# Algorithm Design Methods

# Algorithm Design Techniques

- An algorithm can be defined as a finite sequence of steps to solve a problem.
- Given a problem, the algorithm is largely influenced by the choice of data structure.
- With a chosen data structure, one can develop a number of different algorithms for the same problem.
- The first intuitive algorithm may not be the best one as far as memory and time efficiency are concerned.
- There are some general techniques for development of algorithms, some of which are discussed in the following slides.

# Divide – and - Conquer

- Break the problem of size n into smaller problems in such a way that from solutions to smaller problems, the solution to the entire problem can be easily constructed.

- Apportioning a problem into equal or nearly equal parts/sub-problems is a crucial factor in obtaining good performance.

- e.g.
- 1. Binary Search
- 2. Mergesort
- 3. Quicksort
- 4. Maximum finding

# Greedy Algorithms

- At any individual stage, select the option which is locally optimal in some particular sense. Not all greedy algorithms succeed in producing the best overall result.

- But they produce "good" solutions with high probability. Frequently, a sub-optimal solution with a cost a few % above the optimal is quite satisfactory.

- For exhaustive search problems, the greedy algorithms may be the only technique for obtaining a "good" solution within a short time. This solution can then be improved.

- e.g.
- 1. Coin change
- 2. Dijkstra's shortest path algo

# Dynamic Programming

- A problem may not be subdivided, but there are a manageable number of smaller problems whose solutions lead to the solution of the original problem.

- A table can be created and filled up with solutions to smaller sub-problems without regard to whether or not a particular sub-problem is actually needed in the overall solution. This technique is called dynamic programming.

- There is an order in which the entries of the table are to be filled.

- e.g.
- 1) Fibonacci Series
- 2) Factorial

# Backtracking

- In exploratory problem solving, go ahead by exploiting /choosing options at each stage till a blind alley comes. Then move one stage backward and choose from the unexplored options.

- This is a general method of solving any constraint-satisfaction problem.

- This produces a backtrack tree.

- e.g.
- 1) n-queens problem
- 2) Engineering design problems

# Branch and Bound

- Here, at each node of the backtrack tree, a lower bound is computed on the best solution that can be found farther down the tree. If that lower bound is greater than a known solution, backtracking is done.

- Effectively, the branches of the backtrack-tree are pruned online.

- e.g. Chess

  Other techniques

  Search

  Traversal

  Randomized Agorithms

# Qualities of an Algorithm

Using different techniques, one can develop many algorithms for the same problem.

There are some qualitative aspects of programs/algorithms, e.g., simplicity, readability, modularity, modifiability, well-documented, etc. But based on these, there can only be some subjective judgment about programs/algorithms.

More objective criteria are :

- (1) Amount of main memory used
- (2) Execution time

# Factors affecting execution time

- Programmer skill
- Compiler Options
- Hardware characteristics of machine [instruction set, clock speed]
- Algorithm used
- Size of input

- Algorithms are to be analysed to compute the objective criteria for different input size before actual implementation.

# Summary

- Different primary methods for algorithm development discussed
- Advanced types of algorithms like randomized algorithms mentioned
- Objective criteria for comparison of algorithm to solve the same problem introduced