Homework 4 – Due Thursday, October 29, 2015

Please refer to the course website for the full homework policy and options.

Reminders

- Your solutions are due in class. Please also submit a soft copy on the dropbox on Owlspace. Late homework will not be accepted.
- For the set of problems to be solved outside of class, collaboration is permitted, but you must write the solutions by yourself without assistance, and be ready to explain them orally to a member of the course staff if asked. You must also identify your collaborators. Getting solutions from outside sources such as the Web or students not enrolled in the class is strictly forbidden.
- To facilitate grading, please write down your solution to each problem on a separate sheet of paper. Make sure to include all identifying information and your collaborators on each sheet.
- When in doubt, email the TAs or the instructor.
- For problems that require you to provide an algorithm, you must give a precise description of the algorithm, together with a correctness argument and an analysis of its running time. You may use algorithms from class as subroutines. You may also use any facts that we proved in class.

1. (Number of shortest paths) Suppose we are given a directed graph \( G = (V, E) \), with costs on the edges. The costs may be positive or negative, but every cycle in the graph has strictly positive cost. We are also given two nodes \( v, w \in V \). Give an efficient algorithm that computes the number of shortest \( v-w \) paths in \( G \).

2. (Gerrymandering) 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 into two districts, each consisting of \( n/2 \) of the precincts. Now, for each precinct, we have information on how many voters are assigned to two political parties. (We assume that each 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 into two districts in such a way that the same political party holds a majority in both districts.

Give an algorithm, polynomial in \( n \) and \( m \), to decide whether a given set of precincts is susceptible to gerrymandering.

3. (k-shot strategies) Consider \( n \) consecutive days of a given stock, at some point in the past. The days are numbered 1, 2, \ldots, \( n \); for each day, you have a price \( p(i) \) per share for the stock on that day.

For some (possibly large) value of \( k \), you want to study a so-called \( k \)-shot strategy. A \( k \)-shot strategy is a collection of \( m \) pairs of days \( (b_1, s_1), \ldots, (b_m, s_m) \), where \( 0 \leq m \leq k \) and

\[
1 \leq b_1 < s_1 < b_2 < s_2 < \cdots < b_m < s_m \leq n.
\]
We view these as a set of \( k \) non-overlapping intervals, during each of which the investors buy 1000 shares of the stock (on day \( b_i \)) and then sell it (on day \( s_i \)). The return of the strategy is the profit obtained from the transactions in this interval, namely,

\[
1000 \sum_{i=1}^{m} p(s_i) - p(b_i).
\]

Give an efficient algorithm (polynomial in both \( n \) and \( k \)) to determine, given the sequence of stock prices, the \( k \)-shot strategy with the maximum possible return.

4. (Oscillating subsequence) Call a sequence \( X[1\ldots n] \) of numbers oscillating if \( X[i] < X[i + 1] \) for all even \( i \), and \( X[i] > X[i + 1] \) for all odd \( i \). Describe an efficient algorithm to compute the length of the longest oscillating subsequence of an arbitrary array \( A \) of integers.

Note: a subsequence of an array \( A \) is a sequence \( \langle a_1, \ldots, a_k \rangle \) of elements of \( A \) that appear in the same order as in \( A \). A subsequence need not be a contiguous subarray. For example, \( \langle 1, 7, 3 \rangle \) is a subsequence of the array \( [5, 1, 9, 7, 23, 30, 3] \).

5. (Marble cutting) You have mined a large slab of marble from your quarry. For simplicity, suppose the marble slab is a rectangle measuring \( n \) inches in height and \( m \) inches in width. You want to cut the slab into smaller rectangles of various sizes — some for kitchen countertops, some for large sculpture projects, others for memorial headstones. You have a marble saw that can make either horizontal or vertical cuts across any rectangular slab. At any given time, you can query the spot price \( p_{x,y} \) of an \( x \)-inch by \( y \)-inch marble rectangle, for any positive integers \( x \) and \( y \). These prices will vary with demand, so do not make any assumptions about them; in particular, larger rectangles may have much smaller spot prices. Given the spot prices, describe an efficient algorithm to compute how to subdivide an \( n \times m \) marble slab to maximize your profit.