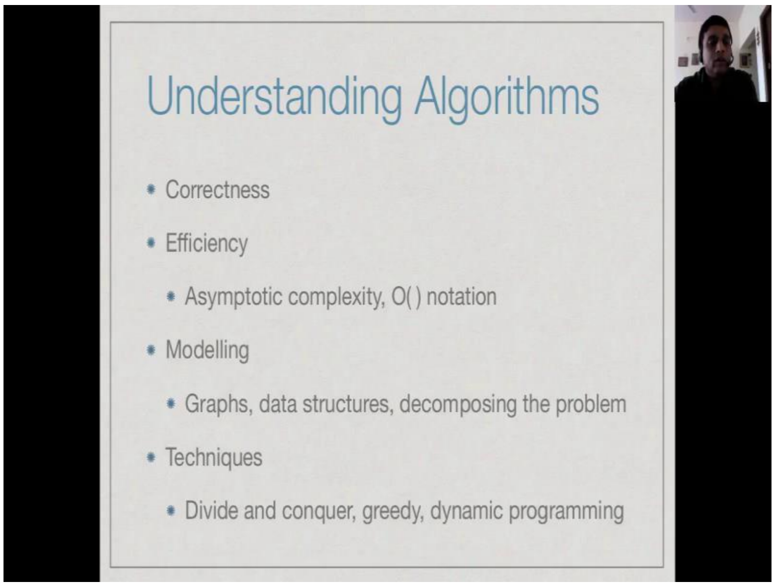
Design and Analysis of Algorithms, Chennai Mathematical Institute

Prof. Madhavan Mukund

Department of Computer Science and Engineering,

Week - 01
Module - 01
Lecture - 01

(Refer Slide Time: 00:06)



Understanding Algorithms

- Correctness
- Efficiency
  - Asymptotic complexity, O( ) notation
- Modelling
  - Graphs, data structures, decomposing the problem
- Techniques
  - Divide and conquer, greedy, dynamic programming

So, welcome to the NPTEL MOOC on the design and the analysis of algorithms. So, here are some of the things that we would be looking at in this course. When we study algorithms, the first thing that we need to convince our selves is that the algorithm is correct and it is doing the job that we expected. So, we look at the strategies for proving the correctness of algorithms.

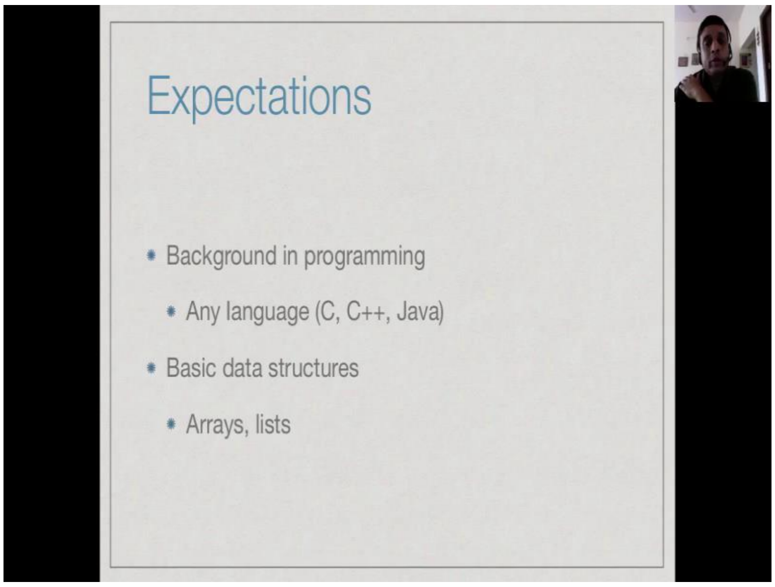
The other important aspect of algorithm is of course its efficiency. How much time does the algorithms take on inputs? Now, of course we have to factor in the size of the input. And we need a notation are a way of comparing two different algorithms, which operates on the same types of inputs and produce the same type of outputs. So, this will be achieved through the concept called asymptotic complexity, which measures the running time of an algorithm as inputs grow larger and larger as the function of the inputs. And we will develop some notation, typically the big O notation in order to smoothen out some ((Refer Time: 01:00)) of algorithms and group them into large chunks, which are equivalent.

An important part of problem solving in any domain and in particular algorithms is the art of modelling the problem at a suitable level of detail. In most algorithms that we will see we need to find a suitable mathematical model. One of these will be graphs. We need a way of representing the concepts of these models in our algorithm. For this, we need appropriate data structures. And of course typically in order to solve a problem we need to break it down into manageable sub problems.

So, we will look at strategies to decompose problems into smaller problems and see how to put them together to solve the problem it has. Over the course of time, many generic techniques have been developed to solve the large number of problems. And we will see examples of how these techniques can be applied to very standard problems that we come across repeatedly. Among the techniques are divide and conquer. Where, we break up the problem into individual components which do not overlap with each other and then combine these solutions in order to get the solution for the overall problems.

In some cases, we can identify a strategy which looks at the local state of the problem and chooses an optimal path and arrives at the final solution without having to look at all possibilities. These kind of greedy algorithms are there. It is important to know how to prove such an algorithms correct. But if a greedy algorithms does exists, it is typically much more efficient than other types of algorithms. When greedy does not work, we need a systematic way of exploring all the possibilities and choosing the best one. In this process, sometimes we have to discard overlapping problems and make sure that we do not waste fully recomputed things. So, this is covered by the concept of dynamic programming.

(Refer Slide Time: 03:06)

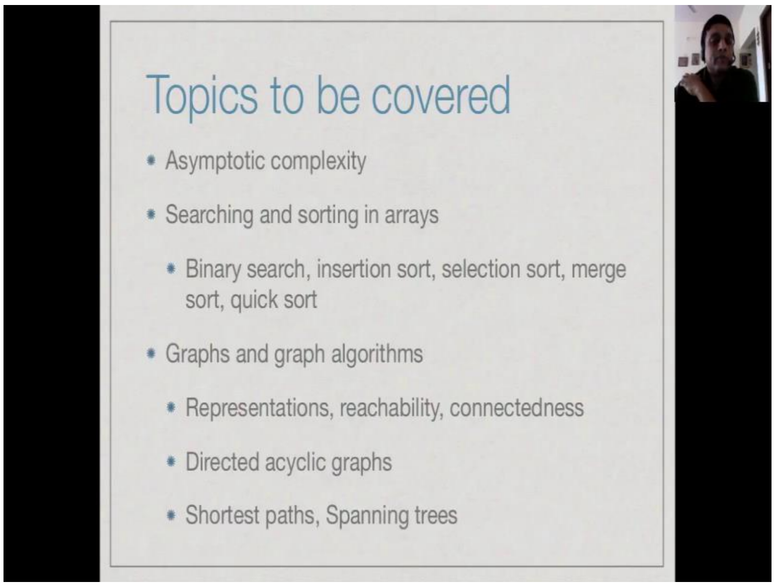


Expectations

- Background in programming
  - Any language (C, C++, Java)
- Basic data structures
  - Arrays, lists

Now along with the theory, in this course we will have some programming assignments. So, we do expect that all of you have some background in programming. This could be in C or C plus plus or java. We are flexible about the language we use, but you should be able to standard programs and implement some of the basic algorithms that we are using in our course. In order to do this, we will of course cover some new data structures in this course. But we do expect that in the language that you use, you are familiar with basic concepts like arrays and lists and also things like stacks and queues, which build up on these. Of course, when we come to stacks and queues we will try to give some detail about how they are used. But we do expect that you have seen them before. So, this not the first time you are seeing these concepts.

(Refer Slide Time: 03:54)



Topics to be covered

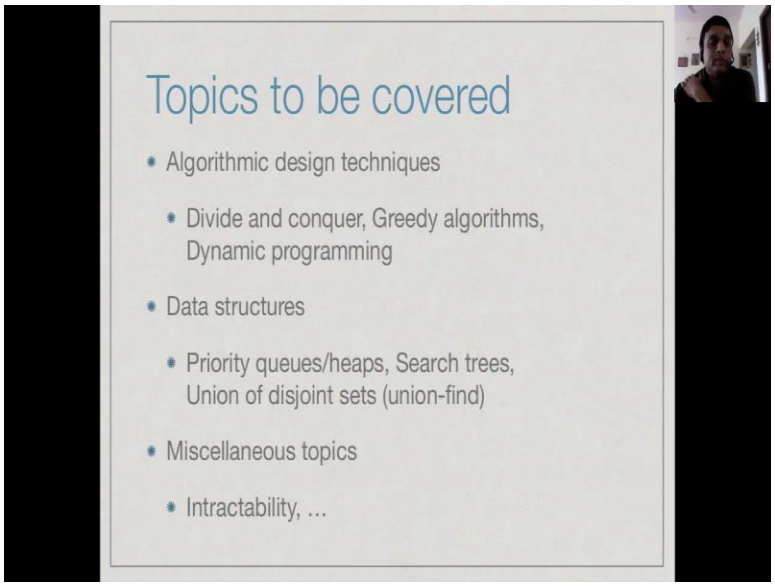
- Asymptotic complexity
- Searching and sorting in arrays
  - Binary search, insertion sort, selection sort, merge sort, quick sort
- Graphs and graph algorithms
  - Representations, reachability, connectedness
  - Directed acyclic graphs
  - Shortest paths, Spanning trees

Here is a kind of approximate list of the topics we expect to cover in the course. So, after looking at a few examples we will start with asymptotic complexity, which is the way of measuring the efficiency of algorithms and writing down this measure in a way that we can compare easily across algorithms. We will then move to the most basic problem that we have for arrays, which is to search an array for an element. And in the process we will realize that it is important to be able to sort the array efficiently in order to arrange the elements in a way where we can search in an effective manner.

So, we will look at searching and sorting. So, we will look at binary search and we will look at different notions of sorting. Some of which are more elementary and intuitive like insertion sort and selection sort. But I am not the best in terms the efficiency. And then we will look at more efficient algorithms like merge sort, quick sort; which are not obvious to begin with.

Moving on from searching and sorting, we come to graphs and graph algorithms. So, we will introduce graphs. We will see how we can use them to model certain types of problems. We need to know how to represent a graph. How do you? Graph is essentially a picture, so how do you translate this picture into something that an algorithm can manipulate? So, that is representation. We will look at standard problems in graphs; reachability and connectedness. We will look at a special class of graphs called directed acyclic graphs, which are very useful for modelling certain kinds of problems. And then we will look at other canonical problems on graphs including shortest paths and spanning trees.

(Refer Slide Time: 05:31)

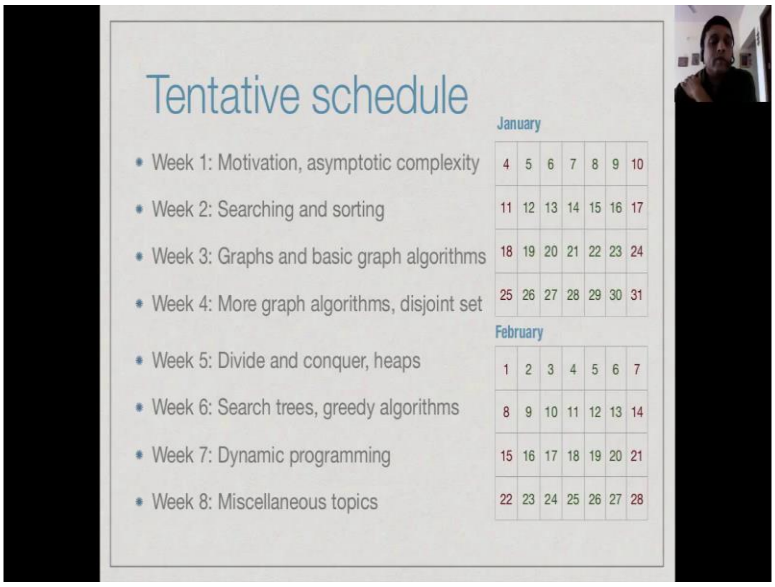


Topics to be covered

- Algorithmic design techniques
  - Divide and conquer, Greedy algorithms, Dynamic programming
- Data structures
  - Priority queues/heaps, Search trees, Union of disjoint sets (union-find)
- Miscellaneous topics
  - Intractability, ...

As we mentioned before, there are some basic algorithmic design techniques; which are applied across a class of problems. So, we will look in particular divide and conquer, greedy algorithms and dynamic programming. Among the data structures that we will encounter in this course are priority queues, which are often implemented to heaps. You will look at binary search trees, which are efficient ways of maintaining information in a sorted order dynamically as information comes and goes. And we will look at a very useful data structures to maintain a partition of the set into a disjoint collection of subsets; the so called union-find algorithm. And finally, depending on how much time is left we will look at some miscellaneous topics. It is important to realize that not every algorithm admits an efficient solution.

(Refer Slide Time: 06:34)



Tentative schedule

- Week 1: Motivation, asymptotic complexity
- Week 2: Searching and sorting
- Week 3: Graphs and basic graph algorithms
- Week 4: More graph algorithms, disjoint set
- Week 5: Divide and conquer, heaps
- Week 6: Search trees, greedy algorithms
- Week 7: Dynamic programming
- Week 8: Miscellaneous topics

January

4	5	6	7	8	9	10
11	12	13	14	15	16	17
18	19	20	21	22	23	24
25	26	27	28	29	30	31

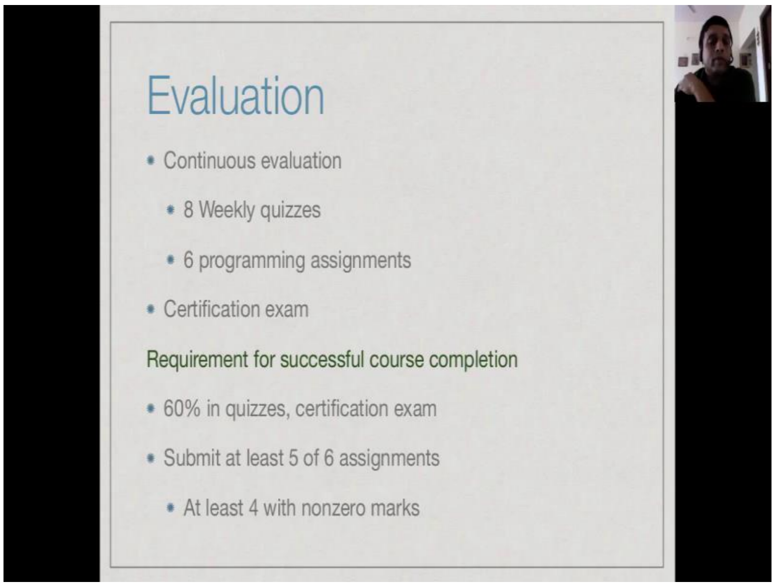
February

1	2	3	4	5	6	7
8	9	10	11	12	13	14
15	16	17	18	19	20	21
22	23	24	25	26	27	28

So, we will look at problem are intractability and some provably hard problems and some other problems of which this state is unknown. This is an eight week course. The tentative schedule is as follows: in the first week, we will do some motivating examples and we will look at some notations and work out some problems involving asymptotic complexity. In the second week, we will move on to searching and sorting. In the third week, we will introduce graphs and look at basic graph algorithms. We will continue with more graph algorithms. And then in the process, introduce the disjoint set data structures. We will then formally look at divide and conquer, which you have already seen along the way. But we will look at it again. And then we will introduce heaps. In the sixth week, we will do search trees and some examples of greedy algorithms and their proofs. In the seventh week, we will be devoted to dynamic programming. And then the last week we will cover miscellaneous topics.

Now, you should remember that this is only an approximate schedule and there will be some variations. But roughly this is the sequence and the speed at which you planned to cover the material in the course. You can also see alongside the weeks. The calendar weeks that these correspond to. So, to the first week of the course is the current week which is January fifth to ninth. And this course will go on to the last week of February.

(Refer Slide Time: 07:53)


Evaluation

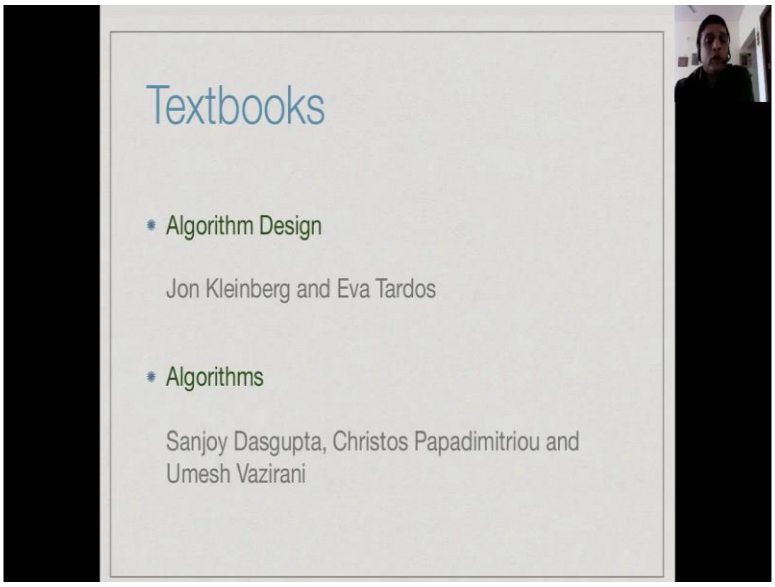
- Continuous evaluation
  - 8 Weekly quizzes
  - 6 programming assignments
- Certification exam

Requirement for successful course completion

- 60% in quizzes, certification exam
- Submit at least 5 of 6 assignments
  - At least 4 with nonzero marks

Now as part of the evaluation for the course, there will be continuous evaluations. Every week, there will be quizzes. You will also be programming assignments; roughly six programming assignments across the eight weeks. After the course ends, there will be a certification exam. In order to get a certificate that you successfully completed this course, you need to score 60 percent in the quizzes and in the certification exam. You need to submit at least five of six; out of the six assignments. And at least, four of them you must do something ((Refer Time: 08:27)) review.

(Refer Slide Time: 08:30)


Textbooks

- Algorithm Design

  Jon Kleinberg and Eva Tardos

- Algorithms

  Sanjoy Dasgupta, Christos Papadimitriou and Umesh Vazirani

We will be following two main text books. Although, I will not use the text books directly in the course, but if you want to look at some materials you will find them in these two books. The first book is called “Algorithm design” by Jon Kleinberg and Eva Tardos. And the second book is just called “Algorithms” by Sanjay Dasgupta, Christos Papadimitriou and Umesh Vazirani. The book by Kleinberg and tardos is more detail and it has a number of really nice examples to illustrate many of the concepts and quite detail proofs. The book by Dasgupta Papadimitriou and Vazirani is a slimmer book. It is more easy and accessible to read on your own. But on the other hand it does leave a lot of details as exercises. So, in order to really understand the material, you really need to spend more time working out the exercises and not just go by the materials that present in the chapters.