Homework #1

(Due: Oct 10)

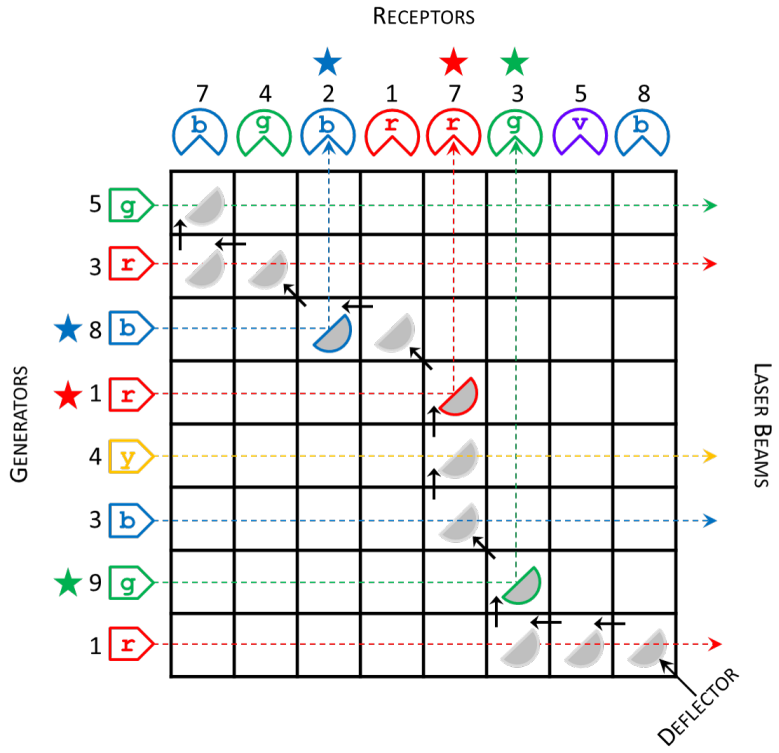


Figure 1: The laser game.

Task 1. [60 Points] Laser Game

Consider the following game played on an $n \times n$ board, where $n = 2^k$ for some integer $k \geq 0$ (see Figure 1). There are three (3) types of game pieces:

Generators. A laser beam generator is placed to the left of each row which generates and directs a laser beam of one of the following colors horizontally to the right: **red** (**r**), **green** (**g**), **blue** (**b**), **yellow** (**y**) and **violet** (**v**). Each generator is worth some points $\in [1, 100]$. The number of available generators is $n_g \gg n$, but only n of them can be used during any given game.

Receptors. At the top of each column a laser receptor is placed which is activated when a laser beam of a specific color (i.e., **r**/**g**/**b**/**y**/**v**) hits its sensor. Each receptor is also worth some points $\in [1, 100]$. During any given game exactly n receptors must be chosen for use from a set of $n_r \gg n$ available receptors.

Deflector. The beam deflector moves through the grid. When it is placed inside a grid cell its mirror surface must be placed on a cell-diagonal so that it faces the top-left corner of that cell. When a horizontal laser beam hits the deflector the beam changes direction and moves upward and hits the receptor placed at the top of the column. Only one (1) deflector is available.

Game Rules.

1. For every row on the board choose a generator uniformly at random from the set of available generators and place it to the left of the row facing horizontally to the right.
2. For every column on the board choose a receptor uniformly at random from the set of available receptors and place it at the top of the column facing vertically downward.
3. The deflector is initially placed inside the cell at the bottom-right corner of the grid.
4. At each step of the play the deflector must be moved to a neighboring cell either to the left or above or diagonally to the top-left of the current cell.
5. The game ends when the deflector reaches the top-left corner cell of the grid.
6. When inside a cell the deflector deflects the beam it receives from the beam generator placed to its left on the same row so that the beam hits the receptor placed at the top of the column. If the deflected laser has the right color (i.e., has the color that the receptor can accept), the receptor is activated.
7. As soon as a receptor becomes activated both that receptor and the activating generator are removed from the board and the player collects all points associated with them.
8. The deflector must never be on an empty row (i.e., without a generator) or column (i.e., without a reflector).
9. The player's goal is to move the deflector from the bottom-right corner of the grid to the top-left corner in a way that maximizes the total number of points he can collect.

Figure 1 shows an example game on an 8×8 game board. The path taken by the deflector allows the player to collect $(9+3)+(1+7)+(8+2) = 30$ points which is not necessarily the highest possible score on this board.

Now please answer the following questions.

(a) [**20 Points**] Given an initial state of the board with all generators and receptors are chosen and placed and the deflector at the bottom-right corner, explain how in $\Theta\left(n^2\right)$ time and space you can find a best deflector path, i.e., a deflector path that maximizes the total number of points collected by the player.

(b) [**5 Points**] Explain how you can generalize your algorithm from part (a) to use $\Theta\left(n^{1+\epsilon}\right)$ space and run in time $\Theta\left(n^{3-\epsilon}\right)$ for any given $\epsilon \in [0,1]$. Observe that when $\epsilon = 0$, this algorithm uses $\Theta(n)$ space, but runs in $\Theta\left(n^3\right)$ time.

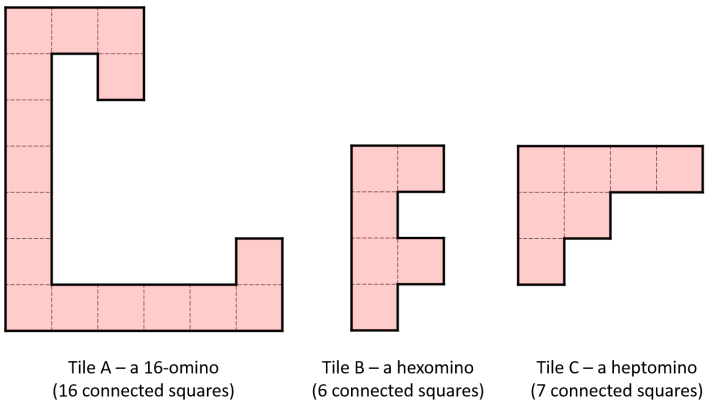
(c) [**35 Points**] Design a recursive divide-and-conquer algorithm to find the best deflector path in $\Theta\left(n^2\right)$ time and $\Theta(n)$ space Find and solve the recurrences for running time and space usage.

Task 2. [50 Points] A Game of Polyominoes

In this task we will consider a game of polyominoes played by two (2) players (**red** and **green**) on an $n \times n$ square board, where $n = 2^k$ for some integer $k \geq 0$. Each game piece is a polyomino which is nothing but a tile that can exactly cover a finite connected subset of the square cells of the game board. In other words, each game piece is a geometric figure formed by putting one or more square cells edge to edge, where each cell is exactly equal to a cell of the board. Each such polyomino will be composed of $m \in [1, n^2]$ square cells, and must also completely fit inside the game board. Figure 2 shows an example.

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

(a) Game board



(b) Game pieces (each piece comes in both **red** and **green**)

Figure 2: An example 8×8 game board with three (3) types of game pieces (i.e., polyominoes).

Each square of the game board has a number written on it which is the number of points a player wins if she can occupy that cell using one of her game pieces (e.g., see Figure 2(a)). Initially, the board is empty (i.e., no one occupies any of its cells), and both players have exactly the same set of polyominoes (i.e., one **red** set and one **green** set).

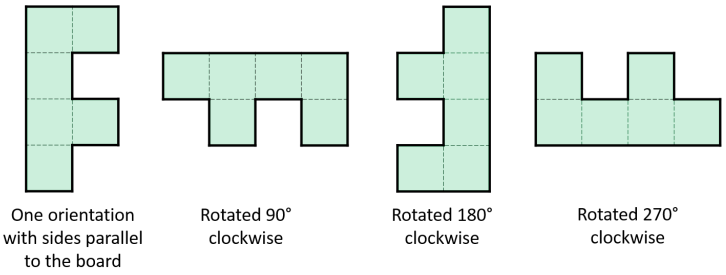


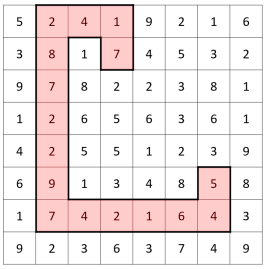
Figure 3: Each game piece can be used in one of four (4) orientations.

The **red** player makes the first move, and then the two players take turns in making the moves. In her turn a player covers a subset of unoccupied cells using one of her game pieces. The newly placed polyomino must not cover an already occupied cell. Though the entire polyomino does not need to

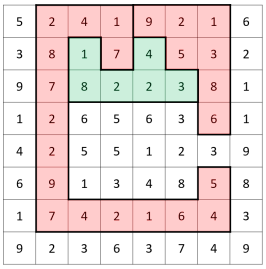
be placed completely inside the board, it must cover at least one board cell. All four orientations of a polyomino that keep its sides parallel to the sides of the board are legal (see Figure 3). The game ends as soon as either both players run out of game pieces or one of the players fails to make a legal move.

When the game ends each player adds up all points written on the cells she has occupied, and the player with the higher total score wins.

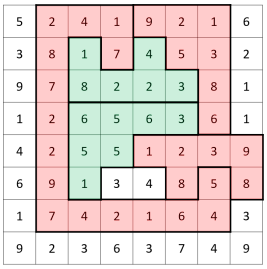
Figure 4 shows an example game in which **red** wins with 136 points while **green** scores only 84 points.



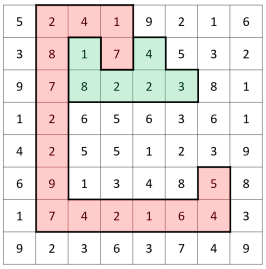
(a) after **red**'s first move



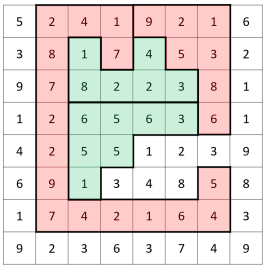
(c) after **red**'s second move



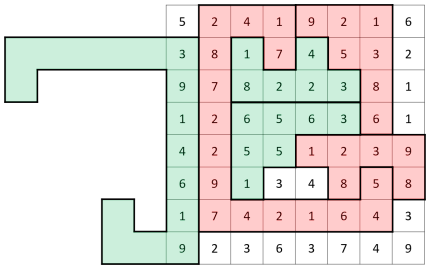
(e) after **red**'s third move



(b) after **green**'s first move



(d) after **green**'s second move



(f) after **green**'s third move

Figure 4: State of the game board after each move made by the players on the game board shown in Figure 2(a) using the game pieces shown in Figure 2(b).

Now please answer the following questions.

(a) [**10 Points**] Suppose you are given an $n \times n$ polyomino game board with a number written on each cell, where $n = 2^k$ for some integer $k \geq 0$. Some of the cells of the board are marked as occupied as a result of moves already made by the players. You are also given a polyomino that the player who is going to make the next move is planning to use. The polyomino can have anywhere between 1 and n^2 square cells, but completely fits inside the game board. You are asked to help the player by finding a location on the board where this game piece can be legally placed and in what orientation so that the total number of points in all cells covered by it is maximized. Show that this problem can be trivially solved in $\mathcal{O}\left(n^4\right)$ time.

(b) [**40 Points**] Show that the problem given in part (a) can be solved in $\mathcal{O}\left(n^2 \log n\right)$ time.

Task 3. [70 Points] Multiplying Fractal Matrices

In this task we will consider matrices in which some of the entries are guaranteed to be zeros, and the cells that may contain nonzero entries form a fractal shape. We will call them *fractal matrices*.

We will only consider square matrices, and assume that for each such $n \times n$ matrix n will be a power of 2, i.e., $n = 2^k$ for some integer $k \geq 1$. Row and column numbers will be from 0 to $n - 1$.

The following two types of fractal matrices will be used.

Δ-Fractal. A cell (i, j) will definitely contain a zero provided the bitwise AND of i and j is nonzero.

∇-Fractal. A cell (i, j) will definitely contain a zero provided the bitwise AND of $(n - i - 1)$ and $(n - j - 1)$ is nonzero.

We will analyze running times of the three matrix multiplication algorithms we have seen in the class, namely the standard iterative algorithm, the standard recursive algorithm and Strassen's algorithm, for computing products XY, where X is a Δ-fractal matrix and Y is either a Δ-fractal or a ∇-fractal or a standard matrix. The major question we will ask is if Strassen's algorithm is still asymptotically faster than the other two algorithms.

(a) [**5 Points**] Draw Δ-fractal matrices of sizes 2×2, 4×4, 8×8 and 16×16, and only mark the cells that are not guaranteed to be zeros leaving all other cells empty. Do the same for ∇-fractal matrices.

(b) [**5 Points**] If X is an $n \times n$ Δ-fractal matrix and Y is an $n \times n$ standard matrix, what is the running time of the standard recursive algorithm for computing XY?

(c) [**10 Points**] How will you modify the layout of the matrices so that the standard iterative algorithm can compute the product in part (b) within the same asymptotic time bound as the standard recursive algorithm?

(d) [**10 Points**] What will be the running time of Strassen's algorithm for computing the product in part (b)?

(e) [**20 Points**] Repeat parts (b)–(d) assuming Y to be another $n \times n$ Δ-fractal matrix.

(f) [**20 Points**] Repeat parts (b)–(d) assuming Y to be an $n \times n$ ∇-fractal matrix.