15-451/651 Algorithm Design & Analysis

Spring 2023, Recitation #5

Objectives

- Practice dynamic programming algorithms for sequences and trees (using small steps).
- Understand how to analyze the complexity of a dynamic programming algorithm.

Recitation Problems

1. **(Removal Game)** You are playing a game with Wurzelbrunft. There is a list of n numbers L, and players move alternately (with you going first). For each move, the player either removes the first element or the last element from the list, and adds that value to their score. Both of you are trying to maximize your score. Come up with an algorithm to find what your score would be, assuming both of you play optimally.

   This can be solved in $O(n^2)$ using Dynamic Programming.

   (a) Find an appropriate sub-problem.

   (b) Given a sub-problem, come up with a recurrence to determine its answer.

   (c) Given this, finish the solution by describing how to compute the answer for the whole array.

2. **(Closest Depot in a Tree)** You're given a rooted tree T with n vertices. There are $m \leq n$ special vertices called *depots*. You are to compute, for every node v of T, the distance from v to the nearest depot. The distance is the number of tree edges that must be traversed to get there. (If v is a depot the distance is 0.)

   Come up with an $O(n)$ to compute the distance from every node to the nearest depot.

3. **(Cheapest Tree Separation)** There are N cities, numbered from 1 through N, connected by $N-1$ roads, forming a weighted tree. Countries A and B each occupy a set of cities (no city is occupied by both countries, and some cities may not be occupied at all).

   To stop fighting between the two countries, you want to destroy roads such that no city occupied by country A is connected to a city in country B. Destroying a road of length x costs x dollars. What is the minimum cost required? Given a linear time algorithm.

Further Review

1. **(Vacuums)** Suppose that you are a door-to-door salesman, selling the latest innovation in vacuum cleaners to less-than-enthusiastic customers. Today, you are planning on selling to some of the n houses along a particular street. You are a master salesman, so for each house, you have already worked out the amount c_i of profit that you will make from the person in house i if you talk to them. Unfortunately, you cannot sell to every house, since if a person's neighbour sees you selling to them, they will hide and not answer the door for you. Therefore, you must select a subset of houses to sell to such that none of them are next to each other, and such that you make the maximum amount of money.

   For example, if there are 10 houses and the profits that you can make from each of them are 50, 10, 12, 65, 40, 95, 100, 12, 20, 30, then it is optimal to sell to the houses $1, 4, 6, 8, 10$ for a total profit of \$252. Devise a dynamic programming algorithm to solve this problem.

   (a) Define a set of subproblems that we could use to apply dynamic programming

   (b) What are the base case subproblems and what are their values?

   (c) Write a recurrence relation that describes the solutions to the subproblems

2. **(Shortest Common Supersequence)** Consider a pair of sequences $a_1, ..., a_n$ and $b_1, ..., b_m$ of length n and m. A supersequence of a sequence a is a sequence that contains a as a subsequence. Devise an algorithm that finds the length of a shortest common supersequence of a and b. A common supersequence is a sequence that is both a supersequence of a and of b. Your algorithm should run in $O(nm)$.

3. **(Matrix)** You are given a positive integer N and a 26 by 26 matrix A whose entries are either 0 or 1. How many strings of length N, consisting of lowercase English letters, are there such that for all $1 \leq i, j \leq 26$ where $A_{ij} = 0$, character j does not appear directly after character i?

   (a) Define a suitable set of subproblems to apply dynamic programming

   (b) Give a recurrence relation with base cases that show how to compute the value of a subproblem

   (c) Argue that your dynamic program solves the problem in $O(N)$ time

4. **(Number of Increasing Partitions)** Define the value of an array to be the value of its maximum element. We are given an array $A = [A_1, A_2, \ldots, A_n]$. How many ways are there to partition A into subarrays, or "pieces", such that the values of the pieces, from left to right, are nondecreasing? For example $A = [1, 2, 1, 3, 2, 1]$ has a valid partition $[1, 2], [1, 3, 2, 1]$, but $[1, 2, 1, 3], [2, 1]$ is an invalid partition.

   (a) Define a set of subproblems to which we can apply dynamic programming.

   (b) Give a recurrence relation with base cases that show how to compute the value of a subproblem

(c) Show that your dynamic programming solution can be evaluated in $O(n^2)$ time, then show that it can be improved to $O(n)$ time

5. **(Bin Packing)** You are given a collection of n items, and each item has size $s_i \in [0, 1]$. You have many bins, each of unit size, and you want to pack the n items into as few bins as possible. (Each bin can take a subset of items, whose total size is at most 1.)

   Show that you can solve this problem in time $O(4^n)$. Then improve this bound to $O(3^n)$.