ĐẠI HỌC ĐÔNG Á
Tạo dựng con đường thành công

Introduction to Data Structures & Algorithms

Instructor introduction

Name: Ths. Luong Anh Tuan

- Email: tuanla@donga.edu.vn
- Skype: tuanlase02874
- Mobile: 0983095376

What is data structure?

In computer science, simply speaking a ***data structure is a way of storing data*** in a computer so that ***it can be used efficiently***. A data structure is a way of organizing data that considers ***not only the items stored, but also their relationship*** to each other. Advance knowledge about the relationship between data items allows designing of efficient algorithms for the manipulation of data. Often a ***carefully chosen data structure will allow a more efficient algorithm*** to be used. The choice of the data structure often begins from the choice of an abstract data structure. A well-designed data structure allows a variety of critical operations to be performed on using as little resources, both execution time and memory space, as possible. Different kinds of data structures are suited to different kinds of applications, and some are highly specialized to certain tasks. For example, B-trees are particularly well-suited for implementation of databases, while compiler implementations usually use hash tables to look up identifiers.

Why data structure is important in computer science?

Data structures are used in almost every program or software system. Data structures provide a means to manage huge amounts of data efficiently, such as large databases and internet indexing services. Usually, efficient data structures are a key to designing efficient algorithms. Some formal design methods and programming languages emphasize data structures, rather than algorithms, as the key organizing factor in software design. **Niklaus Emil Wirth (born February 15, 1934) is a Swiss computer scientist said:**

Algorithms + Data Structures = Programs

Basic types of data structure

In programming, the term *data structure* refers to a scheme for organizing related pieces of information. The basic types of data structures include:

- **files**
- **lists**
- **arrays**
- records
- trees
- tables

Each of these basic structures has many variations and allows different operations to be performed on the data.

What is algorithm?

A formula or set of steps for solving a particular problem. To be an algorithm, a set of rules must be unambiguous and have a clear stopping point. Algorithms can be expressed in any language, from natural languages like English or French to programming languages like FORTRAN.

- We use algorithms every day. For example, a recipe for baking a cake is an algorithm.
- Most programs, with the exception of some artificial intelligence applications, consist of algorithms.
- Inventing elegant algorithms -- algorithms that are simple and require the fewest steps possible -- is one of the principal challenges in programming.

The word “algorithm” derives from the name of the mathematician, Abu Ja'far Mohammed ibn-Musa al-Khwarizmi, who was an Arab, probably from what is now Southern Uzbekistan, who taught at the Caliph's Palace of Wisdom in Baghdad in the 9th Century, and is one the most important mathematicians in history. Through Latin translations of his work, al-Khwarizmi introduced to Europe the Hindu-Arabic base 10 numerals - the use of which came to be known as algorithm in English.

Algorithm example

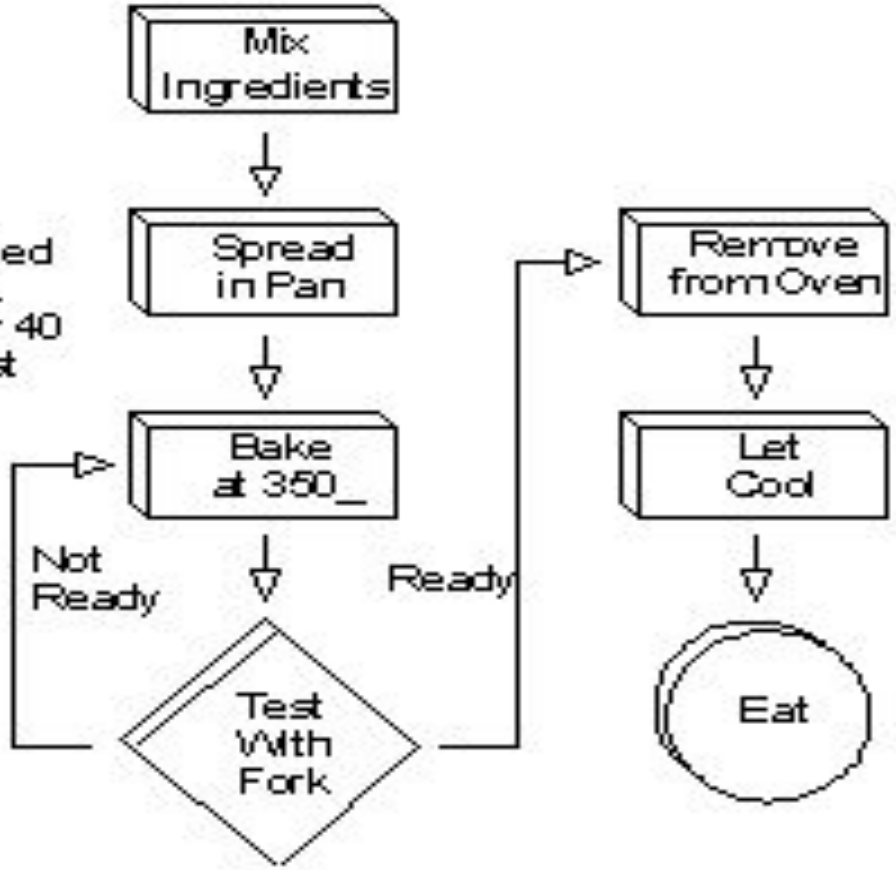
Recipe
CHOCOLATE CAKE

4 oz. chocolate	3 eggs
1 cup butter	1 tsp. vanilla
2 cups sugar	1 cup flour

Melt chocolate and butter. Stir sugar into melted chocolate. Stir in eggs and vanilla. Mix in flour. Spread mix in greased pan. Bake at 350_ for 40 minutes or until inserted fork comes out almost clean. Cool in pan before eating.

Program Code

```
Declare variables:
    chocolate   eggs        mix
    butter      vanilla
    sugar       flour
mix = melted ((4*chocolate) + butter)
mix = stir (mix + (2*sugar))
mix = stir (mix + (3*eggs) + vanilla)
mix = mix + flour
spread (mix)
While not clean (fork)
bake (mix, 350)
```



What are the properties of an algorithm?

1. **Input** - an algorithm accepts zero or more inputs
2. **Output** - it produces at least one output.
3. **Finiteness** - an algorithm terminates after a finite numbers of steps.
4. **Definiteness** - each step in algorithm is unambiguous. This means that the action specified by the step cannot be interpreted (explain the meaning of) in multiple ways & can be performed without any confusion.
5. **Effectiveness** - it consists of basic instructions that are realizable. This means that the instructions can be performed by using the given inputs in a finite amount of time.
6. **Generality** - it must work for a general set of inputs.

How to represent algorithms?

- Use natural languages
  - too verbose
  - too "context-sensitive"- relies on experience of reader
- Use formal programming languages
  - too low level
  - requires us to deal with complicated syntax of programming language
- Pseudo-Code (alias programming language) - natural language constructs modeled to look like statements available in many programming languages.
- Flowchart (diagram) - A flowchart is a common type of chart, that represents an algorithm or process, showing the steps as boxes of various kinds, and their order by connecting these with arrows. Flowcharts are used in analyzing, designing, documenting or managing a process or program in various fields.

Algorithm representation examples

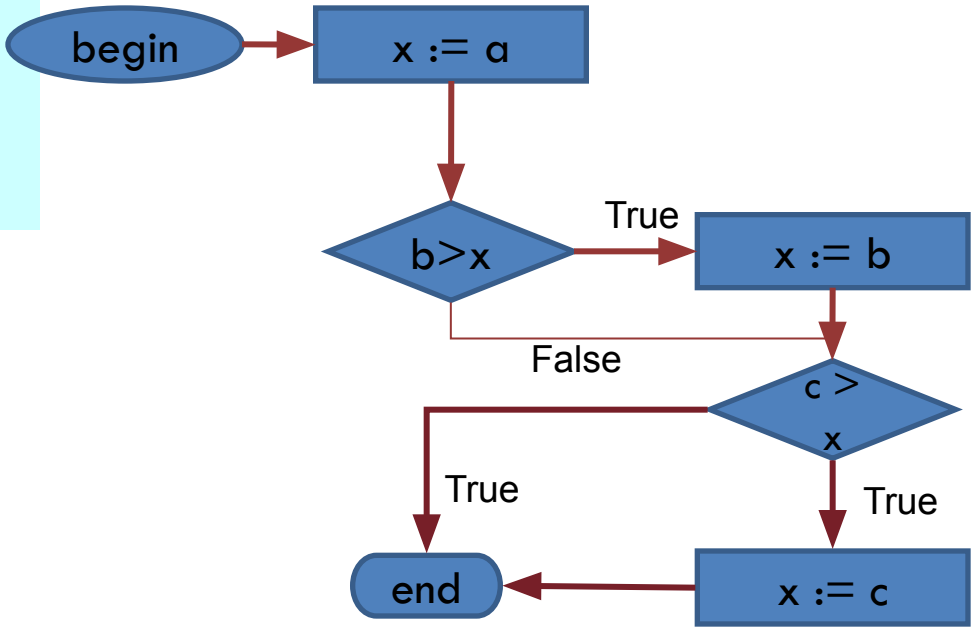
Problem: find the greatest of three numbers

Natural language:

find max(a,b,c)

- Assign x = a
- If b great than x then assign x=b
- If c great than x then assign x=c;
- Result: x => max(a,b,c)

Flowchart:



Pseudo-code:

```
function max(a,b,c)
Input: a, b, c
Output: max(a,b,c)
        x = a;
        if b > x then x = b;
        if c > x then x = c;
        return x;
```

Data Structures and Algorithms

- Most algorithms operate on data collections, so define
- Collection Abstract Data Type (ADT)
  - Methods
    - Constructor / Destructor
    - Add / Edit / Delete
    - Find
    - Sort
    - ….