02-24-00108
Data Structures and Algorithms

Lecture 1: Course Introduction

Introducing the Instructors

Prof. Dr. Amr El Masry

Head of Computer Science Department

Faculty of Engineering

Alexandria University

Email: amr.a.elmasry@googlemail.com

Introducing the Instructors

Dr. Mervat Mikhail

Computer Science Department CSD

Assistant Professor of Engineering Math Department

Faculty of Engineering, Alexandria University

Email: mervat.mikhail80@gmail.com

Office hours: Wednesday 11:30 AM -12:00 PM

Thursday 11:30 AM -12:00 PM

Teaching Assistants

<u>Special Program</u>

1. *Eng. Rania Ismail Mohamed raniaismail2016@gmail.com*
2. *Eng. Fady Nabil Samy fadynabilyacoub@gmail.com*
3. *Eng. Heba Allah Tarek Fathy heba.tarek@rocketmail.com*

<u>General program</u>

1. *Eng. Ahmed Mohamed Shokry shokry.success@gmail.com*
2. *Eng. Samy Mamdouh Sami.mamdouh@gmail.com*
3. *Eng. Mohamed Sherif Hosny msherif449@gmail.com*

Course content

Lecture 01: Introduction - Big O notation

Lecture 02:Array & Pointers & linked list

Lecture 03:Stack & queue

Lecture 04: Hash tables

Lecture 05: Tree & Binary tree

Lecture 06: Tree & Binary tree &AVL tree (cont.)

Lecture 07: Heaps

Lecture 08: Sorting Techniques

Lecture 09: Sorting Techniques (cont.) + Searching

Lecture 10: Graph BFS+DFS

Lecture 11: MST

Lecture 12: Shortest Path Algorithm

Grade Breakdown

- **25%** Assignments & Projects
- **25%** Mid-term exam
- **50%** Final exam
- **5%** Bonus for participation in lecture

Resources

- Lecture slides
- Textbooks (Many!!)
- And More Online Materials

Communication Channels:

- Email
- Piazza Class for announcements, lectures, labs and sheets, course questions.

http://piazza.com/faculty_of_computers_and_data_science_alexandria_university/spring2020/022400108

access code : 02-24-00108

Introduction to Algorithms

Good Computer Program

A computer program is a series of instructions to carry out a particular task written in a language that a computer can understand.

The process of preparing and feeding the instructions into the computer for execution is referred as programming.

There are a number of features for a good program

- Run efficiently and correctly
- Have a user friendly interface
- Be easy to read and understand
- Be easy to debug
- Be easy to modify
- Be easy to maintain

Good Computer Program

Programs consists of two things: Algorithms and data structures

A Good Program is a combination of both algorithm and a data structure

An algorithm is a step by step recipe for solving an instance of a problem

A data structure represents the logical relationship that exists between individual elements of data to carry out certain tasks

A data structure defines a way of organizing all data items that consider not only the elements stored but also stores the relationship between the elements

Algorithm

To make a computer do anything, you have to write a computer program. To write a computer program, you have to tell the computer, step by step, exactly what you want it to do. The computer then "executes" the program, following each step mechanically, to accomplish the end goal.

When you are telling the computer *what* to do, you also get to choose *how* it's going to do it. That's where **computer algorithms** come in. The algorithm is the basic technique used to get the job done. Let's follow an example to help get an understanding of the algorithm concept.

Goal : you want to arrive your home

The taxi algorithm:

1. Go to the taxi stand.
2. Get in a taxi.
3. Give the driver my address.

The bus algorithm:

1. Outside baggage claim, catch bus number 70.
2. Transfer to bus 14 on Main Street.
3. Get off on Elm street.
4. Walk two blocks north to my house.

The rent-a-car algorithm:

1. Take the shuttle to the rental car place.
2. Rent a car.
3. Follow the directions to get to my house.

All three of these algorithms accomplish exactly the same goal, but each algorithm does it in completely different way.

Each algorithm also has a different cost and a different travel time.

Taking a taxi, for example, is probably the fastest way, but also the most expensive. Taking the bus is definitely less expensive, but a whole lot slower. <u>You choose the algorithm based on the circumstances.</u>

Algorithm Properties

An algorithm possesses the following properties:

- It must be correct.
- It must be composed of a series of concrete steps.
- There can be no ambiguity as to which step will be performed next.
- It must be composed of a finite number of steps.
- It must terminate.
- It takes zero or more inputs
- It should be efficient and flexible
- It should use less memory space as much as possible
- It results in one or more outputs

Efficiency of an algorithm

Writing efficient programs is what every programmer hopes to be able to do. But what kinds of programs are efficient? The question leads to the concept of generalization of programs.

Algorithms are programs in a general form. An algorithm is an idea upon which a program is built. An algorithm should meet three things:

It should be independent of the programming language in which the idea is realized

Every programmer having enough knowledge and experience should understand it

It should be applicable to inputs of all sizes

Algorithm Performance

Two areas are important for performance:

Space efficiency **-** the memory required, also called, space complexity

Time efficiency **-** the time required, also called time complexity

Space efficiency

Components of space/memory use:

1. Instruction space

Affected by: the compiler, compiler options, target computer (CPU)

2. Data space

Affected by: the data size/dynamically allocated memory, static program variables

3. Run-time stack space

Affected by: the compiler, run-time function calls and recursion, local variables, parameters

Time efficiency

The actual running time depends on many factors:

1. The speed of the computer: CPU.
2. The compiler, compiler options .
3. The quantity of data - ex. search a long list or short.
4. The actual data - ex. in the sequential search if the name is first or last.

Time Complexity

Worst Case: It is the longest time that an algorithm will use over all instances of size n for a given problem to produce a desired result.

Average Case: It is the average time(or average space) that the algorithm will use over all instances of size n for a given problem to produce a desired result. It depends on the probability distribution of instances of the problem.

Best Case: It is the shortest time (or least space) that the algorithm will use over all instances of size n for a given problem to produce a desired result.

Introduction to Data Structures

The Need for Data Structures

Data structures organize data

$\Rightarrow$ more efficient programs.

More powerful computers $\Rightarrow$ more complex applications.

More complex applications demand more calculations.

Complex computing tasks are unlike our everyday experience.

More typically, a *data structure* is meant to be an *organization for a collection of data items*.

Any organization for a collection of records can be searched, processed in any order, or modified.

The choice of data structure and algorithm can make the difference between a program running in a few seconds or many days. A data structure requires a certain amount of:

space for each data item it stores

time to perform a single basic operation

programming effort.

Selecting a Data Structure

Select a data structure as follows:

1. Analyze the problem to determine the resource constraints a solution must meet.
2. Determine the basic operations that must be supported. Quantify the resource constraints for each operation.
3. Select the data structure that best meets these requirements.

Data Structures

DS includes

- Logical or mathematical description of the structure and Implementation of the structure on a computer
- Quantitative analysis of the structure, which includes determining the amount of memory needed to store the structure and the time required to process the structure.

Classification of Data Structures

"*Data Structures* "deals with the study of how the data is organized in the memory, how efficiently the data can be retrieved and manipulated, and the possible ways in which different data items are logically related.

Types:

Primitive Data Structure: Ex. int,float,char

Non primitive Data Structures:
Ex.Arrays,Structures,stacks

Linear Data Structures: Ex.Stacks,queues,linked list

Non Linear Data Structures: Ex.Trees,Graphs.

Asymptotic Notation

How we measure the complexity ?

Asymptotic Complexity

- The notations we use to describe the asymptotic running time of an algorithm are defined in terms of functions whose domains are the set of natural numbers N={0,1,2,3,…….}

- Running time of an algorithm as a function of input size *n* **for large *n***.
- Expressed using only the **highest-order term** in the expression for the exact running time.
  - Instead of exact running time, say $\Theta(n^2)$.
- Written using ***Asymptotic Notation***.

Θ, O, Ω, o, ω

When we use asymptotic notation to apply to the running time of an algorithm, we need to understand which running time we mean. Sometimes we are interested in the worst-case running time. Often, however, we wish to characterize the running time no matter what the input.

Big O-notation (Asymptotic upper bound)

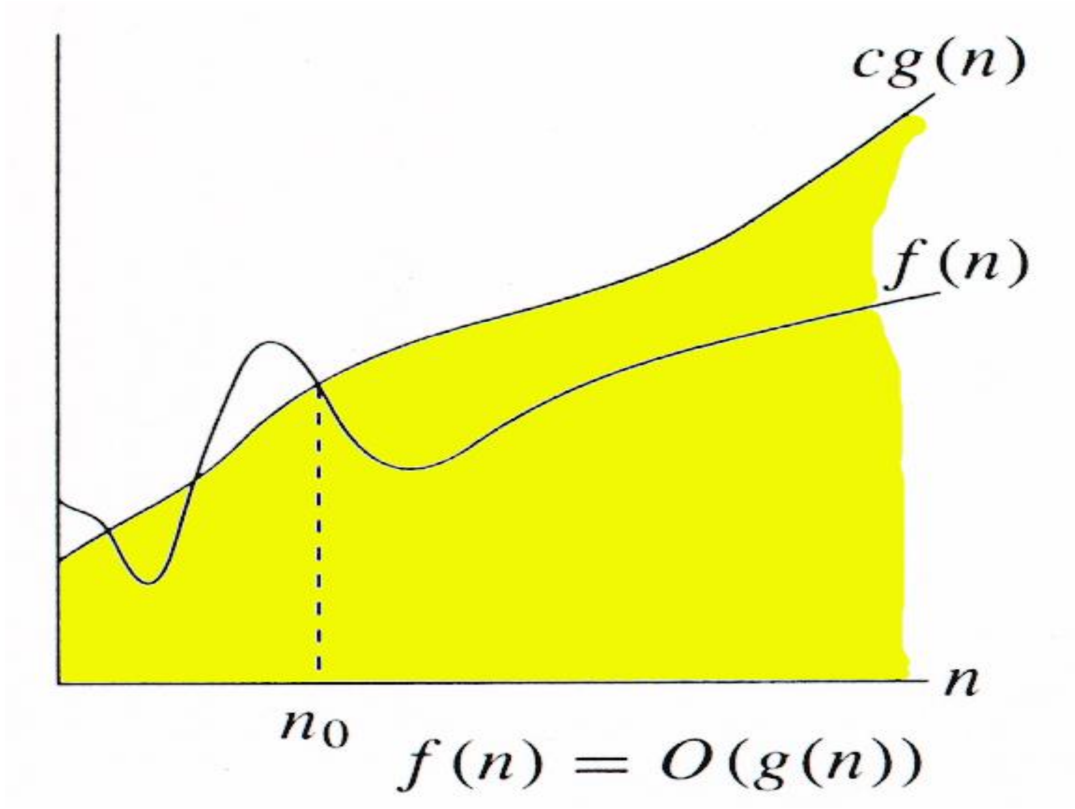
Given two nonnegative functions *f* and *g*, ***f* = *O*(*g*)** if and only if *g* asymptotically dominates *f*

$O(g(n)) = \{f(n)$:

$\exists$ **positive constants *c* and n_0, such that** $\forall n \geq n_0$,

we have $0 \leq f(n) \leq cg(n)$ $\}$

***g*(*n*) is an *asymptotic upper bound* for *f*(*n*).**



Examples

1. Is $4n^2+16n+2 = O(n^4)$?
2. Is $an+b = O(n^2)$? For any a,b ?
3. Is $3n^3 = O(n^4)$?
4. Is $3n^2+25 = O(n^2)$?
5. Is $2^{n-1} = O(2^n)$?
6. Is $2^{2n} = O(2^n)$?

Hint : Solutions are in tab Notes pdf