➢Be able to express an algorithm / interpret an algorithm using pseudocode

➢Be able to trace the important steps to demonstrate you understand how the algorithm works

➢Be able to ascertain the growth rate of an algorithm’s time wrt the input size

➢Know the algorithms mentioned

➢Work Problems similar to the assigned homeworks

Chapter 8: Dynamic Programming

•What is Dynamic programming?

1. A technique for solving problems with overlapping subproblems. This method suggests solving each of the smaller subproblems only once and recording the results in a table from which a solution to the original problem can be obtained.
2. Not limited to special types of optimization problems.
3. Principle of optimality – an optimal solution to any instance of an optimization problem is composed of optimal solutions to its sub instances.

•What are the benefits of storing partial solutions?

1. Storing partial solutions makes it to where the program does not have to spend time re-calculating those solutions. On the other hand however, storing the solutions to the partial problems is space expensive. They can be refined to not take up as much space though.

•Section 8.1: Example

1. **Coin-row problem**: Pick up the max amount of money subject to the constraint that no two coins adjacent in the initial row can be picked up. Omega(n) time and space.
2. **Change Making Problem**: Give change for amount n using the minimum number of coins of denominations d1 < d2 < … < dm. O(n\*m) time and Omega(n) space.
3. **Coin Collecting Problem**: Several coins are placed in cells of an n x m board, no more than one coin per cell. Robot can move one right and one down and picks up the maximum amount of coins on the board. Time and space is Omega (n\*m).

•Section 8.2: The Knapsack Problem

Maximum value of n items with w weight and j value. Time and space for dynamic is Omega(n\*w) and time to find optimal solution is in O(n).

O Using memory function to store partial, overlapping information

1. The memory function seeks to solve only necessary subproblems and only do it once. Top-down but maintains the table of bottom-up.

O Why is this better than top-down recursive solutions?

1. This is better than top down because it does not solve unnecessary instances of the problem and it also does not solve subproblems more than once. It is more efficient for time and space.

O Problem 12 in this section –world series odds

•Section 8.4: Warshall's and Floyd's Algorithms – both of these are designed to exploit a relationship between a problem and its simpler rather than smaller version.

1. **Warshall’s:** Computing the transitive closure of a directed graph. This constructs the transitive closure through a series of n x n Boolean matrices.
2. Transitive closure of a digraph allows us to determine in constant time whether the jth vertex is reachable from the ith vertex. This can be used for investigating data flow and control flow dependencies as well as for inheritance testing of object-oriented programming.
3. **Floyd’s:** The shortest all-pairs shortest-paths problem. This problem asks to find all the distances from each vertex to all other vertices. Time efficiency is cubic. Same as Warshall’s.
   * 1. **Distance matrix:** Record the lengths of shortest paths in an n x n matrix D.

Chapter 9: Greedy Technique

1. **Greedy:** This approach suggests constructing a solution through a sequence of steps, each expanding a partially constructed solution obtained so far until a complete solution is found.
2. ***feasibl****e*, i.e., it has to satisfy the problem’s constraints
3. ***locally optimal***, i.e., it has to be the best local choice among all feasible choices available on that step
4. ***irrevocable***, i.e., once made, it cannot be changed on subsequent steps of the algorithm

•**What kinds of problems is the greedy technique used for?** It is applicable to optimization problems only. problems where choosing locally optimal also leads to global solution are best fit for Greedy.

•**Section 9.1: Prim's Algorithm**

1. Constructs a minimum spanning tree through a sequence of expanding subtrees.

•**Section 9.2: Kruskal's Algorithm:**

1. https://www.geeksforgeeks.org/kruskals-minimum-spanning-tree-algorithm-greedy-algo-2/

•**Section 9.3: Djikstra's Algorithm**

1. https://www.geeksforgeeks.org/dijkstras-shortest-path-algorithm-greedy-algo-7/

Chapter 12: Coping with Limitations

•Section 12.1: Backtracking – construct solutions one component at a time and evaluate such partially constructed candidates as followed.

1. If a partially constructed solution can be developed further without violating the problem’s constraints, it is done by taking the first remaining legitimate option for the next component. If there are no legit options for the next component, no alternatives for any remaining component need to be considered. In this case, the algorithm backtracks to replace the last component of the partially constructed solution with its next option.
2. **State Space Tree:** nodes reflect specific choices made for a solution’s components. **Promising:** if the tree corresponds to a partially constructed solution that may still lead to a complete solution.
3. Usually used for no optimization problems. Tree is usually developed depth-first.

o What kinds of problems is this technique suited for?

1. Generally, every [constraint satisfaction problem](https://en.wikipedia.org/wiki/Constraint_satisfaction_problem) which has clear and well-defined constraints on any objective solution, that incrementally builds candidate to the solution and abandons a candidate (“backtracks”) as soon as it determines that the candidate cannot possibly be completed to a valid solution, can be solved by Backtracking.

o n-Queens Problem

1. Find the board of 4 queens where the queens cannot hit each other.
2. A single solution can be found in linear time (n >= 4).

Diagram, engineering drawing

Description automatically generated

o Knight’s tour Problem

1. Given a N\*N board with the Knight placed on the first block of an empty board. Moving according to the rules of chess knight must visit each square exactly once. Print the order of each the cell in which they are visited.
2. https://www.geeksforgeeks.org/the-knights-tour-problem-backtracking-1/

o Subseb-sum problem: Find a subset of a given set of n positive integers whose sum is equal to a given positive integer D.

Diagram

Description automatically generated