Data Structures and Algorithms Homework 14

Due Wednesday Dec 11; Joseph Sepich (jps6444)

1 Problem 1

1.1 Part a

The following sequence would be suboptimal to follow the greedy hueristic:

- 4
- 6
- 4
- 4
- 1
- 2

In this case the first player would grab 4, 4, 2, and the second player would grab 6, 4, 1. This gives a score of 10 to 11 in favor of player two.

Collaborators: None

1.2 Part b

To solve this problem we can create a matrix of precomputed values given a certain situation. The rows (indexed with i) will be the "first" card on the deck and the columns (indexed with j) will be the last card on the deck. We have two clear choices for each point in the problem. We choose the first card in the deck, and the opponent chooses the next card to minimize our score, or we choose the last card and the opponent choose the next card to minimize our score. This can be written as the following dynamic programming problem:

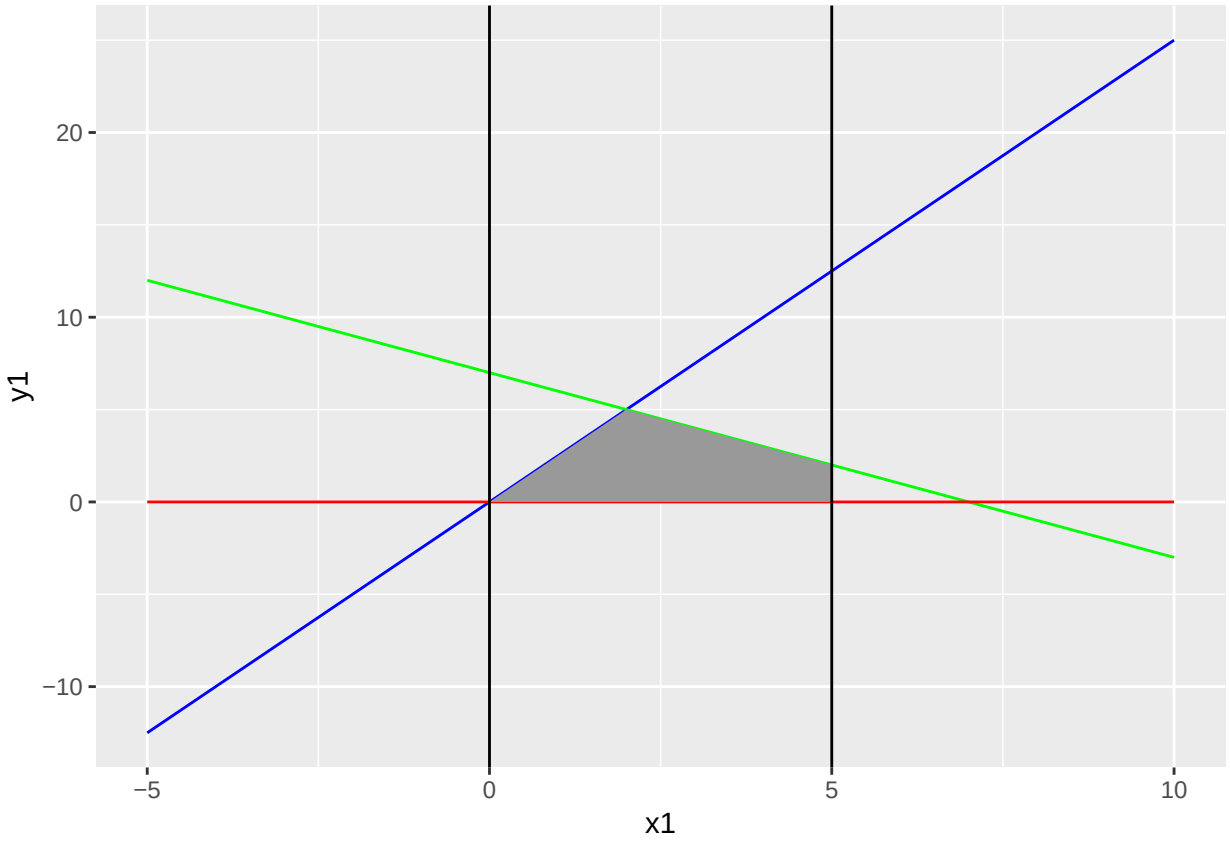
```
def maxValueCards(int[] cards):

    # Create a table to store solutions
    table = [[0 for i in range(n)] for i in range(n)]

    # Fill table
    for diff in range(n): # fill in diagonally up left
        for j in range(diff, n):
            i = j - diff

            # make sure to hop coins -> opponent choice
            # x = table(i+2, j)
            # y = table(i+1, j-1)
            # z = table(i, j-2)
            x = 0
            if((i + 2) <= j):
                x = table[i + 2][j]
            y = 0
            if((i + 1) <= (j - 1)):
                y = table[i + 1][j - 1]
            z = 0
            if(i <= (j - 2)):
                z = table[i][j - 2]
            table[i][j] = max(cards[i] + min(x, y), cards[j] + min(y, z))
    return table[0][n - 1]
```

2 Problem 2 Linear Program



Maximizing the function 5x + 3y, the (5,0) vertex gives a value of 25. Going up the vertical line to the next vertex (5, 2) the value is 31. This value is clearly higher. Going to the interecting vertex (2, 5) the value is 25. Our highest value vertex is (5, 2) with 31. If we go towards the top vertex with (4, 3) we get 29, which is less, and if we go towards the axis vertex with (5, 1) we get 28, which is less. Therefore our optimal solution is 31 with x = 5 and y = 2.

3 Problem 3

The equation for our dual is as follows:

$$(j+k)x + (j+k)y \leq j*3 + k*5$$

So our dual LP would be:

$$\min(3j+5k)$$
$$j+k \geq 1$$
$$j,k \geq 0$$

The minimization of this dual is clearly just j=1 to get a value of 3. If you plug this into the primal LP, an upper bound of the problem is 3. 3 could be obtained by x = 1 and y = 2, which would be the optimal solution, since both the primal and dual LP have the same value. Again this solution would be:

- j = 1
- k = 0
- x = 1
- y = 2
- value = 3

4 Problem 4

We must formulate a linear program to minimize the chance that our components fail. x will be units in compartment 1, y will be compartment 2, and z will be compartment 3. First let us write our objective function:

$$\min(0.3^x * 0.4^y * 0.2^z) = \min(x * ln(0.3) + y * ln(0.4) + z * ln(0.2))$$

Note that in order to find probability of independent random variables we must multiply them together, so the probability of them all failing at once is their product together; however we need a linear function, so we use logarithmic properties to reduce this to a linear problem.

$$\text{Space constraint} : 40x + 50y + 30z \leq 500$$

$$\text{Weight constraint} : 15x + 20y + 10z \leq 200$$

$$\text{Cost constraint} : 30x + 35y + 25z \leq 400$$

$$\text{Probability constraints} : x * ln(0.3), y * ln(0.4), z * ln(0.2) \leq ln(0.05)$$

Now we can solve! If we want it to say maximize, just take the same objective function and multiply by negative one.

5 Probelm 5

5.1 Part 1

We have a variation on the maximum flow problem. Each edge in this case has both a capacity (upper bound) and a lower bound. This implies that and edge must have at least the lower bound flowing through it. Recall that we could set up the vanilla max flow problem as such:

$$\max(\Sigma_{(e \in E)} f_e)$$

$$\text{All edges:} f_e \leq c_e$$

$$\text{All edges:} f_e \geq 0$$

$$\text{All vertices(v):} \Sigma_{(w,v) \in E} f_{(w,v)} - \Sigma_{(v,u) \in E} f_{(v,u)} = 0$$

To adapt this to include a lower bound were merely need to add an additional constraint (where k is the lower bound):

$$\text{All edges:} f_e \geq k; -f_e \leq -k$$

This would be a proper reduction of this variance of the max flow problem.

5.2 Part 2

We have a variation of the maximum flow problem. Each node u in this case has a loss coefficient ε_u. This loss coefficient will adjust our linear programming reduction of our max flow problem. Below is the reduction of the original linear flow problem.

$$\max(\Sigma_{(e \in E)} f_e)$$

$$\text{All edges:} f_e \leq c_e$$

$$\text{All edges:} f_e \geq 0$$

$$\text{All vertices(v):} \Sigma_{(w,v) \in E} f_{(w,v)} - \Sigma_{(v,u) \in E} f_{(v,u)} = 0$$

We do not want the last constraint in our variation, but we want a revised version of it:

$$\text{All vertices(v):} \Sigma_{(w,v) \in E} f_{(w,v)} - \Sigma_{(v,u) \in E} f_{(v,u)} = \varepsilon_u$$

In this revised constraint the difference of flow in and out of a given vertex is not 0, rather the loss coefficient ε_u. This would be a proper reduction of this variance of the max flow problem.