Running time ✓
Execution time ✗

RAM model

basic operation
complex
i++

Program
- specific H/w : processor & memory
- O/S
- PL

(x) ms

Assumption
- Each basic operation takes one unit time

multiplication : repeated addition

0

a x b
a * b

1) i = i + 1 two unit time

1) for i = 1 to n
2) C[i] ← a[i] * b[i]

i = 1
i++
i < n

$T(n) = C \cdot n + k$

linear running time

$n \to \infty$

Standard methods for designing algos.

1) D & C
2) Greedy
3) DP
4) Backtracking
5) Branch & bound

- Design
- Analysis

Types of algorithms:

- Deterministic algo.
- Non-deterministic algo

Deterministic algo.

- wrt time & place

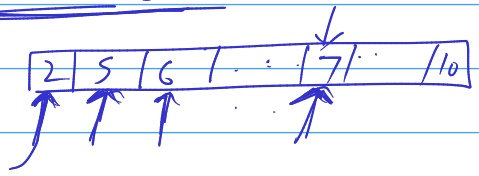
→ on different runs

A

I/P — O/P

X — Y

Linear search



Key = 7

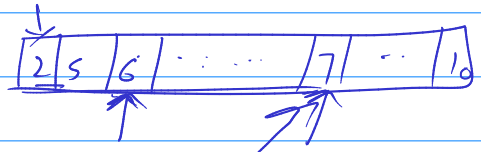
Non-deterministic algo.

Algo. A

different behaviour on different runs

Searching

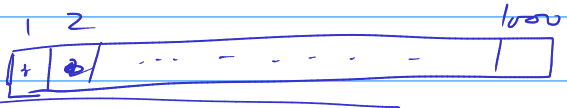
Randomization



Key = 7

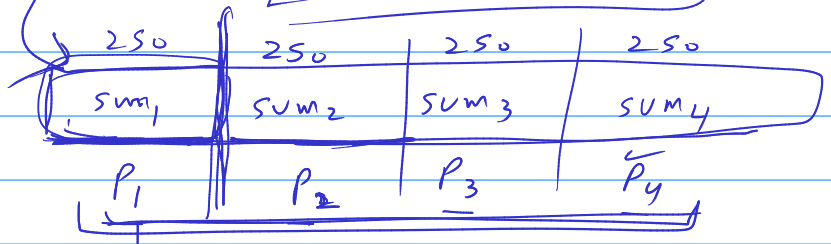
4 processors

Concurrent algos.



Seq. algo.

Seq. algo: sum = 0; sum = sum + A(i)



sum = sum1 + sum2 + sum3 + sum4

1st run: P2, P3, P4, P1

2nd run: P1, P2, P4, P3

sum = sum1 + sum2

Quality of algorithm: is mainly decided by two parameters

time and space

time complexity → time taken by an algo

space complexity → space

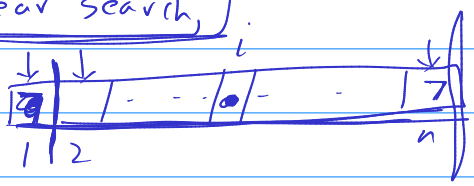
Analysis of algorithms:

- Best case → I/P is provided in such a way that we have min. time to process.
- Worst case → max. time
- Average case → all inputs are equally likely to occur

Example: Searching problem

A[ ], key

Linear search



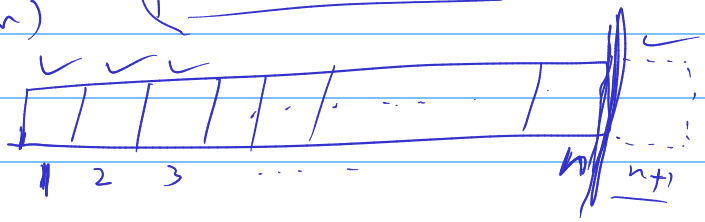
key = 7

Best case : constant $O(1)$

worst case : $O(n)$

average case : $O(n)$

key with A[i]



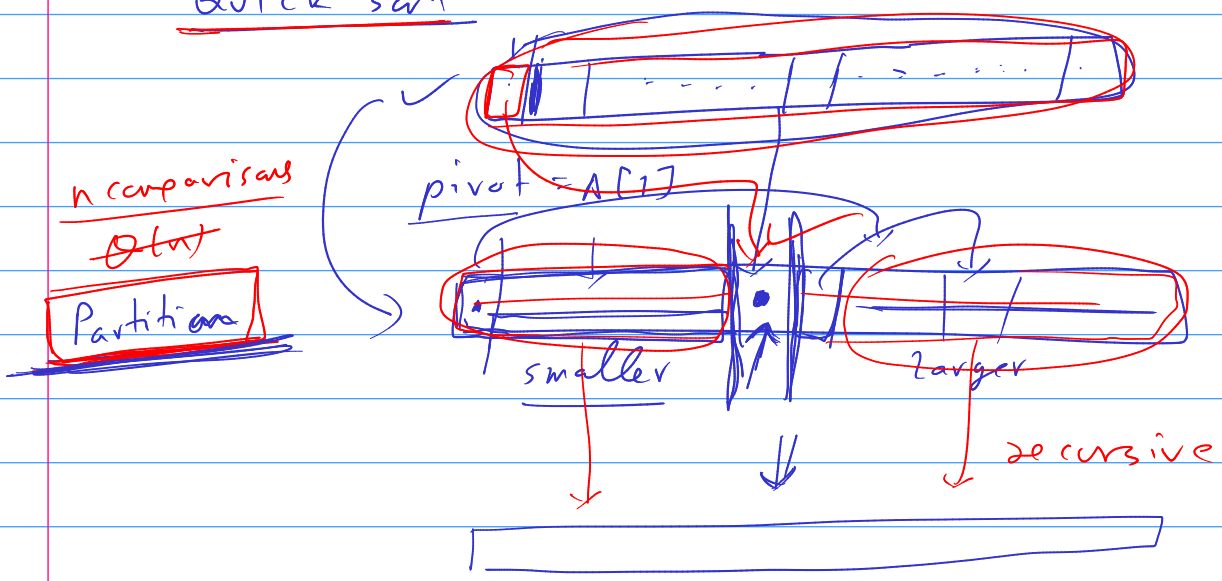
$$T_n = \frac{1+2+3+\dots+n}{n} = O(n)$$

$$= \frac{n+1/2}{} = \frac{n}{2} + \frac{1}{2}$$

$$= O(n)$$

Non-recursive implementation of Quick sort
Merge

Quick sort



Merge sort: $\rightarrow O(n \lg n)$

merge procedure

$n = 2^k$