MCAC 301

Design and Analysis of Algorithms

Neelima Gupta

ngupta@cs.du.ac.in

Disclaimer: Every lecture will be recorded. If anyone has any objection, you are requested to put your objection in the chat box.

Introduction

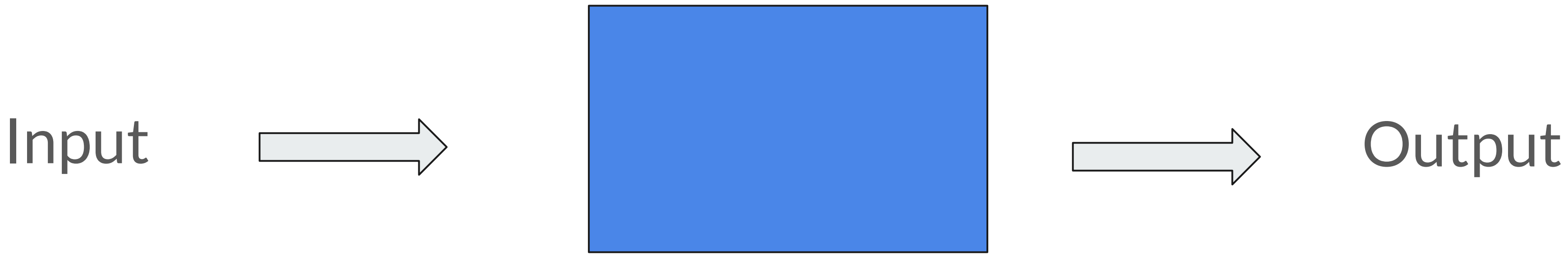
Structure:

- Lecture on a topic,
- Quiz: 5 -15 mins,
- Discussion on Quiz: 5 -10 mins,
- Programming Assignment on every topic
- Next Topic
- Minor and Major will be as per department policy

Pre-requisite: Familiarity with basic Maths : sets, functions, relations, log functions, polynomials, exponents etc.

Familiarity with basic data structures like arrays, stacks, queues.

What is an Algorithm?

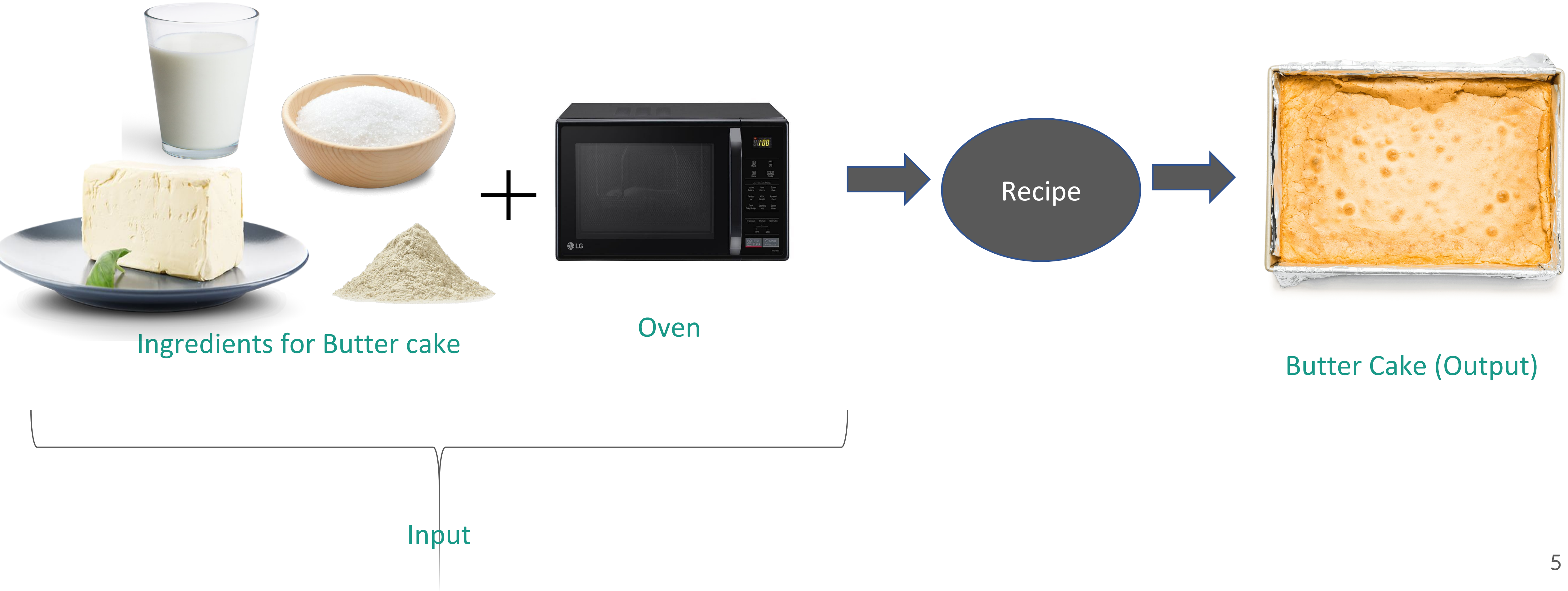


Input

Output

- Algorithm is a finite sequence of instructions that uses the given input and produces the desired output

Algorithms Everywhere



Ingredients for Butter cake

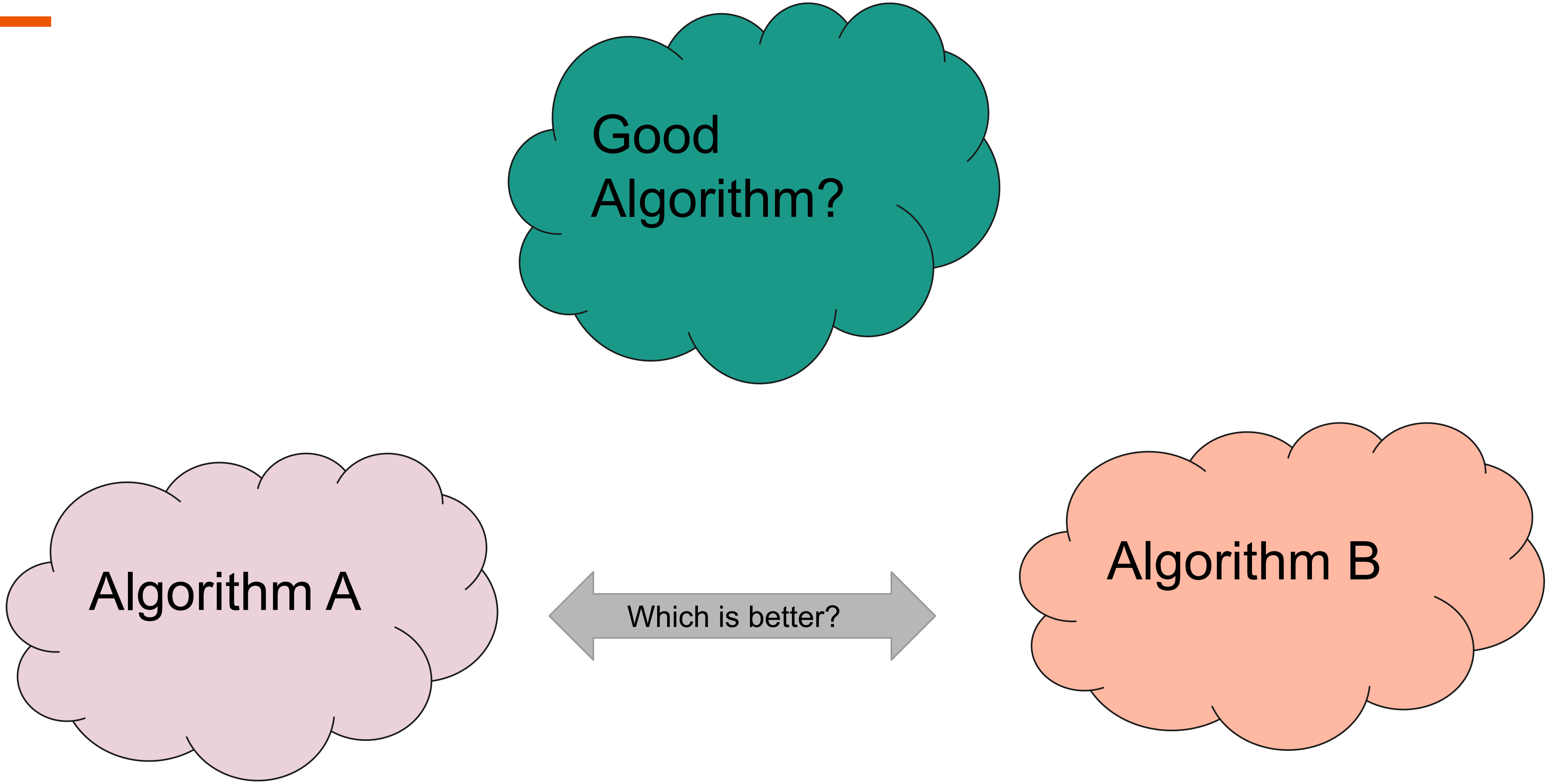
Oven

Recipe

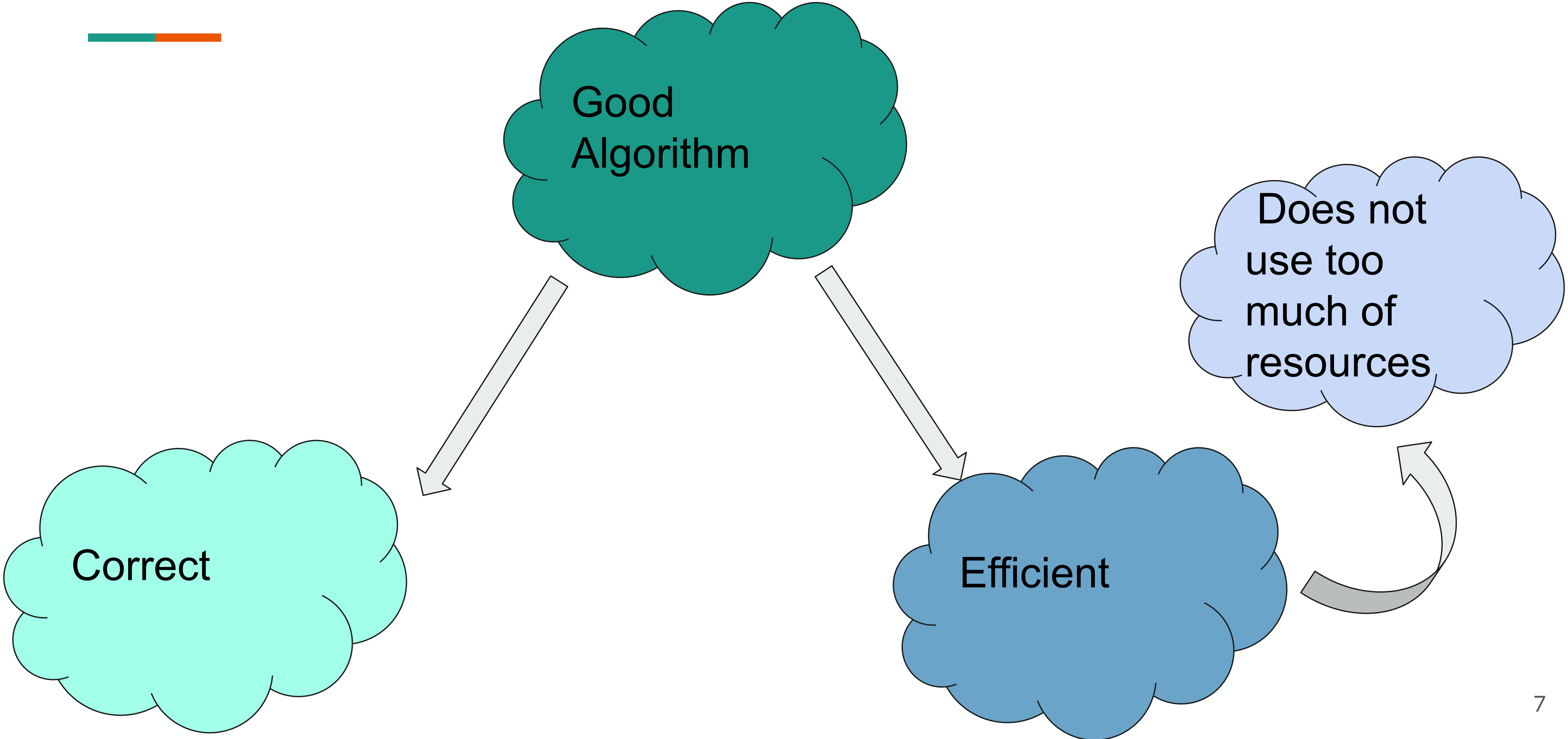
Butter Cake (Output)

Input

Questions we ask



When do we say that an Algorithm is good?

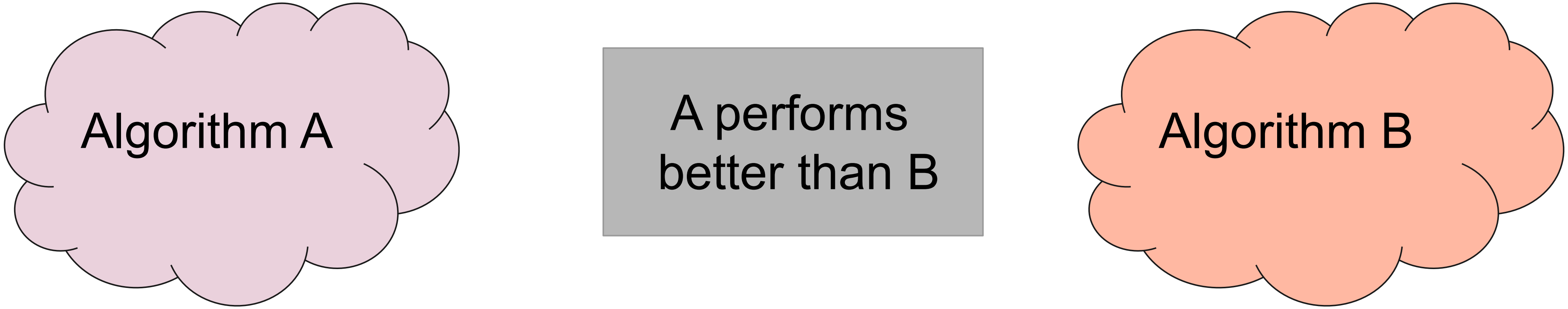


Correctness

- Experimental Studies:

  - Limitations:
    - Effort to Code,
    - inexhaustive on inputs and,
    - if the output is large, how do you verify that it is correct?
- Mathematical Tools: Proofing Techniques ✓

Compare Two Algorithms



- If A performs better than B on large input irrespective of their relative performance on small inputs

By “performs better” we mean “uses lesser resources”.

Question I promised to answer

- What do we mean by “not too much” in “not too much of resources”?

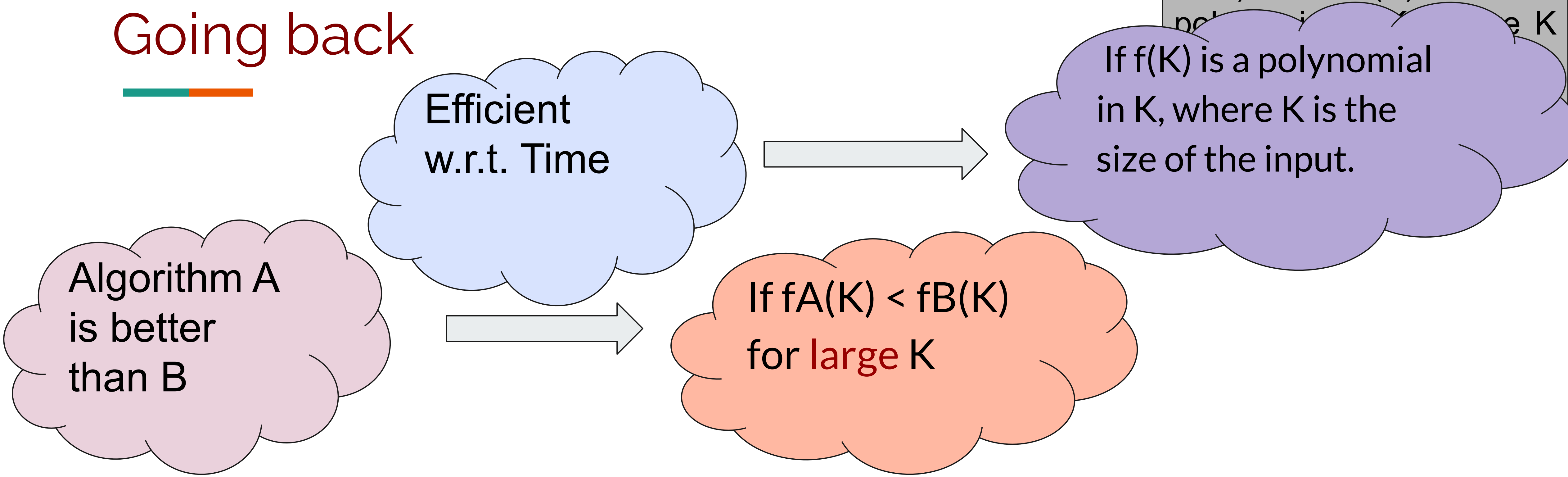
Let f(K) denotes the amount of resource used by an algorithm A on an input of size K.

For example, let the running time of A ,on an input of size n, be expressed as $f(n) = n^2$

For the purpose of this course, we say that an algorithm does not use too much of a resource if f(n) is a polynomial in ‘n’.

Going back

An algorithm is efficient(with respect to time) if f(K) is a po i e K



Algorithm A is better than B

Efficient w.r.t. Time

If fA(K) < fB(K) for large K

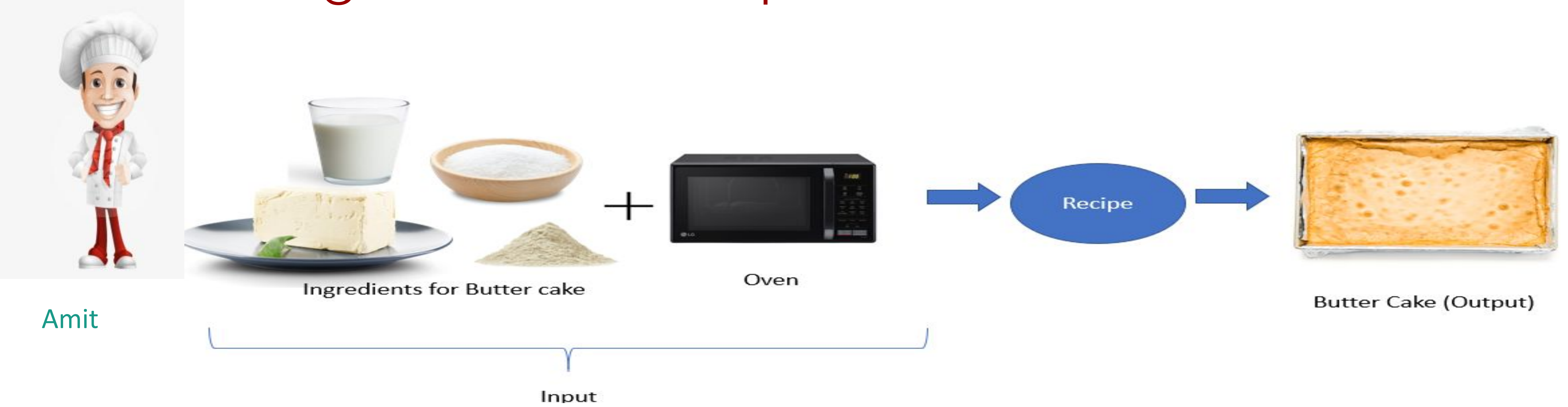
If f(K) is a polynomial in K, where K is the size of the input.

Qs: 1. How to compute the time f(K)?

2. What do we mean by the input size K?

Input Size is the number of memory words required to store the input. WIll study more in detail later.

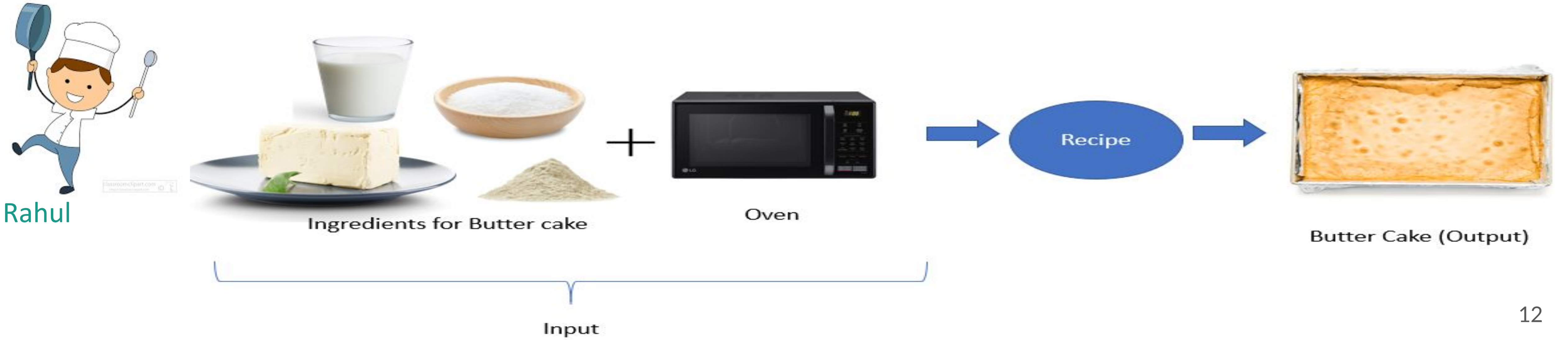
The Big Q: How to compute f(B) - the time?



Amit

Ingredients for Butter cake + Oven → Recipe → Butter Cake (Output)

Input



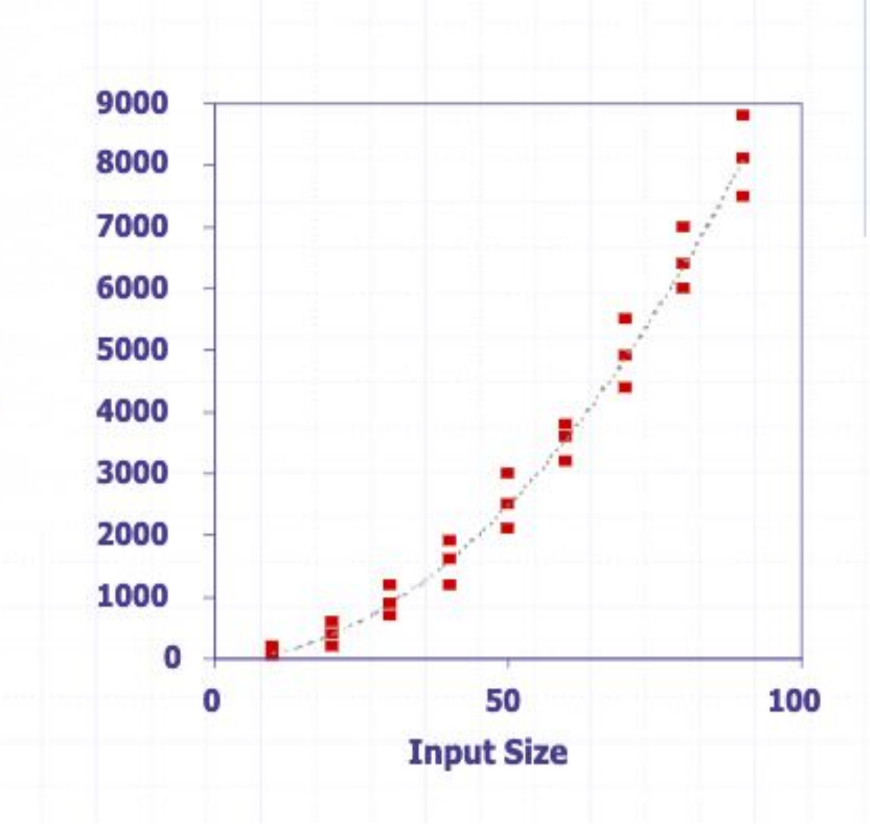
Rahul

Ingredients for Butter cake + Oven → Recipe → Butter Cake (Output)

Input

The Big Q: How to compute f(B) - the time?

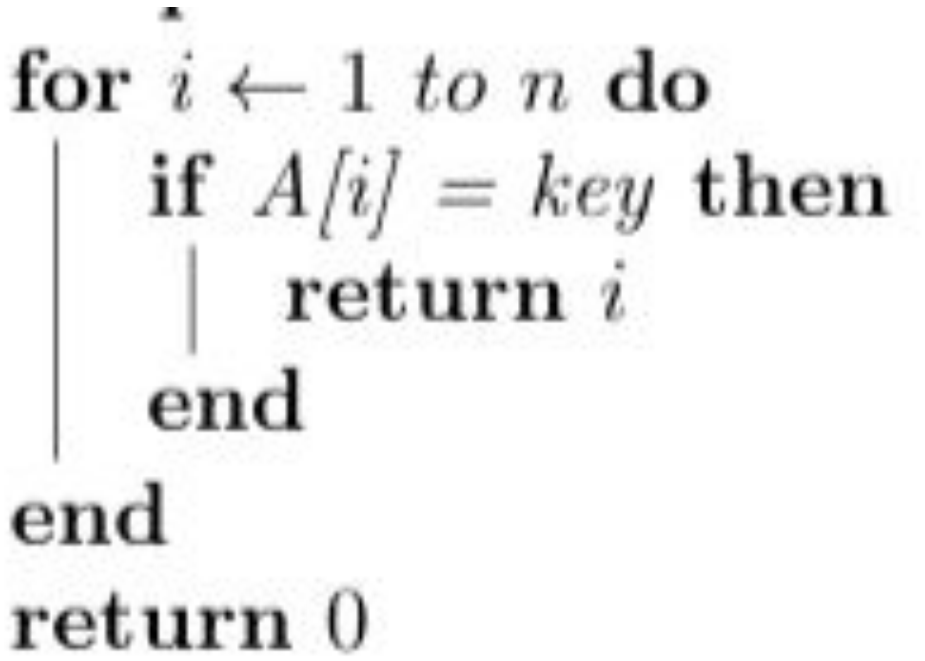
- Experimental Studies:



- Limitations:
  - Effort to Code,
  - inexhaustive on inputs and,
  - maintaining same environment for both the algorithms, which is nearly impossible
- Theoretical Analysis: Count the Primitive steps ✓

Theoretical Analysis (Counting the Primitive Steps)

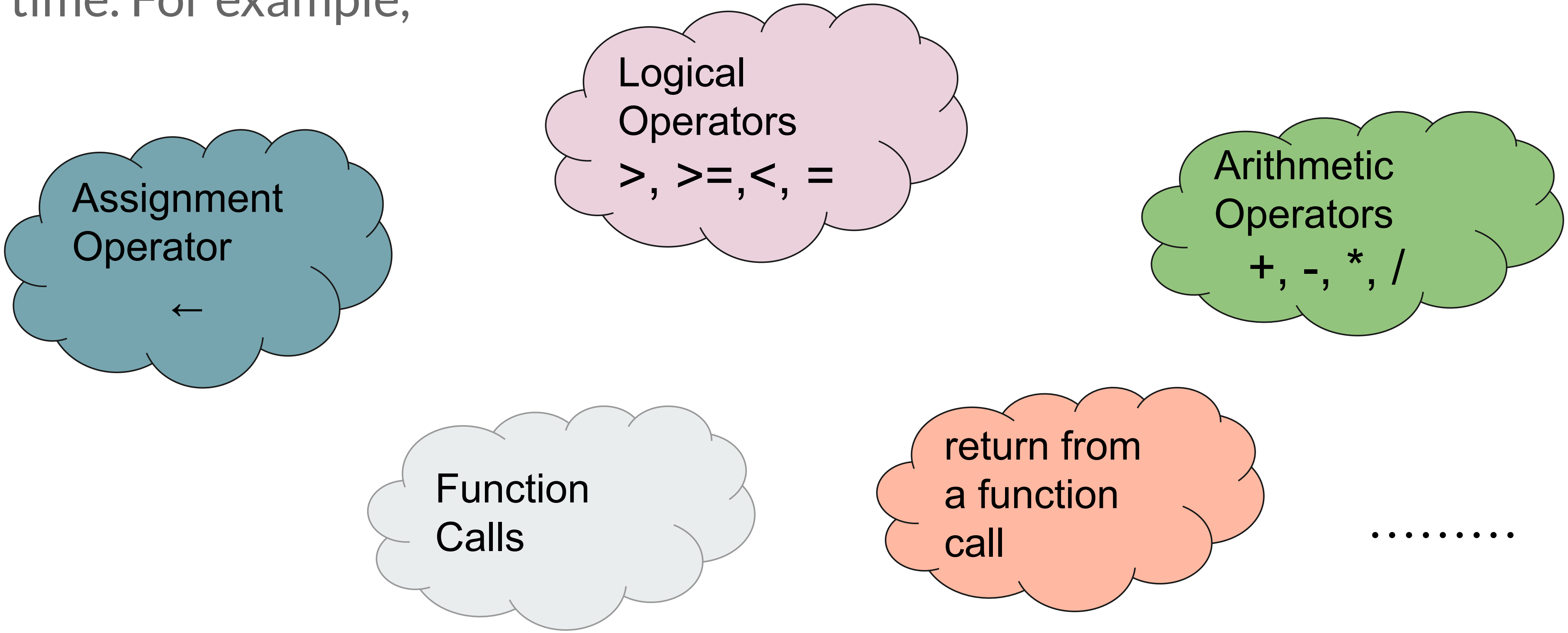
- Uses a high-level description of the algorithm instead of a detailed implementation
- Considers all possible inputs
- Allows us to evaluate(estimate) the time taken by an algorithm independent of the hardware/ software environment



```
for $i \leftarrow 1$ $to$ $n$ do
    if $A[i] = key$ then
        return $i$
    end
end
return $0$
```

Primitive Steps

Primitive Steps are the steps that can be performed in constant amount of time. For example,



Logical Operators
>, >=,<, =

Assignment Operator
←

Arithmetic Operators
+, -, *, /

Function Calls

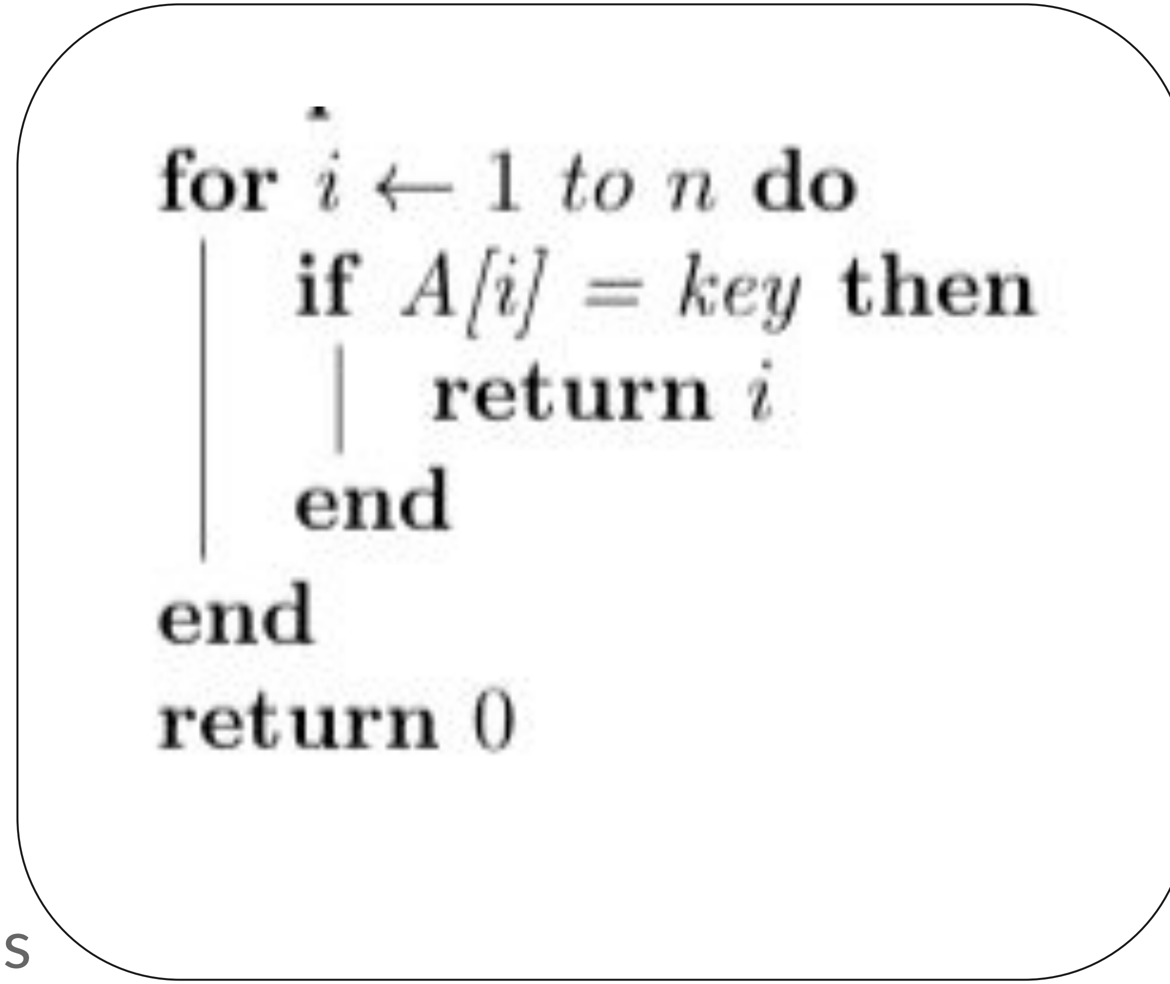
return from a function call

....

Pseudocode : A high-level description of an algorithm

- ❑ Less Structured than programs
  - ○ Only high-level description of an algorithm - hides details of implementation
  - ○ Doesn’t require programming language support
  - ○ Hides Syntactical issues

- ❑ More Structured than English prose
  - ○ Convenient to prove correctness and
  - ○ Count the number of primitive operations



```
for $i \leftarrow 1$ to $n$ do
    if $A[i] = key$ then
        return $i$
    end
end
return 0
```

Pseudocode Standards

- Control flow
  - **if** … **then** … [**else** …]
  - **while** … **do** …
  - **repeat** … **until** …
  - **for** … **do** …
  - Indentation replaces braces
- Method declaration

```
Algorithm method (arg [, arg…])
    Input …
    Output …
```

- Method call

  method (*arg* [, *arg*…])

- Return value

  return *expression*

- Expressions:

  ← Assignment

  = Equality testing

  n^2 Superscripts and other mathematical formatting allowed