IIT KGP
Dept. of Computer Science & Engineering

CS 30053
Foundations of Computing

Debdeep Mukhopadhyay

Overview on the Course

Text Books + References

1. Foundations of Computer Science, C Edition, Alfred V. Aho and Jeffrey D. Ullman
2. Discrete Mathematics in Computer Science, Donald F. Stanat and David F. McAllister
3. Kenneth H. Rosen, Discrete Mathematics and its Applications, Tata McGraw-Hill.
4. Other materials shall be distributed

Syllabus

- Logic,
- Sets, Relations
- Functions
- Induction
- Iteration and recursion
- Graphs.
- Algebraic structures,
- Combinatorics.
- Grammars and languages
- Automata,
- Turing machines
- Undecidability.
- Algorithms and their correctness
- Complexity
- Intractability.

Tentative Evaluation Scheme

1. Assignments: 10 marks
2. Quiz: 10 marks
3. Mid Term Examination : 30 marks
4. End Term Examination : 50 marks

What is Mathematics, really?

- It's *not* just about numbers!
- Mathematics is *much* more than that:

> Mathematics is, most generally, the study of any and all *absolutely certain* truths about any and all *perfectly well-defined* concepts.

- But, these concepts can be about numbers, symbols, objects, images, sounds, *anything*!

Then, why are some mathematics useful?

- **Applications:** Certain field of mathematics finds useful applications.
- Todays maths may lead to tomorrows, or may be day after tomorrows applications.
  - Example : Galois Theory, **Évariste Galois**(October 25, 1811 – May 31, 1832)
- You can also develop your own mathematics, if you can lay down well formed laws (axioms) to start with and certain inference laws.
- In this course, we shall see certain mathematical structures and principles, which are used very frequently in computer science.

Foreword

- Computer Science is a science of abstraction: creating a correct model for thinking about a problem and devising an appropriate mechanism to solve it.
- Create abstraction of real world phenomenon that can be understood and manipulated by computers and their users.
- We can model electronic circuits, by something which is called propositional logic. Abstraction, always simplifies certain details, which are not required for the job at hand.

An Example

- Problem: Allot slots to the courses, so that no two subjects are in the same slot if there is a student taking both.
- Abstraction: *Course conflict graph*
  - Modeling :
    - Represent courses by circles or nodes
    - Draw an edge between 2 nodes if there is a student taking both
    - Define Independent Set to be the collection of nodes, which has no edge between them. When the Independent set cannot grow further, it is called maximal

Example (contd.)

- Method to solve (Algorithm):

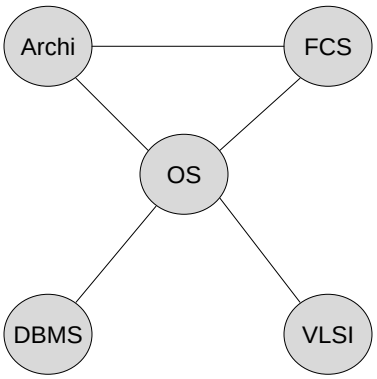
ITERATION:

1. Assign first slot to the first Maximal Independent Set.
2. Then remove the nodes in the maximal independent set.
3. Repeat the above steps (iteration).

RECURSION:

If there are still courses left to be scheduled, form a Maximal Independent Set, assign the next available slot and then apply the same algorithm on the left over courses (left after removing the courses in the Maximal Independent Set)

Example of the example



First Maximal Independent Set:

{Archi, DBMS, VLSI}

Time Slot	Course
1	Archi, DBMS, VLSI
2	OS
3	FCS

Example (contd.)

- Is this algorithm the best?
  - Analysis of algorithms.
  - Typically, running time, space required for storing and manipulating data etc.
- Do we always have an algorithm ?

  There are many problems in computer science, which are not solvable. There are some for which solution exists, but are very hard to find. While there are others which for which nice solutions exist.

Proofs are wonderful

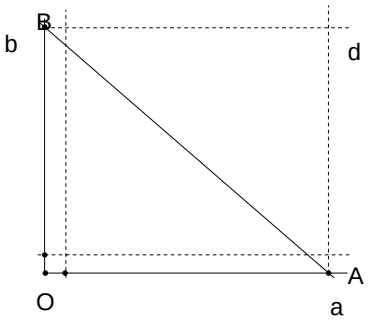
- An example of a proof based on counting discrete points: If a and b are two positive integers, then

$$d = \gcd(a,b) = 2\sum_{i=1}^{a-1}\left\lfloor i\frac{b}{a}\right\rfloor + a + b - ab$$

Recap of gcd:

1. If d must be the common factor of a and b.
2. d must be the largest common factor of a and b.

Proofs are absolute truths



Points on legs of ΔAOB: $a+b+1$

Points inside or on AB: $\sum_{i=1}^{a-1}\left\lfloor -i\frac{b}{a}+b\right\rfloor = \sum_{i=1}^{a-1}\left\lfloor \frac{b}{a}(a-i)\right\rfloor = \sum_{i=1}^{a-1}\left\lfloor \frac{b}{a}i\right\rfloor$

$\therefore$ Points inside on or inside ΔAOB: $s = a+b+1+\sum_{i=1}^{a-1}\left\lfloor \frac{b}{a}i\right\rfloor$

Points on AB: no of integers x ($0 \le x \le a$), s.t y=(-b/a)x+b is an integer

= no of elements in the set: $\{0, \frac{a}{d}, 2\frac{a}{d}, ..., (d-1)\frac{a}{d}, a\} = d+1.$

$\therefore$ No of elements on or inside rectangle OADB

$= s + (s-(d+1)) = 2s-(d+1)$

But it is also equal to $(a+1)(b+1)$

$\therefore 2s-(d+1) = (a+1)(b+1)$

$\Rightarrow d = 2\sum_{i=1}^{a-1}\left\lfloor i\frac{b}{a}\right\rfloor + a + b - ab.$

QED: (Quod erat demonstrandum, thus it is proved.)

If you remember this proof then you can deduce the expression also. So, it gives a better understanding.

Objectives of this course

- Upon completion of this course, the student should be able to:
  - Check validity of simple logical arguments (proofs).
  - Check the correctness of simple algorithms.
  - Creatively construct simple valid logical arguments.
  - Creatively construct simple correct algorithms.
  - Describe the definitions and properties of a variety of specific types of discrete structures.
  - Correctly read, represent and analyze various types of discrete structures using standard notations.

A Gentle Reminder

- While solving assignments, please try to think. Group work is always encouraged, but your final solutions should be independent. Any form of copying or cheating shall be dealt with strictly. Please cooperate to make this course fruitful.