Q → sort a running integer stream ∀ integer intake.

I/P → 2 10 8 7 15 1 6

1 2 7 8 10 15

0	1	2	3	4	
1	2	3	4	5	....

6 ?

TC to insert → O(N)

Total # elements = N

then $TC = O(N^2)$

min # shifts = 0

max # shifts = # elements = O(N)

I/P → 9 8 3 1

1 3 8 9

shifts = $\overbrace{0 + 1 + 2 + 3}$ = 6

max total # shifts = 0 + 1 + 2 + ... N-1

$= \frac{(N-1) * N}{2}$

Insertion Sort

$TC = O(N^2)$

$SC = O(1)$

```
n = 0    // # elements
for (∀ input x) {
    for (i = n ; i > 0 ; i--) {
        if (A[i-1] > x)
            A[i] = A[i-1]
        else
            break
    }
    A[i] = x
    n += 1
}
```

[9 8 6 7 12 7]

n = ~~0 1 2 3 4 5~~ 6

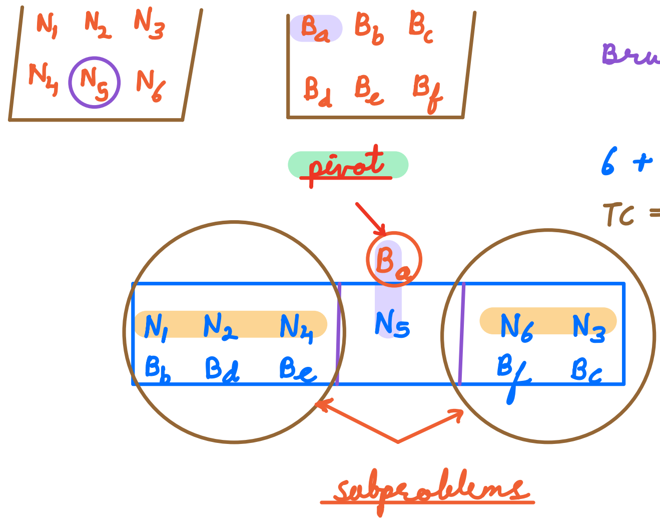
0	1	2	3	4	5
6	7	7	8	9	12

i = ~~5 4 3~~ 2

Nuts & Bolts

Given N nuts of different sizes & N bolts of different sizes.
There is a 1:1 mapping b/w nuts & bolts.
Match nuts & bolts with a constraint that comparing a nut with itself & a bolt with itself is not allowed.

Compare a nut & a bolt
- → exactly fits
- → nut is small
- → nut is big



Bruteforce → compare all nuts with all bolts.

$6+5+4+3+2+0 = \underline{20}$

$TC = \underline{O(N^2)}$ ✓

Partitioning

Q → Given an integer array, rearrange the elements in it s.t. ∀i if $A[i] < X$ then it is on left side else on right side of array. ($A[i] >= X$)

$A = [9 \quad 8 \quad 1 \quad 6 \quad 5 \quad 8]$ $\quad \underline{X = 6}$ ✓ $\quad SC = \underline{O(1)}$

pivot

$[1 \quad 5 \quad 9 \quad 8 \quad 6 \quad 8]$

$[1 \quad 5 \quad 8 \quad 8 \quad 9 \quad 6]$

$[5 \quad 1 \quad 9 \quad 8 \quad 6 \quad 8]$

Sol 1 → sorting

$TC = \underline{O(N \log(N))}$

$A = [1 \quad 2 \quad 3 \quad 4 \quad 5]$ $\quad X = \underline{4}$

$\{1, 2, 3\} \quad \{4, 5\}$

$j \rightarrow$

$$\begin{array}{cccccccc} & 0 & 1 & 2 & 3 & 4 & 5 & \\ A = [& \cancel{9}\,1 & \cancel{8}\,5 & \cancel{1}\,9 & 6 & \cancel{5}\,8 & 8 &] \end{array}$$

$\uparrow i$

$\underline{X = 6}$ ✓

$j \rightarrow$

$$\begin{array}{ccccccccc} & 0 & 1 & 2 & 3 & 4 & 5 & 6 & \\ A = [& 3 & \cancel{5}\,1 & \cancel{1}\,\cancel{5}\,0 & \cancel{5}\,2 & \cancel{0}\,5 & 7 & \cancel{2}\,5 &] \end{array}$$

$X = \underline{4}$ pivot

$\uparrow$

$3 \quad 1 \quad 0 \quad 2 \quad i\ 5 \quad 7 \quad 5$

$TC = \underline{O(N)}$ ✓

$SC = \underline{O(1)}$ ✓

Partitioning

Quick Sort → Divide & Conquer

10 : 40 PM

	0	1	2	3	4	5	6
A =	7	3	2	5	1	6	4

[3 2 1] 4 [7 5 6]

TC = O(N)

	0	1	2	3	4	5	6
A =	7	3	2	5	1	6	4

[3 2 5 1 6 4] 7

TC = O(N)

N

N/2 N/2

N/4 N/4 N/4 N/4

⋮

$\frac{N}{2^K} = 1$

$\Rightarrow \boxed{K = \log_2(N)}$

TC = O(N log (N))

N

N-1

N-2

⋮

N - K = 1

⇒ K = N-1

TC = O(N²)

O(N log(N)) <= TC of Quick Sort <= O(N²)

(Best) (Worst)

	0	1	2	3	4	5	6
A =	7	3	2	5	1	6	4

[3 2 1] 4 [7 5 6]

1 [3 2] [5] 6 [7]

2 [3]

```
void quickSort(A, st, end) {        // st → 0, end → N-1
    if (st >= end)
        return
    pi = partition(A, st, end)
    quickSort(A, st, pi-1)
    quickSort(A, pi+1, end)
}
```

SC = ~~O(1)~~ ✗ = O(# levels)

O(log(N)) O(N) ✓

(recursion)

```
int partition(A, st, end) {
    pe = A[end]  // pivot element
    i = st
    for (j → st to (end-1)) {
        if (A[j] < pe) {
            swap A[i] A[j])
            i += 1
        }
    }
    swap(A[i], A[end])
```

<pe

pe

i

(A, 0, 6)

(A, 0, 2) (A, 4, 6)

(A, 0, -1) (A, 1, 2)

(A, 1, 1) (A, 3, 2)

| return i

}

TC = O(N) SC = O(1)

j → st 0, 1, end 2, pi 3, 4, 5) 6

A = [~~7~~ 3, ~~3~~ ~~7~~ 2, ~~2~~ 1, ~~5~~ 4, ~~1~~ 7, 6, ~~4~~ 5]

pe, i

j, pi 0, 1, 2

~~3~~ 1, 2, ~~1~~ 3

pe, i

j → pi 1, 2

2, 3

pe, i

1, 2, 3 --- 9, 10, 11, ... 49, 50, 51, 52, --- 89, 90, 91, ... 99, 100

(in any order)

worst pivot = {1, 100}

best pivot = {50, 51}

N = 100

< 90% of N < 90% of N

(< 90 elements) (< 90 elements)

[11 90]

\# elements = 80

probability of selecting a pivot that make sure subproblems are < 90% of original problem = $\frac{80}{100}$ = 0.8 or 80%

20: (1 – 19) (21 – 100); 19, 80

5 ×: 4, 95

10 ×: 9, 90

90: 89, 10

N

$\frac{N}{10}$ $\frac{9N}{10}$

$\frac{N}{100}$ $\frac{9N}{100}$ $\frac{9N}{100}$ $\left(\frac{9}{10}\right)^2 N$

$\left(\frac{9}{10}\right)^K N = 1$

$\Rightarrow N = \left(\frac{10}{9}\right)^K \Rightarrow \boxed{K = \log_{(10/9)}(N)}$

$N = 10^5$

$\log_{(10/9)}(10^5) = 109 \approx 10^2$

$N * \log_{(10/9)} N$

$10^5 * 10^2 = 10^7$

80% of times we do better than

$SC = O(\log_{10/9}(N))$

$TC = O(N \log_{(10/9)}(N))$