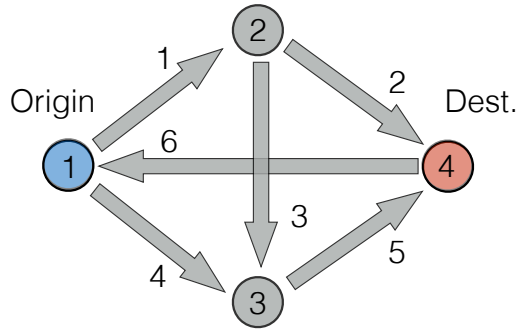
EE578B - Convex Optimization - Winter 2021

Homework 6

Due Date: Wednesday, Feb 24th, 2021 at 11:59 pm

Consider the following graph structure for each of the following problems.



1. **Graph Constraints**

- **(PTS:0-2)** Write down the incidence matrix for the graph, $E \in \mathbb{R}^{4\times 6}$. Write down the source-sink vector for the origin and the destination nodes, $b \in \mathbb{R}^4$.

  Solution:

$$E = \begin{bmatrix} -1 & 0 & 0 & -1 & 0 & 1 \\ 1 & -1 & -1 & 0 & 0 & 0 \\ 0 & 0 & 1 & 1 & -1 & 0 \\ 0 & 1 & 0 & 0 & 1 & -1 \end{bmatrix}, \qquad b = \begin{bmatrix} -1 \\ 0 \\ 0 \\ 1 \end{bmatrix},$$

- **(PTS:0-2)** Write down a route indicator matrix $\mathbf{R}^{6\times 3}$ for the three routes from the origin to the destination that follow the direction of the edges given in the graph and do not include cycles. Compute $E\mathbf{R}$. How does this quantity relate to b?

  Solution:

$$\mathbf{R} = \begin{bmatrix} 1 & 1 & 0 \\ 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \\ 0 & 1 & 1 \\ 0 & 0 & 0 \end{bmatrix}, \qquad E\mathbf{R} = \begin{bmatrix} -1 & -1 & -1 \\ 0 & 0 & 0 \\ 0 & 0 & 0 \\ 1 & 1 & 1 \end{bmatrix} = b\mathbf{1}^T$$

  where $\mathbf{1}^T = [1\ 1\ 1]$

- **(PTS:0-2)** Show that E is not full row-rank. (Hint: compute $\mathbf{1}^T E$ where $\mathbf{1}$ is a vector of 1's.) Show also that $\mathbf{1}^T b = 0$. What does this say about the constraint $Ex = b$?

Solution:

$$\mathbf{1}^T E = \mathbf{0}$$

and thus $\mathbf{1}^T$ is in the left nullspace of E. We also have that $\mathbf{1}^T b = -1 + 1 = 0$. The result of this is that the constraint $Ex = b$ has a redundant row.

- **(PTS:0-2)** Now consider the matrix

$$U = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 1 & 1 & 1 & 1 \end{bmatrix}$$

What is U^{-1}?

Solution:

$$U^{-1} = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ -1 & -1 & -1 & 1 \end{bmatrix}$$

- **(PTS:0-2)** The term $v^T(Ex-b)$ shows up in the Lagrangian of the network flow optimization problem we discussed in class, where $v \in \mathbb{R}^{|\mathcal{S}|}$ can be interpreted as a value function on the nodes. Rewrite this component of the Lagrangian as $v^T(Ex - b) = v'^T(E'x - b')$ where $v'^T = v^T U^{-1}$ What is E'? What is b'? Compute E' and b' for the example given. Intuitively, what do the values of v' represent relative to v? Find the useless row in the equation $E'x = b'$. After you remove this row is the equation full row rank?

Solution:

$$E' = UE = \begin{bmatrix} -1 & 0 & 0 & -1 & 0 & 1 \\ 1 & -1 & -1 & 0 & 0 & 0 \\ 0 & 0 & 1 & 1 & -1 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 \end{bmatrix}, \qquad b' = Ub = \begin{bmatrix} -1 \\ 0 \\ 0 \\ 0 \end{bmatrix}$$

The values of v' are given by

$$v'^T = v^T U^{-1} \begin{bmatrix} v'_1 & v'_2 & v'_3 & v'_4 \end{bmatrix} = \begin{bmatrix} v_1 - v_4 & v_2 - v_4 & v_3 - v_4 & v_4 \end{bmatrix}$$

The coordinate transform $v'^T = v^T U^{-1}$ transforms the dual variables to be measured relative to the dual variable v_4. The transformation U on the constraint $Ex = b$ isolates the redundant row. Last row of E' (and b') can be removed since they are vacuous. The dual variable $v_4 = v'_4$ does not show in the Lagrangian and can be assumed to be any value (though $v_4 = 0$ is the most sensible for v_i to have the interpretation of distance to the destination). Since v_i serves to represent the cost-to-go from each node to the destination the relative value $v_i - v_4$ is what really matters and the value of v_4 is just an offset. The v' coordinates make this fact explicit.

2. **Shortest Path: Explicit Path Enumeration**

Consider the shortest path linear program with the routes enumerated using the routing matrix computed in the previous part.

$$\begin{aligned} \min_{z \in \mathbb{R}^3} \quad & \ell^T z = c^T \mathbf{R} z \\ \text{s.t..} \quad & \mathbf{1}^T z = 1,\ z \geq 0 \end{aligned} \tag{1}$$

- **(PTS:0-2)** What does z represent? What does each element of ℓ represent?

  Solution: z_r represents the mass assigned to each route. $c \in \mathbb{R}^6$ is the cost of traveling on the edges of the graph. ℓ_r is then the cost on route r which is the sum of the edge costs along that route

$$\ell^T = c^T \mathbf{R}, \quad \Rightarrow \quad \ell_r = \sum_{e \in r} c_e$$

$$\begin{bmatrix} \ell_1 & \ell_2 & \ell_3 \end{bmatrix} = \begin{bmatrix} c_1 & c_2 & c_3 & c_4 & c_5 & c_6 \end{bmatrix} \begin{bmatrix} 1 & 1 & 0 \\ 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \\ 0 & 1 & 1 \\ 0 & 0 & 0 \end{bmatrix} = \begin{bmatrix} c_1 + c_2 & c_1 + c_3 + c_5 & c_4 + c_5 \end{bmatrix}$$

- **(PTS:0-2)** Write the dual of this optimization problem using $\lambda \in \mathbb{R}$ for the summation constraint and $u \in \mathbb{R}^3_+$ for the positivity constraint. What does λ represent? What does each element of u represent?

  Solution:

  The Lagrangian is given by

$$\mathcal{L}(z, \lambda, u) = \ell^T z - \lambda(\mathbf{1}^T z - 1) - u^T z$$

  The dual optimization problem is given by

$$\begin{aligned} \max_{\lambda, u} \quad & \lambda \\ \text{s.t.} \quad & \lambda \mathbf{1}^T = \ell^T - u^T,\ u \geq 0 \end{aligned}$$

  λ represents a lower bound on the travel cost in general and the minimum travel cost at optimum. u_r at optimum represents the inefficiency of route r.

- **(PTS:0-2)** At optimum the complementary slackness condition is given by $z_i u_i = 0$ for $z_i \geq 0$, $u_i \geq 0$. Given the meaning of z_i and u_i, how does this condition intuitively ensure that the mass distribution vector z puts all mass on the optimal route?

Solution:

The complementary slackness condition $u_r z_r = 0$ implies that at least one of u_r or z_r is 0 at optimum. In English, this can be interpreted as if a route is inefficient, ie. $u_r > 0$, then no mass is assigned to it, $z_r = 0$. Mass is only assigned to efficient routes, ie. $z_r > 0$ only if $u_r = 0$.

3. **Shortest Path: Edge Formulation**

Consider the shortest path linear program using the incidence matrix.

$$\begin{aligned} \min_{x \in \mathbb{R}^6} \quad & c^T x \\ \text{s.t..} \quad & Ex = b,\ x \geq 0 \end{aligned} \tag{2}$$

for $c \in \mathbb{R}^6$.

- **(PTS:0-2)** Write the Lagrangian of this optimization problem using $v \in \mathbb{R}^{|\mathcal{S}|}$ as the dual variable for the equality constraint and $\mu \in \mathbb{R}^{|\mathcal{E}|}$ as the dual variable for the positivity constraint.

  Solution:

$$\mathcal{L}(x, v, \mu) = c^T x + v^T (Ex - b) - \mu^T x$$

- **(PTS:0-2)** Compute the KKT (optimality) conditions for this problem

$$\frac{\partial \mathcal{L}}{\partial x} = 0, \quad \frac{\partial \mathcal{L}}{\partial v} = 0, \quad x \geq 0, \quad \mu \geq 0, \quad \mu^T x = 0$$

  Solution: The KKT optimality conditions are given by

$$\begin{aligned} \frac{\partial \mathcal{L}}{\partial x} &= c^T + v^T E - \mu^T = 0 \\ \frac{\partial \mathcal{L}}{\partial v} &= Ex - b = 0 \\ x \geq 0, \quad \mu \geq 0, \quad & \mu^T x = 0 \end{aligned}$$

  Element-wise, the first condition can be written as

$$v_o = c_e + v_i - \mu_e = 0 \tag{3}$$

  for edge e traveling out of node o and into node i. Again c_e is the cost on edge e, μ_e is the inefficiency of edge e, v_i is the cost to go from node i, and v_o is the cost to go from node o. The optimality condition relates v_o to the edge costs and future cost to go.

- **(PTS:0-2)** Use the KKT conditions at optimum to show that for any route from the origin to the destination the total (summed) travel cost is greater than or equal to the minimum travel cost
- **(PTS:0-2)** Use the KKT conditions to show that, at optimum, mass only chooses optimal routes.

  Solution:

For a given route from origin to destination r, summing up Equation (3) along the route gives

$$v_{\text{orig}} - v_{\text{dest}} = \sum_{e \in r} c_e - \sum_{e \in r} \mu_e$$

Note that the quantity $v_{\text{orig}} - v_{\text{dest}}$ is path independent and that the cost of a route is given by

$$\ell_r = \sum_{e \in r} c_e = v_{\text{orig}} - v_{\text{dest}} + \sum_{e \in r} \mu_e \geq v_{\text{orig}} - v_{\text{dest}}$$

Note that the optimal route r^* will have mass on each edge, ie. $x_e > 0$ for every $e \in r^*$ and thus $\mu_e = 0$ for all $e \in r^*$. It follows that

$$\ell_{r^*} = \sum_{e \in r^*} c_e = v_{\text{orig}} - v_{\text{dest}}$$

and it follows that

$$\ell_r = \sum_{e \in r} c_e \geq \sum_{e \in r^*} c_e = \ell_{r^*}$$

These computations can be done in compact form for two edge flow vectors corresponding to the optimal mass flow x^* and any other feasible mass flow x.

$$\begin{aligned} c^T x + v^T E x - \mu^T x &= 0 \\ c^T x + v^T b - \mu^T x &= 0 \end{aligned}$$

For x^*, $\mu^T x^* = 0$ and thus

$$c^T x = -v^T b + \mu^T x \geq -v^T b = c^T x^*$$

- **(PTS:0-2)** Derive the dual linear program. (You should end up with something closely related to the following.)

$$\begin{aligned} \max_{v} \quad & v^T b \\ \text{s.t.} \quad & v^T E \leq c^T \end{aligned}$$

Solution:

The game form of the primal problem is given by

$$\min_x \max_{v, \mu \geq 0} \; \mathcal{L}(x, v, \mu) = \min_x \max_{v, \mu \geq 0} \; c^T x + v^T (Ex - b) - \mu^T x$$

The dual problem is given by

$$\max_{v, \mu \geq 0} \min_x \; \mathcal{L}(x, v, \mu) = \max_{v, \mu \geq 0} \min_x \; c^T x + v^T (Ex - b) - \mu^T x$$

Solving the inner problem gives the dual program as

$$\begin{aligned} \max_{v,\mu} \quad & -v^T b \\ \text{s.t.} \quad & c^T + v^T E - \mu^T = 0,\ \mu \geq 0 \end{aligned}$$

or

$$\begin{aligned} \max_{v} \quad & -v^T b \\ \text{s.t.} \quad & c^T + v^T E \geq 0 \end{aligned}$$

- **(PTS:0-2)** Rewrite this linear program in terms of v', E', and b' from the first problem. Take a stab at interpreting the meaning of the constraints and objective in this reformulated linear program. How does this relate to the simplex optimization program we solved last week?

  Solution: Applying the coordinate transformation $v'^T = v^T U^{-1}$ to the dual program gives

$$\begin{aligned} \max_{v} \quad & -v'^T U b \\ \text{s.t.} \quad & c^T + v'^T U E \geq 0 \end{aligned}$$

  Plugging in U gives

$$\begin{aligned} \max_{v} \quad & v'_{\text{orig}} \\ \text{s.t.} \quad & v'_o \leq c_e + v'_i, \quad \text{for ea. edge } e \text{ from } o \text{ to } i \end{aligned}$$

  The constraint indicates that the optimal cost-to-go (relative to the destination) from each node v'_o must be less than the cost to go from each subsequent node v_i plus the cost to reach that node along edge e, c_e. The dual optimization problem then tries to raise the cost-to-go from the origin as much as possible while maintaining this constraint. The minimum cost route is the first route that stops the cost-to-go from the origin from increasing anymore. The route choices parallel the corners of the simplex in the simplex optimization problem and the dual optimization problem maximizes the lower bound on the cost of each route.

4. **Numerical optimization**

- **(PTS:0-4)** Use `cvx` or `cvxpy` to solve Problem (1) and it's dual for the cost vectors $c^T = [1\ 3\ 1\ 3\ 1\ 1]$ and $c^T = [1\ 2\ 1\ 3\ 1\ 1]$ and the graph structure given above. What are the optimal primal and dual variables for each case? Describe the meaning of each of these values intuitively. How are the two scenarios different?

```
import numpy as np
import cvxpy as cp

n = 3;
c1 = np.array([1,3,1,3,1,1]);
c2 = np.array([1,2,1,3,1,1]);
R  = np.array([[1.,1.,0.],
               [1.,0.,0.],
               [0.,1.,0.],
               [0.,0.,1.],
               [0.,1.,1.],
               [0.,0.,0.]])
l1 = c1@R;
l2 = c2@R;
l1 = l1;
# Define and solve the CVXPY problem.
z = cp.Variable(n)
obj = c1.T@R@z;
constraints = [np.ones(3)@z == 1, z >= 0];
primal = cp.Problem(cp.Minimize(obj),constraints)
primal.solve()

print('Primal Problem: ')
print("The optimal value for the primal problem is ", np.round(primal.value,4))

print('Route cost vector (l): ', l1)
print("Optimal z: ",np.round(z.value,4), '(optimal mass distribution - all mass on minimum cost route)
print('Equality constraint dual variable: ',np.round(primal.constraints[0].dual_value,4), '(optimal t
print('Inequality constraint dual variable: ',np.round(primal.constraints[1].dual_value,4), '(ineff. c


lam = cp.Variable(1);
u   = cp.Variable(3);
dual_obj = lam;
dual_constraints = [lam*np.ones(3) + u == l1, u >= 0]
dual = cp.Problem(cp.Maximize(dual_obj),dual_constraints);
dual.solve();


print(' ');print(' ');
print('Dual Problem: ')
print("The optimal value for the dual problem is ", np.round(dual.value,4))
print("Optimal lam: ",np.round(lam.value,4),' (same as the equality dual variable from the primal prob
print("Optimal u: ",np.round(u.value,4), ' (same as the inequality dual variable from the primal prob.
print('Equality dual variable: ',np.round(dual.constraints[0].dual_value,4), ' (same as the previous p
print('Inequality dual variable: ',np.round(dual.constraints[1].dual_value,4), ' (same as the previous
```

```
Primal Problem:
The optimal value for the primal problem is  3.0
Route cost vector (l):  [4. 3. 4.]
Optimal z:  [0. 1. 0.] (optimal mass distribution - all mass on minimum cost route)
Equality constraint dual variable:  -3.0 (optimal travel cost (up to a sign))
Inequality constraint dual variable:  [1. 0. 1.] (ineff. of each route.)


Dual Problem:
The optimal value for the dual problem is  3.0
Optimal lam:  [3.]  (same as the equality dual variable from the primal prob.)
Optimal u:  [ 1. -0.  1.]  (same as the inequality dual variable from the primal prob.)
Equality dual variable:  [0. 1. 0.]  (same as the previous primal variable)
Inequality dual variable:  [0. 1. 0.]  (same as the previous primal variable)
```

```
import numpy as np
import cvxpy as cp

n = 3;
c1 = np.array([1,3,1,3,1,1]);
c2 = np.array([1,2,1,3,1,1]);
R  = np.array([[1.,1.,0.],
               [1.,0.,0.],
               [0.,1.,0.],
               [0.,0.,1.],
               [0.,1.,1.],
               [0.,0.,0.]])
l1 = c1@R;
l2 = c2@R;
l1 = l2;
# Define and solve the CVXPY problem.
z = cp.Variable(n)
obj = c2.T@R@z;
constraints = [np.ones(3)@z == 1, z >= 0];
primal = cp.Problem(cp.Minimize(obj),constraints)
primal.solve()

print('Primal Problem: ')
print("The optimal value for the primal problem is ", np.round(primal.value,4))

print('Route cost vector (l): ', l1)
print("Optimal z: ",np.round(z.value,4), '(optimal mass distribution - distributed over rou
print('Equality constraint dual variable: ',np.round(primal.constraints[0].dual_value,4), '
print('Inequality constraint dual variable: ',np.round(primal.constraints[1].dual_value,4),


lam = cp.Variable(1);
u  = cp.Variable(3);
dual_obj = lam;
dual_constraints = [lam*np.ones(3) + u == l1, u >= 0]
dual = cp.Problem(cp.Maximize(dual_obj),dual_constraints);
dual.solve();


print(' ');print(' ');
print('Dual Problem: ')
print("The optimal value for the dual problem is ", np.round(dual.value,4))
print("Optimal lam: ",np.round(lam.value,4),' (same as the equality dual variable from the
print("Optimal u: ",np.round(u.value,4), ' (same as the inequality dual variable from the p
print('Equality dual variable: ',np.round(dual.constraints[0].dual_value,4), ' (same as the
print('Inequality dual variable: ',np.round(dual.constraints[1].dual_value,4), ' (same as t
```

```
Primal Problem:
The optimal value for the primal problem is  3.0
Route cost vector (l):  [3. 3. 4.]
Optimal z:  [0.5 0.5 0. ] (optimal mass distribution - distributed over routes 1 and 2
Equality constraint dual variable:  -3.0 (optimal travel cost (up to a sign))
Inequality constraint dual variable:  [0. 0. 1.] (ineff. of each route.)


Dual Problem:
The optimal value for the dual problem is  3.0
Optimal lam:  [3.]  (same as the equality dual variable from the primal prob.)
Optimal u:  [-0. -0.  1.]  (same as the inequality dual variable from the primal prob.)
Equality dual variable:  [0.5 0.5 0. ]  (same as the previous primal variable)
Inequality dual variable:  [0.5 0.5 0. ]  (same as the previous primal variable)
```

- **(PTS:0-4)** Use `cvx` or `cvxpy` to solve Problem (2) and it's dual for the cost vectors $c^T = [1\ 3\ 1\ 3\ 1\ 1]$ and $c^T = [1\ 2\ 1\ 3\ 1\ 1]$ and the graph structure given above. What are the optimal primal and dual variables for each case? Describe the meaning of each of these values intuitively. Again, how are the two scenarios different?

  Note: You may have to use the constraint $E'x = b'$ if `cvx` chokes on the fact that the rows of E are not linearly independent. You can try both tho.

```
import numpy as np
import cvxpy as cp


n = 6;
cl = np.array([1,3,1,3,1,1]);
E = np.array([[-1. , 0. , 0. , -1., 0. , 1.],
              [ 1. , -1., -1., 0. , 0. , 0.],
              [ 0. , 0. , 1. , 1. , -1., 0.],
              [ 0. , 1. , 0. , 0. ,  1. ,-1.]])
b = np.array([-1.,0.,0.,1.]);
x = cp.Variable(n)
obj = cl.T@x;
constraints = [E@x==b, x >= 0];
primal = cp.Problem(cp.Minimize(obj),constraints)
primal.solve()
print('Primal Problem: ')
print("The optimal value for the primal problem is ", np.round(primal.value,4))
print('Edge cost vector (c): ', cl)
print("Optimal x: ",np.round(x.value,4), '(optimal mass distribution over edges')
print('Equality constraint dual variable: ',np.round(primal.constraints[0].dual_value,4), 'value functic
print('Inequality constraint dual variable: ',np.round(primal.constraints[1].dual_value,4), 'ineff. of e
print('')
print('removing the last row of E and b forces v_4 to 0...')
print('')
E = np.array([[-1. , 0. , 0. , -1., 0. , 1.],
              [ 1. , -1., -1., 0. , 0. , 0.],
              [ 0. , 0. , 1. , 1. , -1., 0.]]);
b = np.array([-1.,0.,0.]);
x = cp.Variable(n)
obj = cl.T@x;
constraints = [E@x==b, x >= 0];
primal = cp.Problem(cp.Minimize(obj),constraints)
primal.solve()
print('Primal Problem: ')
print("The optimal value for the primal problem is ", np.round(primal.value,4))
print('Edge cost vector (c): ', cl)
print("Optimal x: ",np.round(x.value,4), '(optimal mass distribution over edges')
print('Equality constraint dual variable: ',np.round(primal.constraints[0].dual_value,4), 'value functic
print('Inequality constraint dual variable: ',np.round(primal.constraints[1].dual_value,4), 'ineff. of e
```

```
Primal Problem:
The optimal value for the primal problem is  3.0
Edge cost vector (c):  [1 3 1 3 1 1]
Optimal x:  [ 1.  0.  1.  0.  1. -0.] (optimal mass distribution over edges
Equality constraint dual variable:  [ 1.5  0.5 -0.5 -1.5] value function on the nodes (relative to node 4)
Inequality constraint dual variable:  [0. 1. 0. 1. 0. 4.] ineff. of each edge

removing the last row of E and b forces v_4 to 0...

Primal Problem:
The optimal value for the primal problem is  3.0
Edge cost vector (c):  [1 3 1 3 1 1]
Optimal x:  [ 1.  0.  1.  0.  1. -0.] (optimal mass distribution over edges
Equality constraint dual variable:  [3. 2. 1.] value function on the nodes (relative to node 4)
Inequality constraint dual variable:  [0. 1. 0. 1. 0. 4.] ineff. of each edge
```

```
p.array([1,3,1,3,1,1]);
.array([[-1. , 0. , 0. , -1., 0. , 1.],
        [ 1. , -1., -1., 0. , 0. , 0.],
        [ 0. , 0. , 1. , 1. , -1., 0.],
        [ 0. , 1. , 0. , 0. ,  1. ,-1.]])
.array([-1.,0.,0.,1.]);
.Variable(4);
cp.Variable(6);
bj = -v@b;
onstraints = [cl + v@E - mu == 0, mu >= 0]
 cp.Problem(cp.Maximize(dual_obj),dual_constraints);
olve();
' ');print(' ');
'Dual Problem: ')
"The optimal value for the dual problem is -v@b ", np.round(dual.value,2))
"Optimal v: ",np.round(v.value,2),' (same as the equality dual variable from the primal prob.)')
"Optimal mu: ",np.round(mu.value,2), ' (same as the inequality dual variable from the primal prob.)')
'Equality dual variable: ',np.round(dual.constraints[0].dual_value,2), ' (same as the previous primal vari
'Inequality dual variable: ',np.round(dual.constraints[1].dual_value,2), ' (same as the previous primal va
'');print('')
'removing the last row of E and b forces v_4 to 0...')
'')
.array([[-1. , 0. , 0. , -1., 0. , 1.],
        [ 1. , -1., -1., 0. , 0. , 0.],
        [ 0. , 0. , 1. , 1. , -1., 0.]]);
.array([-1.,0.,0.]);
.Variable(3);
cp.Variable(6);
bj = -v@b;
onstraints = [cl + v@E - mu == 0, mu >= 0]
 cp.Problem(cp.Maximize(dual_obj),dual_constraints);
olve();
'Dual Problem: ')
"The optimal value for the dual problem is -v@b ", np.round(dual.value,2))
"Optimal v: ",np.round(v.value,2),' (same as the equality dual variable from the primal prob. with v_4 for
"Optimal mu: ",np.round(mu.value,2), ' (same as the inequality dual variable from the primal prob.)')
'Equality dual variable: ',np.round(dual.constraints[0].dual_value,2), ' (same as the previous primal vari
'Inequality dual variable: ',np.round(dual.constraints[1].dual_value,2), ' (same as the previous primal va
```

```
Dual Problem:
The optimal value for the dual problem is -v@b  3.0
Optimal v:  [ 1.5  0.5 -0.5 -1.5]  (same as the equality dual variable from the primal prob.)
Optimal mu:  [ 0.  1. -0.  1.  0.  4.]  (same as the inequality dual variable from the primal prob.)
Equality dual variable:  [-1. -0. -1. -0. -1.  0.]  (same as the previous primal variable - up to the sign)
Inequality dual variable:  [1. 0. 1. 0. 1. 0.]  (same as the previous primal variable)


removing the last row of E and b forces v_4 to 0...

Dual Problem:
The optimal value for the dual problem is -v@b  3.0
Optimal v:  [3. 2. 1.]  (same as the equality dual variable from the primal prob. with v_4 forced to be 0)
Optimal mu:  [ 0.  1. -0.  1.  0.  4.]  (same as the inequality dual variable from the primal prob.)
Equality dual variable:  [-1. -0. -1. -0. -1.  0.]  (same as the previous primal variable - up to the sign)
Inequality dual variable:  [1. 0. 1. 0. 1. 0.]  (same as the previous primal variable)
```

```
import numpy as np
import cvxpy as cp
n = 6;
c2 = np.array([1,2,1,3,1,1]);
E = np.array([[-1. , 0. , 0. , -1., 0. , 1.],
              [ 1. , -1., -1., 0. , 0. , 0.],
              [ 0. , 0. , 1. , 1. , -1., 0.],
              [ 0. , 1. , 0. , 0. ,  1. ,-1.]])
b = np.array([-1.,0.,0.,1.]);
x = cp.Variable(n)
obj = c2.T@x;
constraints = [E@x==b, x >= 0];
primal = cp.Problem(cp.Minimize(obj),constraints)
primal.solve()
print('Primal Problem: ')
print("The optimal value for the primal problem is ", np.round(primal.value,2))
print('Edge cost vector (c): ', c2)
print("Optimal x: ",np.round(x.value,2), 'optimal mass distribution over edges - flow on routes 1 and 2...')
print('Equality constraint dual variable: ',np.round(primal.constraints[0].dual_value,2), 'value function on th
print('Inequality constraint dual variable: ',np.round(primal.constraints[1].dual_value,2), 'ineff. of each edg
print('')
print('removing the last row of E and b forces v_4 to 0...')
print('')
E = np.array([[-1. , 0. , 0. , -1., 0. , 1.],
              [ 1. , -1., -1., 0. , 0. , 0.],
              [ 0. , 0. , 1. , 1. , -1., 0.]]);
b = np.array([-1.,0.,0.]);
x = cp.Variable(n)
obj = c2.T@x;
constraints = [E@x==b, x >= 0];
primal = cp.Problem(cp.Minimize(obj),constraints)
primal.solve()

print('Primal Problem: ')
print("The optimal value for the primal problem is ", np.round(primal.value,2))

print('Edge cost vector (c): ', c2)
print("Optimal x: ",np.round(x.value,2), 'optimal mass distribution over edges - flow on routes 1 and 2...')
print('Equality constraint dual variable: ',np.round(primal.constraints[0].dual_value,2), 'value function on th
print('Inequality constraint dual variable: ',np.round(primal.constraints[1].dual_value,2), 'ineff. of each edg
```

```
Primal Problem:
The optimal value for the primal problem is  3.0
Edge cost vector (c):  [1 2 1 3 1 1]
Optimal x:  [ 1.    0.47  0.53  0.    0.53 -0.  ] optimal mass distribution over edges - flow on routes 1 and 2...
Equality constraint dual variable:  [ 1.5  0.5 -0.5 -1.5] value function on the nodes (relative to node 4)
Inequality constraint dual variable:  [0. 0. 0. 1. 0. 4.] ineff. of each edge + v_4

removing the last row of E and b forces v_4 to 0...

Primal Problem:
The optimal value for the primal problem is  3.0
Edge cost vector (c):  [1 2 1 3 1 1]
Optimal x:  [ 1.    0.47  0.53  0.    0.53 -0.  ] optimal mass distribution over edges - flow on routes 1 and 2...
Equality constraint dual variable:  [3. 2. 1.] value function on the nodes (relative to node 4)
Inequality constraint dual variable:  [0. 0. 0. 1. 0. 4.] ineff. of each edge
```

```
.array([1,2,1,3,1,1]);
array([[-1. , 0. , 0. , -1., 0. , 1.],
       [ 1. , -1., -1., 0. , 0. , 0.],
       [ 0. , 0. , 1. , 1. , -1., 0.],
       [ 0. , 1. , 0. , 0. ,  1. ,-1.]])
array([-1.,0.,0.,1.]);


Variable(4);
p.Variable(6);
j = -v@b;
nstraints = [c2 + v@E - mu == 0, mu >= 0]
cp.Problem(cp.Maximize(dual_obj),dual_constraints);
lve();


 ');print(' ');
Dual Problem: ')
The optimal value for the dual problem is -v@b ", np.round(dual.value,4))
Optimal v: ",np.round(v.value,2),' (same as the equality dual variable from the primal prob.)')
Optimal mu: ",np.round(mu.value,2), ' (same as the inequality dual variable from the primal prob.)')
Equality dual variable: ',np.round(dual.constraints[0].dual_value,2))
Inequality dual variable: ',np.round(dual.constraints[1].dual_value,2))
');print('')
removing the last row of E and b forces v_4 to 0...')
')
array([[-1. , 0. , 0. , -1., 0. , 1.],
       [ 1. , -1., -1., 0. , 0. , 0.],
       [ 0. , 0. , 1. , 1. , -1., 0.]]);
array([-1.,0.,0.]);
Variable(3);
p.Variable(6);
j = -v@b;
nstraints = [c2 + v@E - mu == 0, mu >= 0]
cp.Problem(cp.Maximize(dual_obj),dual_constraints);
lve();
Dual Problem: ')
The optimal value for the dual problem is -v@b ", np.round(dual.value,2))
Optimal v: ",np.round(v.value,2),' (same as the equality dual variable from the primal prob. with v_4 forced to be
Optimal mu: ",np.round(mu.value,2), ' (same as the inequality dual variable from the primal prob.)')
Equality dual variable: ',np.round(dual.constraints[0].dual_value,2), ' (same as the previous primal variable - up
Inequality dual variable: ',np.round(dual.constraints[1].dual_value,2), ' (same as the previous primal variable)')
```

```
Dual Problem:
The optimal value for the dual problem is -v@b  3.0
Optimal v:  [ 1.5  0.5 -0.5 -1.5]  (same as the equality dual variable from the primal prob.)
Optimal mu:  [ 0.  0. -0.  1.  0.  4.]  (same as the inequality dual variable from the primal prob.)
Equality dual variable:  [-1.   -0.47 -0.53 -0.   -0.53  0.  ]
Inequality dual variable:  [1.   0.47 0.53 0.   0.53 0.  ]


removing the last row of E and b forces v_4 to 0...

Dual Problem:
The optimal value for the dual problem is -v@b  3.0
Optimal v:  [3. 2. 1.]  (same as the equality dual variable from the primal prob. with v_4 forced to be 0)
Optimal mu:  [ 0.  0. -0.  1.  0.  4.]  (same as the inequality dual variable from the primal prob.)
Equality dual variable:  [-1.   -0.47 -0.53 -0.   -0.53  0.  ]  (same as the previous primal variable - up to the sig
n)
Inequality dual variable:  [1.   0.47 0.53 0.   0.53 0.  ]  (same as the previous primal variable)
```