CS 341, Spring 2016
Timothy Chan, Semih Salihoglu

Assignment 3 (due June 24th Friday noon)

Please first read the instructions at `https://www.student.cs.uwaterloo.ca/~cs341/#assignments`.

1. [*25 marks*] **Programming question.** Implement the divide-and-conquer algorithm from Assignment 2, Question 2 which runs in time $O(n \log n)$, as described in the model solutions. The input to your program is given in the following format:

   $$n\ x_0\ y_0\ c_0\ x_1\ y_1\ c_1\ \ldots\ x_{n-1}\ y_{n-1}\ c_{n-1}$$

   where the x_i, y_i are integers and $c_i = 0$ or 1 for $0 \le i \le n-1$. $c_i = 0$ indicates that the point (x_i, y_i) is red and $c_i = 1$ indicates that the point (x_i, y_i) is blue.

   You may assume that no two x-coordinates are the same, and no two y-coordinates are the same. You can also assume that the points are given in increasing order of their x-coordinates.

   The output to your program *should be a single number* (in a single line), namely, the total number of pairs (r, b) such that r is a red point, b is a blue point, and r dominates b. Do not include any other information or text in the output.

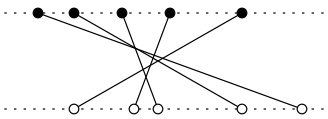
   (a) [*20 marks*] Hand in a printed copy of your program and electronically submit the source file, named a3.cpp or a3.java.

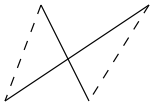
   (b) [*5 marks*] Compare the performance (i.e., actual runtimes) of your divide-and-conquer algorithm with a naive $O(n^2)$ brute-force algorithm (which just computes the count by checking all pairs (r, b)) on randomly generated input of various sizes. (Use the UNIX time method to measure actual runtimes; e.g., type /usr/bin/time -p your-command and watch for user time.) Describe (on paper) how your input is generated, present the results in a table or graph, and briefly comment on whether the experimental results agree with the theoretical analysis. (You do not need to submit the brute-force program or your input-generating program.)

2. [*13 marks*] Consider the following geometric matching problem: Given a set A of n points and a set B of n points in 2D, form pairs $(a_1, b_1), \ldots, (a_n, b_n)$, with $\{a_1, \ldots, a_n\} = A$ and $\{b_1, \ldots, b_n\} = B$, maximizing $\sum_{i=1}^{n} d(a_i, b_i)$. Here, $d(a_i, b_i)$ denotes the Euclidean distance between a_i and b_i.

   Assume that all points in A have y-coordinate equal to 0 and all points in B have y-coordinate equal to 1. (Thus, all the points lie on two horizontal lines.)

   Consider the following greedy strategy: Pick pair (a, b) where a is the leftmost point in A and b is the rightmost point in B. Then remove a and b, and repeat. (See the left figure.)





(a) [*3 marks*] How fast can this greedy algorithm be implemented?

(b) [*10 marks*] Prove that this algorithm always gives a correct optimal solution.
Hint: you may use the following fact (which easily follows from the "triangle inequality"): if two line segments ab and $a'b'$ intersect, then $d(a,b) + d(a',b') > d(a,b') + d(a',b)$. (In other words, in the right figure, the total length of the solid line segments exceed that of the dashed line segments.)

3. [*12 marks*] The manager of a large student union on campus comes to you with the following problem. She is in charge of a group of n students, each of whom is scheduled to work one shift during the week. There are different jobs associated with these shifts (tending the main desk, helping with package delivery, rebooting cranky information kiosks, etc.), but we can view each shift as a single contiguous interval of time. There can be multiple shifts going on at once. She is trying to choose a subset of these n students to form a supervising committee that she can meet with once a week. She considers such a committee to be complete if, for every student not on the committee, that student's shift overlaps (at least partially) the shift of some student who is on the committee. In this way, each student's performance can be observed by at least one person who is serving on the committee. Give an efficient greedy algorithm that takes the schedule of n shifts and produces a complete supervising committee containing as few students as possible.

   Example. Suppose $n = 3$, and the shifts are:

   - Monday 4 P.M. – Monday 8 P.M.
   - Monday 6 P.M. – Monday 10 P.M.
   - Monday 9 P.M. – Monday 11 P.M.

   Then the smallest complete supervising committee would consist of just the second student, since the second shift overlaps both the first and the third.

4. [*20 marks*] 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 us think a bit about what

these latter issues look like. Suppose we have a set of n precincts, each containing m registered voters. We want 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 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 into two districts in such a way that the same party holds a majority in both districts.

Example. Suppose we have $n = 4$ precincts, and the following information on registered voters.

Precinct	1	2	3	4
Number registered for party A	55	43	60	47
Number registered for party B	45	57	40	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 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.

(a) [*10 marks*] Let a_k denote the number of voters registered for party A in precinct k. (Then $m - a_k$ is the number of voters registered for party B in precinct k.) Define the following subproblems ($i = 0, \ldots, n$, $j = 0, \ldots, n$, $\ell = 0, \ldots, m$):

$$C[i, j, \ell] = \begin{cases} 1 & \text{if there exists a subset } S \subseteq \{1, \ldots, i\} \text{ such that } |S| = j \text{ and } \sum_{k \in S} a_k = \ell \\ 0 & \text{else} \end{cases}$$

First give a dynamic programming algorithm to compute $C[i, j, \ell]$ for all i, j, ℓ. Remember to derive the base cases and recursive formula, provide justification for your formula, and write and analyze pseudocode. The running time and space should be polynomial in n and m.

(b) [*5 marks*] Now using the table $C[\cdot, \cdot, \cdot]$ computed in part (a), solve the gerrymandering problem, i.e., decide whether there exists a subset $S \subset \{1, \ldots, n\}$ of size $n/2$ such that the average of $\{a_k : k \in S\}$ is more than $m/2$ and the average of $\{a_k : k \notin S\}$ is more than $m/2$. The running time and space should be polynomial in n and m.

(c) [*5 marks*] Write pseudocode to recover a subset S with the stated property in (b) if it exists, and analyze the running time.