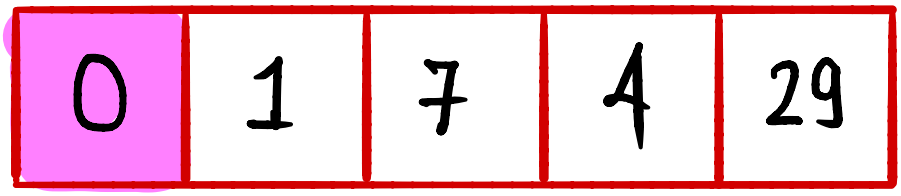




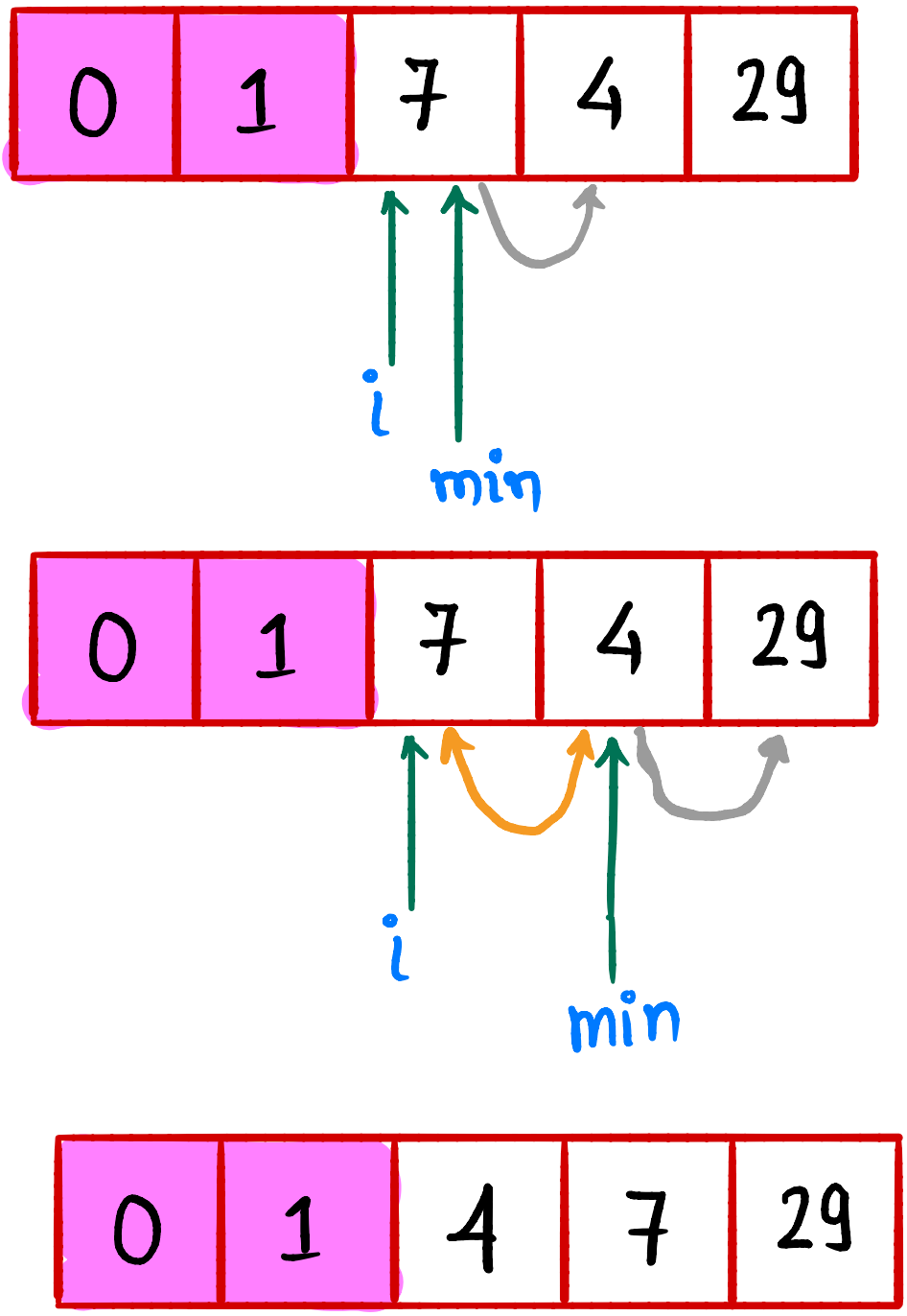


i = 1

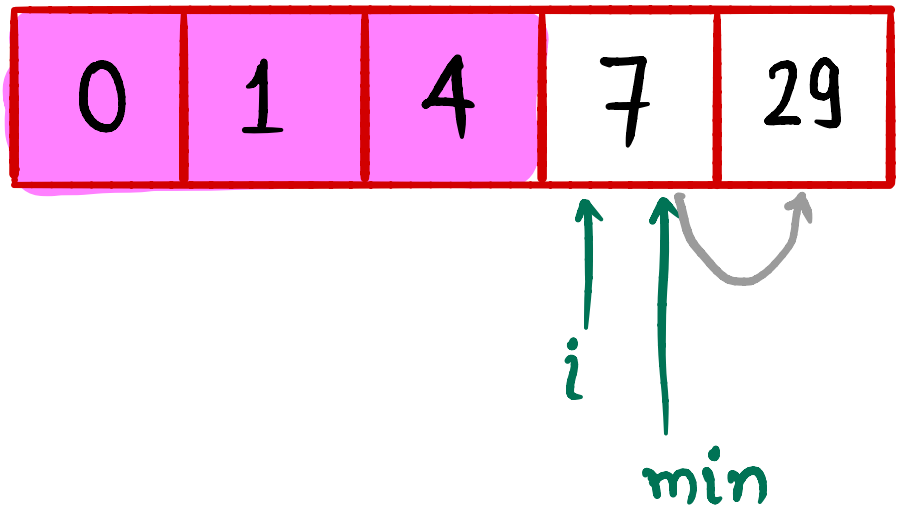




$i = 2$



i=3

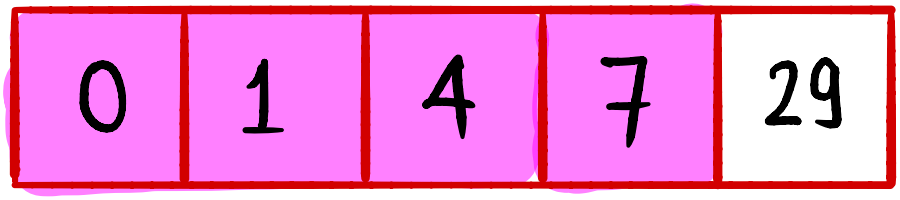


No Swaping Required

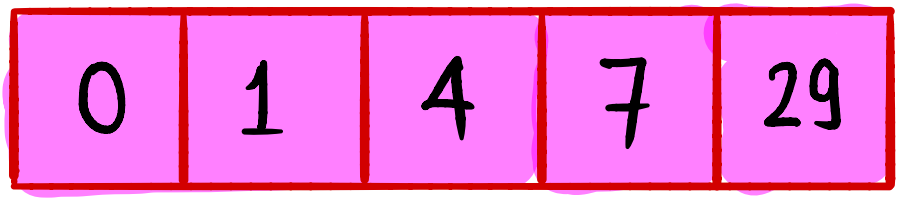
Observation:

At i^{th} iteration, $A[0, i-1]$ is sorted.

$i = 4$



No need of $i = 4$ as we are left with one one element and no element for comparisons.



(Sorted Array)

Code	Cost (Approx.)
`void SelectionSort(int A[], int n)`	
`{`	
`int min_idx;`	C_1
`for(int i=0; i<n-1; i++){`	nC_2
`min_idx = i;`	$(n-1)C_3$
main loop: `for(int j=i+1; j<n; j++){`	$\frac{n(n+1)}{2}C_4$
`if(A[min_idx] > A[j])`	$\frac{n(n-1)}{2}C_5$
`{`	
`min_idx = j;`	$\left(\sum_{k=1}^{(n-1)} t_k\right)C_6$
`}`	
`}`	
`if(min_idx != i)`	$(n-1)C_7$
`swap(&A[min_idx], &A[i]);`	PC_8 $(0 \le P \le n-1)$
`}`	
`}`	

min_idx is executed $\sum_{k=1}^{n-1} t_k$ times.

t_k : is no. of times the condition $A[min\_idx] > A[j]$ evaluates to true. in k^{th} pass. of main loop

$$T(n) = C_1 + nC_2 + (n-1)C_3 + \left(\frac{n^2+n}{2}\right)C_4$$

$$+ \left(\frac{n^2-n}{2}\right)C_5 + \left(\sum_{k=1}^{n-1} t_k\right)C_6 + (n-1)C_7 + PC_8$$

Best Case Inputs (Best Case Analysis)

$t_k = 0 \quad \forall k = \{1, \ldots, n-1\}$

i.e $P = 0 \longrightarrow$ No swapping required

Hence,

$$T(n) = C_1 + nC_2 + (n-1)C_3 + \left(\frac{n^2+n}{2}\right)C_4 + \left(\frac{n^2-n}{2}\right)C_5 + (n-1)C_7$$

$$= (C_1 - C_3 - C_7) + n(C_2 + C_3 + C_4' - C_5' + C_7) + (C_4' + C_5')n^2$$

$\left(C_4' = \frac{C_4}{2}, \quad C_5' = \frac{C_5}{2}\right)$

$T(n) = \Theta(n^2)$ (Best Case Analysis.)

(using polynomial lemma)

Worst Case Analysis

Let, $t_1 = n-1$

$t_2 = n-2$ .... $t_{n-1} = 1$

$P = (n-1)$ (Just taking worst values independently)

$$T(n) = C_1 + nC_2 + (n-1)C_3 + \left(\frac{n^2+n}{2}\right)C_4 + \left(\frac{n^2-n}{2}\right)C_5 + \left(\sum_{k=1}^{n-1} k\right)C_6 + (n-1)C_7 + (n-1)C_8$$

$$T(n) = (C_1 - C_3 - C_7 - C_8) + n(C_2 + C_3 + C_4' - C_5' + C_7 + C_8) + n^2(C_4' + C_5') + \left(\frac{n^2-n}{2}\right)C_6$$

$$= (C_1 - C_3 - C_7 - C_8) + n(C_2 + C_3 + C_4' - C_5' + C_7 + C_8, - C_6') + n^2(C_4' + C_5' + C_6')$$

$T(n) = \Theta(n^2)$ (using polynomial lemma)

Qns :-

Give a problem instance for (worst-case scenario and best-case) for the Selection sort.

Proof of correctness :- (Using loop invariant property covered in class)

Insertion Sort

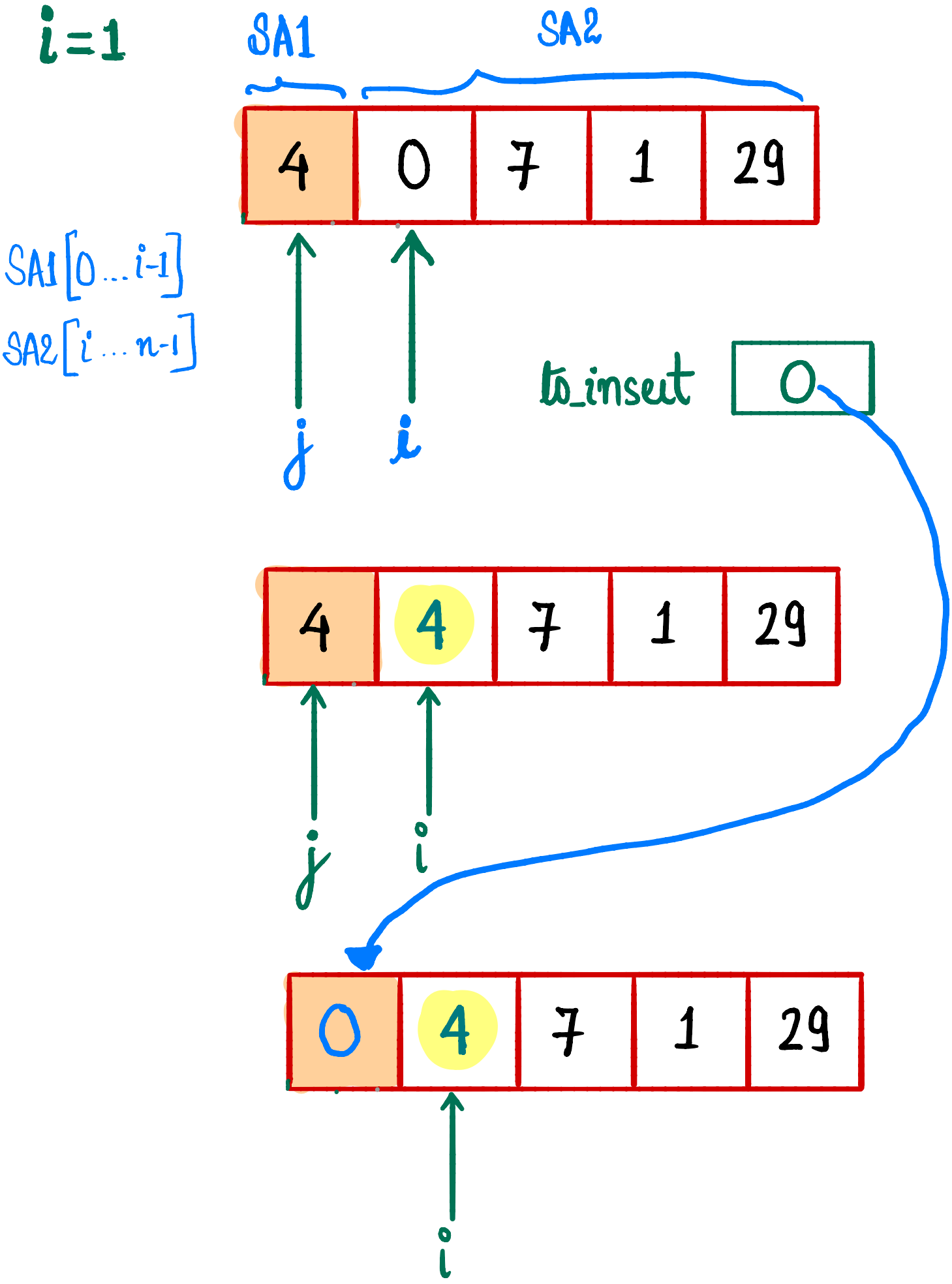
LIST 1	LIST 2
⟨4, 0, 7, 1, 29⟩	⟨ ⟩
⟨0, 7, 1, 29⟩	⟨4⟩
⟨7, 1, 29⟩	⟨0, 4⟩
⟨1, 29⟩	⟨0, 4, 7⟩
⟨29⟩	⟨0, 1, 4, 7⟩
⟨⟩	⟨0, 1, 4, 7, 29⟩

Delete → Insert at Right Place (So that List2 is sorted)

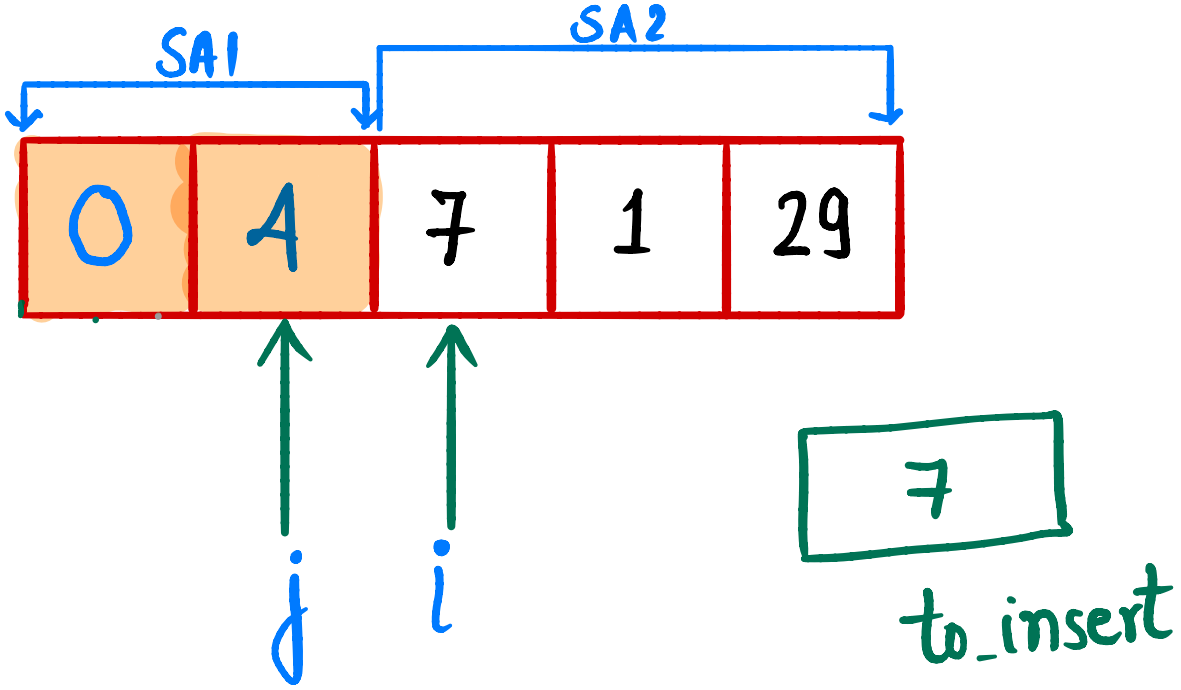
From List 1

in list 2

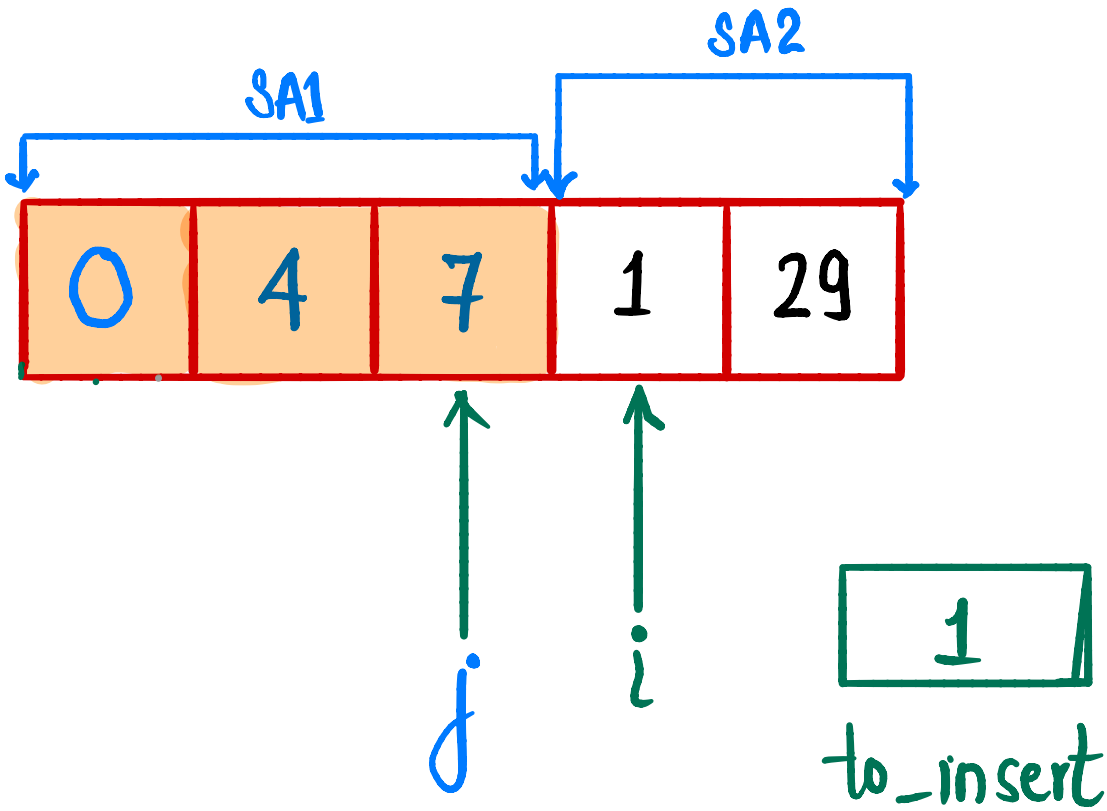
Next :- Inplace Insertion Sort →


i = 1
SA1
SA2
4
0
7
1
29
SA1[0...i-1]
SA2[i...n-1]
j
i
to_insert
0
4
4
7
1
29
j
i
0
4
7
1
29
i

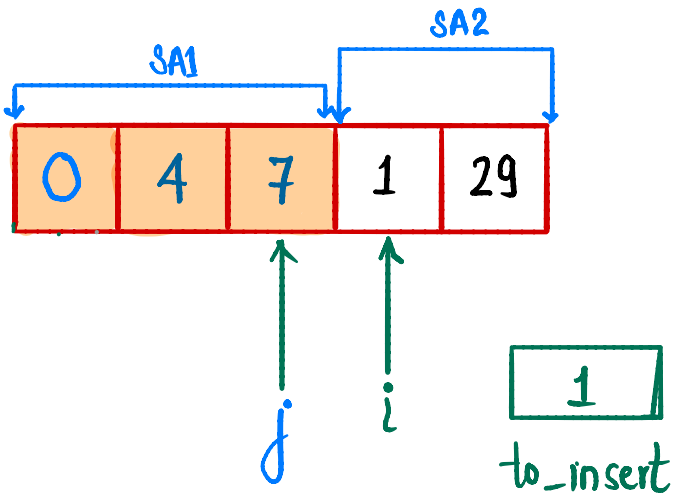
i=2

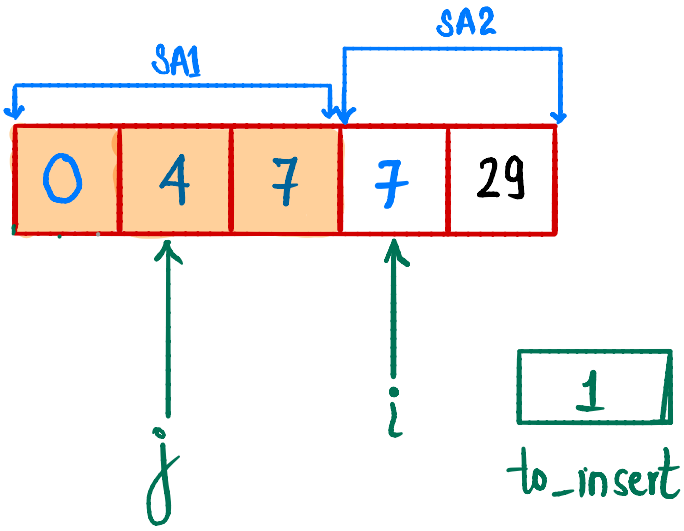


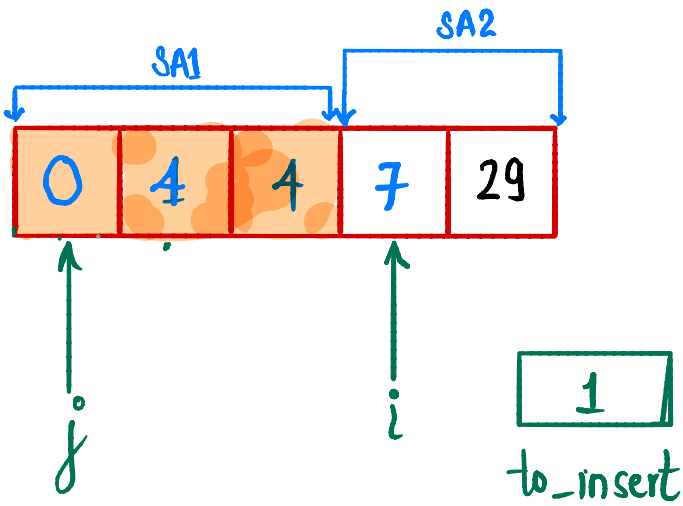
i=3

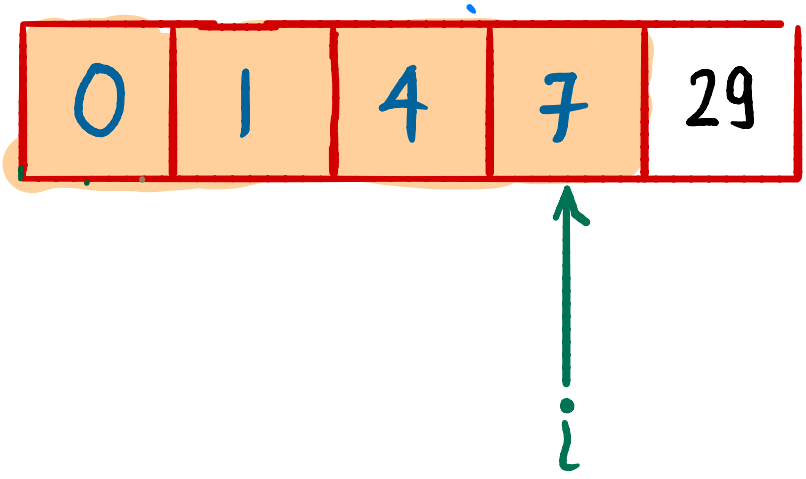


finding the right Position in SA1.

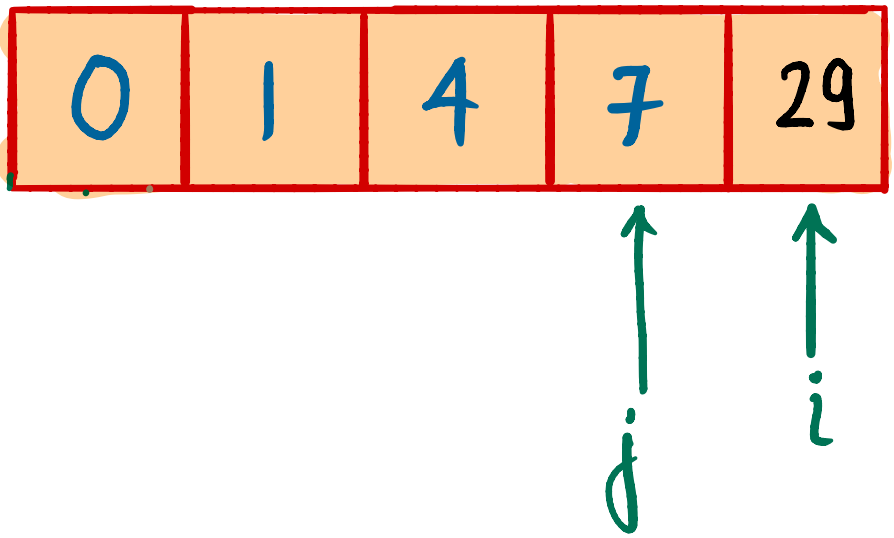








$i = 4$



```
void InsertionSort(int A[], int n)
{
    int to_insert, j;                                   C1
    for(int i=1; i<n; i++){                             n C2
        to_insert = A[i];                               (n-1) C3
        j = i-1;                                        (n-1) C4
        while(j>=0 && A[j] > to_insert){                (n-1) C5
            A[j+1] = A[j];                              (C6) (Σ_{k=1}^{n-1} t_k)
            j = j-1;                                    (C7) (Σ_{k=1}^{n-1} t_k)
        }
    A[j+1] = to_insert;                                 (n-1) · C8
    }
}
```

Finding and creating right Position

Inserting

t_k: Total no. of times while condition evaluates to true in k^{th} pass of main loop.

$$T(n) = C_1 + nC_2 + (n-1)C_3 + (n-1)C_4 + (n-1)C_5 +$$

$$\left(\sum_{k=1}^{n-1} t_k\right) C_6 + C_7 \cdot \left(\sum_{k=1}^{n-1} t_k\right) + (n-1)C_8$$

Best Case $t_k = 0$ i.e., array is already sorted.

$$T(n) = C_1 + nC_2 + (n-1)C_3 + (n-1)C_4 + (n-1)C_5 + (n-1)C_8$$

$$T(n) = (C_1 - C_8 - C_3 - C_4 - C_5) + n(C_2 + C_3 + C_4 + C_5 + C_8)$$

Clearly, $C_2 + C_3 + C_4 + C_5 + C_8 > 0$

$$T(n) = C_1 + C_2 n \quad \text{(using polynomial lemma)}$$

$C_2 > 0$

Hence, $T(n) = \Theta(n)$

Worst Case

while Condition is evaluated true

$(n-1)$ times if $k=1$

$(n-2)$ times if $k=2$

$\vdots$ $\vdots$ $\vdots$

1 times. if $k=n-1$

So,

$$\sum_{k=1}^{n-1} t_k = 1+2+\cdots+(n-1)$$

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

$$T(n) = c_1 + nc_2 + (n-1)c_3 + (n-1)c_4 + (n-1)c_5 + \sum_{k=1}^{n-1} t_k(c_6+c_7) + (n-1)c_8$$

$$T(n) = c_1 + nc_2 + (n-1)(c_3+c_4+c_5) + \frac{n(n-1)}{2}(c_6+c_7) + (n-1)c_8$$

$$T(n) = An^2 + Bn + C$$

$$T(n) = \Theta(n^2) \quad \text{(using polynomial lemma)}$$

Proof of Correctness :- Homework.