Algorithms & AI – Lectures, Projects, Q&A

Lectures, Projects, Q&A etc.

Project 3

Dynamic Programming and DFS/BFS Problems

// Option 1 ==> ID ends in 1, Option 2 ==> ID ends in 2, Option 0 ==> ID ends in 0.

0. Teleportation in Astro haunted galaxies
1. Box Stacking
2. Magical eggs and floors
3. Longest common subsequence
4. Maximum Value but Limited Neighbors
5. Fast response k-server problem
6. Diameter of a graph
7. Pseudo-polynomial partition
8. Bi-connectivity
9. Maze using DFS

Detailed Descriptions

"Teleportation in Astro Haunted Galaxies"
You have a teleporter that can take you from galaxy i to galaxy j. Cost to teleport is given by c(i,j), which can be arbitrary. Some galaxies are "astro-haunted" – this is specified by a(i) which can be 0 or 1 (1 means that that galaxy is "astro-haunted"). Give a polynomial time algorithm that minimizes the cost of going from galaxy 1 to galaxy n, such that you pass through no more than k astro-haunted galaxies. (You can assume that galaxies 1 and n are not astro-haunted.)

Box Stacking
You are given a set of n types of rectangular 3-D boxes, where the i^th box has height h(i), width w(i) and depth d(i) (all real numbers). You want to create a stack of boxes which is as tall as possible, but you can only stack a box on top of another box if the dimensions of the 2-D base of the lower box are each strictly larger than those of the 2-D base of the higher box. Of course, you can rotate a box so that any side functions as its base. It is also allowable to use multiple instances of the same type of box.

Magical eggs and tiny floors (*aka* "The Cellphone Drop Testing Problem")

You are given m eggs and an n floor building. You need to figure out the highest floor an egg can be dropped without breaking, assuming that (i) all eggs are identical, (ii) if an egg breaks after being dropped from one floor, then the egg will also break if dropped from all higher floors, and (iii) if an egg does not break after being thrown from a certain floor, it retains all of its strength and you can continue to use that egg. Your goal is to minimize the number of throws. Describe an algorithm to find the floor from which to drop the first egg.

Longest common subsequence

Given two strings (sequences of characters), the longest common subsequence (LCS) problem is to find the longest subsequence (not necessarily contiguous) that exists in both of the input strings. For example, given strings "mangoes" and "mementos", the subsequence "mnos" is common in both and is in fact the longest common subsequence. Given two strings of sizes n1 and n2 respectively, find a dynamic programming algorithm to find the longest common subsequence in O(n1n2) time.

Maximum Value But Limited Neighbors

You are given an array a[1..n] of positive numbers and an integer k. You have to produce an array b[1..n], such that: (i) For each j, b[j] is 0 or 1, (ii) Array b has adjacent 1s at most k times, and (iii) sum_{j=1 to n} a[j]*b[j] is maximized. For example, given an array [100, 300, 400, 50] and integer k = 1, the array b can be: [0 1 1 0], which maximizes the sum to be 700. Or, given an array [10, 100, 300, 400, 50, 4500, 200, 30, 90] and k = 2, the array b can be [1, 0, 1, 1, 0, 1, 1, 0, 1] which maximizes the sum to 5500.
To be precise about the definition of adjacency: sequence [0, 1, 0, 1, 0, 1, 1, 1] has two adjacent 1s. Sequence [0, 1, 0, 0, 1, 1, 1, 1] has 3 adjacent 1s. Sequence [1, 0, 1, 1, 0, 1, 1, 1] also has 3 adjacent 1s.

Fast Response k-Server Problem

A series of client machines [1, 2, ... n] are located along a linear network. The *i*-th client generates amount of traffic that is given by *w[i]*. You want to place *k* servers along the linear network that minimizes the total amount of traffic carried by the network. Total traffic is given by sum of each client's individual traffic, multiplied by the distance (the number of hops) from the server. Provide a polynomial time algorithm to identify the optimal locations for *k* servers.

Diameter of a Graph

Diameter of a graph is defined as the largest distance between any pair of vertices of G. Give an efficient polynomial algorithm to find the diameter of the graph.

Pseudo-polynomial Partition

Given a set consisting of *n* integers $[a_1, a_2, \ldots a_n]$, you want to partition into two parts so that the sum of the two parts is equal. Suppose $s = a_1 + a_2 \ldots + a_n$. The time complexity of your algorithm should be *O(ns)* or better. **[Note:** Due to the presence of the term *s* in the time complexity, such an algorithm is called pseudo polynomial algorithm.]

Biconnectivity

Given a graph G, check if the graph is biconnected or not. If it is not, identify all the articulation points. The al-

gorithm should run in linear time. Use the given input sets as tests.

Maze using DFS

Model a given maze as a graph. Find the target using depth first search. The algorithm should run in linear time. Use the given input sets as tests.