Algorithms Lab 8

(More dynamic programming)

Laura Toma, csci2200, Bowdoin College

This week's topics

- Dynamic programming and greedy: weighted interval scheduling, activity selection

Review the topics discussed in class this week. This is a great time to understand all the details and ask questions.

In-class (COLLABORATION LEVEL 0[1])

1. **Longest increasing subsequence:**[2] A *subsequence* of a sequence (for example, an array, linked list or string), is obtained by removing zero or more elements and keeping the rest in the same order. A subsequence is callde a *substring* if its elements are contiguous in the original sequence. For example:

   (a) GORIT, ALGO, RITHMS are all substrings of ALGORITHMS

   (b) GRM, LORI, LOT, AGIS,GORIMS are all subsequences of ALGORITHMS

   (c) ALGOMI, OG are *not* subsequences of ALGORITHMS

   The problem: Given an array $A[1..n]$ of integers, compute the length of a *longest increasing subsequence* (A sequence $B[1..k]$ is *increasing* if $B[i] < B[i+1]$ for all $i = 1..k-1$). For example, given the array

   $$\{5, 3, 6, 2, 1, 5, 3, 1, 2, 5, 1, 7, 2, 8\}$$

   your algorihm should return 5 (for e.g. correponding to the subsequence $\{1, 3, 5, 7, 8\}$; there are other subsequences of length 5).

   Describe the subproblem and the recursive formulation.

[1] Collaboration level 0: everything allowed!

[2] Credit: This is a classical problem. This particular formulation is from Jeff Erickson, UIUC, `http://www.cs.illinois.edu/ jeffe/teaching/algorithms`. I have seen it phrased in the context of financial markets where the sequence represents stock values.

2. **Matching points on a line:**[3] You are given two arrays of n points in one dimension: red points $r_1, r_2, ..., r_n$ and blue points $b_1, b_2, ..., b_n$. You may assume that all red points are distinct and all blue points are distinct. We want to pair up red and blue points, so that each red point is associated uniquely with a blue point and vice versa. Given a pairing, we assign it a score which is the sum of distances between each pair of matched points.

   Find the pairing (matching) of minimum score. Hint: Aim for an $O(n \lg n)$ algorithm. Draw examples for small valus of n to get intuition.

   Example: Consider the input where the red points are $r_1 = 8, r_2 = 1$ and the blue points are $b_1 = 3$ and $b_2 = 9$. There are two possible matchings:

   - match r_1 to b_1 and r_2 to b_2: the score of this matching is $|r_1 - b_1| + |r_2 - b_2| = |8 - 3| + |1 - 9| = 13$
   - match r_1 to b_2 and r_2 to b_1: the score of this matching is $|r_1 - b_2| + |r_2 - b_1| = |8 - 9| + |1 - 3| = 3$

   The algorithm shoud return the cost of the optimal matching, which is 3.

Homework: String Shuffling[4]

A *shuffle* of two strings A and B is formed by interspersing the characters into a new string, keeping the characters from A and B in the same order.

For example, the string BANANAANANAS is a shuffle of the string BANANA and ANANAS (in several different ways, actually: BANANAANANAS, BANANAANANAS and also BANANAANANAS).
Similarly, the strings ANEVEGARIN and ANEGAVERIN are both shuffles of NEVER and AGAIN.

The problem: Given three strings $A[1..m]$, $B[1..n]$ and $C[1..m + n]$, come up with an efficient algorithm to determine whether C is a shuffle of A and B.

1. Argue that the problem has optimal substructure.
2. Define your subproblem. Clearly state in words what the arguments represent, and what the subproblem represents.
3. Give a recursive definition of the subproblem.
4. Imagine you write a function to compute the subproblem using the formula above. Briefly argue what the running time would be.
5. Give pseudocode for the recursive, dynamic programming approach and analyze it.

[3]Credit: reproduced from Stanford University, cs161, fall 2016-2017
[4]Credit: this problem is from Jeff Erickson, UIUC

6. **Pair programming:** Translate your efficient dynamic programming algorithm above into either Java or Python code. Your code should be able to take 3 arguments (preferably on the command line), which represent the three strings, and report whether the third string is a shuffle of the first two.

Pair-programming honor code

Everyone is required to work with a partner. No exceptions please, unless the number of students in the class is odd. Which it is. So this leaves us two choices: either someone works alone, or a team of three. And, your partner has to be different than the partner you had for the previous programming assignment. The honor code for pair-programming is the following:

1. You must work together, physically in the same place.
2. The overall lab must be a true joint effort, equally owned, created and understood by both partners.
3. Specifically splitting the lab into parts and working on them separately is not allowed and violates the honor code for the class.
4. If you start together as a team and are unable to complete the project together, then you will each inherit the shared code, and split up to work individually on the remainder of the project. You would then submit individually, with a note on what happened.
5. The two partners in a team obviously share everything, obviously. Across teams, collaboration is allowed at COLLABORATION LEVEL: 1.

What to turn in

1. Problems 1 through 5 (one per team, can be all on one page).
2. Push your code in GitHub. A link for the Github assignment will be posted on piazza.
3. Bring a hard copy of the code (one per team). Make sure you include a brief header that describes how to run it.