CS 311: Algorithm Design and Analysis

Introduction

Asymptotic notation

The Course

- Purpose: a rigorous introduction to the design and analysis of algorithms
  - Not a lab or programming course
  - Not a math course, either
- Textbook: *Introduction to Algorithms*, Cormen, Leiserson, Rivest, Stein 3rd edition
  - An excellent reference you should own

The Course

- Instructor: Majid Ahmed Askar
  - *Majid.askar@aun.edu.eg*
  - Office: Administration bulding room 305
  - Office hours: TBD
- TA: Eng. Aya Mahmoud
  - Office hours and location TBA

The Course

- Grading policy:
  - Class work: 25%
  - Midterm Exam : 25%
  - Final: 50%

The Course

- Prerequisites:
  - CS 211 Data Structures and Algorithms

The Course

- Upon completing this course the student will have learned, through appropriate classroom and laboratory experiences, the following.
  - The main classic algorithms in various domains.
  - Techniques for designing efficient algorithms.
  - Applying the algorithms and design techniques to solve problems.
  - Having a sense of the complexities of various problems in different domains.

Asymptotic Performance

- In this course, we care most about *asymptotic performance*
  - How does the algorithm behave as the problem size gets very large?
    - Running time
    - Memory/storage requirements

Asymptotic Notation

- By now you should have an intuitive feel for asymptotic (big-O) notation:
  - *What does $O(n)$ running time mean? $O(n^2)$? $O(n \lg n)$?*
  - *How does asymptotic running time relate to asymptotic memory usage?*
- Our first task is to define this notation more formally and completely

Analysis of Algorithms

- Analysis is performed with respect to a computational model
- We will usually use a generic uniprocessor random-access machine (RAM)
  - All memory equally expensive to access
  - No concurrent operations
  - All reasonable instructions take unit time
    - Except, of course, function calls
  - Constant word size
    - Unless we are explicitly manipulating bits

Input Size

- Time and space complexity
  - This is generally a function of the input size
    - E.g., sorting, multiplication
  - How we characterize input size depends:
    - Sorting: number of input items
    - Multiplication: total number of bits
    - Graph algorithms: number of nodes & edges
    - Etc

Running Time

- Number of primitive steps that are executed
  - Except for time of executing a function call most statements roughly require the same amount of time
    - y = m * x + b
    - c = 5 / 9 * (t - 32)
    - z = f(x) + g(y)
- We can be more exact if need be

Analysis

- Worst case
  - Provides an upper bound on running time
  - An absolute guarantee
- Average case
  - Provides the expected running time
  - Very useful, but treat with care: what is “average”?
    - Random (equally likely) inputs
    - Real-life inputs

The End