CS41 Lab 9: Dynamic Programming

November 4, 2022

In typical labs this semester, you'll be working on a number of problems in groups of 3-4 students. You will not be handing in solutions; the primary purpose of these labs is to have a low-pressure space to discuss algorithm design. However, it will be common to have some overlap between lab exercises and homework sets.

The learning goals of lab this week are (i) to better understand the power of dynamic programming and (ii) to practice building DP algorithm design skills.

1. **Testing RNA Substructure Implementations.** Last week, we introduced the RNA Substructure problem and developed an efficient algorithm for RNA Substructure that uses dynamic programming.In this lab problem, you'll see this solution in practice.

   In /home/brody/public/cs41/, you'll find two executables: `rna`-A, and `rna`-B. One uses dynamic programming to solve the RNA Substructure problem, and one solves it without storing solutions to overlapping subproblems in a table. Each implementation takes in the name of a file containing a single string representing an RNA molecule, and returns the size of the largest matching (following the RNA substructure rules discussed in class).

   For this exercise, you'll use the UNIX `time` command to examine the runtime of each implementation. For example, to measure how much time `rna`-A takes on input `rna_test_data`/test1, execute

   ```
   $ time /home/brody/public/cs41/rna-A /home/brody/public/cs41/rna_test_data/test1
   ```

   (a) Using the test files in `rna_test_data` and your own test files, determine which program uses dynamic programming and which does not.

   (b) **How large can inputs be?** For both `rna`-A and `rna`-B, create input files of different sizes and determine how large the input can be if the implementation must run in at most 30 seconds.

   (c) **How does the runtime scale?** Again for each implementation, create some test files of different lengths, and measure the execution time and how it scales with the size of the inputs. Use this to guess what the implementation's runtime is. Is `rna`-A an $O(n^2)$ algorithm? or $O(n^3)$ or $O(n^4)$? $O(2^n)$? Do the same for `rna`-B.

2. **Subset Sum.** In this problem, you are given an integer *weight threshold* $W > 0$ and a list of n items $\{1, \ldots, n\}$ each with nonnegative weight w_i. Your task is to output a subset of items $S \subseteq \{1, \ldots, n\}$ such that $\sum_{i \in S} w_i$ is as large as possible, subject to $\sum_{i \in S} w_i \leq W$.

   Design an analyze a dynamic program to solve **Subset Sum**. Your algorithm should run in $O(nW)$ time.

3. **Gerrymandering (K& T 6.24)** *Gerrymandering* is the practice of carving up electoral districts in very careful ways so as to lead to outcomes that favor a particular political party. Recent court challenges to the practice have argued that through this calculated redistricting, large numbers of voters are being effectively (and *intentionally*) disenfranchised.

   Suppose we have a set of n precincts $P_1, \ldots, P_n$, each containing m registered voters. We're supposed to divide these precincts in to two *districts*, each consisting of $n/2$ precincts. Now, for each precinct, we have information on how many voters are registered to each of two political parties. Say that the set of precincts is *susceptible* to gerrymandering if it is possible to perform the division into two districts in such a way that the same party holds a majority in both districts.

Give an algorithm to determine whether a given set of precincts is susceptible to gerrymandering. The running time of your algorithm should be polynomial in n and m.

For example, suppose there are four precincts, and two political parties A and B. Letting A_i and B_i be the number of voters in precinct i of each political party, Suppose we have

$$A_1 = 55, A_2 = 43, A_3 = 60, A_4 = 47 \textit{ and}$$

$$B_1 = 45, B_2 = 57, B_3 = 40, B_4 = 53\ .$$

This set of precincts is susceptible to gerrymandering since pairing precincts 1 and 4 together and $2, 3$ together gives party A a 102 - 98 majority in the first district and a 103 - 97 majority in the second.

Hint: Focus on the choice you need to make as you're building up a partial solution to this problem (in this case, an assignment of precincts to districts). You will likely need to maintain extra information about the partial solution.