Advanced Algorithmics

Description

The course shows that asserting that an algorithm will complete with the correct result is not enough: it also must give the result in a “reasonable” amount of time.

After reviewing a set of mathematical tools, the students will be taught how to describe an algorithm in an implementation-language-independent way, then to evaluate its time complexity

Learning Objectives and Outcomes

- design an algorithm in a language-independent way
- evaluate the time needed for a given algorithm to complete
- decide whether it is worth coding this algorithm or find another, more performant

Course Schedule and Contents

Session#1

- Why bother with writing an algorithm before coding?
- Example: searching in an (un)sorted array -> notion of complexity
- Mathematical tools to evaluate complexity:
  - limit of an increasing sequence
  - Landau notation
  - Properties of O, o and Θ
  - Classes of equivalence for Θ
- First examples of simple iterative algorithms: importance of parameter and unit operations definition

Session#2

- What about recursive algorithms?
  - notion of recurrence equation
  - Examples of main classes of complexity, from log(N) to exp(N)
  - Simple example of "divide & conquer" : MergeSort

- general recurrence for divide & conquer (master theorem)
- Introduction to Abstract Data Types (ADT): the TUOPA formalism
- importance of constructors
- simple examples : Boolean, Natural, Counter, Stack

Session#3
- Linear ADTs (Stack, Queue, list): basic definitions
- Fundamental linear complexity for most list operations

Session #4
- Non-linear operations on list (ex. Reverse)
- General trees: a solution to bypass the linear limit
  - basic operations and measures (size, depth, …)
  - classical tree exploration algorithms

Session #5
- Binary trees (BT): equivalence with general trees
- Proof by induction on BT main properties
- Binary search trees (BST): definition and basic operations
- height = log(size) for random BST
- importance of inorder traversal

Session #6
- Insertion and removal in BSTs
- using BSTs to efficiently sort data

Session #7
- AVL as an “always balanced” BST
- Introduction to hashing approach
- Comparison between trees and hash table to manage data

Grading

Written exam, all documents allowed (no electronic devices)

Policies

- Lecture notes are given at the end of each session
- Most exercises in class are excerpts from preceding exams: it is important to try to solve them!

Good Luck!