CS 256: Algorithm Design and Analysis

Assignment 5 (due 10/24/2020)

Instructor: Sam McCauley

Note on Dynamic Programs. For full credit on a dynamic program, you must clearly state the following parts.

(a) *Subproblem definition:* your subproblem must have an optimal substructure.
(b) *Recurrence:* how should the next subproblem be computed using the previous ones? This is the core of your algorithm and its correctness. A less ideal alternative to a recurrence is clear pseudocode for the final iterative dynamic-programming algorithm.
(c) *Base case(s):* you need to start somewhere!
(d) *Final output:* in terms of your subproblem.
(e) *Memoization data structure:* this is often obvious but should not be skipped.
(f) *Evaluation order:* describes the dependencies between the subproblems.
(g) *Time and space analysis.*

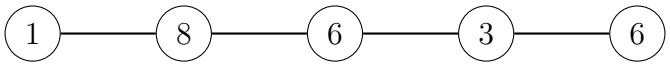
For this assignment, these are sufficient to argue correctness (that is to say, if you explain why the above parts are correct, that's sufficient to show that your algorithm works properly).

The problems in this assignment are in (approximate) order of difficulty. We'll be seeing a number of dynamic programming examples over the next few lectures. Some of the later, more diffiuclt problems will likely become easier as you get more practice with dynamic programming.

Problem 1. (Kleinberg Tardos 6.1) Let $G = (V, E)$ be an undirected graph with n nodes. A subset of the nodes is called an *independent set* if no two of them are joined by an edge. Finding large independent sets is difficult in general; but here we'll see that it can be done efficiently if the graph is simple enough.

Call a graph $G = (V, E)$ a path if its nodes can be written as $v_1, v_2, ..., v_n$, with an edge between v_i and v_{i+1}, for $i \in \{1, 2, \ldots, n-1\}$. With each node v_i, we associate a positive integer weight w_i. The problem we want to solve is the following: Find an independent set in a path G whose total weight is as large as possible.

For example, given the following path



The maximum weight of an independent set is 14.

(a) Give a counterexample to show that the following "pick the heaviest weight" greedy algorithm does not always work.

- Start with $S = \emptyset$
- While some node remains in G
  - Pick a node v_i of maximum weight and v_i to S
  - Delete v_i and its neighbors from G
- Return S

(b) Give a simple exponential-time recursive algorithm that gives a correct solution.

(c) Give a dynamic-programming algorithm that takes an n-node path G with weights and returns the *value* of the independent set of maximum total weight.

Solution. □

Problem 2. (Erickson 3.6) Solving part (a) is sufficient for full credit on this problem; part (b) is extra credit. However, I encourage you to (at least) think very hard about how to solve part (b), as it's excellent dynamic programming practice.

A shuffle of two strings X and Y is formed by interspersing the characters into a new string, keeping the characters of $X$and Yin the same order.For example, the string BANANAANANAS is a shuffle of the strings BANANA and ANANAS in several different ways.

$$\mathsf{BANANA_{ANANAS}} \qquad \mathsf{BAN_{ANA}ANA_{NAS}} \qquad \mathsf{B_{AN}AN_{A}A_{NA}NA_{S}}$$

Similarly, the strings PRODGYRNAMAMMIINCG and DYPRONGARMAMMICING are both shuffles of DYNAMIC and PROGRAMMING:

$$\mathsf{PRO_{D}G_{Y}R_{NAM}AMMI_{I}N_{C}G} \qquad \mathsf{DY_{PRO}PRO_{N}G_{A}R_{M}AMM_{IC}ING}$$

(a) Given three strings $A[1..m]$, $B[1..n]$, and $C[1..m+n]$, describe and analyze an algorithm to determine whether C is a shuffle of A and B.

(b) **(Extra credit: 5 pts)** A **smooth** shuffle of X and Y is a shuffle of X and Y that never uses more than two consecutive symbols of either string. For example,

- $\mathsf{PR_{D}O_{Y}G_{NA}RA_{M}MM_{I}I_{C}NG}$ is a smooth shuffle of the strings DYNAMIC and PROGRAMMING.
- $\mathsf{DY_{PR}N_{OGR}A_{A}M_{MM}IC_{ING}}$ is a shuffle of DYNAMIC and PROGRAMMING, but it is not a smooth shuffle (because of the substrings OGR and ING).
- $\mathsf{XX_{X}X_{X}X_{XX}XX_{XX}XX_{X}X_{X}XX}$ is a smooth shuffle of the strings XXXXXXX and XXXXXXXXXXX.
- There is no smooth shuffle of the strings XXXX and XXXXXXXXXXXX.

Describe and analyze an algorithm to decide, given three strings X, Y,and Z, whether Z is a smooth shuffle of X and Y.

Hint: What do you need to change in order to build up a smooth shuffle rather than a normal shuffle? What do you need to keep track of to ensure that you can make this distinction?

Solution. □

Problem 3. (From Steve Skiena's *Algorithm Design Manual*) Consider the problem of storing n books on shelves in a library. The order of the books is fixed by the cataloging system and so cannot be rearraged. Let book b_i have thickness t_i and height h_i, for $1 \leq i \leq n$. Let the length of each bookshelf at this library be L. Suppose we have the freedom to adjust the height of each shelf to fit the tallest book on it. The cost of a particular layout is the sum, over each shelf, of the height of the largest book on that shelf. (So if shelf 1 has books with heights $(1, 5, 3)$ and shelf 2 has books with heights $(2, 4)$, the total cost is $5 + 4 = 9$.)

(a) Give an example to show that the greedy algorithm of stuffing each shelf as full as possible (that is, fill the first shelf with as many books as possible until book b_i does not fit, and then repeat the same process on subsequent shelves) does not always give the minimum overall height.

(b) Give a dynamic programming algorithm that computes the height of the optimal arrangement, and analyze its time and space complexity. (*Hint. We have done a similar example in class with a different cost function and constraints.*)

Solution. □