EC504 Homework Assignment 4 (due November 9 in class)

The following problems can be solved using dynamic programming. You must clearly specify the set of sub-problems you are using, and the recurrence you are using—describing both in English as well as any notation you define. You should also explain why your recurrence leads to the correct solution of the sub-problems. Finally, describe the complete algorithm that makes use of the recurrence and sub-problems.

1. Let $G = (V, E)$ be a directed graph with nodes $v_1, ..., v_n$. We say that G is a *line-graph* if it has the following property:

   - Each edge in the graph goes from a node with a lower index to a node with a higher index.

   Given a line-graph G and an integer $k \leq n$, give a polynomial-time algorithm to find a directed path in G from v_1 to v_n using exactly k edges if such a path exists.

2. Recall the least squares problem for linear regression, discussed briefly in class. Given a set of points in the plane $P = \{(x_1, y_1), ..., (x_n, y_n)\}$ where $x_1 < x_2 < ... < x_n$ (we assume the x_i values are scalars), we aim to find a line L defined by the equation $y = ax + b$ to minimize the error function:

   $$Err(L, P) = \sum_{i=1}^{n} (y_i - ax_i - b)^2.$$

   We can solve for a and b in closed-form as

   $$a = \frac{n \sum_i x_i y_i - (\sum_i x_i)(\sum_i y_i)}{n \sum_i x_i^2 - (\sum_i x_i)^2} \text{ and } b = \frac{\sum_i y_i - a \sum_i x_i}{n}.$$

   In class we considered an extension where we partition the points P into some number of segments, where each segment is a subset of P that represents a contiguous set of x-coordinates (for example, if $n = 10$, we might segment the points into three segments where the first three points are in the first segment, the next four points are in the second segment, and the last three points are in the third segment). Within each segment we compute the line minimizing the error with respect to the points in that segment, using the formulas above.

   We defined a new penalty function that is the sum of the following terms:

   - The number of segments into which we partition P, multiplied by a fixed value $\lambda > 0$.
   - For each segment, the error value of the optimal line through that segment.

   We defined $OPT(j)$ to denote the optimum solution to this penalty function for points 1 through j, and we let e_{ij} be the minimum error of any line with respect to points i through j, inclusive. Then, for the subproblem on the points 1 through j, we saw that

   $$OPT(j) = \min_{1 \leq i \leq j} (e_{ij} + \lambda + OPT(i - 1)),$$

   and the segment containing points i through j, inclusive, is used in an optimum solution if and only if the minimum is obtained using index i. (See Kleinberg and Tardos, section 6.3 for a more complete description of this approach.)

In the above problem, we did not fix the number of line segments, but rather placed a cost of λ for each line segment, which was incorporated into the penalty function. Here we will consider a problem when we know how many lines should fit the data, so we will not have any penalty parameter λ. Instead, we have an integer k, and we need to find the best fit for the points with exactly k line segments. So, given $k \leq n$, the problem is to find a partition P into exactly k segments so as to minimize the sum of the least squares error over the segments in the partition. Note that when $k = 1$, this reduces to the formula described above for simple least squares regression.

(a) Give a polynomial-time algorithm to find the optimal solution to this problem.

(b) Load in the points in the file `hw4test.txt` (first column is x-coordinates, and second column is y-coordinates) and determine the optimal partition of the points into $k = 4$ partitions. Give the partition determined by your algorithm. What is the total error produced by your solution? (Note: the running time is dominated by the time to pre-compute the e_{ij} values. One can compute these values in $O(n^2)$ time, but for the purposes of this exercise you can compute these values in a less efficient manner.)

3. (*In honor of the election.*) 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, it turns out, have been implicated as the source of some of the "villainy" in the news coverage on this topic: thanks to powerful software, gerrymandering has changed from an activity carried out by a bunch of people with maps, pencil, and paper into the industrial-strength process that it is today. Why is gerrymandering a computational problem? There are database issues involved in tracking voter demographics down to the level of individual streets and houses; and there are 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 n *precincts* $P_1, P_2, ..., P_n$, each containing m registered voters. We're supposed to divide each of 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'll 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.

Example. Suppose we have $n = 4$ precincts, and the following information on registered voters. For party A, the number of registered voters in precinct 1 is 55, in precinct 2 is 43, in precinct 3 is 60, and in precinct 4 is 47. For party B, the number of registered voters in precinct 1 is 45, in precinct 2 is 57, in precinct 3 is 40, and in precinct 4 is 53. This set of precincts is susceptible since, if we grouped precincts 1 and 4 into one district, and precincts 2 and 3 into the other, then part 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.