Purdue University
CS 381

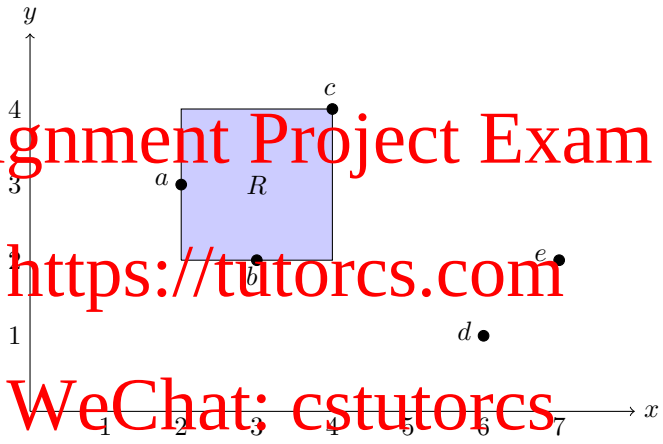
Grigorescu, Darivandpour
Spring 2019

Homework 3 (100 points)

Due: Thursday, February 28, 2019 at 11:59 pm. Upload the pdf file on Blackboard before 11:59 pm; no exceptions will be granted regarding the submission rules. **Read and follow the submission instructions in the syllabus carefully. Make sure you include CR statements for each problem.**

Problem 1 (30 points). An axis-aligned rectangle is a rectangle with edges parallel to the coordinate axes. Given n points with their x and y coordinate values, design an $O(n \log n)$ time Divide-and-Conquer algorithm which finds an axis-aligned rectangle with **minimum perimeter** that covers at least 3 points out of the n given points.

In the following example, $n = 5$, $a = (2, 3)$, $b = (3, 2)$, $c = (4, 4)$, $d = (6, 1)$, and $e = (7, 2)$. The minimum perimeter axis-aligned rectangle R covers a, b, and c. The vertices of R are $(2, 2)$, $(2, 4)$, $(4, 2)$, and $(4, 4)$.



Problem 2 (30 points). You are given k distinct bins arranged in an ordered sequence. Your task is to throw n balls (in order) into these k bins. However, every ball has an associated weight with it. The task is set up in a way such that:

- You may throw a **consecutive** sequence of balls into the same bin.
- Once you move on from bin i, you cannot come back to throw later balls into it.

Define the **load** of a bin as the total weight of all balls in it. Design and analyze an efficient DP algorithm to determine the balls contributing to the load on every bin so as to *minimize the maximum load over all bins*. In your solution, make sure to explicitly state:

1. the optimal substructure and why it is optimal,
2. the DP recurrence formula including base cases,
3. whether your algorithm is bottom-up or top-down DP,
4. the description of your algorithm, and
5. the time and space complexity of your algorithm.

For example, suppose $B = [20, 9, 15, 10, 32]$ are the weights of the balls and $k = 3$, then your algorithm should return $l_1 = [20, 9]$, $l_2 = [15, 10]$, and $l_3 = [32]$, where l_i represents the list of balls corresponding to the load on bin i. The *minimum max load* is 32 for the given input B, and there are no other ways to throw the balls differently so that the minimum max load over all bins is any lower. In case of multiple optimal solutions, your algorithm may output any one of them.

Problem 3 (40 points). You and your friend decide to play a game of Connect 4[1]. After you set up your friend in a trap, they lose the game and get frustrated. In their frustration they propose the following "Disconnect" game to you. The game "Disconnect" is played on the same board as a Connect 4 game but does not have the same rules. Specifically, on a filled-up Disconnect board of size $n \times 7$, you and your opponent will alternate removing pieces from the board and collecting points for them. The "filled-up" board is not necessarily in an end-game position of Connect 4. It could have more than 4 pieces lined up or less than 4 lined up for either color. Also, the number of red pieces may not necessarily equal the number of blue pieces. The player with the most points in the end wins. The rules of "Disconnect" are as follows:

1. A player can remove any piece from the board, G, if and only if no other pieces sit on top of it. Specifically piece $G_{i,j}$ can be removed if and only if pieces $G_{k,j}$ have been removed for all $k > i$ or if $i = n$.

2. If a player removes a piece $G_{i,j}$ of their own color, then they lose 1 point and their opponent gains 1 point.

3. If a player removes a piece $G_{i,j}$ of the opponent's color, then they gain k points, where k is equivalent to the length of the longest *connection* that piece $G_{i,j}$ is a part of at that current state, and their opponent loses k points.

A *connection* is defined as a contiguous straight-line sequence of same-colored pieces (diagonal, horizontal, or vertical). For instance in Figure 2, the blue piece $G_{n,2}$ is a part of two connections: $\{G_{n,2}, G_{n-1,3}, G_{n-2,4}\}$ and $\{G_{n,2}, G_{n,3}\}$. The former being of length 3 and the latter being of length 2.

Figure 1: Example Board Game State for "Disconnect"

Top of Game Board						
$n,1$	$n,2$	$n,3$	$n,4$	$n,5$	$n,6$	$n,7$
$n-1,1$	$n-1,2$	$n-1,3$	$n-1,4$	$n-1,5$	$n-1,6$	$n-1,7$
$n-2,1$	$n-2,2$	$n-2,3$	$n-2,4$	$n-2,5$	$n-2,6$	$n-2,7$
⋮	⋮	⋮	⋮	⋮	⋮	⋮
$2,1$	$2,2$	$2,3$	$2,4$	$2,5$	$2,6$	$2,7$
$1,1$	$1,2$	$1,3$	$1,4$	$1,5$	$1,6$	$1,7$
Bottom of Game Board						

To give a few more examples of points, consider the configuration in Figure 2, where a blank space represents a piece that has been removed in a prior turn. If it is Red's turn, then removing piece:

- $G_{n,1}$ would result in -1 points for Red and +1 points for Blue.

[1]For this problem, you only need to read the first paragraph of the linked article.

- $G_{n,2}$ would result in +3 points for Red and -3 points for Blue.
- $G_{n,3}$ would result in +2 points for Red and -2 points for Blue.
- $G_{n-1,4}$, $G_{n-2,5}$, or $G_{n-1,6}$ would result in -1 points for Red and +1 points for Blue.
- $G_{n,7}$ would result in +2 points for Red and -2 points for Blue.

Your goal is to create an optimal DP solution for Disconnect 4. Specifically, describe and analyze a polynomial-time Dynamic Programming algorithm to determine your optimal final score under the assumption that your opponent also plays optimally. You can assume you are the Red player and you will always go first. In your solution, make sure to explicitly state:

1. the optimal substructure and why it is optimal,
2. the DP recurrence formula including base cases,
3. whether your algorithm is bottom-up or top-down DP,
4. the description of your algorithm, and
5. the time and space complexity of your algorithm.