

Algorithms I

Definition

An **algorithm** is a *well defined* computation procedure that takes a set of values as input and produces a set of values as output.

Note: the term *well defined* is itself, not well defined.

Definitions

- **Problems** have specific inputs and outputs, input must be finite and not a stream of data.
- **Problem instances** is a specific set of inputs for a problem. A problem can have a Big-O but not a problem instance.
- A program is **correct** if for every input instance, it terminates with the correct output.

Note

- Randomised algorithms is a branch of incorrect algorithms.
- Some algorithms produce incorrect outputs with a probability (e.g. quantum computing)
- Some algorithms loops infinitely for some inputs, but runs a lot faster than an algorithm that guarantees termination for cases where it terminates. It might be possible to determine whether it will terminate for a specific input before running it.
- Some algorithms gives an output within a margin of error (e.g. A* vs Dijkstra)

Notation

Arrays

- $A[1]$ is the first item
- $A[1..n]$ is an array of length n
- $A.length$ is the number of items in the array

We write pseudocode that is

- Imperative
- Block structured
- Fixed form (indentation matters)
- Parameters are passed by values, objects are passed by pointers
- Loop induction (for loops) increments after the final loop

```
for i=1 to 10
  // do stuff
```

After this loop, consider $i=11$

Sorting

Each **key** may have attached payloads.

Insertion Sort

```
for j = 2 to A.length
  Key = A[j]
  i = j - 1
  while i > 0 && A[i] > Key
    A[i + 1] = A[i]
    i = i - 1
  A[i + 1] = Key
```

Use proof by induction for algorithms:

- **Initialisation:** find a property that is true at the start of the program

P : at the start of each loop, $A[1 \dots j - 1]$ contains the $1 \dots j - 1$ items in sorted order.

At the start of the first loop, that is just $[a_1]$, true.

Note

Define “the start of the loop” as: after assigning the value of j , but before running the first line of code in the loop.

- **Maintenance:** show that the property is maintained as the program is running.
- **Termination:** when the program terminates, show the output is correct.

After the last loop, $A[1 \dots A.length]$ would have been containing all the items $1 \dots A.length$ in order.

And then we can also show the program terminates as it only needs to complete the loop $A.length$ items.

Note

Which is the same as the following Hoare logic proof.

Let P, Q be pre and post-conditions, B be body of the loop, C be condition for the loop.

Given:

1. $\{P\} B \{P\}$
2. $P \wedge \neg C \implies Q$

Then $\{P\} \text{ while } C \text{ do } B \{Q\}$
