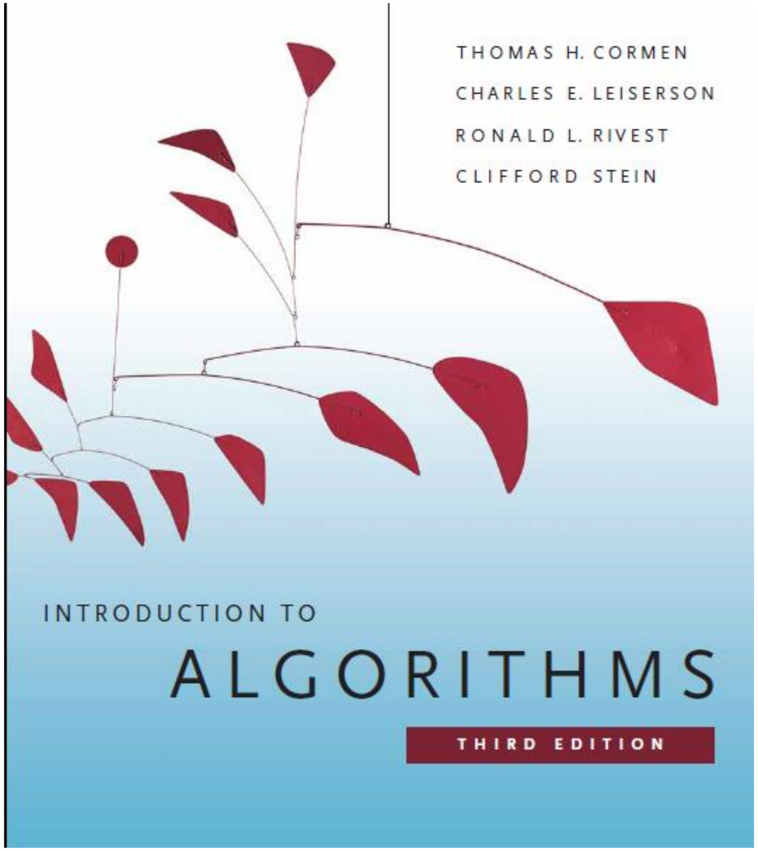
Design and Theory of Algorithms

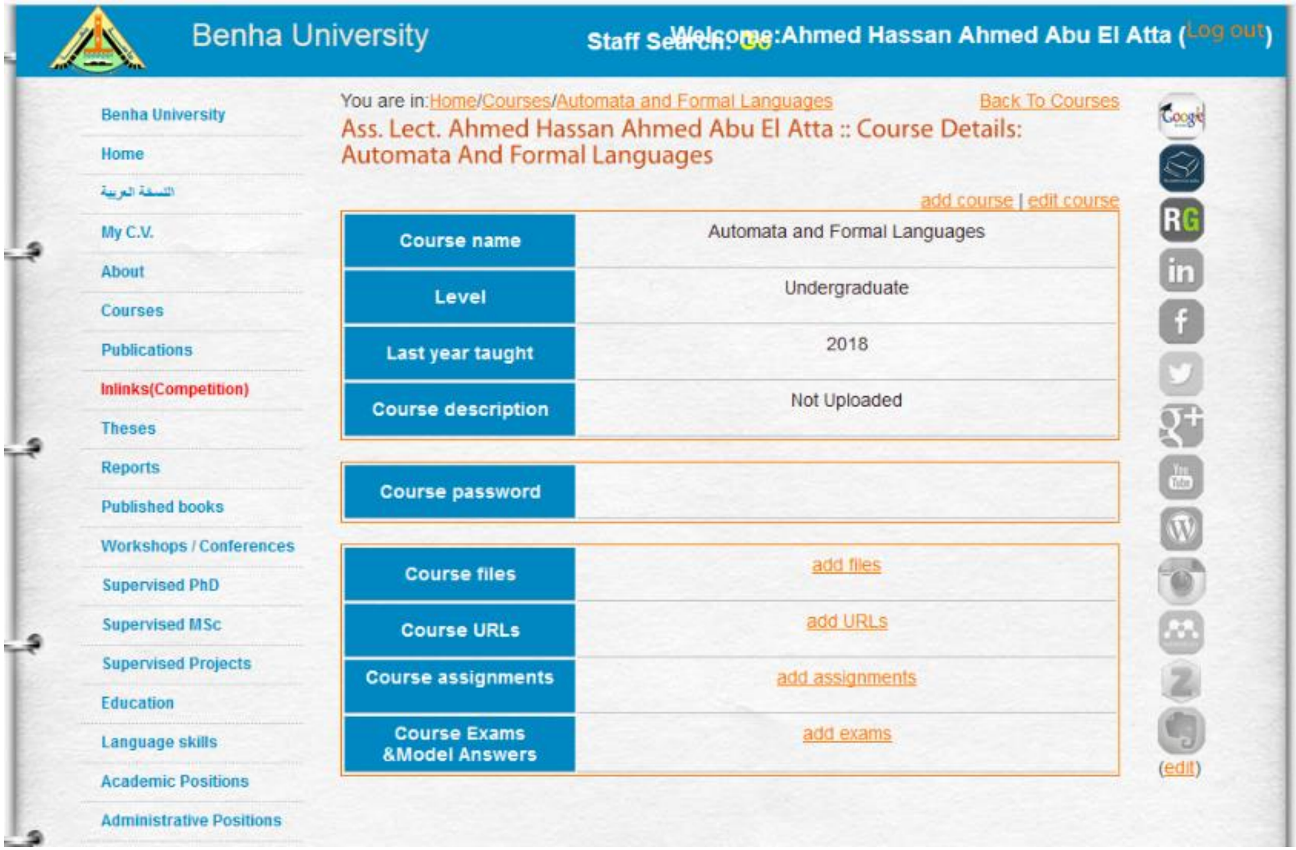
Lecture 01

Books


THOMAS H. CORMEN
CHARLES E. LEISERSON
RONALD L. RIVEST
CLIFFORD STEIN
INTRODUCTION TO
ALGORITHMS
THIRD EDITION

PowerPoint

http://www.bu.edu.eg/staff/ahmedaboalatah14-courses/14767



Introduction

Outline

Definitions

Algorithms

Problems

Course Objectives

Analysis of Algorithms

Examples

What is an Algorithm?

Well-defined computation procedure that takes some value, or a set of values as input and produces some value or a set of values as output

An algorithm is the thing that stays the same whether the program is in C,BASIC, etc.

An algorithm has to solve a general, specified problem.

What is a problem?

Problem Specification

- Specify what a typical input instance is
- Specify what the output should be in terms of the input instance

Example: Sorting

- **Input**: A sequence of "n" numbers $a_1 \ldots a_n$
- **Output**: the permutation (reordering) of the input sequence such that $a_{s(1)} \leq a_{s(2)} \leq \ldots \leq a_{s(n)}$.

Types of Problems

Search: find X in the input satisfying property Y — Max

Structuring: Transform input X to satisfy property Y — Sort

Construction: Build X satisfying Y — Scheduling

Optimization: Find the best X satisfying property Y — TSP

Decision: Does X satisfy Y? — Odd or Even

Adaptive: Maintain property Y over time. — Insert in sorted list

Two desired properties of algorithms

Correctness

- Always provides correct output when presented with legal input

Efficiency

- What does efficiency mean?

Example: Odd Number

Input: A number n
Output: Yes if n is odd, no if n is even
Which of the following algorithms solves Odd Number best?

- Count up to that number from one and alternate naming each number as odd or even.
- Factor the number and see if there are any 2 in the factorization.
- Keep a lookup table of all numbers from 0 to the maximum integer.
- Look at the last bit (or digit) of the number.

Course Objectives

1. Learning classic algorithms
2. How to devise correct and efficient algorithms for solving a given problem
3. How to express algorithms
4. How to analyze algorithms
5. How to prove (or at least indicate) no correct, efficient algorithm exists for solving a given problem

How to devise algorithms

Something of an art form

We will describe some general techniques and try to illustrate when each is appropriate

Expressing Algorithms

Implementations

Pseudo-code

English

Verifying algorithm correctness

Proving an algorithm generates correct output for all inputs

One technique covered in textbook

- Loop invariants

Examples

Problem 1

Write an algorithm to find set of prefix sums S = {s_1, s_2, …, s_n } for a set of “n” numbers A = {a_1, a_2, a_3, …, a_n }

(hint: prefix sum $s_k = \sum_{i=1}^{k} a_i$).

Algorithm 1.1

For k = 1 to n do

- $s_k = 0$
- For i = 1 to k do
  - $s_k = s_k + a_i$
- End For

- End For

Algorithm 1.2

- $s_1 = a_1$

For $k = 2$ to n do

  - $s_k = s_{k-1} + a_k$

- End For

What is the best one?

- Algorithm 1.1 takes approximately “$n^2/2$” steps.
- Algorithm 1.2 takes approximately “n” steps.

Problem 2

Write an algorithm to find the intersection set $C = \{c_1, c_2, c_3, \ldots, c_h\}$ between two sets $A = \{a_1, a_2, a_3, \ldots, a_n\}$ and $B = \{b_1, b_2, b_3, \ldots, b_m\}$

(hint: c_i belongs to C if c_i belongs to A and c_i belongs to B).

Algorithm 2.1

For i = 1 to n do

- For j = 1 to m do
  - If a_i equals to b_j then
    - Add a_i to C
  - End If
- End For

- End For

Algorithm 2.2

- Let two sets A and B are sorted.
- i = j =1

While (i ≤ n and j ≤ m) do

- If (ai = bj) then
  - Add ai to C
  - i = i + 1
  - j = j + 1
- Else If (ai < bj)
  - i = i + 1
- Else
  - j = j + 1
- End If
- End If

- End While

What is the best one?

- Algorithm 2.1 takes approximately “n*m” steps.

- Algorithm 2.2 takes approximately “[n + m + (sorting steps)]” steps.

Problem 3

Write a program that compute ***e*** for given number ***i*** **.**

You can approximate ***e*** using the following series:

$$e = 1 + \frac{1}{1!} + \frac{1}{2!} + \frac{1}{3!} + \frac{1}{4!} + \ \dots \ + \frac{1}{i!}$$

(***hint:*** $i! = i \times (i - 1) \times \ldots\ldots \times 3 \times 2 \times 1$).

Algorithm 3.1

Let ***e*** =1

For k = 1 to *i* do

- Let fact =1
- For j = 1 to k do
  - fact = fact + j
- End For
- ***e*** = ***e*** +1/fact

- End For

Algorithm 3.2

- Let ***e*** =1
- Let fact =1

For k = 1 to *i* do

  - fact = fact * k
  - ***e*** = ***e*** +1/fact

- End For