prob 1

(a) infeasible, the screen shot is below.

(b) c_{ij} : the cost of edge (i,j) x_{ij} : the flow on edge (i,j) Δ_{ij} : the delta on edge (i,j)

$$\min \sum_{i=1}^{5} \sum_{j=6}^{9} c_{ij} \, \Delta_{ij} \qquad i = 1, 2, 3, 4, 5 \quad j = 6, 7, 8, 9$$

s.t.

$$x_{16} + x_{17} + x_{18} + x_{19} = 208$$
$$x_{26} + x_{27} + x_{28} + x_{29} = 193$$
$$x_{36} + x_{37} + x_{38} + x_{39} = 195$$
$$x_{46} + x_{47} + x_{48} + x_{49} = 209$$
$$x_{56} + x_{57} + x_{58} + x_{59} = 4031$$

$$-(x_{16} + x_{26} + x_{36} + x_{46} + x_{56}) = -1530$$
$$-(x_{17} + x_{27} + x_{37} + x_{47} + x_{57}) = -1583$$
$$-(x_{18} + x_{28} + x_{38} + x_{48} + x_{58}) = -1562$$
$$-(x_{19} + x_{29} + x_{39} + x_{49} + x_{59}) = -161.$$

$$x_{16} \le 7407 + \Delta_{16}$$
$$x_{17} \le 3546 + \Delta_{17}$$
$$x_{18} \le 5072 + \Delta_{18}$$
$$x_{19} \le 1932 + \Delta_{19}$$
$$x_{26} \le 87 + \Delta_{26}$$
$$x_{27} \le 90 + \Delta_{27}$$
$$x_{28} \le 29 + \Delta_{28}$$
$$x_{29} \le 902 + \Delta_{29}$$
$$x_{36} \le 13 + \Delta_{36}$$
$$x_{37} \le 8413 + \Delta_{37}$$
$$x_{38} \le 8719 + \Delta_{38}$$
$$x_{39} \le 7439 + \Delta_{39}$$
$$x_{46} \le 5047 + \Delta_{46}$$
$$x_{47} \le 83 + \Delta_{47}$$
$$x_{48} \le 58 + \Delta_{48}$$
$$x_{49} \le 76 + \Delta_{49}$$
$$x_{56} \le 83 + \Delta_{56}$$
$$x_{57} \le 7904 + \Delta_{57}$$
$$x_{58} \le 73 + \Delta_{58}$$
$$x_{59} \le 65 + \Delta_{59}$$

$x_{ij} \ge 0, \ \forall i, \forall j.$

$\Delta_{ij} \ge 0 \ \ \forall i, \forall j.$

solved by gurobi:

obj value = 1749022

$\Delta_{28} = 144$

$\Delta_{56} = 1372$

$\Delta_{58} = 855$

otherwise $= 0$.

#

prob 1 (continuous).

©.

$$\min \sum_{i=1}^{5}\sum_{j=6}^{9} x_{ij}\, c_{ij} \qquad i = 1,2,3,4,5 \quad j = 6,7,8,9$$

s.t.

$$x_{16} + x_{17} + x_{18} + x_{19} = 208$$
$$x_{26} + x_{27} + x_{28} + x_{29} = 193$$
$$x_{36} + x_{37} + x_{38} + x_{39} = 195$$
$$x_{46} + x_{47} + x_{48} + x_{49} = 209$$
$$x_{56} + x_{57} + x_{58} + x_{59} = 4031$$
$$-(x_{16} + x_{26} + x_{36} + x_{46} + x_{56}) = -1530$$
$$-(x_{17} + x_{27} + x_{37} + x_{47} + x_{57}) = -1583$$
$$-(x_{18} + x_{28} + x_{38} + x_{48} + x_{58}) = -1562$$
$$-(x_{19} + x_{29} + x_{39} + x_{49} + x_{59}) = -161$$

$x_{16} \le 7407$
$x_{17} \le 3546$
$x_{18} \le 5072$
$x_{19} \le 1932$
$x_{26} \le 87$
$x_{27} \le 90$
$x_{28} \le 29 + 144$
$x_{29} \le 902$
$x_{36} \le 13$
$x_{37} \le 8413$
$x_{38} \le 8719$
$x_{39} \le 7439$
$x_{46} \le 5047$
$x_{47} \le 83$
$x_{48} \le 58$
$x_{49} \le 76$
$x_{56} \le 83 + 1372$
$x_{57} \le 7904$
$x_{58} \le 73 + 855$
$x_{59} \le 65$

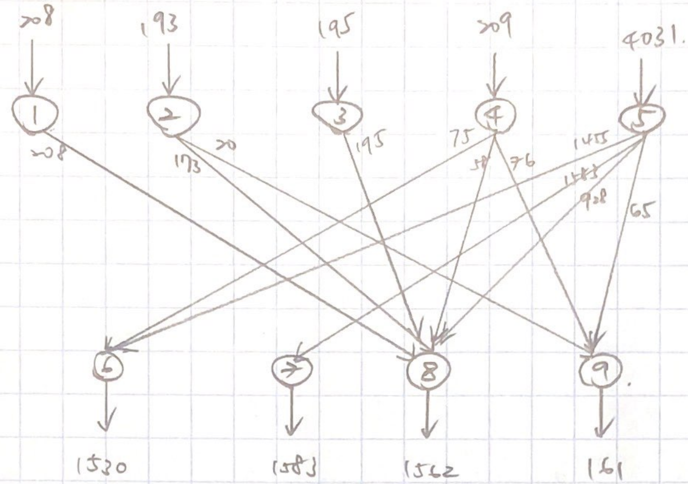
$x_{ij} \ge 0 \quad \forall i, \forall j$

solved by gurobi:

obj value = 4600787

$x_{18} = 208$
$x_{28} = 173$
$x_{29} = 20$
$x_{38} = 195$
$x_{46} = 75$
$x_{48} = 58$
$x_{49} = 76$
$x_{56} = 1455$
$x_{57} = 1583$
$x_{58} = 928$
$x_{59} = 65$
otherwise $= 0$ #

d



1 → 8 × 208
2 → 8 × 173
2 → 9 × 20
3 → 8 × 195
4 → 6 × 75
4 → 8 × 58
4 → 9 × 76
5 → 6 × 1455
5 → 7 × 1583
5 → 8 × 928
5 → 9 × 65 #

Problem 1

December 3, 2019

0.1 name: Yi Ping Tseng

0.2 uni: yt2690

0.3 email: yt2690@columbia.edu

0.4 github: https://github.com/r50206v/Optimization-Homework/tree/master/homework5

[ ]:

```
```

[ ]:

```
```

0.4.1 a.

Gurobi .lp file

[ ]:

```
Minimize
 + 541.0 x_1,6 + 386.0 x_1,7 + 25.0 x_1,8 + 1512.0 x_1,9 + 234.0 x_2,6
 + 899.0 x_2,7 + 103.0 x_2,8 + 1256.0 x_2,9 + 543.0 x_3,6 + 257.0 x_3,7
 + 1653.0 x_3,8 + 1085.0 x_3,9 + 1785.0 x_4,6 + 227.0 x_4,7 + 1670.0 x_4,8
 + 823.0 x_4,9 + 490.0 x_5,6 + 1233.0 x_5,7 + 1242.0 x_5,8 + 1841.0 x_5,9
Subject To
balance_1:  + x_1,6 + x_1,7 + x_1,8 + x_1,9 = 208.0
balance_2:  + x_2,6 + x_2,7 + x_2,8 + x_2,9 = 193.0
balance_3:  + x_3,6 + x_3,7 + x_3,8 + x_3,9 = 195.0
balance_4:  + x_4,6 + x_4,7 + x_4,8 + x_4,9 = 209.0
balance_5:  + x_5,6 + x_5,7 + x_5,8 + x_5,9 = 4031.0
balance_6:  - x_1,6 - x_2,6 - x_3,6 - x_4,6 - x_5,6
 = -1530.0
balance_7:  - x_1,7 - x_2,7 - x_3,7 - x_4,7 - x_5,7
 = -1583.0
balance_8:  - x_1,8 - x_2,8 - x_3,8 - x_4,8 - x_5,8
 = -1562.0
balance_9:  - x_1,9 - x_2,9 - x_3,9 - x_4,9 - x_5,9
 = -161.0
Bounds
```

```
x_1,6 <= 7407.0
x_1,7 <= 3546.0
x_1,8 <= 5072.0
x_1,9 <= 1932.0
x_2,6 <= 81.0
x_2,7 <= 90.0
x_2,8 <= 29.0
x_2,9 <= 902.0
x_3,6 <= 13.0
x_3,7 <= 8413.0
x_3,8 <= 8719.0
x_3,9 <= 7439.0
x_4,6 <= 5047.0
x_4,7 <= 83.0
x_4,8 <= 58.0
x_4,9 <= 76.0
x_5,6 <= 83.0
x_5,7 <= 7904.0
x_5,8 <= 73.0
x_5,9 <= 65.0
END
```

output in terminal

[ ]:
```
Academic license - for non-commercial use only

Gurobi Optimizer version 8.1.1 build v8.1.1rc0 (mac64)
Copyright (c) 2019, Gurobi Optimization, LLC

Read LP format model from file prob1-a.lp
Reading time = 0.01 seconds
: 9 rows, 20 columns, 40 nonzeros
Optimize a model with 9 rows, 20 columns and 40 nonzeros
Coefficient statistics:
  Matrix range     [1e+00, 1e+00]
  Objective range  [2e+01, 2e+03]
  Bounds range     [1e+01, 9e+03]
  RHS range        [2e+02, 4e+03]
Presolve time: 0.00s

Solved in 0 iterations and 0.00 seconds
Infeasible model
```

[ ]:

[ ]:

0.4.2 b.

[ ]:
```
from gurobipy import *


# create a model
m = Model()

# create variables
x16 = m.addVar(vtype=GRB.CONTINUOUS, name="x16", lb=0)
x17 = m.addVar(vtype=GRB.CONTINUOUS, name="x17", lb=0)
x18 = m.addVar(vtype=GRB.CONTINUOUS, name="x18", lb=0)
x19 = m.addVar(vtype=GRB.CONTINUOUS, name="x19", lb=0)
x26 = m.addVar(vtype=GRB.CONTINUOUS, name="x26", lb=0)
x27 = m.addVar(vtype=GRB.CONTINUOUS, name="x27", lb=0)
x28 = m.addVar(vtype=GRB.CONTINUOUS, name="x28", lb=0)
x29 = m.addVar(vtype=GRB.CONTINUOUS, name="x29", lb=0)
x36 = m.addVar(vtype=GRB.CONTINUOUS, name="x36", lb=0)
x37 = m.addVar(vtype=GRB.CONTINUOUS, name="x37", lb=0)
x38 = m.addVar(vtype=GRB.CONTINUOUS, name="x38", lb=0)
x39 = m.addVar(vtype=GRB.CONTINUOUS, name="x39", lb=0)
x46 = m.addVar(vtype=GRB.CONTINUOUS, name="x46", lb=0)
x47 = m.addVar(vtype=GRB.CONTINUOUS, name="x47", lb=0)
x48 = m.addVar(vtype=GRB.CONTINUOUS, name="x48", lb=0)
x49 = m.addVar(vtype=GRB.CONTINUOUS, name="x49", lb=0)
x56 = m.addVar(vtype=GRB.CONTINUOUS, name="x56", lb=0)
x57 = m.addVar(vtype=GRB.CONTINUOUS, name="x57", lb=0)
x58 = m.addVar(vtype=GRB.CONTINUOUS, name="x58", lb=0)
x59 = m.addVar(vtype=GRB.CONTINUOUS, name="x59", lb=0)
theta_x16 = m.addVar(vtype=GRB.CONTINUOUS, name="theta_x16", lb=0)
theta_x17 = m.addVar(vtype=GRB.CONTINUOUS, name="theta_x17", lb=0)
theta_x18 = m.addVar(vtype=GRB.CONTINUOUS, name="theta_x18", lb=0)
theta_x19 = m.addVar(vtype=GRB.CONTINUOUS, name="theta_x19", lb=0)
theta_x26 = m.addVar(vtype=GRB.CONTINUOUS, name="theta_x26", lb=0)
theta_x27 = m.addVar(vtype=GRB.CONTINUOUS, name="theta_x27", lb=0)
theta_x28 = m.addVar(vtype=GRB.CONTINUOUS, name="theta_x28", lb=0)
theta_x29 = m.addVar(vtype=GRB.CONTINUOUS, name="theta_x29", lb=0)
theta_x36 = m.addVar(vtype=GRB.CONTINUOUS, name="theta_x36", lb=0)
theta_x37 = m.addVar(vtype=GRB.CONTINUOUS, name="theta_x37", lb=0)
theta_x38 = m.addVar(vtype=GRB.CONTINUOUS, name="theta_x38", lb=0)
theta_x39 = m.addVar(vtype=GRB.CONTINUOUS, name="theta_x39", lb=0)
theta_x46 = m.addVar(vtype=GRB.CONTINUOUS, name="theta_x46", lb=0)
theta_x47 = m.addVar(vtype=GRB.CONTINUOUS, name="theta_x47", lb=0)
theta_x48 = m.addVar(vtype=GRB.CONTINUOUS, name="theta_x48", lb=0)
theta_x49 = m.addVar(vtype=GRB.CONTINUOUS, name="theta_x49", lb=0)
theta_x56 = m.addVar(vtype=GRB.CONTINUOUS, name="theta_x56", lb=0)
theta_x57 = m.addVar(vtype=GRB.CONTINUOUS, name="theta_x57", lb=0)
```

```
theta_x58 = m.addVar(vtype=GRB.CONTINUOUS, name="theta_x58", lb=0)
theta_x59 = m.addVar(vtype=GRB.CONTINUOUS, name="theta_x59", lb=0)


# integrate new variables
m.update()

# set objective
m.setObjective(
    541.0*theta_x16 + 386.0*theta_x17 + 25.0*theta_x18 + 1512.0*theta_x19 + 234.
 ↪0*theta_x26 + 899.0*theta_x27 + 103.0*theta_x28 + 1256.0*theta_x29 + 543.
 ↪0*theta_x36 + 257.0*theta_x37 + 1653.0*theta_x38 + 1085.0*theta_x39 + 1785.
 ↪0*theta_x46 + 227.0*theta_x47 + 1670.0*theta_x48 + 823.0*theta_x49 + 490.
 ↪0*theta_x56 + 1233.0*theta_x57 + 1242.0*theta_x58 + 1841.0*theta_x59,
    GRB.MINIMIZE
)

# add constraints
m.addConstr(x16 + x17 + x18 + x19 == 208.0)
m.addConstr(x26 + x27 + x28 + x29 == 193.0)
m.addConstr(x36 + x37 + x38 + x39 == 195.0)
m.addConstr(x46 + x47 + x48 + x49 == 209.0)
m.addConstr(x56 + x57 + x58 + x59 == 4031.0)
m.addConstr(-1*(x16 + x26 + x36 + x46 + x56) == -1530.0)
m.addConstr(-1*(x17 + x27 + x37 + x47 + x57) == -1583.0)
m.addConstr(-1*(x18 + x28 + x38 + x48 + x58) == -1562.0)
m.addConstr(-1*(x19 + x29 + x39 + x49 + x59) == -161.0)
m.addConstr(x16 <= 7407.0 + theta_x16)
m.addConstr(x17 <= 3546.0 + theta_x17)
m.addConstr(x18 <= 5072.0 + theta_x18)
m.addConstr(x19 <= 1932.0 + theta_x19)
m.addConstr(x26 <= 81.0 + theta_x26)
m.addConstr(x27 <= 90.0 + theta_x27)
m.addConstr(x28 <= 29.0 + theta_x28)
m.addConstr(x29 <= 902.0 + theta_x29)
m.addConstr(x36 <= 13.0 + theta_x36)
m.addConstr(x37 <= 8413.0 + theta_x37)
m.addConstr(x38 <= 8719.0 + theta_x38)
m.addConstr(x39 <= 7439.0 + theta_x39)
m.addConstr(x46 <= 5047.0 + theta_x46)
m.addConstr(x47 <= 83.0 + theta_x47)
m.addConstr(x48 <= 58.0 + theta_x48)
m.addConstr(x49 <= 76.0 + theta_x49)
m.addConstr(x56 <= 83.0 + theta_x56)
m.addConstr(x57 <= 7904.0 + theta_x57)
m.addConstr(x58 <= 73.0 + theta_x58)
m.addConstr(x59 <= 65.0 + theta_x59)
```

```
# optimize
m.optimize()
print("Model status: ", m.status)

# print out decision variables
for v in m.getVars():
    print(v.varName, v.x, "\n")

print("-"*15)
print("Obj Value: ", m.objVal)
```

[ ]:

```
x16 0.0
x17 0.0
x18 208.0
x19 0.0
x26 0.0
x27 0.0
x28 173.0
x29 20.0
x36 0.0
x37 0.0
x38 195.0
x39 0.0
x46 75.0
x47 0.0
x48 58.0
x49 76.0
x56 1455.0
x57 1583.0
x58 928.0
x59 65.0
theta_x16 0.0
theta_x17 0.0
theta_x18 0.0
theta_x19 0.0
theta_x26 0.0
theta_x27 0.0
theta_x28 144.0
theta_x29 0.0
theta_x36 0.0
theta_x37 0.0
theta_x38 0.0
theta_x39 0.0
theta_x46 0.0
theta_x47 0.0
theta_x48 0.0
```

```
theta_x49 0.0
theta_x56 1372.0
theta_x57 0.0
theta_x58 855.0
theta_x59 0.0
---------------
Obj Value:  1749022.0
```

[ ]:

```

```

0.4.3 c.

[ ]:

```
from gurobipy import *


# create a model
m = Model()

# create variables
x16 = m.addVar(vtype=GRB.CONTINUOUS, name="x16", lb=0)
x17 = m.addVar(vtype=GRB.CONTINUOUS, name="x17", lb=0)
x18 = m.addVar(vtype=GRB.CONTINUOUS, name="x18", lb=0)
x19 = m.addVar(vtype=GRB.CONTINUOUS, name="x19", lb=0)
x26 = m.addVar(vtype=GRB.CONTINUOUS, name="x26", lb=0)
x27 = m.addVar(vtype=GRB.CONTINUOUS, name="x27", lb=0)
x28 = m.addVar(vtype=GRB.CONTINUOUS, name="x28", lb=0)
x29 = m.addVar(vtype=GRB.CONTINUOUS, name="x29", lb=0)
x36 = m.addVar(vtype=GRB.CONTINUOUS, name="x36", lb=0)
x37 = m.addVar(vtype=GRB.CONTINUOUS, name="x37", lb=0)
x38 = m.addVar(vtype=GRB.CONTINUOUS, name="x38", lb=0)
x39 = m.addVar(vtype=GRB.CONTINUOUS, name="x39", lb=0)
x46 = m.addVar(vtype=GRB.CONTINUOUS, name="x46", lb=0)
x47 = m.addVar(vtype=GRB.CONTINUOUS, name="x47", lb=0)
x48 = m.addVar(vtype=GRB.CONTINUOUS, name="x48", lb=0)
x49 = m.addVar(vtype=GRB.CONTINUOUS, name="x49", lb=0)
x56 = m.addVar(vtype=GRB.CONTINUOUS, name="x56", lb=0)
x57 = m.addVar(vtype=GRB.CONTINUOUS, name="x57", lb=0)
x58 = m.addVar(vtype=GRB.CONTINUOUS, name="x58", lb=0)
x59 = m.addVar(vtype=GRB.CONTINUOUS, name="x59", lb=0)


# integrate new variables
m.update()

# set objective
m.setObjective(
```

```
    541.0*x16 + 386.0*x17 + 25.0*x18 + 1512.0*x19 + 234.0*x26 + 899.0*x27 + 103.
 ↪0*x28 + 1256.0*x29 + 543.0*x36 + 257.0*x37 + 1653.0*x38 + 1085.0*x39 + 1785.
 ↪0*x46 + 227.0*x47 + 1670.0*x48 + 823.0*x49 + 490.0*x56 + 1233.0*x57 + 1242.
 ↪0*x58 + 1841.0*x59,
    GRB.MINIMIZE
)

# add constraints
m.addConstr(x16 + x17 + x18 + x19 == 208.0)
m.addConstr(x26 + x27 + x28 + x29 == 193.0)
m.addConstr(x36 + x37 + x38 + x39 == 195.0)
m.addConstr(x46 + x47 + x48 + x49 == 209.0)
m.addConstr(x56 + x57 + x58 + x59 == 4031.0)
m.addConstr(-1*(x16 + x26 + x36 + x46 + x56) == -1530.0)
m.addConstr(-1*(x17 + x27 + x37 + x47 + x57) == -1583.0)
m.addConstr(-1*(x18 + x28 + x38 + x48 + x58) == -1562.0)
m.addConstr(-1*(x19 + x29 + x39 + x49 + x59) == -161.0)
m.addConstr(x16 <= 7407.0 + 0.0)
m.addConstr(x17 <= 3546.0 + 0.0)
m.addConstr(x18 <= 5072.0 + 0.0)
m.addConstr(x19 <= 1932.0 + 0.0)
m.addConstr(x26 <= 81.0 + 0.0)
m.addConstr(x27 <= 90.0 + 0.0)
m.addConstr(x28 <= 29.0 + 144.0)
m.addConstr(x29 <= 902.0 + 0.0)
m.addConstr(x36 <= 13.0 + 0.0)
m.addConstr(x37 <= 8413.0 + 0.0)
m.addConstr(x38 <= 8719.0 + 0.0)
m.addConstr(x39 <= 7439.0 + 0.0)
m.addConstr(x46 <= 5047.0 + 0.0)
m.addConstr(x47 <= 83.0 + 0.0)
m.addConstr(x48 <= 58.0 + 0.0)
m.addConstr(x49 <= 76.0 + 0.0)
m.addConstr(x56 <= 83.0 + 1372.0)
m.addConstr(x57 <= 7904.0 + 0.0)
m.addConstr(x58 <= 73.0 + 855.0)
m.addConstr(x59 <= 65.0 + 0.0)

# optimize
m.optimize()
print("Model status: ", m.status)

# print out decision variables
for v in m.getVars():
    print(v.varName, v.x, "\n")

print("-"*15)
```

```python
print("Obj Value: ", m.objVal)
```

[ ]:
```
x16 0.0
x17 0.0
x18 208.0
x19 0.0
x26 0.0
x27 0.0
x28 173.0
x29 20.0
x36 0.0
x37 0.0
x38 195.0
x39 0.0
x46 75.0
x47 0.0
x48 58.0
x49 76.0
x56 1455.0
x57 1583.0
x58 928.0
x59 65.0
---------------
Obj Value:  4600787.0
```

[ ]:
```
```

prob 2.

to make the problem remains maximum flow problem, we can add a new node before t, and set the edge capacity as 10.

By doing the adjustment, we can guarantee all flow in the optimal path will be restricted in 10 units. #

prob 3.

ⓐ W_i : # of wining of team i

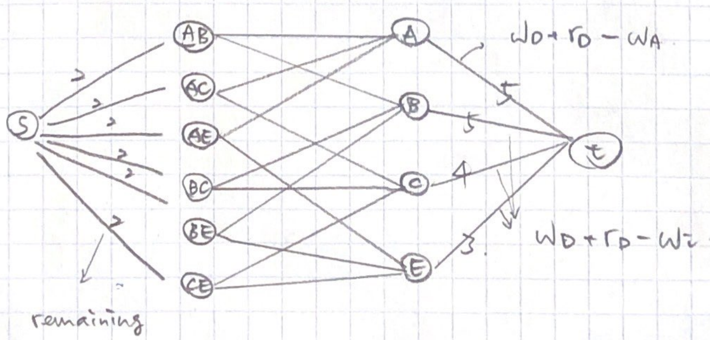
r_i : remaining # of wining of team i.

if D wins the game, this means

$$\begin{aligned} W_D + r_D &\geq W_A + r_A \\ W_D + r_D &\geq W_B + r_B \\ W_D + r_D &\geq W_C + W_C \\ W_D + r_D &\geq W_E + W_E \end{aligned} \Rightarrow$$

assume $r_D = 8$ which means D wins in all the remaining games.

convert to maximum flow.



remaining # of games.

other edges do not have upper bound of weight capacity prob 3 (continue)

if we can find a solution in this maximum flow $s \to t$, then this means D can still win in the game. we can find if there is any feasible solution by applying linear programming on maximum flow problem.

$$
\begin{aligned}
\max \quad & z \\
\text{s.t.} \quad & (sab + sac + sae + sbc + sbe + sce) \times -1 = -z \\
& at + bt + ct + et = z \\
& sab = aba + abb \\
& sac = aca + acc \\
& sae = aea + aee \\
& sbc = bcb + bcc \\
& sbe = beb + bee \\
& sce = cec + cee \\
& at = aba + aca + aea \\
& bt = abb + bcb + beb \\
& ct = acc + bcc + cec \\
& et = aee + bee + cee
\end{aligned}
$$

$$
\begin{aligned}
& sab \le 2 \qquad at \le 5 \\
& sac \le 2 \qquad bt \le 5 \\
& sae \le 2 \qquad ct \le 4 \\
& sbc \le 2 \qquad et \le 3 \\
& sbe \le 2 \\
& sce \le 2
\end{aligned}
$$

all variables ≥ 0.

Solved by gurobi

$sac = sac = sae = sbc = sbe = sce = 2$

$abb = 1$ $aee = 2$ $aba = 1$ $aca = 2$

$bcb = 2$ $beb = 2$ $cec = 2$ $\boxed{\text{other edges} = 0}$

$at = 3$ $bt = 5$ $ct = 2$ $et = 2$

$z = 12$ $\Rightarrow$ since it is feasible, then there is a possibility that D wins. #

prob 3

(b). if D is the only winner in this game

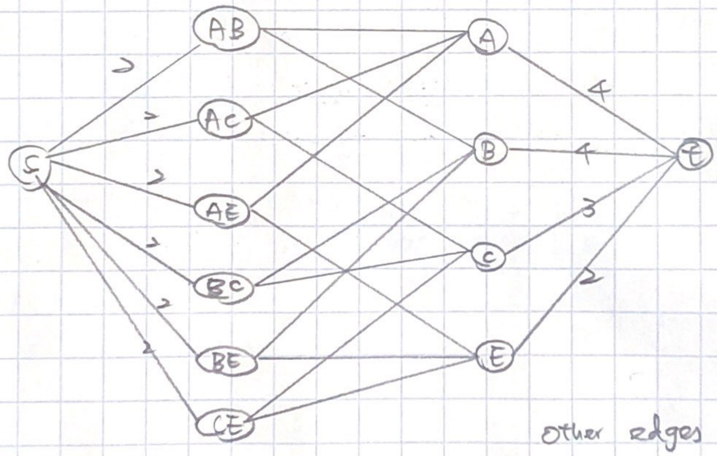
than
$$w_D + r_D > w_A + r_A$$
$$w_D + r_D > w_B + r_B$$
$$w_D + r_D > w_C + r_C$$
$$w_D + r_D > w_E + r_E$$

can be converted to

$$w_D + r_D \geq w_A + r_A + 1$$
$$w_D + r_D \geq w_B + r_B + 1$$
$$w_D + r_D \geq w_C + r_C + 1$$
$$w_D + r_D \geq w_E + r_E + 1$$

Assuming D wins in all the remaining games ($r_D = 8$).

then the graph should be



other edges do not have upper bound of weight capacity

formulate the same LP in prob3 (a) and solve it by Gurobi

sab = sac = sae = sbc = sbe = sce = 2

abb = 1 acc = 1 bcc = 2 cee = 2 aba = 1 aca = 1

aea = 2 beb = 2 $\boxed{\text{other edges} = 0}$

at = 4 bt = 3 ct = 3 et = 2

z = 12

Since it is feasible, there is a chance that only D wins in the game. #

Problem 3

December 3, 2019

0.1 name: Yi Ping Tseng

0.2 uni: yt2690

0.3 email: yt2690@columbia.edu

0.4 github: https://github.com/r50206v/Optimization-Homework/tree/master/homework5

[ ]:

```

```

[ ]:

```

```

0.4.1 a.

[ ]:

```
from gurobipy import *



# create a model
m = Model()

# create variables
sab = m.addVar(vtype=GRB.CONTINUOUS, name="sab", lb=0)
sac = m.addVar(vtype=GRB.CONTINUOUS, name="sac", lb=0)
sae = m.addVar(vtype=GRB.CONTINUOUS, name="sae", lb=0)
sbc = m.addVar(vtype=GRB.CONTINUOUS, name="sbc", lb=0)
sbe = m.addVar(vtype=GRB.CONTINUOUS, name="sbe", lb=0)
sce = m.addVar(vtype=GRB.CONTINUOUS, name="sce", lb=0)
abb = m.addVar(vtype=GRB.CONTINUOUS, name="abb", lb=0)
acc = m.addVar(vtype=GRB.CONTINUOUS, name="acc", lb=0)
aee = m.addVar(vtype=GRB.CONTINUOUS, name="aee", lb=0)
bcc = m.addVar(vtype=GRB.CONTINUOUS, name="bcc", lb=0)
bee = m.addVar(vtype=GRB.CONTINUOUS, name="bee", lb=0)
cee = m.addVar(vtype=GRB.CONTINUOUS, name="cee", lb=0)
aba = m.addVar(vtype=GRB.CONTINUOUS, name="aba", lb=0)
aca = m.addVar(vtype=GRB.CONTINUOUS, name="aca", lb=0)
```

```python
aea = m.addVar(vtype=GRB.CONTINUOUS, name="aea", lb=0)
bcb = m.addVar(vtype=GRB.CONTINUOUS, name="bcb", lb=0)
beb = m.addVar(vtype=GRB.CONTINUOUS, name="beb", lb=0)
cec = m.addVar(vtype=GRB.CONTINUOUS, name="cec", lb=0)
at = m.addVar(vtype=GRB.CONTINUOUS, name="at", lb=0)
bt = m.addVar(vtype=GRB.CONTINUOUS, name="bt", lb=0)
ct = m.addVar(vtype=GRB.CONTINUOUS, name="ct", lb=0)
et = m.addVar(vtype=GRB.CONTINUOUS, name="et", lb=0)
z = m.addVar(vtype=GRB.CONTINUOUS, name="z", lb=0)


# integrate new variables
m.update()

# set objective
# sum of outflow of s
m.setObjective(
    z,
    GRB.MAXIMIZE
)

# add constraints
# input/output constraints
m.addConstr(-1*(sab + sac + sae + sbc + sbe + sce) == -z)
m.addConstr(at + bt + ct + et == z)
# capacity constraints
m.addConstr(sab <= 2)
m.addConstr(sac <= 2)
m.addConstr(sae <= 2)
m.addConstr(sbc <= 2)
m.addConstr(sbe <= 2)
m.addConstr(sce <= 2)
m.addConstr(at <= 5)
m.addConstr(bt <= 5)
m.addConstr(ct <= 4)
m.addConstr(et <= 3)
# inflow equals to outflow constraints
m.addConstr(sab == aba + abb)
m.addConstr(sac == aca + acc)
m.addConstr(sae == aea + aee)
m.addConstr(sbc == bcb + bcc)
m.addConstr(sbe == beb + bee)
m.addConstr(sce == cec + cee)
m.addConstr(at == aba + aca + aea)
m.addConstr(bt == abb + bcb + beb)
m.addConstr(ct == acc + bcc + cec)
m.addConstr(et == aee + bee + cee)
```

```
# optimize
m.optimize()
print("Model status: ", m.status)

# print out decision variables
for v in m.getVars():
    print(v.varName, v.x, "\n")

print("-"*15)
print("Obj Value: ", m.objVal)
```

[ ]:

```
sab 2.0
sac 2.0
sae 2.0
sbc 2.0
sbe 2.0
sce 2.0
abb 1.0
acc 0.0
aee 2.0
bcc 0.0
bee 0.0
cee 0.0
aba 1.0
aca 2.0
aea 0.0
bcb 2.0
beb 2.0
cec 2.0
at 3.0
bt 5.0
ct 2.0
et 2.0
z 12.0
---------------
Obj Value:  12.0
```

[ ]:

[ ]:

0.4.2 b.

[ ]:
```
from gurobipy import *


# create a model
m = Model()

# create variables
sab = m.addVar(vtype=GRB.CONTINUOUS, name="sab", lb=0)
sac = m.addVar(vtype=GRB.CONTINUOUS, name="sac", lb=0)
sae = m.addVar(vtype=GRB.CONTINUOUS, name="sae", lb=0)
sbc = m.addVar(vtype=GRB.CONTINUOUS, name="sbc", lb=0)
sbe = m.addVar(vtype=GRB.CONTINUOUS, name="sbe", lb=0)
sce = m.addVar(vtype=GRB.CONTINUOUS, name="sce", lb=0)
abb = m.addVar(vtype=GRB.CONTINUOUS, name="abb", lb=0)
acc = m.addVar(vtype=GRB.CONTINUOUS, name="acc", lb=0)
aee = m.addVar(vtype=GRB.CONTINUOUS, name="aee", lb=0)
bcc = m.addVar(vtype=GRB.CONTINUOUS, name="bcc", lb=0)
bee = m.addVar(vtype=GRB.CONTINUOUS, name="bee", lb=0)
cee = m.addVar(vtype=GRB.CONTINUOUS, name="cee", lb=0)
aba = m.addVar(vtype=GRB.CONTINUOUS, name="aba", lb=0)
aca = m.addVar(vtype=GRB.CONTINUOUS, name="aca", lb=0)
aea = m.addVar(vtype=GRB.CONTINUOUS, name="aea", lb=0)
bcb = m.addVar(vtype=GRB.CONTINUOUS, name="bcb", lb=0)
beb = m.addVar(vtype=GRB.CONTINUOUS, name="beb", lb=0)
cec = m.addVar(vtype=GRB.CONTINUOUS, name="cec", lb=0)
at  = m.addVar(vtype=GRB.CONTINUOUS, name="at", lb=0)
bt  = m.addVar(vtype=GRB.CONTINUOUS, name="bt", lb=0)
ct  = m.addVar(vtype=GRB.CONTINUOUS, name="ct", lb=0)
et  = m.addVar(vtype=GRB.CONTINUOUS, name="et", lb=0)
z = m.addVar(vtype=GRB.CONTINUOUS, name="z", lb=0)


# integrate new variables
m.update()

# set objective
# sum of outflow of s
m.setObjective(
    z,
    GRB.MAXIMIZE
)

# add constraints
# input/output constraints
m.addConstr(-1*(sab + sac + sae + sbc + sbe + sce) == -z)
```

```python
m.addConstr(at + bt + ct + et == z)
# capacity constraints
m.addConstr(sab <= 2)
m.addConstr(sac <= 2)
m.addConstr(sae <= 2)
m.addConstr(sbc <= 2)
m.addConstr(sbe <= 2)
m.addConstr(sce <= 2)
m.addConstr(at <= 4)
m.addConstr(bt <= 4)
m.addConstr(ct <= 3)
m.addConstr(et <= 2)
# inflow equals to outflow constraints
m.addConstr(sab == aba + abb)
m.addConstr(sac == aca + acc)
m.addConstr(sae == aea + aee)
m.addConstr(sbc == bcb + bcc)
m.addConstr(sbe == beb + bee)
m.addConstr(sce == cec + cee)
m.addConstr(at == aba + aca + aea)
m.addConstr(bt == abb + bcb + beb)
m.addConstr(ct == acc + bcc + cec)
m.addConstr(et == aee + bee + cee)


# optimize
m.optimize()
print("Model status: ", m.status)

# print out decision variables
for v in m.getVars():
    print(v.varName, v.x, "\n")

print("-"*15)
print("Obj Value: ", m.objVal)
```

[ ]:

```
sab 2.0
sac 2.0
sae 2.0
sbc 2.0
sbe 2.0
sce 2.0
abb 1.0
acc 1.0
aee 0.0
bcc 2.0
bee 0.0
```

```
cee 2.0
aba 1.0
aca 1.0
aea 2.0
bcb 0.0
beb 2.0
cec 0.0
at 4.0
bt 3.0
ct 3.0
et 2.0
z 12.0
---------------
Obj Value:  12.0
```

[ ]:

```

```

problem 4:

(a)

$$f_i(j) = f_i(j-1) + f_{i-1}(j)$$

if either i or $j = 0$, $f_i(j) = 1$.

0	1	1	1	1	1	1
1	2	3	4	5	6	7
1	3	6	10	15	21	28
1	4	10	20	35	56	84

(b)

$$f_i(j) = \max(f_i(j-1), f_{i-1}(j)) + I(i,j)$$

$$I(i,j) = \begin{cases} 1, & \text{if there is a coin at } (i,j) \\ 0, & \text{o.w.} \end{cases}$$

0	0	0	0	0	1	1
0	0	1	2	2	2	3
0	1	1	2	3	3	3
0	1	2	2	3	4	4

path should be this