Let's begin at 9:02PM sharp

L29
Sorting : Introduction

If interested, check out the System Design course.
Early Bird discount is ON.

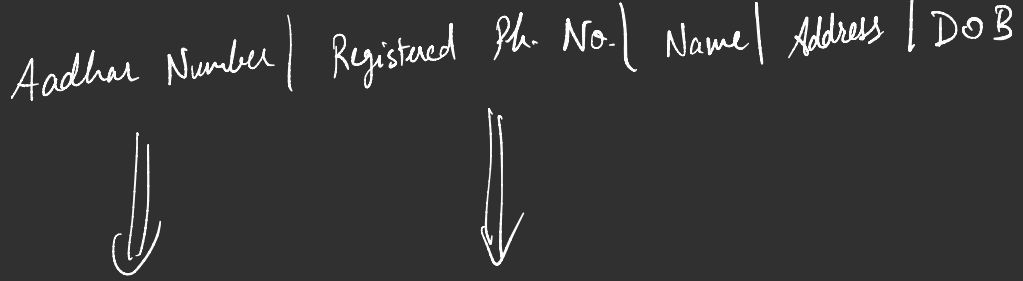
Join Discord - https://bit.ly/ly-discord

RECAP

What is Sorting?

Simply a way of arranging data
in a particular way.

Example



List of cuboids

l, b, h

Basis + order

⇓

Comparator

⇒ inc. order of length

⇒ inc. order of volume

⇒ dec. order of breadth

⇒ inc. order of surface area, [If surface area is equal then compare the perimeter]

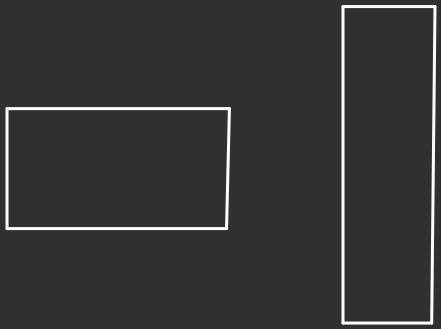
e_1 , e_2

e_1 will come before e_2 in the sorted order if

$e_1.len * e_1.width * e_1.height < e_2.len * e_2.width * e_2.height$

Comparator has to be valid

list of rectangles $\Longrightarrow$ (r_1, r_2) $\Rightarrow$ r_1 can fit into r_2.



Given a list of integers, can you think of different possible comparators?

1) inc order of value
2) dec order of value
3) inc order of sum of digits
4) Sort in lexicographical order.

Sorting Algorithms

A lot of them are there in the world.
Few of them are the famous ones.

Inserting an element in an already sorted list

Eg. -5, 1, 2, 3, 5, 6, 7, 10, 11, 13, 15

num = 7, 15, -5

Let's use this to sort a given array → Insertion Sort

Eg. [3 | 1, 5, 2, 1, 10, 5]

[1, 3 | 5, 2, 1, 10, 5]

[1, 3, 5 | 2, 1, 10, 5]

[1, 2, 3, 5 | 1, 10, 5]

[1, 1, 2, 3, 5 | 10, 5]

[1, 1, 2, 3, 5, 10 | 5]

[1, 1, 2, 3, 5, 5, 10]

Sorted

num = 5

Time Complexity analysis of Insertion Sort

[10, 5, 5, 3, 2, 1, 1]

Time $\Rightarrow O(N^2)$

Extra Space $\Rightarrow O(1)$

Bubble Sort $\Rightarrow$ At most $N-1$ steps

Eg: [9, 5, 1, 6, 2, 3]

Time $\Rightarrow O(N^2)$

Space $\Rightarrow O(1)$

Step 1: [5, 1, 6, 2, 3, 9]

Step 2: [1, 5, 2, 3, 6, 9]

Step 3: [1, 2, 3, 5, 6, 9]

Step 4: [1, 2, 3, 5, 6, 9]

Step 5: [1, 2, 3, 5, 6, 9]

Selection Sort

Time ⇒ $O(N^2)$

Space ⇒ $O(1)$

Eg: [3, 1, 5, 2, 1, 10]

Step 1 : i = 0 → [1, 3, 5, 2, 1, 10]

Step 2 : i = 1 → [1, 1, 5, 2, 3, 10]

Step 3 : i = 2 → [1, 1, 2, 5, 3, 10]

Step 4 : i = 3 → [1, 1, 2, 3, 5, 10]

Step 5 : i = 4 → [1, 1, 2, 3, 5, 10]

(Sorted)

1. What is the best auxiliary space complexity a sorting algorithm can have?

A. O (n log n)
B. O (n)
C. O (1)
D. O (n^2)

1. What is the best auxiliary space complexity a sorting algorithm can have?

A. O (n log n)
B. O (n)
C. O (1)
D. O (n^2)

Bubble Sort = Insertion Sort

Solution : the discussed selection sort is an **in-place** sorting algorithm, and one can't go better than O(1).

2. Sort [🍎, 🍊, 🍊, 🍎] in ascending order.

A. [🍎, 🍊, 🍊, 🍎]
B. [🍊, 🍊, 🍎, 🍎]
C. [🍊, 🍎, 🍊, 🍎]
D. Can't sort.

2. Sort [🍎, 🍊, 🍊, 🍎] in ascending order.

A. [🍎, 🍊, 🍊, 🍎]
B. [🍊, 🍊, 🍎, 🍎]
C. [🍊, 🍎, 🍊, 🍎]
D. Can't sort. ✓

Solution : one can't compare apple and oranges. More formally, the comparator function is not defined.

3. Sort [🍎, 🍊, 🍊, 🍎] in ascending order, if 🍊 < 🍎

A. [🍎, 🍊, 🍊, 🍎]
B. [🍊, 🍊, 🍎, 🍎]
C. [🍎, 🍎, 🍊, 🍊]
D. [🍊, 🍎, 🍊, 🍎]

3. Sort [🍎, 🍊, 🍊, 🍎] in ascending order, if 🍊 < 🍎

A. [🍎, 🍊, 🍊, 🍎]

B. [🍊, 🍊, 🍎, 🍎] ✓

C. [🍎, 🍎, 🍊, 🍊]

D. [🍊, 🍎, 🍊, 🍎]

Solution : The comparator function is now defined.

4. Which of the following best describes a stable sorting algorithm (single answer)?

A. Any algorithm that doesn't crash for all possible inputs.
B. An algorithm that correctly sorts the input array correctly for all possible inputs.
C. An algorithm that sorts the input and maintains relative order of those elements that are equal to each other at the end for all possible inputs.
D. An algorithm that never allows two elements to be equal to each other.

4. Which of the following best describes a stable sorting algorithm (single answer)?

A. Any algorithm that doesn't crash for all possible inputs.
B. An algorithm that correctly sorts the input array correctly for all possible inputs.
C. An algorithm that sorts the input and maintains relative order of those elements that are equal to each other at the end for all possible inputs.
D. An algorithm that never allows two elements to be equal to each other.

Solution : Definition

5. Is bubble sort a stable sorting algorithm?

A. Yes
B. No

5. Is bubble sort a stable sorting algorithm?

A. Yes
B. No

Solution : The order is not changed when swapping the values since only adjacent elements are swapped when the first element is strictly greater than the second.

6. Is the following sorting algorithm stable?

A. Yes
B. No
C. It is not a correct sorting algorithm

```
for i do in range 0...n-1
    for j do in range 1...n-1-i
        if then A_j ≤ A_{j-1}
            swap(A_{j-1}, A_j)
        end if
    end for
end for
```

6. Is the following sorting algorithm stable?

```
for i do in range 0...n − 1
    for j do in range 1...n − 1 − i
        if then A_j ≤ A_{j−1}
            swap(A_{j−1}, A_j)
        end if
    end for
end for
```

A. Yes
B. No ✓
C. It is not a correct sorting algorithm

Solution : Consider (1, 1, 3), after the first iteration it becomes (1, 3, 1) first element moves to the end and relative position of the 1's change.

Thank You!

Reminder: Going to the gym & observing the trainer work out can help you know the right technique, but you'll muscle up only if you lift some weights yourself.

So, PRACTICE, PRACTICE, PRACTICE!