DATA STRUCTURE & ALGORITHM

Introduction

Lecture 1

Instructor: Engr. Tayyab Abdul Hannan
Email: tayyab.abdulhannan@gmail.com

Outline

- **Prerequisites:** Object Oriented Paradigms (Basic I/O functions, Conditional Statement, Loop
- Introduction to data structures ; **Function, Pointers** Arrays, Stacks, Queues, Priority Queues, Linked Lists, Trees, and Graphs. Recursion, sorting and searching algorithms, Hashing, Storage and retrieval properties and techniques for the various data structures. Algorithm Complexity, Polynomial and Intractable Algorithms, Classes of Efficient Algorithms, Divide and Conquer, Dynamic, Greedy

Reference Material:

- *Data Abstraction and Problem Solving with C++, 2nd ed, Frank M. Carrano, Paul Helman, Robert Veroff, Addison-Wesley, 1998.*
- *Data Structures and Algorithms* (SAMS teach yourself), Lafore, Sams Publishing, 1999.
- *Fundamentals of Data Structures in C++*, Horowitz, Sahni, and Mehta, Computer Science Press, 1995.
- *Data Structures in JAVA*, Standish, Addison Wesley, 2000

- **WHY STUDY DATA STRUCTURE & ALGORITHMS**

Introduction

- Let’s discuss why we need data structures and what sort of problems can be solved with their use.
- Data structures help us to organize the data in the computer, resulting in more efficient programs.
- An efficient program executes faster and helps minimize the usage of resources like memory, disk.
- Computers are getting more powerful with the passage of time with the increase in CPU speed in GHz, availability of faster network and the maximization of disk space

Introduction

- Therefore people have started solving more and more complex problems.
- As computer applications are becoming complex, so there is need for more resources.
- This does not mean that we should buy a new computer to make the application execute faster.
- Our effort should be to ensue that the solution is achieved with the help of programming, data structures and algorithm.
- What does organizing the data mean? It means that the data should be arranged in a way that it is easily accessible. The data is inside the computer and we want to see it

Introduction

- We may also perform some calculations on it.
- Suppose the data contains some numbers and the programmer wants to calculate the average, standard deviation etc.
- May be we have a list of names and want to search a particular name in it.
- To solve such problems, data structures and algorithm are used.
- Sometimes you may realize that the application is too slow and taking more time.
- There are chances that it may be due to the data structure used, not due to the CPU speed and memory.

Introduction

- A solution is said to be efficient if it solves the problem within its resource constraints.
- What does it mean? In the computer, we have hard disk, memory and other hardware. Secondly we have time.
- Suppose you have some program that solves the problem but takes two months. It will be of no use.
- Usually, you don’t have this much time and cannot wait for two months.
- Suppose the data is too huge to be stored in disk. Here we have also the problem of resources.
- This means that we have to write programs considering the resources to achieve some solution as soon as
- possible.

Introduction

- There is always cost associated with these resources.
- We may need a faster and better CPU which can be purchased. Sometimes, we may need to buy memory.
- As long as data structures and programs are concerned, you have to invest your own time for this.
- While working in a company, you will be paid for this.
- All these requirements including computer, your time and computer time will decide that the solution you have provided is suitable or not.
- If its advantages are not obtained, then either program or computer is not good.

Introduction

- So the purchase of a faster computer, while studying this course, does not necessarily help us in the resolution of the problem.
- In the course of “Computer Architecture” you will see how the more efficient solutions can be prepared with the hardware.
- In this course, we will use the software i.e. data structures, algorithms and the recipes through which the computer problems may be resolved with a faster solution.

Selecting a Data Structure

- How can we select the data structure needed to solve a problem?
- You have already studied where to use array and the size of array and when and where to use the pointers etc.
- First of all, we have to analyze the problem to determine the resource constraints that a solution must meet.
- Suppose, the data is so huge i.e. in Giga bytes (GBs) while the disc space available with us is just 200 Mega bytes.
- This problem can not be solved with programming.
- Rather, we will have to buy a new disk.

Selecting a Data Structure

- Suppose you have to insert the data in the computer or database and have to search some data item.
- Let's take the example of telephone directory. Suppose there are eight million names in the directory.
- Now someone asks you about the name of some particular person.
- You want that this query should be answered as soon as possible.
- You may add or delete some data. It will be advisable to consider all these operations when you select some data structure.

Selecting a Data Structure

- It is necessary to determine the basic operations that must be supported.
- Finally select the data structure that meets these requirements the maximum.
- Without, sufficient experience, it will be difficult to determine which one is the best data structure.
- We can get the help from internet, books or from someone whom you know for already getting the problems solved.
- We may find a similar example and try to use it.

Selecting a Data Structure

- Now you have selected the data structure.
- Suppose a programmer has inserted some data and wants to insert more data.
- This data will be inserted in the beginning of the existing data, or in the middle or in the end of the data.
- Let’s talk about the arrays and suppose you have an array of size hundred.
- Data may be lying in the first fifty locations of this array.
- Now you have to insert data in the start of this array.
- What will you do?

Selecting a Data Structure

- You have to move the existing data (fifty locations) to the right so that we get space to insert new data.
- Other way round, there is no space in the start. Suppose you have to insert the data at 25th location.
- For this purpose, it is better to move the data from 26th to 50th locations; otherwise we will not have space to insert this new data at 25th location.
- Now we have to see whether the data can be deleted or not. Suppose you are asked to delete the data at 27th position. How can we do that?
- What will we do with the space created at 27th position?

Selecting a Data Structure

- Thirdly, is all the data processed in some well-defined order or random access allowed?
- Again take the example of arrays. We can get the data from 0th position and traverse the array till its 50th position.
- Suppose we want to get the data, at first from 50th location and then from 13th .
- It means that there is no order or sequence.
- We want to access the data randomly.
- Random access means that we can’t say what will be the next position to get the data or insert the data.

Data Structure Philosophy

- Let’s talk about the philosophy of data structure.
- Each data structure has costs and benefits.
- Any data structure used in your program will have some benefits.
- For this, you have to pay price.
- That can be computer resources or the time.
- Also keep in mind that you are solving this problem for some client.
- If the program is not efficient, the client will not buy it.

Data Structure Philosophy

- In rare cases, a data structure may be better than another one in all situations.
- It means that you may think that the array is good enough for all the problems.
- Yet this is not necessary.
- In different situations, different data structures will be suitable.
- Sometimes you will realize that two different data structures are suitable for the problem.
- In such a case, you have to choose the one that is more appropriate.

Data Structure Philosophy

- There are three basic things associated with data structures.
- A data structure requires:
- – space for each data item it stores
- – time to perform each basic operation
- – programming effort

Goals of this Course

- At times, you may have two suitable data structures for some problem.
- These can be tried one by one to adjudge which one is better one.
- How can you decide which data structure is better than other.
- Firstly, a programmer can do it by writing two programs using different data structure while solving the same problem.
- Now execute both data structures. One gives the result before the other. The data structure that gives results first is better than the other one.

Goals of this Course

- But sometimes, the data grows too large in the problem.
- Suppose we want to solve some problem having names and the data of names grows to10 lakhs (one million).
- Now when you run both programs, the second program runs faster.
- What does it mean? Is the data structure used in program one not correct?
- This is not true. The size of the data, being manipulated in the program can grow or shrink.
- You will also see that some data structures are good for
- small data while the others may suit to huge data.

Objective

- Data structures is concerned with the representation and manipulation of data
- All programs manipulate data So, all programs represent data in some way
- Data manipulation requires an algorithm

  algorithms + data structures = programs
- There are many ways to represent data and manipulate them.
- our objective is to study the most efficient combinations of data representation and algorithms which manipulate them.

Data Structure

- Data Structure is a systematic way of organizing and accessing data.
- Data Structure is a logical or mathematical model of a particular organization of Data.
- A construct that can be defined within a programming language to store a collection of data.
- There are several data structures, while choosing your Data Structure.
- Analyze your data, the data must keep the real world relationship in that structure keeping in view the space requirements.

Basic data structures

- Array: Fixed-size/ Variable
- Linked-list: Variable-size
- Stack: Add to top and remove from top
- Queue: Add to back and remove from front
- Priority queue: Add anywhere, remove the highest priority
- Hash tables: Unordered lists which use a ‘hash function’ to insert and search
- Tree: A branching structure with no loops
- Graph: A more general branching structure, with less strict connection conditions than for a tree

Types of Data Structure

- Data Structures can be classified as
- 1. Linear Data Structures: elements form a sequence or linear list, relationship between elements is linear
- 2. Non Linear Data Structures: such as trees and graphs, relationship between elements is non-linear
- Linear Data Structures can be stored in memory as
- 1. Arrays: linear relationship between the elements is represented by means of sequential memory locations
- 2. Linked List: linear relationship between the elements is represented by means of pointers or links

Kinds of operations on Data Structures

OPERATIONS ON LINEAR STRUCTURES:

- Traversal: Processing each element in the list
- Search: Finding the location of the element with a given value
- Insertion: Adding a new element to the list
- Deletion: Removing an element from the list
- Sorting: Arranging the element in some order
- Merging: Combining two lists into a single list
- We chose a particular linear structure in a given situation depending on the relative frequency of these different operations

Origin of word: *Algorithm*

- The word *Algorithm* comes from the name of the muslim author *Abu Ja'far Mohammad ibn Musa al-Khowarizmi*.
- He was born in the eighth century at Khwarizm (Kheva), a town south of river Oxus in present Uzbekistan.
- Uzbekistan, a Muslim country for over a thousand years, was taken over by the Russians in 1873.
- His year of birth is not known exactly.
- Al-Khwarizmi parents migrated to a place south of Baghdad when he was a child.
- It has been established from his contributions that he flourished under Khalifah Al-Mamun at Baghdad during 813 to 833 C.E. Al-Khwarizmi died around 840 C.E.

Origin of word: *Algorithm*

- Much of al-Khwarizmi's work was written in a book titled *al Kitab al-mukhatasar fi hisab al-jabr wa'l-muqabalah* (The Outstanding Book on Calculation by Completion and Balancing).
- It is from the titles of these writings and his name that the words *algebra* and *algorithm* are derived.
- As a result of his work, al-Khwarizmi is regarded as the most outstanding mathematician of his time

Algorithm: Informal Definition

- An algorithm is any well-defined computational procedure that takes some values, or set of values, as input and produces some value, or set of values, as output.
- An algorithm is thus a sequence of computational steps that transform the input into output.
- A good understanding of algorithms is essential for a good understanding of the most basic element of computer science: programming.
- Unlike a program, an algorithm is a mathematical entity, which is independent of a specific programming language, machine, or compiler.

WRITING AN ALGORITHM

- Algorithm is a step-by-step procedure, which defines a set of instructions to be executed in a certain order to get the desired output.
- Algorithms are generally created independent of underlying languages, i.e. an algorithm can be implemented in more than one programming language.

How to Write an Algorithm?

- There are no well-defined standards for writing algorithms.
- Rather, it is problem and resource dependent.
- Algorithms are never written to support a particular programming code.
- All programming languages share basic code constructs like loops (do, for, while), flow-control (if-else), etc. These common constructs can be used to write an algorithm.
- We write algorithms in a step-by-step manner, but it is not always the case. Algorithm writing is a process and is executed after the problem domain is well-defined. That is, we should know the problem domain, for which we are designing a solution.

How to Write an Algorithm?

- Example
- **Problem** − Design an algorithm to add two numbers and display the result.
- **Step 1** − START
- **Step 2** − declare three integers **a**, **b** & **c**
- **Step 3** − define values of **a** & **b**
- **Step 4** − add values of **a** & **b**
- **Step 5** − store output of step 4 to **c**
- **Step 6** − print **c**
- **Step 7** − STOP

How to Write an Algorithm?

- Example
- **Problem** − Design an algorithm to add two numbers and display the result.
- **Step 1** − START ADD
- **Step 2** − get values of **a** & **b**
- **Step 3** − c ← a + b
- **Step 4** − display c
- **Step 5** − STOP
- Writing **step numbers**, is optional.
- We design an algorithm to get a solution of a given problem.
- A problem can be solved in more than one ways.

How to Write an Algorithm?

- Linear Search (Array A, Value x)
- Step 1: Set i to 1
- Step 2: if i > n then go to step 7
- Step 3: if A[i] = x then go to step 6
- Step 4: Set i to i + 1
- Step 5: Go to Step 2
- Step 6: Print Element x Found at index i and go to step 8
  Step 7: Print element not found
- Step 8: Exit

How to Write an Algorithm?

- Linear Search (Array A, Value x)
- Step 1: Set i to 1
- Step 2: if i > n then go to step 7
- Step 3: if A[i] = x then go to step 6
- Step 4: Set i to i + 1
- Step 5: Go to Step 2
- Step 6: Print Element x Found at index i and go to step 8
  Step 7: Print element not found
- Step 8: Exit

How to Write an Algorithm?

- **Insertion Sort**
- **Step 1** − If it is the first element, it is already sorted. return 1;
- **Step 2** − Pick next element
- **Step 3** − Compare with all elements in the sorted sub-list **Step 4** − Shift all the elements in the sorted sub-list that is greater than the value to be sorted
- **Step 5** − Insert the value
- **Step 6** − Repeat until list is sorted