Sorting Algorithms

→ Sorting means rearrangement in a particular order. (not necessary to be inc or dec)

Brute force

$[a_1 \quad a_2 \quad a_3 \, . \, . \, . \, . \, . \, . \, a_n]$ →

↳ if try to from all possible rearrangements & then select our desired one.

↳ permutations → $n!$

↳ $O(n!)$

1, 2, 3, 4

Selection Sort

→ sorting → asc

→ $\Omega(n^2)$

eg

sorted ⟷ unsorted →

1	3	6	8	11	13	17	20	22	33

given

i) left part sorted, remaining unsorted

2) biggest element of sorted side is lesser than Smallest element of unsorted side

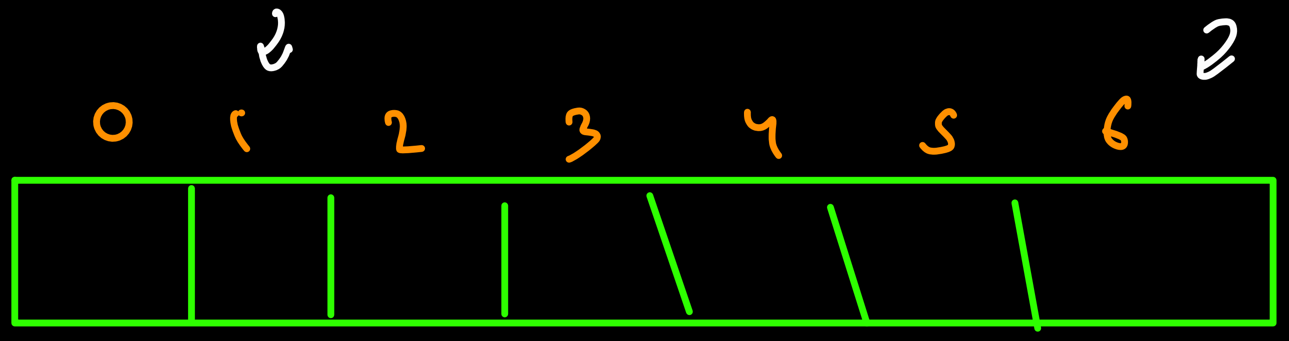
Q→ How to expand sorted region??

					1 j	2 i	
1	2	3	7	8	10	19	5

element = ~~7~~ ~~2~~ 19

first elemt to the left which is less than 7.

i) left part sorted , remaining unsorted



$$\left(1 + 2 + 3 + 4 \dots\dots n-1\right)$$

$$\frac{n \times (n-1)}{2} \rightarrow \underline{\underline{O(n^2)}}$$

1 2 3 4 5

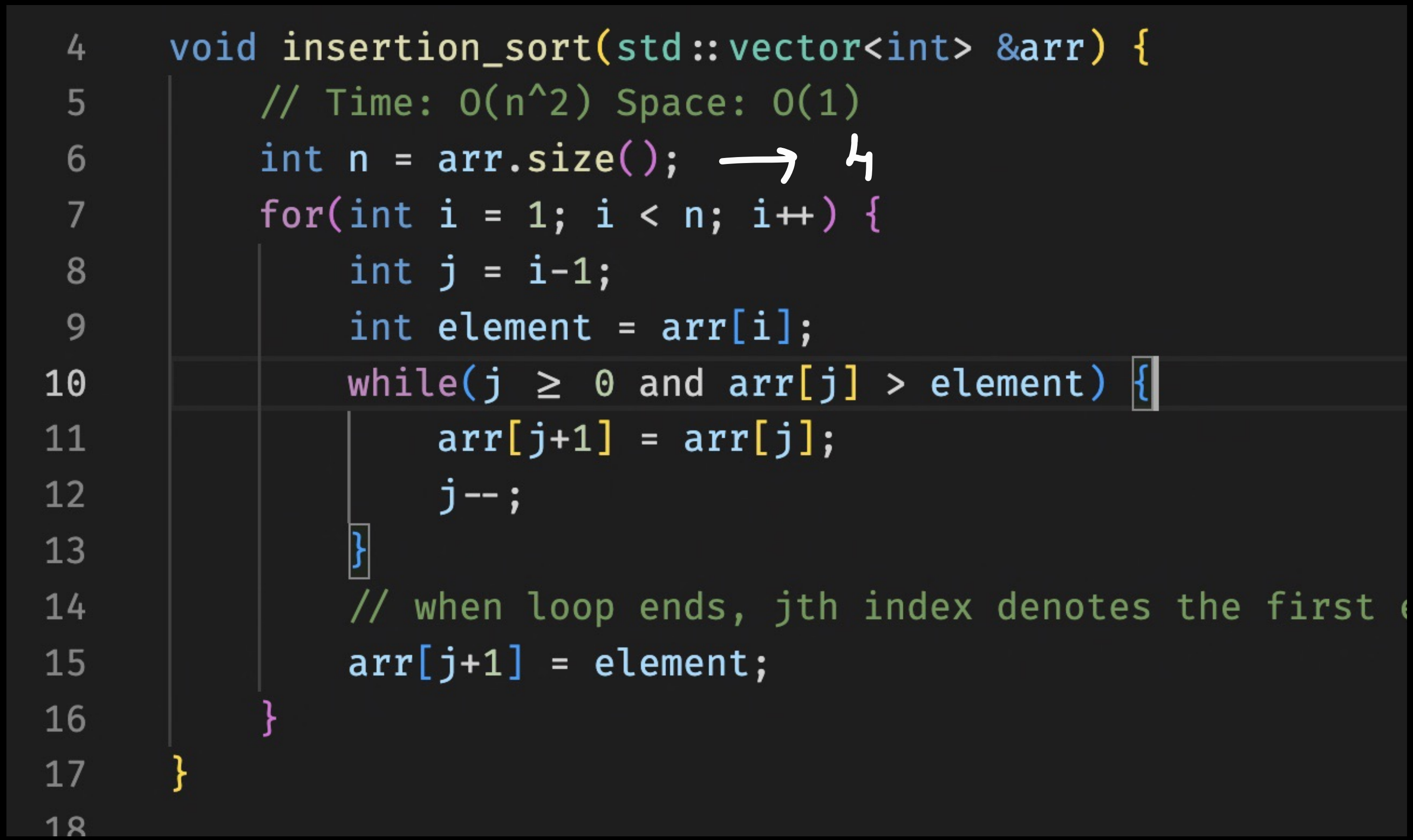
$$(1) + (1) + (1) + (1) \longrightarrow \underline{\underline{\Omega(n)}}$$

$$\underbrace{1, \cancel{3}2, \cancel{4}3, \cancel{5}9, \cancel{6}5, \cancel{7}6, \cancel{1}7}$$

→ almost sorted array

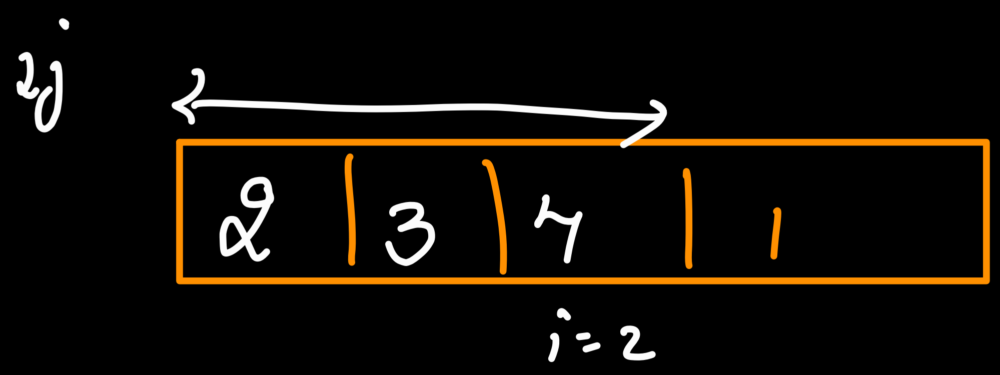
$$\underline{\underline{n}} \quad \underline{\underline{+ 1}}$$

→ $\Omega(n)$



```cpp
void insertion_sort(std::vector<int> &arr) {
    // Time: O(n^2) Space: O(1)
    int n = arr.size();
    for(int i = 1; i < n; i++) {
        int j = i-1;
        int element = arr[i];
        while(j ≥ 0 and arr[j] > element) {
            arr[j+1] = arr[j];
            j--;
        }
        // when loop ends, jth index denotes the first e
        arr[j+1] = element;
    }
}
```

→ 4



j = 1

element = 2

Bubble Sort

→ In one iteration the biggest element moves to the last.

2	3	4	5	11

by doing 2 adjacent comparisons

$1, 2, 3, 4, 5 \rightarrow \underline{\underline{\Omega(n)}}$

$3, 2, 1, 5, 4$

$$\left.\begin{array}{l} \underline{\underline{n-1}} \\ + n-2 \\ + n-3 \\ \quad\vdots \\ \quad 1 \end{array}\right\} \underline{\underline{O(n^2)}}$$