DSAT (Lecture 1)

Administrative Details:

Grading:

- 30% : Two assignments each 15% (Two weeks deadline no extension)
- 25% : Minor
- 45% : Major

Note: Grading will be relative. Score at least 30% to pass the course.

Total no. of lectures: 18

Book: Introduction to Algorithms by CLRS

What's a computational problem?

Given an input (also instance) a comp. problem asks to return as output a solution satisfying some property.

Examples

Prime:

Input: A positive integer n.

Output: Yes/No depending on whether n is prime.

Sorting:

Input: A seq. of n numbers $(a_1, a_2, \ldots, a_n)$.

Output: A permutation $(a_1', a_2', \ldots, a_n')$ of the input seq. s.t. $a_1' \leq a_2' \leq \ldots \leq a_n'$.

What's an algorithm?

An algo. is a seq. of computational steps that transforms the input to output.

An algo for a comp prob is correct, if for every input algo halts and gives the right answer.

Example: Algorithm to solve Prime.

```
Prime (n):
    if n == 1
        return NO

    for i = 2 to n/2
        if n % i == 0
            return No

    return Yes
```

What is a data structure?

A data structure is a way to store and organise data in order to facilitate access & modifications.

Goals of this course:

1) Designing algorithms to solve computational problems.

2) Proving correctness of algorithms.

```
fun1(a,b) :  ⟶ gcd of a & b
    if b == 0
        return a
    else
        for i = max(a,b) to 1
            if a % i == 0 and b % i == 0
                return i
```

```
gcd(a,b)
    if b == 0
        return a
    else
        return gcd(b, a % b)
```

Recursive

$gcd(a,b) = gcd(b, a \% b)$

$a \% b = a - q\,b$

$(15, 3) \rightarrow (3, 0)$

3) Argue about the efficiency of algorithms.

↑ will be mostly concerned with time. sometimes space.

Insertion Sort

5, 2, 4, 6, 1, 3 → 1, 2, 3, 4, 5, 6

Idea: To sort A[1:n], we sort A[1], then A[1:2], then A[1:3], ...

5, 2, 4, 6, 1, -3 → 2, 5, 4, 6, 1, -3

↓

2, 4, 5, 6, 1, -3 ← 2, 4, 5, 6, 1, -3

↓

1, 2, 4, 5, 6, -3 → 1, 2, 3, 4, 5, 6

A[1:6]

A[1:1], A[1:2], A[1:3], ..., A[1:6]

Pseudocode for Insertion sort:

Insertion-sort (A, n):