Data Structures and Algorithms

Introduction

Data structures and Algorithms

- Lecturer: Phạm Bảo Sơn.
- Email: sonpb@vnu.edu.vn
- Consultations: via email or after lecture.
- Course website:
  - courses.uet.vnu.edu.vn
  - uet.vnu.edu.vn/~sonpb/DSA

Aims

- To introduce the basic data structures and algorithms
- To develop skills in the design and analysis of algorithms and data structures
- Focus on concepts using pseudocode. Java or C/C++ for labs and assignments

Outcomes

- Understand the basic data structures and algorithms
- Be able to analyze the complexities of a program
- Be able to *select* appropriate data structures and *design* algorithms for applications at hand

Outcomes

- Become a better programmer
- Sharpen your mathematical and analytical skills
- Start “thinking algorithmically”

“I will, in fact, claim that the difference between a bad programmer and a good one is whether he considers his code or his data structures more important. Bad programmers worry about the code. Good programmers worry about data structures and their relationships.”

— Linus Torvalds (creator of Linux)

Topics

- Analysis of algorithms
- Linked lists, stacks and queues
- Recursion
- Trees and binary search trees
- Maps and dictionaries
- Priority Queue and Heaps
- Sorting
- Text Processing
- Graphs traversal
- Shortest path, minimum spanning trees

Assessment

- Class attendance, participation: 10%
- Labs: 20%
- Midterm exam/Programming Assignment: 10%
- Final Exam: 60%

Readings

- Textbook:
  - Michael T. Goodrich and Roberto Tamassia. *Data structures and Algorithms in Java (6th edition).*
  - Đinh Mạnh Tường: *Cấu trúc dữ liệu và giải thuật*

Overview

Data Structure + Algorithm = Program

- Data structure: arrangement of data.
- Algorithm:
  - A well-defined set of rules for solving a computational problem.
  - Data manipulation

Program

- A Java/C++ has Variables and Functions:
  - Variables <--> Data Structures
  - Functions ←> Algorithms

Example Program

```
#include <iostream>
using namespace std;
int main() {
        int i, j;
        cin >> i >> j;
        cout << i+j << endl;
}
```

Questions

- Will this program correctly add two integers always?
  - If not, under what conditions is it guaranteed to work correctly?
- What are other possible built-in data structures that could have been used in this program?
- Write a program for adding two integers that is guaranteed to always work correctly.
  - In this case, you may need to use your own data structures rather than built-in ones.

Example Program

- Read a triangle and output its area

```
#include <iostream>
using namespace std;
struct point {
        float x,y;
};
struct triangle{
 point p[3];
};
```

```
void read_point(point &p) {
    cin >> p.x >> p.y;
}
void read_triangle(triangle &t) {
    read_point(t.p[0]);
    read_point(t.p[1]);
    read_point(t.p[2]);
}
float area(triangle t) {
    return 0.5*abs((t.p[1].x-t.p[0].x)*(t.p[2].y- t.p[0].y)-(t.p[2].x–
t.p[0].x)*(t.p[1].y-t.p[0].y));
}
int main() {
    triangle t;
    read_triangle(t);
    cout << area(t) << endl;
}
```

Data Structures

- This program uses several data structures
- No built-in data structure for triangles
- C++ gives ways of defining our own data structures for different objects.
- The built-in data structure ***float*** is used to represent the value of a coordinate
- A point represented by its x and y coordinates
- A triangle represented by an array of 3 points

Algorithm

- The program uses 4 functions
- A function to read a point and another to read a triangle
- A function that computes the area of a triangle
- The main function that reads a triangle and outputs its area
- No built-in functions available for these
- Need to define our own functions or algorithms for these

Data structure?

1. What is a data structure?

   A data structure is a particular way of storing and organizing data so that they can be used efficiently.

2. Efficiency?
   - Correctness
   - Memory usage
   - Search and retrieval
   - modification, insertion/deletion
   - Simple and understandable/implementable
3. Standard data structures
   - Struct/Class
   - Array
   - List
   - Tree
   - Hash table
   - Set
   - Dictionary

Algorithms?

1. What is an algorithm?

   An algorithm is a step by step procedure to solve a problem. An algorithm can be described by natural languages (English, Vietnamese...) or programming languages (C++, Java, Python…)

   Example: How to cook rice?

   - Step 1: Get rice
   - Step 2: Combine rice and water
   - Step 3: Boil the rice
   - Step 4: Check the rice

2. A good algorithm?
   - Correctness
   - Fast
   - Memory
   - Simple, understandable/implementable

Why Study DSA

- Important for all other branches of computer science
- Plays a key role in modern technological innovation
- Challenging and Fun

Example 1: Sort a list

Problem: A class has N students. Sort these students in a decreasing order by their average scores.

#	Name	Average score
1	Tuấn	22
2	Thăng	29
3	Vinh	26
4	Ánh	27

#	Name	Average score
1	Thăng	29
2	Ánh	27
3	Vinh	26
4	Tuấn	22

Bubble sort example

Algorithm: Iterate through the list, if there are two consecutive students with the wrong orders, swap them. Repeat this process until we get the correct list.

Step 0

1. (Tuấn, 22)
2. (Thăng , 29)
3. (Vinh, 26)
4. (Ánh , 27)

Step 1

1. (Thăng, 29)
2. (Tuấn, 22)
3. (Vinh, 26)
4. (Ánh, 27)

Step 2

1. (Thăng, 29)
2. (Vinh, 26)
3. (Tuấn, 22)
4. (Ánh, 27)

Step 3

1. (Thăng, 29)
2. (Vinh, 26)
3. (Ánh, 27)
4. (Tuấn, 22)

Step 4

1. (Thăng, 29)
2. (Ánh, 27)
3. (Vinh, 26)
4. (Tuấn, 22)

Bubble sort in a programming language

```
Function bubbleSort (studentList) {
    swapped := false;
    do
        swapped := false;
        for each  i  = 0  to  N-2 do
            if studentList[i].averageScore < StutentList[i + 1].averageScore  {
                swap (studentList[i], studentList[i+1]);
                swapped := true;
            }
    while (swapped  = true)
}
```

Example 2: Sort a list of websites (Google search)

Google has a list *N* websites. Website *x* has the priority value *f*(*x*). Sort the websites decreasingly by their *f(x).*

Question: Can we use bubble sort?

Answer: Yes, but not efficient

Example 3: Phone dictionary

Design a program to manage a phone dictionary with main operations:

1. Check if a phone number is in the phone dictionary?
2. Insert a new phone number
3. Delete a phone number
4. Edit a phone number

Example 4: Integer Multiplication

- Input: two n-digits numbers x and y
- Output: the product x*y
- Primitive operation: add or multiply 2 single-digit numbers

Integer Multiplication

"*Perhaps the most important principle for the good algorithm designer is to refuse to be content*."

Aho, Hopcroft, and Ullman, *The Design and Analysis of Computer Algorithms*, 1974

•Can we do better?

Karatsuba Multiplication

- x = ab, y = cd
- Compute a*c (1)
- Compute b*d (2)
- Compute (a+b)(c+d) (3)
- Compute (3) – (2) – (1) = (4)
- Product: (1)*10000 + (2) + (4)*100
- Recursive Algorithm

What is a good program?

1. Correctness
2. Effective
3. Simple/Understandable
4. Easy to find bugs
5. Easy to change and upgrade/maintain

"Algorithm + Data structure = Program"

N. Wirth