CS 577: Introduction to Algorithms Fall 2020

Homework 4

Instructor: Dieter van Melkebeek TA: Nicollas Mocelin Sdroievski

This homework covers dynamic programming. **Problems 1 and 3 must be submitted for grading by 11:59pm on 10/05.** Problem 1 will be discussed in your discussion section on Friday. Please refer to the homework guidelines on Canvas for detailed instructions.

Warm-up problems

1. [Graded, 2 points] Design an algorithm that takes as input two binary sequences a and b of lengths m and n respectively, and outputs the smallest length of a sequence c such that both a and b are subsequences of c. For example, if $a = 011$ and $b = 0101$, then the answer is 4, as witnessed by $c = 0101$. Your algorithm should run in time $O(m \cdot n)$.

2. Consider the multiplication defined in Table 1. For example, $ab = b$ and $ba = c$. Note that this multiplication is neither associative nor commutative.

   Table 1: Multiplication Table

	a	b	c
a	b	b	a
b	c	b	a
c	a	c	c

   You want to know, given a string of n symbols a, b, c, with $n \geq 1$, whether or not it is possible to parenthesize the string in such a way that the value of the resulting expression is a. For example, the string $bbbbac$ can be parenthesized as $((b(bb))(ba))c$, and that evaluates to a.

   Design an algorithm to solve this problem in time polynomial in n.

Regular problems

3. [Graded, 3 points] A telecommunications company must place n antennas, labeled 1 to n, in $k < n$ separate geographical locations. You'll be visiting locations $1, \ldots, k$ in order, once each, to install the antennas. Due to network specs, you can only install antenna $i+1$ once antenna i has been installed. Still, since $k < n$, we know some antennas will need to be placed in the same location. The problem is that when we place antennas i and j in the same location, they interfere with intensity s_{ij}. If we place more than 2 antennas in the same location, then we only need to account for the strongest interference value. For example, if antennas 1, 2 and 3 are in the same location, then the interference in that location has intensity $\max\{s_{12}, s_{13}, s_{23}\}$. Given a split of the antennas into the k locations, the total interference is the sum over the maximum interference for each location. We want to determine the minimum total interference that can be realized under the given constraints.

   More formally, you are given a number k of locations, a number n of antennas, and nonnegative numbers s_{ij} for every pair of antennas. We need to determine integers $0 \leq t_1 \leq t_2 \leq \ldots \leq$

$t_{k-1} \leq n$ (indicating the last antennas placed in locations 1 through $k-1$) such that

$$\sum_{i=1}^{k} \left(\max_{t_{i-1} < \ell < m \leq t_i} \{s_{\ell m}\} \right)$$

is minimized, where $t_0 = 0$ and $t_k = n$. The max in this expression represents the interference at location i, and we add this value for each location.

For example, consider the instance with $k = 2$, $n = 4$, and interference intensities $s_{12} = 10$, $s_{13} = 5$, $s_{14} = 12$, $s_{23} = 30$, $s_{24} = 40$ and $s_{34} = 15$. Observe that there are 5 possible ways to split the 4 antennas into the 2 locations:

- Split 1: $\{\}\{1, 2, 3, 4\}$
- Split 2: $\{1\}\{2, 3, 4\}$
- Split 3: $\{1, 2\}\{3, 4\}$
- Split 4: $\{1, 2, 3\}\{4\}$
- Split 5: $\{1, 2, 3, 4\}\{\}$

The first and last ones are equivalent and have a total interference value of 40. Split 2 also has total interference 40, while Split 3 has total interference $10 + 15 = 25$ and split 4 has total interference 30. In this case, the answer is 25. Note that if we could change the order of the antennas, the total interference could be reduced even further (laction 1 contains $\{1, 3, 4\}$ and location 2 contains $\{2\}$, for a total interference value of 15), but this is not allowed.

(a) Design an $O(n^2)$ algorithm that outputs the maximum interference values for the subinstances consisting of antennas i through j for *all* $1 \leq i \leq j \leq n$ and $k = 1$.

(b) Design an $O(kn^2)$ algorithm to solve the problem for a given instance with n antennas and a given k. Your algorithm should output the minimum total interference that can be realized under the constraints.

4. In modern origami (the Japanese art of paper folding), one typically starts with a square sheet of paper and attempts to transform this square into a three-dimensional animal, geometric object, or any other sculpture one can think of, using nothing but a sequence of folds. In traditional 17th–18th century origami, however, the starting shape of the paper was less strictly prescribed.

   Hiro has stumbled across a book containing instructions for n origami sculptures from this early period, each of which starts from rectangular paper of size $a_i \times b_i$ where a_i and b_i are positive integers. He would like to make a diorama containing as many of these (not necessarily distinct) sculptures as possible, but he only has access to a single sheet of paper of size $A \times B$ (where A and B are also positive integers) and no scissors. By folding the paper carefully and tearing along the crease, Hiro is confident that he can make perfect horizontal and vertical cuts across an entire sheet of paper, splitting the sheet into two.

   Design an algorithm that on input $A, B, a_1, \ldots, a_n, b_1, \ldots, b_n$, computes the maximum number of sculptures that can be made from the starting sheet of paper. Your algorithm should run in time polynomial in A, B, and n.

5. 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.

Computers, as it turns out, have been implicated as some of the main "villains" in much of the news coverage on this topic: it is only thanks to powerful software that gerrymandering grew from an activity carried out by a bunch of people with maps, pencil, and paper into the industrial-strength process it is today. Why is gerrymandering a computational problem? Partly it is the database issues involved in tracking voter demographics down to the level of individual streets and houses; and partly it is the algorithmic issues involved in grouping voters into districts. Let's think a bit about what these latter issues look like.

Suppose we have a set of precincts $P_1, P_2, \ldots, P_n$, each containing m registered voters. We're supposed to group these precincts into two districts, each consisting of $n/2$ of the precincts. Now, for each precinct, we have information on how many voters are registered to each of two political parties. (Suppose for simplicity that every voter is registered to one of these two.) We say that the set of precincts is susceptible to gerrymandering if it is possible to perform the division in such a way that the same party holds a majority in both districts.

Design 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.

Example Suppose we have $n = 4$ precincts, and the following information on registered voters. Party A has 55, 43, 60, and 47 voters in districts P_1, P_2, P_3, P_4 respectively, and party B has 45, 57, 40, and 53. This set of precincts is susceptible, since if we grouped precincts P_1 and P_4 into one district, and precincts P_2 and P_3 into the other, then party A would have a majority in both districts. (Presumably, the "we" who are doing the grouping here are members of party A.) This example is a quick illustration of the basic unfairness in gerrymandering: although party A holds only a slim majority in the overall population (205 to 195), it ends up with a majority in not one but both districts.

Challenge problem

6. Consider problem 3, but with the modification that the interference at a location with more than 2 antennas is the sum of the pairwise interference values instead of the maximum. That is, if antennas 1, 2 and 3 are in the same location, then the interference in that location has intensity $s_{12} + s_{13} + s_{23}$. Again, we want to minimize the total interference over all locations.

   (a) Design an $O(n^2)$ algorithm that outputs the interference values for the subinstance consisting of antennas i through j for *all* $1 \leq i \leq j \leq n$ and $k = 1$.

   (b) Design an $O(kn \log n)$ algorithm to solve the problem for a given instance with n antennas and a given k when given access to the one-location interference values computed in part (a). Your algorithm should output the minimum total interference that can be realized under the constraints.

Programming problem

7. SPOJ problem Square Brackets (problem code SQRBR).