Sunday, 28 February 2021 12:20 AM

Theory

1. $O(n^2)$ sorting algorithms
2. Merge sort & Iterative Merge sort
3. Quick sort and Counting sort

Practical

1. Quick sorting creation or Problem solving
2. Binary & linear search & binary search
3. Binary search on answer / but on
3. Function sorting

60% theory 40% function sort

Q. what is sorting ??

Arranging things in an order.

accending descending

based on properties of elements.

Eg ① Array → accending / descending

② Permutation of strings

[[a, b], [3, 5], [1, 6], ...]

How to sort ??

→ 3 ways to [sort]

see of a[2] < a[3] [true]

b, c, d

b, c, d

Function

Q. Given an array, what would be the worst (Brute force) way to [sort] ??

Ans → accending

Ans → [2 3 1 4]

Ans → [1 2 3 4]

Sorted version of an array is merely but permutation of the original array.

→ with all permutations and pick the sorted one.

TC → $(n!) \times n$)

$O(n^n)$ for better

Insertion sort

left right

① 6 11 9 8

L R

II 6 11 9 8

[sorted] [unsorted]

6 < 9

11 > 9

III 6 11 9 8

6 < 8 Insertion sort

9 > 8

6 8 9 11

L R

In each iteration, you are shifting one element from left bound and placing it at the right position.

[7 9 8 2]

sorted unsorted

I [7 9 ; 8 2]

sorted unsorted

II [7 8 9 ; 2]

sorted unsorted

III [2 7 8 9]

sorted

In-place sorting algorithm

→ algo which takes constant space complexity

Applications

→ In-place

→ Best case → $O(n)$

→ useful if you want to sort an almost sorted array.

→ 9 7 8 8 7

Bubble Sort

Swapping

① I [7 4 5 2]

[4 7 5 2] 3 swaps

[4 5 7 2]

[4 5 2 7]

II [4 5 2 7] 2 swaps

[4 2 5 7]

III [4 2 5 7] 1 swap

[2 4 5 7]

X

Time Complexity →

(n - 1) iterations

for each iteration (n-1) swaps

$(n-1) + (n-2) + (n-3) \ldots$

$= O(n^2)$

space complexity → $O(1)$

[did not take]

auxiliary space

in-place sorting algo

5 1 4 2 8

1 4 2 5 8

1 2 4 5 8 X swaps

1 2 4 5 8

1 2 4 5 8

Selection Sort

[60 22 12 21 10]

works by repeatedly finding the min element from the unsorted part and putting it at the beginning in unsorted part.

[60 22 12 22 10] 5

unsorted

① find min / min 10

[10 22 12 21 60]

sorted unsorted

② [10 12 22 21 60]

sorted unsorted

③ [10 12 21 22 60]

④ [10 12 21 22 ; 60]

4 iterations

Time Complexity:

① finding a min element from the array of n elements, requires (n-1) comparisons.

→ then n-1 etc, (n-2) comparisons

⇒ Total comparisons = (N-1) + (N-2) + ... = $\frac{N^2(N-1)}{2}$

⇒ $O(n^2)$

Applications

① Memory write is a heavy operation

Time Complexity

→ Space Complexity

→ In-place

Stability

A sorting algorithm is said to be stable if it maintains the relative ordering of the elements.

Name	Age
Joe	2.5
Nowman	2.8
Amit	2.1
Nowman	2.8

Sort by age

Name	Age
Amit	2.1
Joe	2.5
Nowman	2.8
Nowman	2.8

Sort by name

name	age
Amit	2.1
Joe	2.5
Nowman	2.8
Nowman	2.8

(a)

name	age
Amit	2.1
Nowman	2.8
Joe	2.5
Nowman	2.8

(b)

In array terms, same elements

3 5 2 1 5 10

Stable sort

1 2 3 5' 5 10

unstable sort

1 2 3 5' 5 10