Introduction to Algorithms

CELEN086

Seminar 1
(w/c 07/10/2024)

Teaching Hours

Activity	Teaching weeks	Session	Room No.
Lectures	1-13	Thursday 4:00 pm -5:00 pm	DBA05
Seminars	2, 4-13	Check your individual timetable	

Assessment

Type of assessment	Information		Weighting
Mid Sem Exam Lecture 6	Based on Algorithms MCQ +Short answer questions	**20 November 2024**	25%
Coursework (Up to Binary Search Tree)	Based on Algorithms	Given **21 November 2024**	25%
		Collected **5 December 2024 by 5:00 pm**	
Final Examination	Written exam 2 hours, 5 Questions	No calculators	50%

Moodle

- Log in using your University username and password
  - Course announcements
  - Lecture slides
  - Seminar related materials
  - Solutions of homework questions
  - Sample exam paper.

What is expected from you?

- Attend all lectures.
- Attend all seminars.
- Put into practice what you have learnt by doing the exercises.
- Read the textbook.

Self Study and Assistance

- We have exercises in the problem sheet for each week.
- These exercises build up to coursework and exam questions.
- We expect you to complete them.

If you run into problems:

- Consult friends, books, internet.
- Email your Tutors or Visit Office hours:

Day/Time Slot	Room No.
Thrusday, 10:00 to 11:00	PB217
Friday, 14:00 to 15:00	YANG Fujia Building-328+(TB)
Wednesday 11:00 to 12:00	YANG Fujia Building-412(TB)

Outline

In this seminar, we will study and review on following topics:

- Basic computing operators: arithmetic/relational
- Boolean statement/operator
- IF-statement and its basic structure
- How to write proper pseudocode for algorithms

You will also learn useful Math/CS concepts and vocabularies.

Group activity

a daily algorithm

write an algorithm which describes a daily activity or something that you do frequently.

- work in groups of 4.
- discuss your algorithm in English (of course!)
- make sure that the structure, language and the logic are correct!

Group activity (5 minutes)

Work in groups, share your algorithm with your classmates.

- What task does it solve?
- What kind of inputs/outputs/data are there?
- Also, discuss some characteristics of algorithms shown in your example, e.g., finite? feasible?

Each member briefly introduces your algorithm for 1 minute.

Precedence of Operators

Some statements may consist of arithmetic/relational/logical operators. Refer to following table for evaluations.

Highest	Parentheses ()
Precedence ↓	Power ^
	Logical NOT !
	Multiplication * Division /
	Remainder mod, %
	Addition + subtraction -
	Relational <, <=, >, >=, ==, !=
	Logical AND &&
Lowest	Logical OR ||

Arithmetic and relational operator

Evaluate the following statement.

Answers:

- 17 % 4 — 1
- 17 / 4 — 4
- 3 % 4 — 3
- 3 == 2 — False
- 19 / 4 <= 4 — True
- (20 mod 6) != 2 — False

Precedence:

arithmetic > relational

Practice

Given any three digit number n (e.g., 365, 270)

1. Design a statement for representing the last digit (e.g., 5, 0)

   Answer: n%10 or n mod 10

2. Design a statement for representing the first digit (e.g., 3, 2)

   Answer: n/100

Practice (cont'd)

Given any three-digit number n (e.g., 365, 270)

3. Design a statement for representing the middle digit (e.g., 6, 7)

   Answer: (n/10) mod 10 or (n%100) / 10

   Hint: how to represent the first two (or last two) digits?

   n/10 n%100

4. Design a statement (expression) to check if n can be divided by 3 (or we say "n is divisible by 3") or not.

   Answer: n%3 == 0

Complete the following Truth Table

A&&B		
A	B	A && B
T	T	
T	F	
F	T	
F	F	

(A || B) && A			
A	B	A || B	(A || B) && A
F	F		
F	T		
T	F		
T	T		

Boolean (compounded) statement

Let age=16, day='Monday'. Evaluate the following statement:

• (age%20)>=age	16>=16	True
• (age>10) || (age<16)	True || False	True
• !((age%2)>10)	!(False)	True
• (day!='Tuesday') || (age>=16)	True || True	True

- !(age==15) && (day=='Monday')
- (age==16) && (day=='Monday') && (day!='Tuesday')
- (age==16) || (day!='Monday') && (age/5<2)
- ((age%3)>=1) && ((age+5)>20)

Answers: All are True.

If-statement

Structure of (simple) if-else statement:

Condition 1 is a Boolean statement.

```
if (condition 1)          // when condition 1 is satisfied (True),
        statement 1       // we execute statement 1.
else                      // otherwise, condition 1 is not satisfied (False),
        statement 2       // we execute statement 2.
endif
```

Example

Design an algorithm to check if an integer is divisible by 3.

Algorithm: div3(n)
Requires: a positive integer n
Returns: True if n is divisible by 3; False if not divisible by 3

```
1.  if  (??) // what condition can we use here?
2.          return True
3.  else
4.          return False
5.  endif
```

(n%3)==0

Trace div3(365)

Line 1: (365%3)==0 No! (False)

go to Line 3, and Line 4

return False

go to Line 5, done.

Practice

Design an algorithm for the Boolean/logical negation NOT().

Algorithm: NOT(P)
Requires: a Boolean variable P
Returns: the negation of P

P	!(P)
True	**False**
False	**True**

```
1.  if  P // why only P shows up here?
2.          return False
3.  else
4.          return True
5.  endif
```

!P

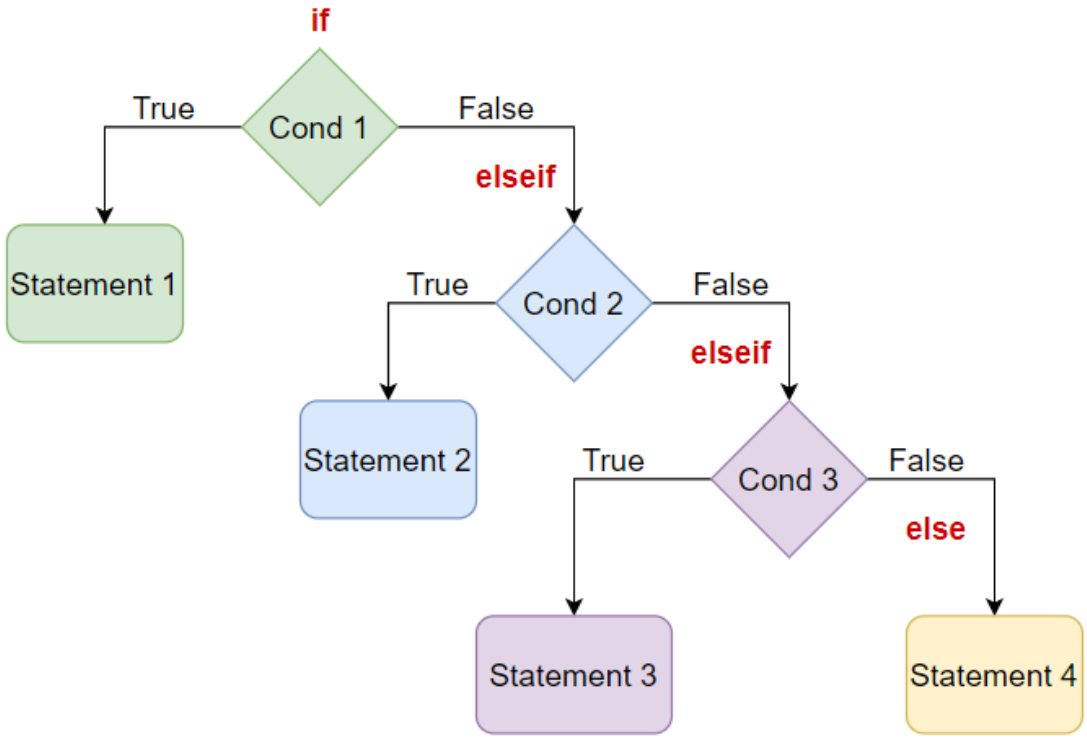
```
1.  if  (??)
2.          return True
3.  else
4.          return False
5.  endif
```

Note: P is already a Boolean variable, it can be directly used as the conditional statement here. With if always use condition having result true or false.

If-statement

In general (more than two cases), we can use if-elseif-else structure.

```
if (condition 1)
        statement 1
elseif (condition 2)
        statement 2
elseif (condition 3)
        statement 3
else
        statement 4
endif
```



Example

Design an algorithm for determining the number of distinct real roots of quadratic equation

$$ax^2 + bx + c = 0.$$

$$x = \frac{-b \pm \sqrt{b^2 - 4ac}}{2a} \qquad \textit{quadratic formula}$$

$$b^2 - 4ac \qquad \textit{discriminant}$$

Think:

- what should be the inputs/outputs?
- How to switch to different cases in using the discriminant?

Solution

Algorithm: rootNum(a,b,c)
Requires: Three real numbers a, b, c ; a is not equal to 0.
Returns: Numbers of real roots of quadratic equation $ax^2 + bx + c = 0$

```
1.  let  D = b*b-4*a*c  // assign the discriminant value to variable D
2.  if  D > 0
3.        return 2  // two distinct real roots
4.  elseif  D == 0  // double equal is to check equality
5.        return 1  // one distinct real roots (two identical real roots)
6.  else
7.        return 0 //  no real roots
8.  endif
```

Practice:Simple If-Else Questions

1. **Temperature Check:**
   - Design an algorithm that takes the current temperature as input. If the temperature is greater than 30°C, return 1 (representing "hot"), otherwise return 0 (representing "cold").
2. **Even or Odd:**
   - Create an algorithm that checks whether a given number is even or odd. If the number is even, return 1 (for "even"), otherwise return 0 (for "odd").
3. **Voting Eligibility:**
   - Develop an algorithm that determines if a person is eligible to vote based on their age. If the age is 18 or above, return 1 (for "eligible"), otherwise return 0 (for "not eligible").
4. **Positive, Negative, or Zero:**
   - Design an algorithm that takes an integer as input and returns:
     - 1 if the number is positive,
     - -1 if it is negative,
     - 0 if it is zero.

Practice: Nested If-Else Questions

1. **Grading System:**
   - Create an algorithm that takes a student's score as input and returns:
     - 1 if the score is 90 or above (representing "Grade A"),
     - 2 if the score is between 80 and 89 (representing "Grade B"),
     - 3 if the score is between 70 and 79 (representing "Grade C"),
     - 4 if the score is between 60 and 69 (representing "Grade D"),
     - 5 if the score is below 60 (representing "Grade F").
2. **Number Classification:**
   - Write an algorithm that takes an integer as input and returns:
     - 1 if it is a positive even number,
     - 2 if it is a positive odd number,
     - -1 if it is a negative even number,
     - -2 if it is a negative odd number,
     - 0 if the number is zero.
3. **Triangle Type:**
   - Design an algorithm that takes the lengths of three sides of a triangle as input and returns:
     - 1 if all sides are equal (equilateral),
     - 2 if exactly two sides are equal (isosceles),
     - 3 if no sides are equal (scalene).

Keywords

- Requires
- Returns
- Assign
- Let
- If else
- Nested if else
- finite
- Mod
- Boolean variable
- Trace
- Negation

Reminder

- Complete Problem Sheet 1 by the end of this week.

- If you have questions, please drop us emails and visit us during the office hours (information is on Moodle).