Design and Analysis of Algorithms (DAA)

Design and Analysis of Algorithms (DAA)

Algorithm:- A sequence of logically related instructions to solve a Computational problem

Computational problems
↳ I/P & O/P

↙ ↘

I/P: Array A, x
? Does $x \in A$
O/P: Yes / No

I/P: Array A
? Sort A

Design and Analysis of Algorithms (DAA) -①-②-③

Algorithm:- A sequence of logically related instructions to solve a Computational problem

Computational problems
↳ I/P & O/P

↙ ↘

I/P: Array A, x
? Does $x \in A$
O/P: Yes / No

I/P: Array A
? Sort A

Any Algorithm must Satisfy (characteristics)

↳ I/P, O/P

↳ Finiteness:- It must terminate after a finite amount of time (1s, 1hr, 1day, 1month)

↳ Definiteness:- Each instruction must be unambiguous → [x=5 ✓ ; y=5 or -4 ✗]

↳ Must involve 1 basic arithmetic instructions!

→ MAX in an integer array

```
i=0
Max= A[i]
For i=1 to n
  if (A[i] > Max)
      Max=A[i]
Return Max
```

→ MAX in an integer array

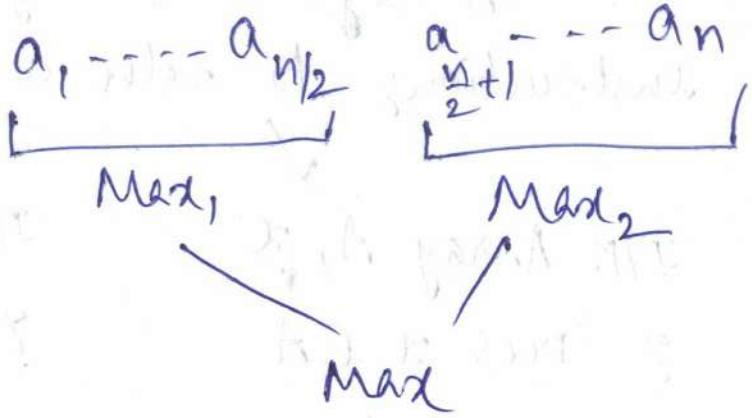
(I)

```
i = 0
Max = A[i]
For i = 1 to n
    if (A[i] > Max)
        max = A[i]
Return Max
```

(II)

```
Sort A
Return A[n]
```

(III)



↳ Many approaches (Algorithms)

If the problem is Solvable (There exists an Algorithm)

↳ How many different algorithms are possible.

CS

↓

A set of Computational problems

/ |

Solvable unsolvable

(There exists an Algo) (No Algo exists)

CS

↓

A set of Computational problems

/ |

Solvable unsolvable

(There exists an Algo) (No Algo exists)

- How many different algos are possible
- How do we Compare Algo
- How to pick the best one (Efficiency)

Design and Analysis of Algorithms (DAA) ⑥-⑦-⑧

CS

↓

A set of Computational problems

/ |

Solvable (There exists an Algo) | unsolvable (No Algo exists)

- L) How many different algos are possible
- L) How do we Compare Algo
- L) How to pick the best one (Efficiency)

① Correctness [Is the Strategy/Approach/Algorithm designed performing the intended task]

② How to measure the time spent / Space occupied (time Complexity) / (Space Complexity)

③ Optimality — A_1 is least { A_1 - 10 s, A_2 - 20 s, A_3 - 30 s }

Can we better A_1

Is there an algorithm whose time is < 10 s

Suppose A_4 (2 s); Is A_4 best ever? ⑧