Algorithm NTS

1) The name algorithm comes from the name of Persian author. Abu Ja'far Mohammed iben Musa al Khowarizmi (C. 825 A.D)

2) An algorithm is a finite set of instruction that if followed, accomplished a particular task. In addition all algorithm must satisfy the following criteria

a) Input :- Zero or more quantities are externally supplied

b) Output :- At least one quantity is produced.

c) Definiteness :- Each instruction is clear and unambiguous

d) Finiteness :- the algorithm terminates after a finite number of steps

e) Effectiveness :- Every instruction must be very basic so that it can be carried out in principle by a person using only pencil and paper.

⇒ Algorithms that are definite and effective are also called computational procedure. Ex:- OS
This procedure is designed to control the execution of jobs, in such a way that when no jobs are available it doesn't terminate but continues in a waiting state until a new job is entered.

★ Sorting and Searching

a: | 1 | 3 | 2 | 7 | 6 | 4 | 8 | 9 | — Unordered

a': | 1 | 2 | 3 | 4 | 6 | 7 | 8 | 9 | — Sorted increase/asc order

$a'_0 \leq a'_1 \leq a'_2 \leq \cdots \leq a'_9$

$a'_0 = [a'[0]$

a'': | 9 | 8 | 7 | 6 | 4 | 3 | 2 | 1 | — Sorted des order

$a''_0 \geq a''_1 \geq a''_2 \geq \cdots \geq a''_i$

★ Searching

	0	1	2	3	4	5	6	7
a =	1	3	2	7	8	9	4	6

S = 3.

② does S(=3) in array a?

→ Yes S=3 exists in a @ position 1.

★ 'Insertion sort' (Key, satellite data.)

Database

id / Columns	name	price	disc
1	n1	P1	d1
2	n2	P2	d2

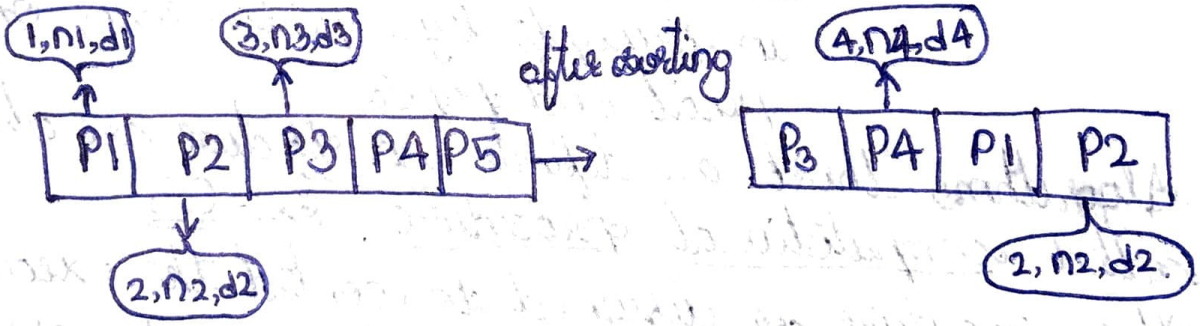
Rows →

P1 → or Price ⇒ Key

satellite data ⇒ id, name, disc

Sorting according to this

★ Satellite data moves along with the condition based on which the key is sorted



★ Insertion sort

① a: 6 5 3 1 8 7 2 4 (K=5)

1 2 3 4 5 6 7 8.

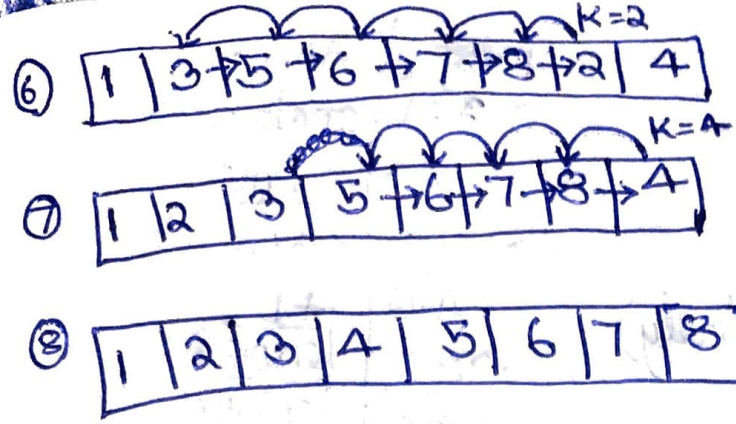
Unsorted array.
default asc-order.

② 5 6 3 1 8 7 2 4 (K=3, K=3)

③ 3 5 6 1 8 7 2 4 (K=1, K=1, K=1)

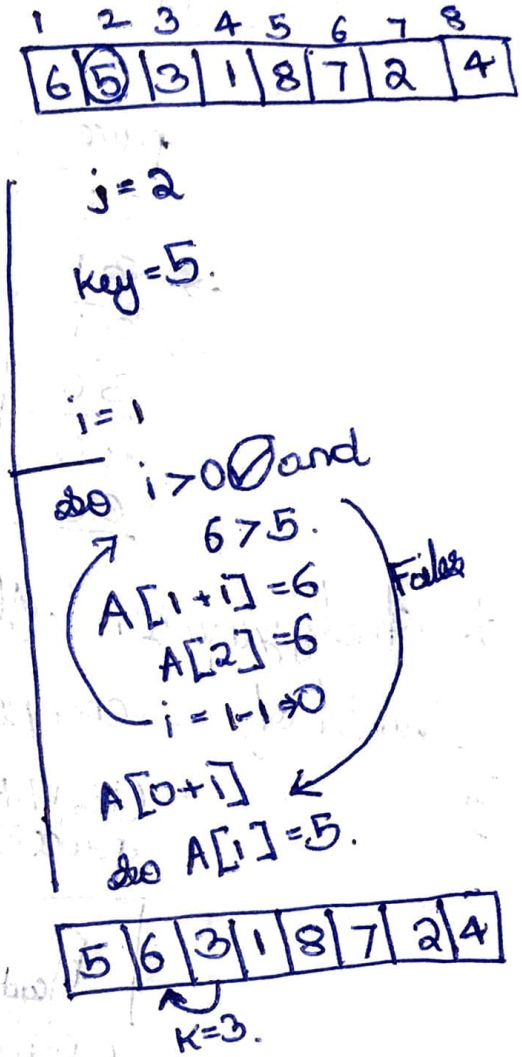
④ 1 3 5 6 8 7 2 4 (K=8)

⑤ 1 3 5 6 8 7 2 4 (K=7)



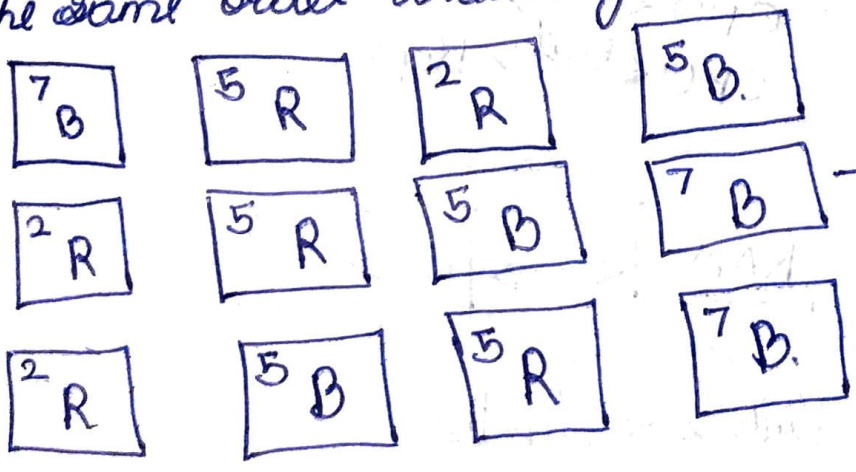
* Pseudo-code → false

1) for j=2 to A.length
2) key = A[j]
3) // Insert A[j] into sorted A[1...j-1]
4) i = j-1
5) while i>0 AND A[i]>key
6) A[i+1] = A[i]
7) i = i-1
8) A[i+1] = key



* Stability Sort.

Stability sort algorithms sort repeated element in the same order that they appear in the input.



Ex:-

price	name
23	apple
→ 36	lenova
31	acer
31	asus
40	HP
→ 60	lenova

① sort by name (alphabetically)

price	name
→ 31	acer
23	apple
→ 31	asus
40	HP
→ 36	lenova
→ 60	lenova

② sort by price stable

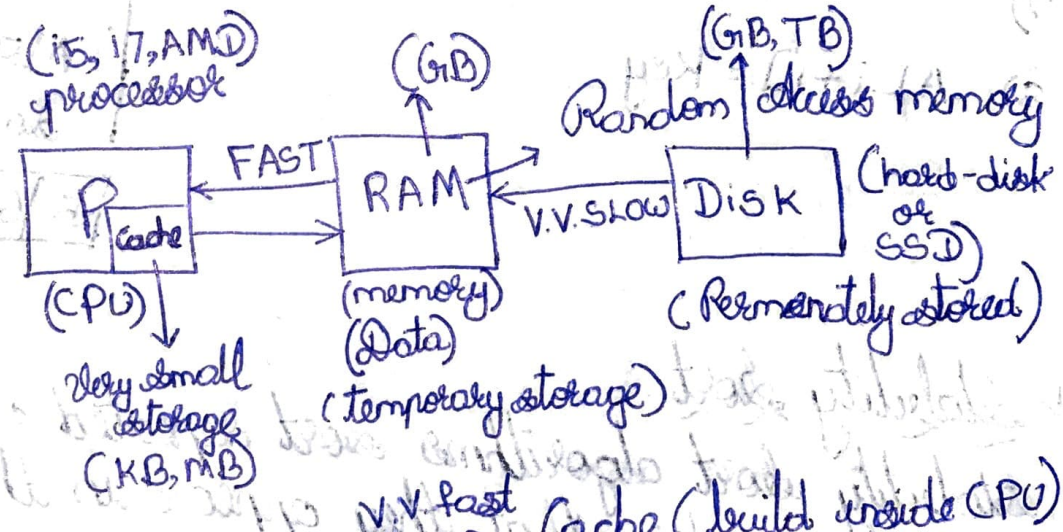
price	name
23	apple
→ 31	acer
→ 31	asus
36	lenova
40	HP
60	lenova

★ Analyzing an Algorithm → (Time, Space / Memory)

Model of a computer

SSD → Solid state drive

1)



CPU ⟷ (V.V fast) Cache (build inside CPU)

CPU ⟷ (Fast) Ram

2)

Multi core algorithm

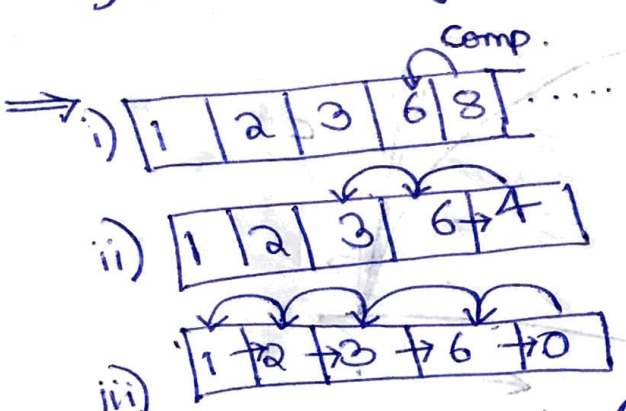
4-core

* Time and space complexity of Insertion sort.

Insertion sort(A).

1)	for j=2 to A=length	→ n	C_1	$C_1 * n$
2)	Key = A[j]	→ n-1	C_2	$C_2 * (n-1)$
3)	// Insert A[j] into A[1,...,j-1]	→ n-1	$C_3 = 0$	$C_3 * (n-1) \Rightarrow 0$
4)	i = j-1	→ n-1	C_4	$C_4 * (n-1)$
5)	while (i>0 AND A[i] > Key) {Compares}	→ n-1 ; $\frac{n(n-1)}{2}$	C_5	$C_5 * (n-1)$, $C_5 \left(\frac{n(n-1)}{2}\right)$
6)	A[i+1] = A[i] → Swap	$\frac{n(n-1)}{2}$	C_6	$C_6 \left(\frac{n(n-1)}{2}\right)$
7)	i = i-1 → Swap	$\frac{n(n-1)}{2}$	C_7	$C_7 \left(\frac{n(n-1)}{2}\right)$
8)	A[i+1] = Key	→ n-1	C_8	$C_8 * (n-1)$

$\mu s = 10^{-6}$ sec.



Comp = 1 Swap = 0 (Best)

Comp → 2 Swap → 1

Comp = 4 Swap = 4 (Worst case)

do min Comp = 1, min Swap = 0

max Comp = 4, max Swap = 4

= j-1, ⇒ j-1

* Comp
  - min = (n-1)
  - max = $\frac{(n-1)n}{2}$
* Swap
  - min = 0
  - max = $\frac{(n-1)n}{2}$

* Comparision

j	j=2	j=3	j=4	...	j=n
min	1	1	1		1
max	1	2	3		n-1

* Swap

	j=2	j=3	.....	j=n
min	0	0	0	0
max	1	2	.....	n-1

3) Scientific computational ML, graphics

GPU → Graphic Processing Unit. (NVIDIA)

1000's of small processors ↔ VRAM ↔ RAM ↔ Disk

(VideoRAM)

do total time

1) $C_1*n + C_2(n-1), C_3(0) + C_4(n-1) + C_5(\overset{n-1}{\quad}) + C_6,(0) + C_7(0) + C_8(n-1)$

$\Rightarrow nC_1 + (n-1)C_2 + C_4(n-1) + C_5(n-1) + C_8(n-1) \Rightarrow \underline{an+b}$ (Best)

2) Max.

$C_1*n + C_2(n-1) + 0 + C_4(n-1) + C_5\left(\frac{n(n-1)}{2}\right) + C_6\left(\frac{n(n-1)}{2}\right) + C_7\left(\frac{n(n-1)}{2}\right) + C_8(n-1)$

$\Rightarrow \underline{an^2+bn+c}$ (Worst)

do (i, j, key) $\Rightarrow$ 3 Variable

★ Insertion Sort :: Big-O notation

A.length = n = input array len.

Best case time complexity = $\phi(n) + b(1) \rightarrow O(n)$

Worst case time complexity = $\phi(n^2) + b(n) + c(1) \rightarrow O(n^2)$

Space Complexity = 3 Variables $\Rightarrow 3(1) \rightarrow O(1)$

do if $n\uparrow = ((\phi n) + b)\uparrow \Rightarrow O(n)$.

if $n\uparrow = \phi n^2 + bn + c$

$\Rightarrow O(n^2)$.

So $n^2\uparrow$ ← then $n\uparrow$. Doesn't makes much difference