MATH-566 HW 10

Due **Dec 9** before class (regularly). Just bring it before the class and it will be collected there.

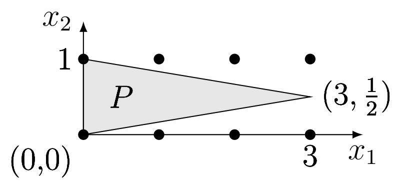
Complete 5 questions.

1: (*Branch and bound*)
Solve the following problem using branch and bound. Draw the branching tree too.

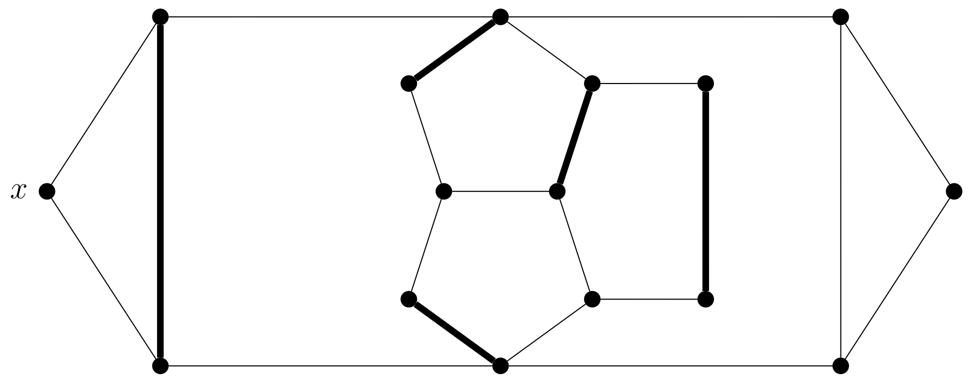
$$(P) = \begin{cases} \text{maximize} & -x_1 + 4x_2 \\ \text{subject to} & -10x_1 + 20x_2 \le 22 \\ & 5x_1 + 10x_2 \le 49 \\ & x_1 \le 5 \\ & x_i \ge 0, x_i \in \mathbb{Z} \text{ for } i \in \{1,2\} \end{cases}$$

You can use any linear programming solver for solving the relaxations.

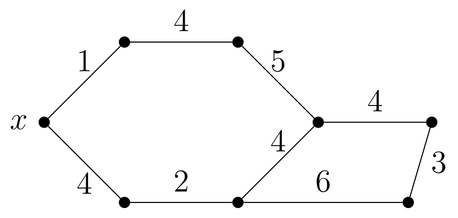
2: (*Cutting planes*)
Let P be a convex hull of $(0,0), (0,1), (k, \frac{1}{2})$. Give an upper bound on Chvátal's rank of P. (Show it is at most $2k$, actually, it is exactly $2k$.)
Hints: Write P as an intersection of half-spaces, use induction *on k. See what we were doing in notes.*
Drawing of P for $k = 3$.



3: (*Can you do Edmonds?*)
Run Edmond's Blossom algorithm on the following graph. Notice that somebody already found a partial matching. What is the largest possible matching? Try to start growing augmenting tree from x, use BFS algorithm for building the tree.



4: (*Can you do weighted bipartite matching?*)
Find minimum-weight perfect matching in the following graph:



A) By using algorithm from class that grows augmenting tree (and keep primal/dual solutions). Start growing x.
B) Formulate the problem using Integer/Linear programming and solve it with your favorite solver.

5: (*Understanding maximum matching*)
Slither is a two-person game played on a graph $G = (V, E)$. The players, called First and Second, play alternatively, with First playing first. At each step the player whose turn it is chooses a previously unchosen edge. The only rule is that at every step the set of chosen edges forms a path. The loser is the first player unable to make a legal move at his or her turn. Prove that if G has a perfect matching, then First can force a win.

6: (*Programming A*)
Implement algorithm for finding maximum matching in bipartite graphs. Test it on the 3D-cube.

7: (*Programming B*)
Implement algorithm for finding maximum matching in any graph.
Test it on the 3D-cube and the graph from question 3.
(Doing this will also solve the previous question - 2 for 1.)