Introduction to Algorithms: 6.006
Massachusetts Institute of Technology April 25, 2017
Professors Constantinos Daskalakis, Piotr Indyk, and Steve Smith Problem Set 5

Problem Set 5

All parts are due on May 9, 2017 at 11:59PM. Please download the .zip archive for this problem set. Remember, your goal is to communicate. Full credit will be given only to a correct solution which is described clearly. Convoluted and obtuse descriptions might receive low marks, even when they are correct. Also, aim for concise solutions, as it will save you time spent on write-ups, and also help you conceptualize the key idea of the problem.

Last, but not least, take a look at the collaboration policy outlined in the handout for this course.

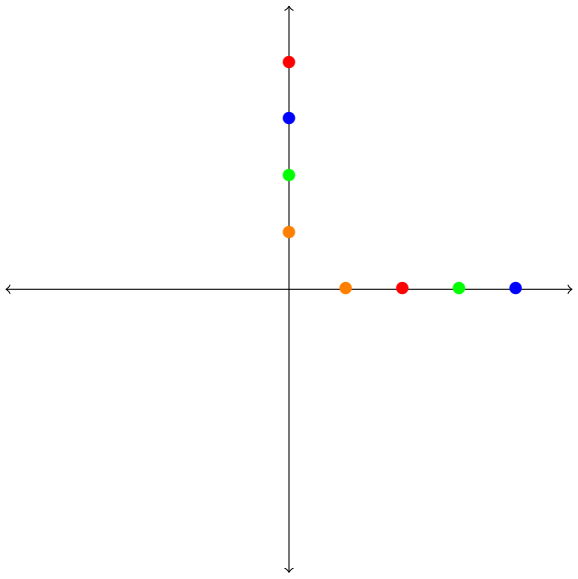
Part A

Problem 5-1. [15 points] **Connect-the-dots**

Ben Bitdiddle has k markers, each of a different color. For each marker, he puts one dot on the positive x axis and one dot on the positive y axis. For each color $i = 1, 2, \ldots, k$, we denote the position of the dot with color i on the x axis as $x(i)$, so that the point is at $(x(i), 0)$. Similarly, the position of the dot with color i on the y axis as $y(i)$, so that that the point is at $(0, y(i))$. All points are distinct (no points have the same coordinates).

Ben wants to draw straight lines between pairs of points with the same color. However, he doesn't want to intersect any of the lines because he is afraid that his markers will get smudged. He wants to connect as many dots as possible. Assume x and y are given as dictionaries keyed by colors i. Find the maximum number of points that he can connect without intersecting any lines, and the colors that he should connect to achieve this maximum, in polynomial time in k.

For example, in the following diagram you can connect 3 colors (all except the red).



Problem 5-2. [15 points] **Fun with Numbers**

You're playing a game with a friend where you are competing to get the smallest score. You start with a large number N with n digits. You start by removing either the first digit or the last digit from the number, and adding that to your score. Your friend continues by doing the same thing to the new $(n-1)$-digit number. You continue taking turns removing digits until there are no more digits in the number.

(a) [5 points] Ben is playing this game with Alyssa, and plays with the following strategy: at every step, he compares the front digit to the last digit, and removes the one that is smaller. What is the runtime of this algorithm? Give a counterexample to show that this strategy is not optimal.

(b) [10 points] Give an algorithm that takes in the number N and returns the score of the first player, given that both players play optimally, with runtime polynomial in n.

Problem 5-3. [20 points] **Lion around**

Simba is looking for the optimal place to sleep. He models the savannah as a n-row by m-column grid. Then for every square at coordinate i, j in the grid, he measures the noise level to be a positive integer $n_{i,j}$. (He defines $(0, 0)$ to be the top left square of the grid, similarly to a Python matrix.) Then for any square at i, j where the square is not on an edge or corner of the grid, he sees that column i and row j will divide the grid into four quadrants. More precisely, he defines the quadrants of a square i, j:

North-west: all squares a, b with $a < i, b < j$

North-east: all squares a, b with $a > i, b < j$

South-west: all squares a, b with $a < i, b > j$

South-east: all squares a, b with $a > i, b > j$

For example, if the jungle looks like this:

1	8	6	3
4	9	3	4
5	7	2	1
8	8	8	7

If Simba was sleeping on the square with the 9, the north-west quadrant would just be the square with value 1, and the minimum would be 1. The north-east quadrant would have the values 6 and 3, and 3 would be the minimum value.

Note that the column i and the row j are not included in any of the quadrants.

Define the noise level of a quadrant as the minimum value of any square in that quadrant. The noise level of each quadrant is denoted as $NW_{i,j}, NE_{i,j}, SW_{i,j}, SE_{i,j}$ (for the northwest, northeast, southwest, southeast quadrants respectively). Simba wants to sleep in the square that minimizes the product of the noise level of its four quadrants. He cannot sleep on the edge or the corner of the grid, because he is afraid of falling off! (and because all four quadrants should be non-empty so that the minimum is well-defined).

(a) [10 points] $NW_{i,j}$ is the minimum value of any square in the northwest quadrant. Find a dynamic programming algorithm that can compute this value for all squares i, j in $O(mn)$ time.

(b) [10 points] Using part (a), find an $O(nm)$ algorithm to find the optimal square to sleep on.

Problem 5-4. [20 points] **String transformations**

You are given an alphabet of k characters $\sigma_1, \sigma_2, \ldots, \sigma_k$ and m rules that apply to the characters. Each rule is of the form: "Replace a single character σ_x with the two-character string $\sigma_y\sigma_z$". You are playing a game where you would like to get to a target string as quickly as possible. At every time-step in the game, for every character in the string so far, you can either leave it alone or apply a rule to it. Several rules can be applied in the same time-step, and each rule can be applied multiple times. Given a target string S with length n, find the sequence of applying rules such that it takes the shortest amount of time.

For example, let the alphabet be A, B, C, and define three rules:

Rule 1: $A \to BC$

Rule 2: $B \to AC$

Rule 3: $C \to AB$

Then let our starting string be A, and our target string be $ACAB$. One optimal possibility is as follows:

Round 1: Apply Rule 1 to get $A \to BC$

Round 2: Apply Rule 2 on B and Rule 3 on C to get $BC \to ACAB$

Note that we apply two rules in round 2 because both B and C are present in the string at the beginning of that round. We can also imagine taking a suboptimal path:

Round 1: Apply Rule 1 to get $A \to BC$

Round 2: Apply Rule 2 on B to get $BC \to ACC$. Do nothing on C.

Round 3: Apply Rule 3 on C to get $ACC \to ACAB$.

Note that we can apply different rules to different instances of the same character in the string, or also choose to leave a character alone.

Find a dynamic programming algorithm that can return the optimal set of moves from the first character of the alphabet, σ_1, to a target string S of length n in $O(n^3mk)$ time.

(a) [12 points] Define a subproblem $P(c, i, j)$ as the number of steps needed to go from some character c to a substring $S[i, j]$ of S. Identify the base cases and the recursive step for a dynamic programming algorithm.

(b) [8 points] Justify why the runtime is $O(n^3mk)$.

Part B

Problem 5-5. [30 points] **Derrick the Frog**

Derrick is a square frog living in a static world of platforms. His goal is to explore this world by moving left and right and hopping. When he lands on a platform for the first time, he adds to his exploration score a bounty associated with that platform. However, moving around is very tiring — he is limited to 30 seconds of activity.

Therefore, he wants to plan very carefully. Your job is to implement `calculate_best_score` in `platforms.py` to compute the best score achievable by Derrick in 30 seconds.

See the provided code for the full details. In particular, you should treat `visualizer.py` as the specification for physics of Derrick's world. Submit your finished `platforms.py` to `alg.csail.mit.edu`. Your code will be run as `python3`.