Introduction to Algorithms

Module Code: CELEN086
Credits: 10
Semester: Sem-1, AY 24-25

Lecturer & Convenor: Manish Dhyani
Email: Manish.dhyani@nottingham.edu.cn

Y-1 computing modules

- CELEN086 in Sem-1

  Introduction to Algorithms
  (10 credits)

- CELEN087 in Sem-2

  Introduction to Mathematical Software and Programming
  (10 credits)

Teaching Staff

Module Convenor: Manish Dhyani Co Convenor: Chenfei Zhang

Lecturer and Seminar Tutor

Manish Dhyani
Email: Manish.Dhyani@nottingham.edu.cn
Office: Trent 341

Seminar Tutor

Chenfei Zhang
Email : Chenfei.Zhang@nottingham.edu.cn
Office : Trent 341

Seminar Tutor

Chenyang Xue
Email : Chenyang.Xue@nottingham.edu.cn
Office : Trent 341

Teaching scheme

- 11 lectures
  - Lecture videos available on Moodle
- 10 seminars
  - Practice on topics covered in previous week's lecture

Note: it has been calculated that this module will take up to 100 hours of your time including approximately 20 hours of lectures and seminars.

Office Hours(From W/c 07/10/2024)

Day Of Week	Time Slot	Venue
Wednesday	11:00-12:00	Yang Fujia Building 412
Thrusday	10:00-11:00	PB 217
Friday	14:00-15:00	Yang Fujia Building 328

- You join office hours without prior appointment.
- During office hours you can get one to one help from tutor for topics covered in the module.

Learning resources

All the learning resources will be uploaded to Moodle page.

- ❑ Module handbook (essential module information)
- ❑ Learning materials
  - Lecture/Seminar slides
  - Problem sheets
  - Reading/Self-study materials
- ❑ Assessment information

What will I learn on this module?

- Computing basics for algorithms and pseudocodes.
- Recursion concept for designing algorithms.
- Techniques for designing algorithms such as divide-and-conquer, helper functions, sub-algorithms.
- Introduction to concepts of data structures: lists, trees, graphs.
- Lists: algorithms on lists, searching and sorting algorithms such as linear search, binary search, insertion sort, merge sort, bucket sort and their time complexities.
- Trees: algorithms on trees; binary trees; binary search trees (BST); traversal schemes on BST; building BST from list.
- Graphs: concepts of different types graphs; Euler path and Euler circuit; topological sorting; minimum spinning tree; shortest path algorithms such as Dijkstra's algorithm, Prim's algorithm, Kruskal's algorithm.

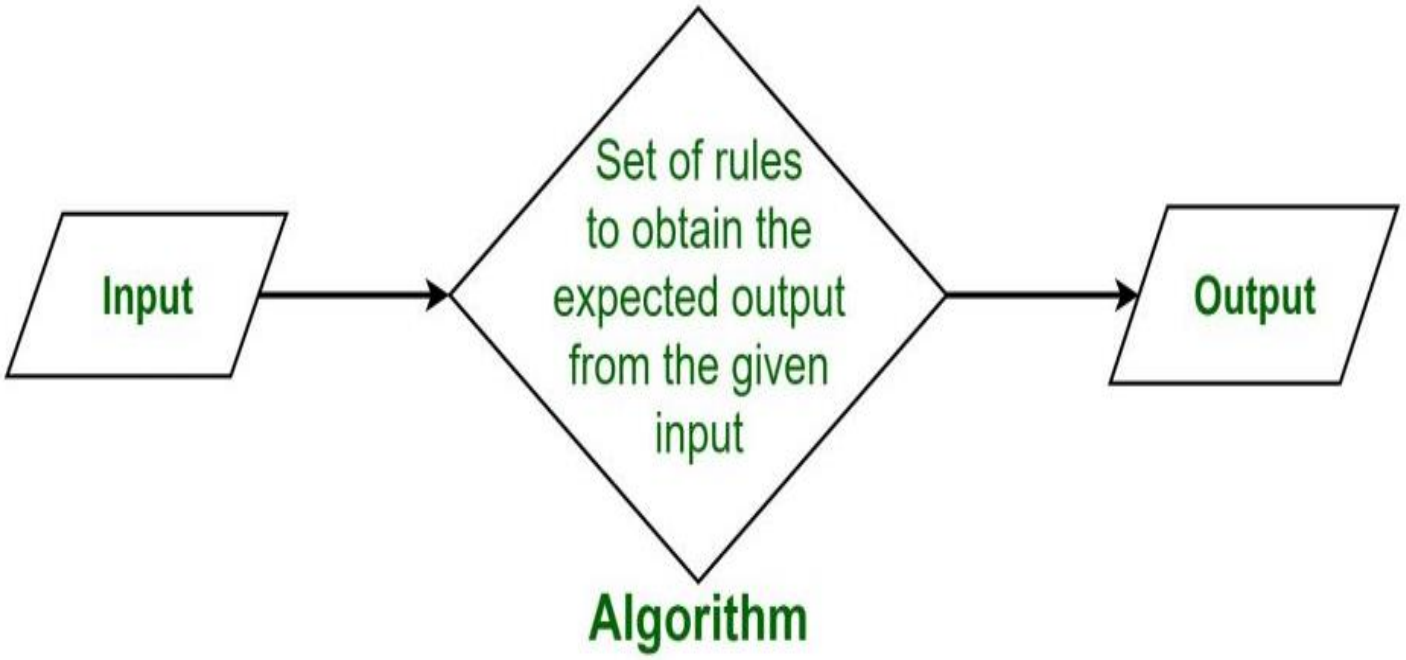
Assessment

Type of Assessment	Information		Weighting
Mid-Semester Exam	Short answer question based on Algorithms up to Lecture 6	20 November 2024 Wednesday	25%
Coursework	Based on Algorithms up to Binary tree	Given 21 November 2024	25%
		Collected 5 December 2024 by 5 pm	
Final Examination	Written exam 2 hours, 5 Questions	No calculators	50%

Plagiarism

- The use of other people's work and the intentional submission of it as though it was one's own is regarded as plagiarism.
- It is a serious academic offence and could result in termination of your University degree course.
- Copying another student's work is not only plagiarism, but it is educationally misguided.
- The penalties and procedures associated with plagiarism are provided in a separate document (Student's Handbook).
- Discussion and working with colleagues and friends is often a productive way of studying and learning. However, Computer Science is a practical subject and to become proficient at it, you must work at the lectures, Seminar work sheets, and coursework.
- Remember, you will be assessed (especially in examinations) on your ability and on what you understand.

What is Algorithm?



Guess Number

Guess a number between 1 and 100.

Tell someone to pick a number between 1 and 100. Your goal is to guess the number.

For each guess you make, your friend tells you one of three things:

- you guessed the number,
- your guess was high, or
- your guess was low.

Winner is the one who will make minimum number of guesses.

Guess Number

Guess a number between 1 and 100.

What is your method to find the correct guess number.

What do you think in how many guesses you can find the correct guess of a number.

What is an algorithm?

Algorithm is a step by step procedure for solving a specific problem or accomplishing a task.

Characteristics of algorithms:

- clear and unambiguous
- well-defined inputs and outputs
- finite
- feasible
- language independent

Applications of Algorithms

Financial Forecasting :
Algorithms help to predict stock prices, detect fraud, and optimize investment strategies.

Medical Diagnosis:
AI Algorithms analyze medical data to aid in disease diagnosis, treatment planning, and drug development .

Transportation Planning:
Route optimization algorithms improve efficiency in transportation and logistics, reducing travel time and cost .

Recommendation Systems:
Algorithms powered personalized recommendations in areas like e-commerce, streaming services, and social media.

Our focus

We will focus on designing algorithms for solving computational problems with different data structures. Prerequisite mathematical topics where necessary will also be covered in this module.

For example:

- finding Greatest Common Divisor
- finding Prime number
- sorting and searching in a List
- searching in a Binary Search Tree
- finding the shortest path in a Graph

Algorithms we will learn

Searching algorithm

Sorting algorithm

Recursive algorithm

Divide-and-conquer algorithm

Greedy algorithm

Brute Force algorithm

Baidu/Google search

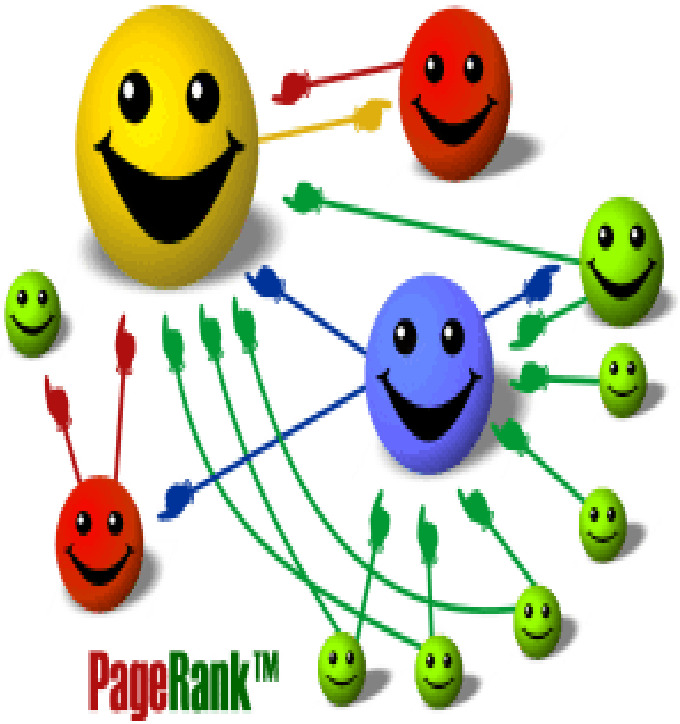
GPS routing

Online shopping

Examples of well –known Algorithms

Google Search engine (Page Rank Algorithm)

- It ranks pages in order of importance.
- Google ranking or numeric value of page is dynamic it changes with demand on web.
- It uses matrices of order 10^{10} or more.



Examples of well –known Algorithms

Edge Rank Algorithm

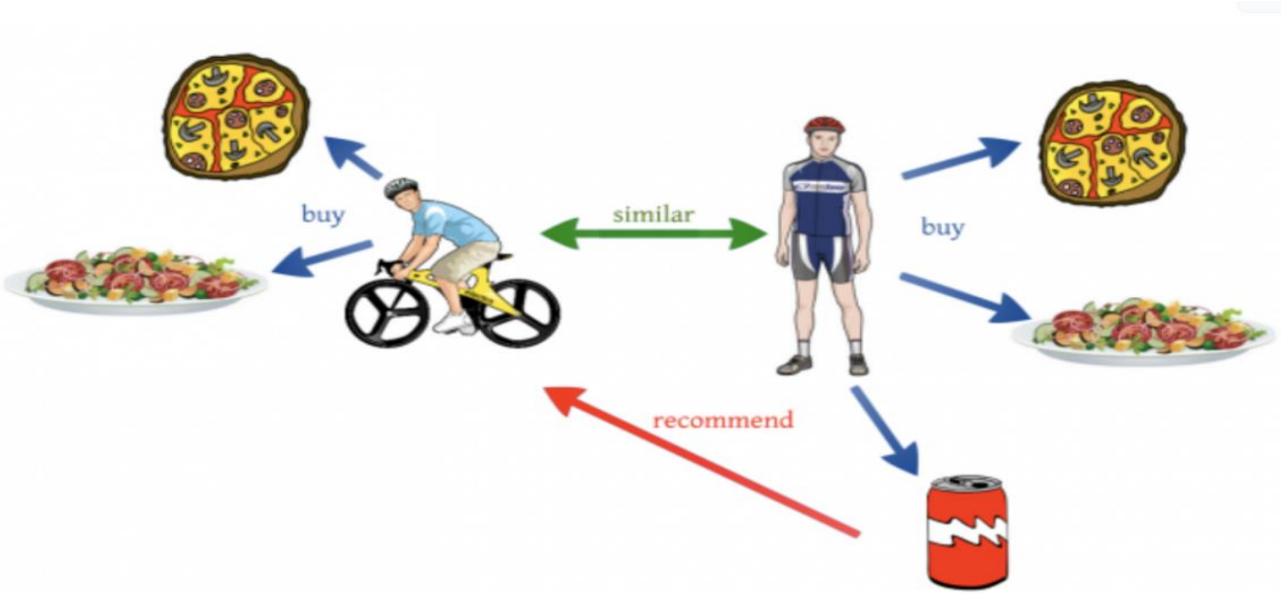
Used in Amazon and Facebook

This personalization of the home page has a big benefit for the online store compared to just displaying top 10 lists or banner ads: the clickthrough and conversion rates are far higher. Customers are more likely to view and buy the suggested products.

Examples of well –known Algorithms

Among all the product recommendation methods, one of the most popular is **collaborative filtering algorithms**, which depends entirely on how other customers and users previously rated the products they purchased.

This method stems from the idea that people with similar past purchases and hobbies may also want to use the same things in the future.

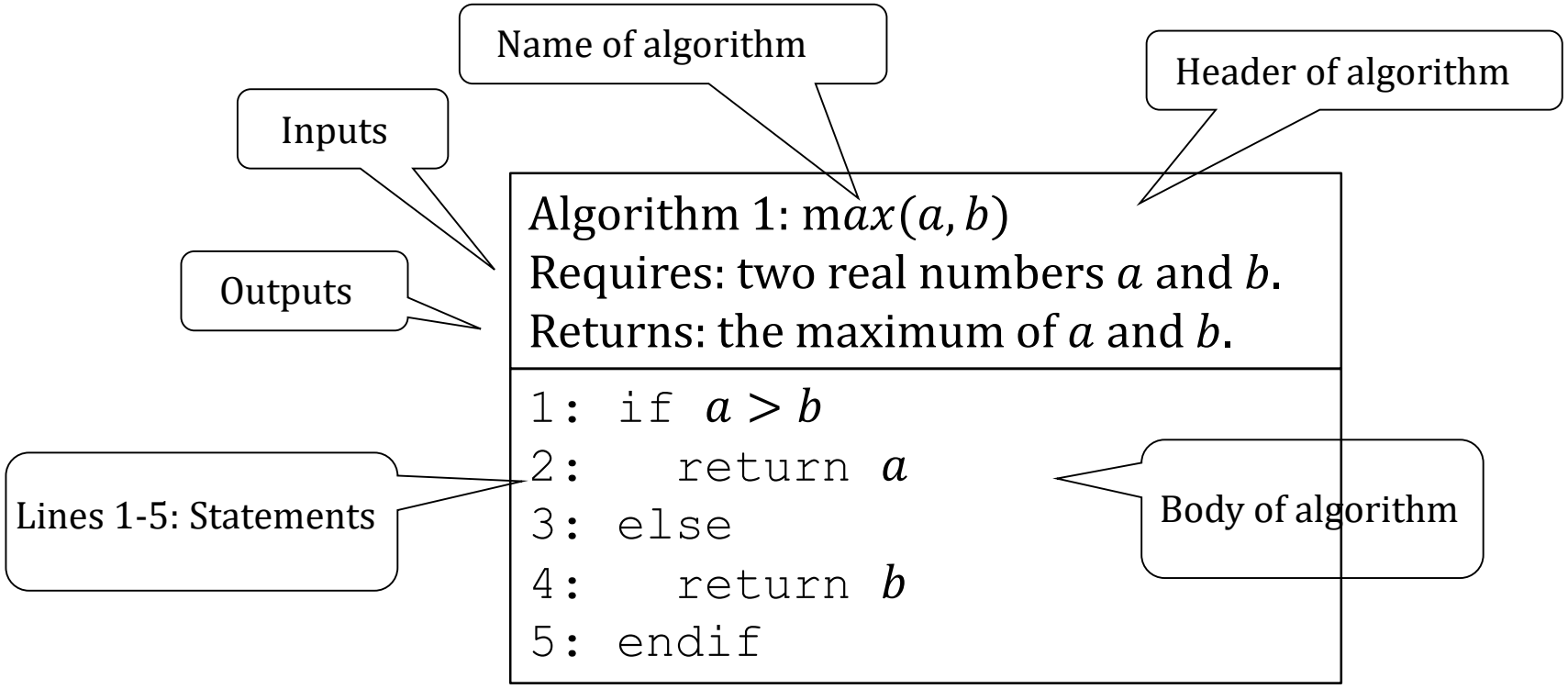


Source Towards Data Science

How to write an algorithm?

Algorithm for finding minimum of two numbers.

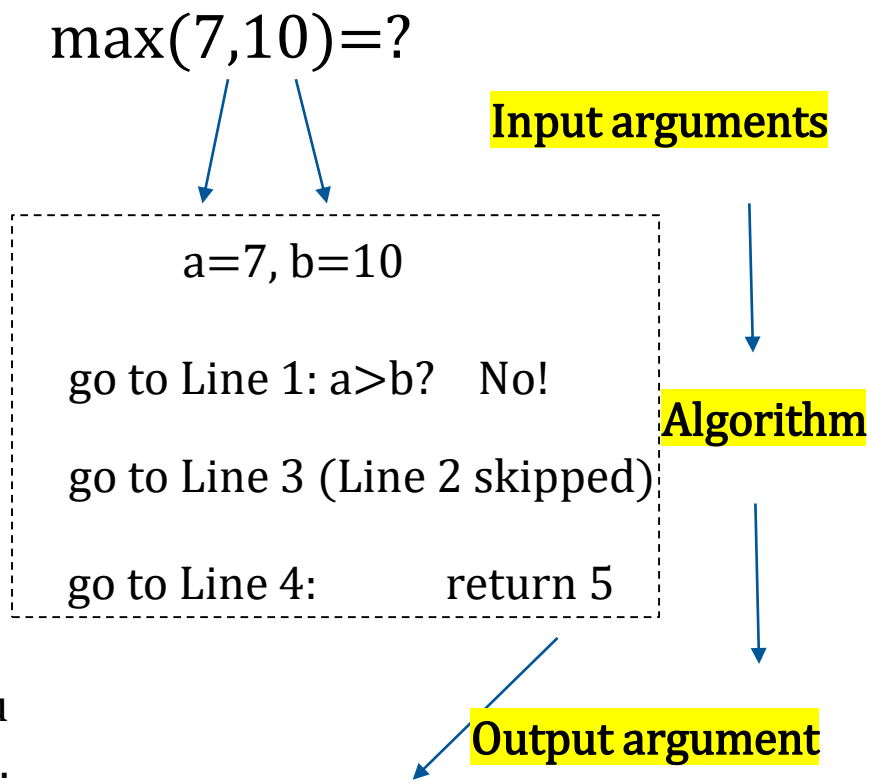
We can design the pseudocode as follows:



Tracing algorithm

Algorithm 1: $max(a, b)$
Requires: two real numbers a and b.
Returns: the smaller one of a and b.

```
1: if a > b
2:    return a
3: else
4:    return b
5: endif
```



To be sure your algorithm works perfectly, you must TRACE it with different input arguments.

Does this algorithm work for two equivalent inputs? e.g., max(5,5)

Computing basic key terms

- Variable
- Arithmetic operator
- Relational operator
- Boolean(logical) value/statement and operator
- Control flow

Variable

Similar to function $y = f(x)$, the input and output arguments are variables that hold different values.

Type of variable	Example
Numerical variable	$X = 3$, $Y = \pi$, $Z = \max(X, Y)$
Character/String	X = 'a', Y = "Hello World"
Boolean variable	True, False

Arithmetic operator

Operator	Action/Operation
=	Assignment operator. Let x = 10 means the numerical value of 10 is assigned to variable x
+	Addition
–	Subtraction
*	Multiplication
/	Integer division (quotient) e.g. 7/2 = 3, not 3.5
% or mod	Remainder of division e.g. 14 % 3 = 2, 5 mod 2 = 1

Relational operator

Operator	Meaning
==	Equality (e.g. (a == b) means if the value of a and b are equal then we get True otherwise False)
!=	Not equal to. e.g. (7 != 6), which is True. (3 != 3), which is False.
>=	Greater than or equal to.
<=	Less than or equal to

Relational operators result in Boolean values/statements.

Boolean(logical) variable/statement

A Boolean variable/statement can only take two values: True or False.

Computers often return 1 for True and 0 for False.

$$P = (4 < 3)$$

How to interpret it?

i. 4<3, not true! This is a Boolean statement.
ii. Assign the value in parentheses to variable P, so P is a Boolean variable taking value false.

$P = (4 < 3)$

Logical operator

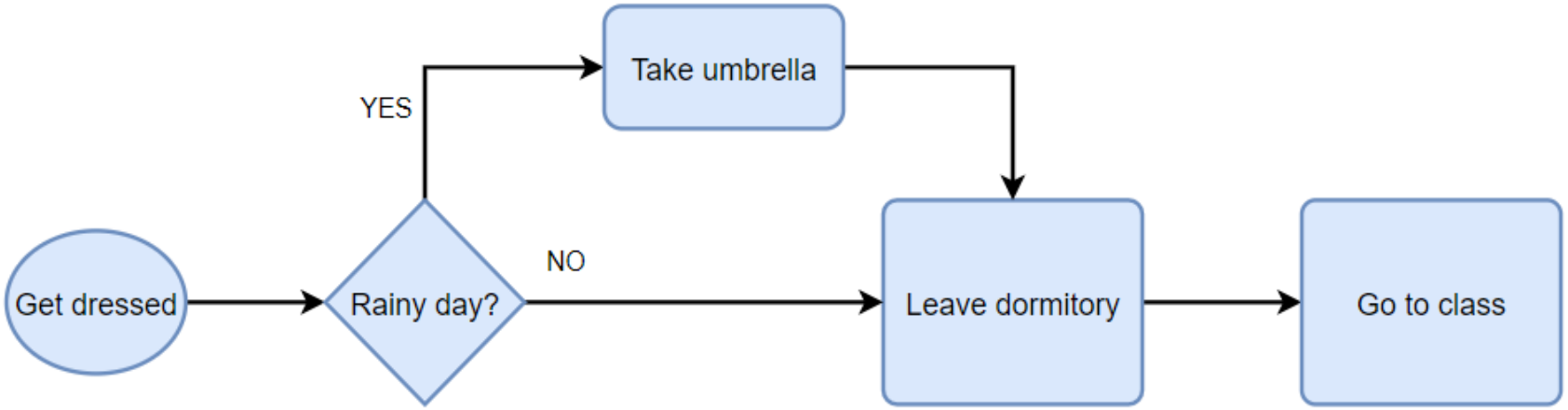
Suppose that P and Q are Boolean variable/statement, we can compound them using logical operator

AND, OR and NOT(negation) :

AND (P,Q)	P AND Q	P && Q
OR (P,Q)	P OR Q	P || Q
NOT P		! P ¬ P

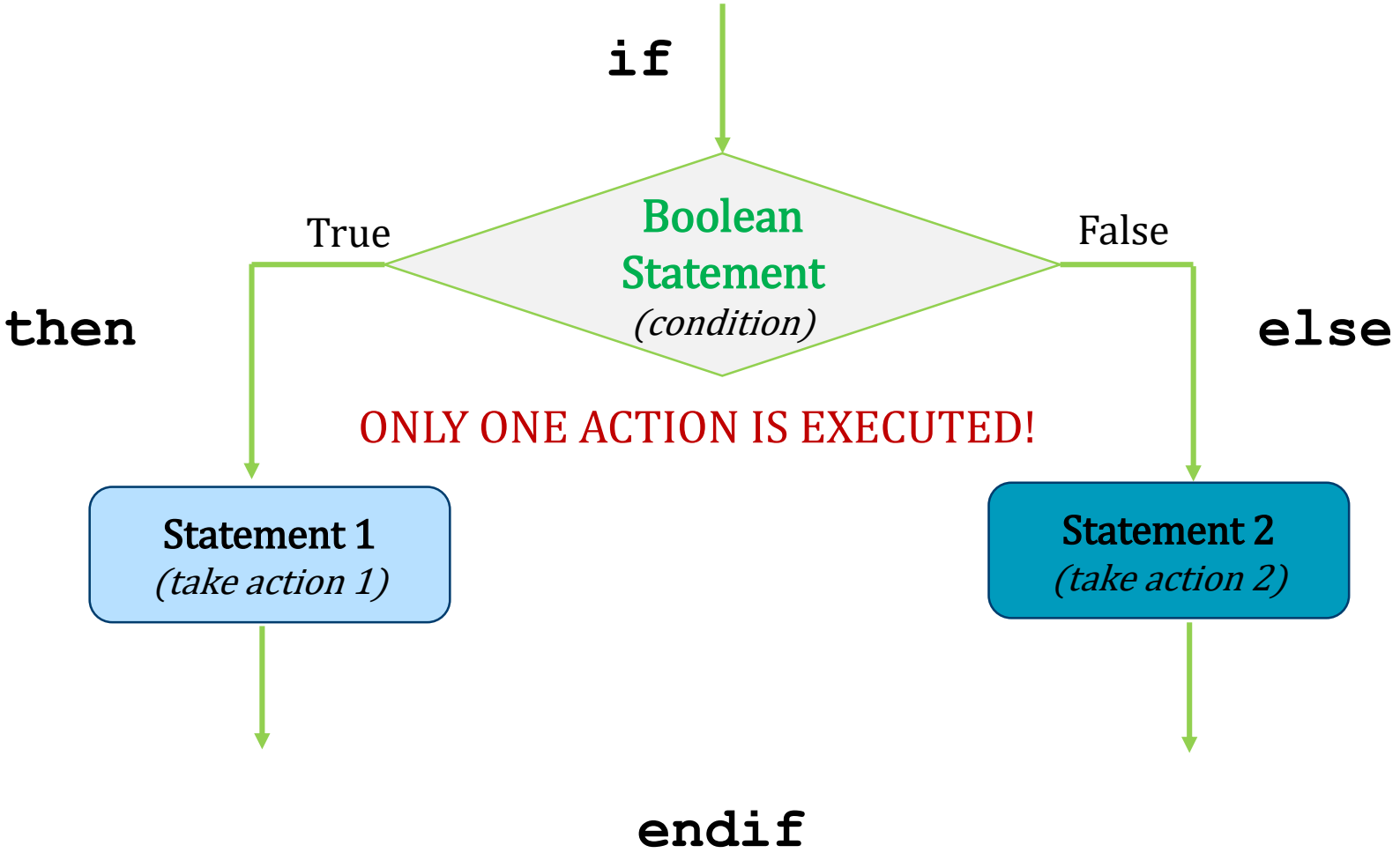
Control flow

A flowchart is a diagram that depicts a process, system or computer algorithm.



- Many programs are composed of statements that are carried out sequentially.
- But sometimes, depending on the choice made by the user, we want our programs to perform different actions.
- There are several statements (called control flow structures) that enable us to change the program flow.

If-else statement (if-block)



Example

Algorithm 3: isEven(n)
Requires: a nonnegative integer n
Returns: True if n is an even number; False if it is odd

```
1. Let x = n mod 2 % compute remainder
2. if x == 0 % n is divisible by 2
3.      return True
4. else % remainder is 1, not divisible by 2
5.      return False
6. endif
```

Note:

- The green statements are called comments.
- It is a good practice in programming to add comments where necessary so that other users of the program/algorithm can understand what the programmer has done.
- Comments will have no effect on the output of a program/algorithm.

Example

We can use if-block to switch to different cases in program/algorithm.

One case:

```
if student_grade < 40
        return 'FAIL'
endif
```

Two cases:

```
if student_grade >= 40
        return 'PASS'
else
        return 'FAIL'
endif
```

More than two cases:

```
if student_grade >= 90
        return 'A'
elseif student_grade >= 80
        return 'B'
elseif student_grade >= 70
        return 'C'
elseif student_grade >= 60
        return 'D'
else
        return 'F'
endif
```

Proper algorithm with header

Algorithm 2: grade_calculator(score)
Requires: an integer between 0 and 100
Returns: letter grade

```
1. if score >= 90
2.     return 'A'
3. elseif score >= 80
4.     return 'B'
5. elseif score >= 70
6.     return 'C'
7. elseif score >= 60
8.     return 'D'
9. else
10.    return 'F'
11.endif
```

Trace the algorithm for grade_calculator(73)

score = 73

Line 1: is score >= 90? False

Go to Line 3

Line 3: is score >= 80? False

Go to Line 5

Line 5: is score >= 70? True

Go to Line 6

Line 6: return 'C'

Go to Line 11 (end if-block)

Practice

Write an algorithm to find area of circle whose radius r given.

```
Algorithm: Area_triangle(r)
Requires: A positive numbers r.
Returns: A positive number A i.e. area of circle.
```

OR

1. return $(3.14 * r * r)$