Caleb Logemann
MATH 566 Discrete Optimization
Homework 5

1. Let S be defined as intersection of halfspaces $x_i \geq 0$ and $(1 - x_i)^k \geq 0$. Suppose $i \in \{1, 2, \ldots, d\}$ and $k \geq 1$ is odd. Compute the analytic center of S. Notice that for x satisfying $(1 - x_i)^k \geq 0$, the function $(1 - x_i)^k$ is convex.

   Each of the halfspaces can be rewritten as $-(1 - x_i)^k \leq 0$ and $-x_i \leq 0$, therefore the logarithmic barrier function for this set is

   $$\Phi(\mathbf{x}) = -\sum_{i=1}^{d} \left(\ln\left((1 - x_i)^k\right)\right) - \sum_{i=1}^{d} \left(\ln(x_i)\right)$$

   This can be simplified to

   $$\Phi(\mathbf{x}) = -k\sum_{i=1}^{d} \left(\ln(1 - x_i)\right) - \sum_{i=1}^{d} \left(\ln(x_i)\right)$$

   Now the analytic center of S is the value $\mathbf{x}^*$ which minimizes $\Phi(\mathbf{x})$. This value can be found using calculus, by finding the value of each x_i that makes $\frac{\partial}{\partial x_i}(\Phi) = 0$ respectively.

   $$\begin{aligned}
   \frac{\partial}{\partial x_i}(\Phi) &= -k\frac{1}{x_i - 1} - \frac{1}{x_i} \\
   0 &= -k\frac{1}{x_i - 1} - \frac{1}{x_i} \\
   0 &= \frac{-kx_i - x_1 + 1}{x_i(x_i - 1)} \\
   0 &= -(k + 1)x_i + 1 \\
   x_i &= \frac{1}{k + 1}
   \end{aligned}$$

   This is true for any x_i with $1 \leq i \leq d$. Also since $(1 - x_i)^k$ is convex function we know that this critical point is a minima. Therefore the analytic center of S is $\left(\frac{1}{k+1}, \cdots, \frac{1}{k+1}\right) \in \mathbb{R}^d$.

2. Compute central path for the following problem

   $$(P) \begin{cases} \text{minimize} & -x_1 \\ \text{subject to} & x_1 \leq 1 \\ & x_2 \leq 1 \\ & x_1 \geq 0 \\ & x_2 \geq 0 \end{cases}$$

   and find the optimal solution using the central path. Plot (sketch) the set of feasible solutions and the computed central path. Lot of calculus...

   The constraints can be rewritten as

   $$(P) \begin{cases} \text{minimize} & -x_1 \\ \text{subject to} & x_1 - 1 \leq 0 \\ & x_2 - 1 \leq 0 \\ & -x_1 \leq 0 \\ & -x_2 \leq 0 \end{cases}$$

The logarithmic barrier function for this linear program

$$\Phi(\mathbf{x}) = -\ln(1-x_1) - \ln(1-x_2) - \ln(x_1) - \ln(x_2)$$

The central path can be found minimizing

$$P_t(x_1, x_2) = t(-x_1) + -\ln(1-x_1) - \ln(1-x_2) - \ln(x_1) - \ln(x_2)$$

with respect to x_1 and x_2. In order to do this we need to take the partial derivatives of P_t with respect to x_1 and x_2.

$$\frac{\partial}{\partial x_1}(P_t) = -t + \frac{1}{1-x_1} - \frac{1}{x_1}$$
$$\frac{\partial}{\partial x_2}(P_t) = \frac{1}{1-x_2} - \frac{1}{x_2}$$

Now setting these derivatives equal to zero we can find a parametrized function for the central path.

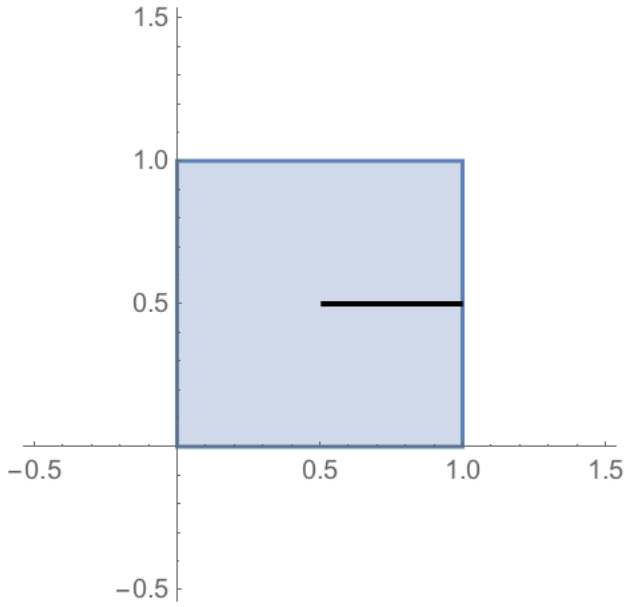
$$\begin{aligned}
\frac{\partial}{\partial x_2}(P_t) &= 0 \\
0 &= \frac{1}{1-x_2} - \frac{1}{x_2} \\
0 &= \frac{x_2 - 1 + x_2}{(1-x_2)x_2} \\
0 &= 2x_2 - 1 \\
x_2 &= \frac{1}{2} \\
\frac{\partial}{\partial x_1}(P_t) &= 0 \\
0 &= -t + \frac{1}{1-x_1} - \frac{1}{x_1} \\
0 &= \frac{-t(1-x_1)x_1 + x_1 - 1 + x_1}{(1-x_1)x_1} \\
0 &= -tx_1^2 + (2-t)x_2 - 1 \\
x_1 &= \frac{t - 2 \pm \sqrt{(2-t)^2 + 4t}}{2t} \\
x_1 &= \frac{1}{2} - \frac{1}{t} \pm \sqrt{\frac{1}{t^2} + \frac{1}{4}}
\end{aligned}$$

For $t \geq 0$, only one of the solution is in the feasible region. Thus the central path is

$$\left(\frac{1}{2} - \frac{1}{t} \pm \sqrt{\frac{1}{t^2} + \frac{1}{4}}, \frac{1}{2}\right)$$

for $t \geq 0$. This central path starts at $\left(\frac{1}{2}, \frac{1}{2}\right)$ and approaches $\left(1, \frac{1}{2}\right)$ as $t \to \infty$.

A plot of the feasible region and central path is shown below.



3. Let $G = (V, E)$ and $|V| = n$.

   Recall that the spanning tree polytope was created by constraints *tree has* $n-1$ *edges* and *tree has no cycles*. Formally,

$$STP = \left\{ \mathbf{x} \in [0,1]^{E(G)} : \sum_{e \in E} x_e = n-1, \sum_{uv \in E, u \in X, v \in X} x_{(u,v)} \leq |X| - 1 \text{ for } \emptyset \subset X \subset V \right\}.$$

   Suppose we try to characterize the spanning tree by assuming that by constraints *tree has* $n-1$ *edges* and *tree is connected*. The tree is connected can be formulated by saying that for every cut, the sum x_e of edges e in the cut is at least one. Formally,

$$P = \left\{ \mathbf{x} \in [0,1]^{E(G)} : \sum_{e \in E} x_e = n-1, \sum_{uv \in E, u \in X, v \notin X} x_{(u,v)} \geq 1 \text{ for } \emptyset \subset X \subset V \right\}.$$

   (a) Prove that the spanning tree polytope is a subset of P. That is, $STP \subseteq P$.

   Proof. Let $x \in STP$, then $\sum_{e \in E}(x_e) = n-1$ and $\sum_{uv \in E, u \in X, v \in X} x_{(u,v)} \leq |X| - 1$ for every $\emptyset \subset X \subset V$. Note that

$$\sum_{uv \in E, u \in X, v \in X} x_{(u,v)} + \sum_{uv \in E, u \in X, v \notin X} x_{(u,v)} = \sum_{e \in E}(x_e)$$

   Simplifying

$$\sum_{uv \in E, u \in X, v \notin X} x_{(u,v)} = n - 1 - \sum_{uv \in E, u \in X, v \in X} x_{(u,v)}$$

   Using $\sum_{uv \in E, u \in X, v \in X} x_{(u,v)} \leq |X| - 1$

$$\begin{aligned} \sum_{uv \in E, u \in X, v \notin X} x_{(u,v)} &\geq n - 1 - |X| + 1 \\ &\geq n - |X| \end{aligned}$$

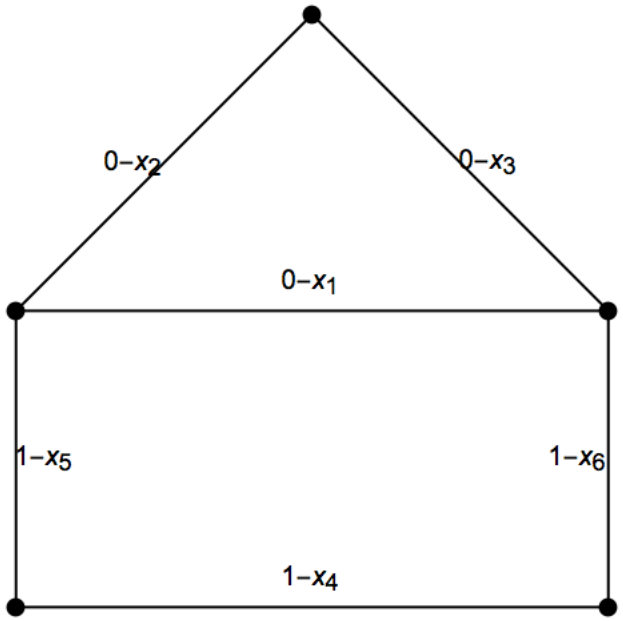
Since $X \subset A$, $|X| \leq n-1$, therefore

$$\sum_{uv \in E, u \in X, v \notin X} x_{(u,v)} \geq 1$$

Thus $x \in P$ as well, and therefore $STP \subset P$. □

(b) Show P does NOT have to be the same as the spanning tree polytope. To do this, show that the polytope P does NOT have to be integral (i.e., P contains a vertex that does not have all coordinates integers).

For this problem, consider the graph with weights shown below.



If we minimize $\sum_{i=1}^{6} (c_i x_i)$ over the polytope P, we will arrive at one of the vertices of P. Note that this linear program is essentially minimize $x_4 + x_5 + x_6$ such that $x_4 + x_5 \geq 1$, $x_5 + x_6 \geq 1$ and $x_4 + x_6 \geq 1$. The values that achieve this minimization are $x_4 = x_5 = x_6 = \frac{1}{2}$. and $x_1 = x_2 = 1$ and Now $x_1 + x_2 + x_3 = 2.5$, so arbitrarily we can let $x_1 = x_2 = 1$ and $x_3 = .5$, since these don't affect the objective function. This is a vertex of the polytope P, but it doesn't contain all integer coordinates. This implies that this is not a vertex of STP as all vertices of STP have integer coordinates. Thus $STP \neq P$.

4. Implement any minimum spanning tree algorithm and test it on random data. You can pick any algorithm you like. You can use ANY programming language but you have to IMPLEMENT the method yourself (calling a library function `RunKruskal` is not acceptable). Obtain data by randomly generating 10 points in range $[0, 10]^2$ and the cost of every edge is the Euclidean distance in $\mathbb{R}^2$. We consider all 45 edges of K_{10}. Finally, create the plot of of the random points and draw edges picked to the spanning tree. You should provide: Name of the algorithm you implemented and short description of implementation, printout of the source code, pictures of two solutions.
Template is provided for Sage, you do not have to use it.
Time complexity DOES NOT matter.

I implemted Kruskal's algorithm in the following function.

```
def kruskal(vertexList, edgeList, costList):
    numVertices = len(vertexList)
    numEdges = len(edgeList)
```

```
    # sort edges by cost
    s = sorted(zip(edgeList, costList), key=lambda pair:pair[1])
    edgeList = [x for (x, y) in s]
    costList = [y for (x, y) in s]

    # create list of edges in tree
    treeEdgeList = []
    # create a list of ids of which tree each vertex is in
    vertexTreeIds = range(numVertices)

    for i in range(numEdges):
        # take edge of minimum cost
        edge = edgeList[i]
        u = edge[0]
        v = edge[1]
        # check if vertices are in the same subtree
        if vertexTreeIds[u] != vertexTreeIds[v]:
            treeEdgeList.append(edge)
            # update tree ids if added edge to spanning tree
            oldId = vertexTreeIds[v]
            for j in range(numVertices):
                if vertexTreeIds[j] == oldId:
                    vertexTreeIds[j] = vertexTreeIds[u]

    return treeEdgeList
```

This function adds minimal edges as long as the edge doesn't connect a subtree to itself. The function does this by keeping track of which vertex is part of which subtree.

The following script runs this function with a random K_{10} graph.

```
# MATH 566 - Minimum spanning tree algorithm
# Notes:
#  - pick any algorithm for minimum spanning three you like
#  - no need to optimize the running time
import itertools as it

# This plots vertices as red dots and blue edges connecting them
def plot_vertices_edges(vertices, edges):
    drawing = line([])
    for x in vertices:
        drawing = drawing + disk(x, 0.1, (0,2*pi), color='red')
    for e in edges:
        drawing = drawing + line([vertices[e[0]], vertices[e[1]]])
    drawing.show()

# Generate 10 random vertices in 10x10 grid
def generate_random_vertices():
    vertexList=[]
    for i in range(10):
```

```
        vertexList.append((random()*10, random()*10))
    return vertexList

vertexList = generate_random_vertices()
edgeList = list(it.combinations(range(10), 2))
costList = []
for edge in edgeList:
    u = vertexList[edge[0]]
    v = vertexList[edge[1]]
    costList.append(sqrt((u[0]-v[0])^2 + (u[1] - v[1])^2))

load('kruskal.sage')
edges = kruskal(vertexList, edgeList, costList)

plot_vertices_edges(vertexList, edges)
```

The following two plots were generated by running this script twice.