1. Project title:

   Police Patrol District Design

2. Team member names:

   Lorraine Chen; Yiling Peng

3. Brief description of the problem:

   The police patrol plays an important role in public safety in city A, and its performance can be greatly affected by patrol district design. Our goal is to improve the effectiveness of lowering the crime rate under some certain circumstances (requirements) by adopting optimal district design.

   We have already known the transportation network in city A, the locations of all patrol centers and all incident spots. There are 15 patrol centers and 68 incident spots. A patrol center can administer an incident spot which is no more than 3000km away. All incident spots need administration from a patrol center.
   Below is part of data showing the distance between each pair <patrol center, incident spot>

Pat. \ Inci.	IP1	IP2	IP3	IP4	...
P1					...
P2					...
P3					...
...					...

   Q1: How do we assign the incident spots to each patrol center as evenly as possible and also satisfy the distance requirement?

   Suppose an urgent incident happens in city A, we need all police force from 20 patrol centers to lock down 13 traffic roads very quickly. In reality, the police force from a center can at most lock down 1 street, and they all have the same speed at 60 km/h. [Data will be collected in a csv file, below is an example]

Patrol center	Traffic roads	Distance (km)
P1	1	....
P2	12	
P16	3	
P7	13	
...	...	

There are total 20 * 13 rows.

Q2: How can we lock down 13 traffic roads with the minimum amount of time?

Q3: If there is a case in some area, and the criminal can escape with speed v, from which patrol center should we send police to intercept what roads that the criminal is possible to reach.

4. Type of model:

Q1 warshall-floyd algorithm:

#thought

We will first use warshall-floyd algorithm to find out the shortest path matrix which will later be used to determined whether A[i] and v[i] is connected. Then we use optimization model to find out how many incident spots a patrol center should take care of.

data

From csv

Step1:

Extract adjacency matrix from csv file. Then replace not connected edges by positive infinity. Then use floyd-warshall algorithm to update the adjacency matrix.

The pseudo code is like this:

```
for k in n:
  for i in n:
        for j in n:
```

```
if(adj[i,k]+adj[k,j)<adj[i,j]):
    dis[i][j]=dis[i,k]+dis[k,j]
```

Step 2 :Draw the graph of flow problem then find Q value that determines the maximum number of incident spots a patrol center should handle using binary search.

pseudo code:
function Qsearch(upper,lower):
nodes = |v| U |A| # create a list of nodes.
Arc=findArc(adj)U(vXA)
Capacity=findCap(q) #find a way to construct the capacity array in which the first |v| entry is 1000 , the |v | th to |v|+|arc|-|U|-|v| th capacity is set to 1, the rest of entries is set to q
Mid=(upper-lower)/2

#variables represent flow on each arc
@variable(m, x[arcs] >= 0)
maximize total flow on arc from sink to source
@objective(m, Min, -x[(:si,:s)])
@constraint(m, cap[a in range(0,v)], x[a] = =capacity[a])
@constraint(m, cap[a in range(0, arc|-|U|-|)], 0<=x[a] =< capacity[a])
@constraint(m, cap[a in range(0,rest)], x[a] <= capacity[a])
@constraint(m, flow[i in nodes], sum(x[a] for a in arcs if a[1] == i) == sum(x[a] for a in arcs if a[2] == i))

set_optimizer_attribute(m, "LogLevel", 0)
Try: optimize!(m)
Catch no primal solution error：
    Return Qsearch(upper,(upper-mid)/2+mid)
If upper<=lower:
    Return upper
Return Qsearch(mid-(upper-mid)/2+,lower)

Step 3： write similar code to print a viable solution though we can further optimize depending on the adjacency matrix.

Q2: LP model
data

```
d[i,j]  # the distance matrix, where i represents the incident spots and j represents the patrol centers

# variables
@variable(m,x[20][13],Bin) # decision variables for locking down or not
@variable(m, for j in [1, 13], qj) # variables for the total number of traffic roads
@variable(m, for i in [1, 20], p_i) # variables for the total number of  patrol centers

# constraints
#each road is handled by at exactly one patrol center
@constraints(m, for j in qj, sum(x[i,j] ==1)
# each patrol center lock down no more than one road
@constraints(m, for i in pi, sum(x[i,j] <=1)

# objectives
@objective(m,Min,for i in pi for j in qj,d[i,j]*x[i,j])
```

Q3: Matching algorithm model

Step 1: set time variable t which increase in each iteration
Step 2: calculate the diameter of the current escape circle r=v*t
Step 3: iterate through each node to calculate the set of reachable area in time t R
Step 4: Now since we have 2 sets R&P with no intersection,we can use a matching algorithm to find a matching so each possible area that the criminal can reach will be covered by the police in 10 minutes.

Suggested pseudocode:

```
 t=0

while(t<escaped){
   t=t+1
  r=v*t
  R=findSet(r)
  for i in R{
       If check (mat)
           return t
```

```
  }

}

Function find(int x,matrix K){
        int i,j
        int[] ret=int[nodes]
        bool[] scanned= bool[nodes]
        for  (j=1:j<|R| ; j++){
                if(K[x][j]&&scaned[j]==false){
                        scanned=true
                        If   (ret[j]==0||find(ret)){
                         ret[j]=x
                        return mat
                        }
                }
        }
        return null
}
 Function check(int[] mat){
        if(mat==null){
                return false
        }
        for(i in mat){
                If (mat[i]==0 ){return false}
        }
        return true
 }
```