Gurobi Optimization - Minimizing Total Distance

In [1]:
```
import gurobipy as gp
from gurobipy import *
import numpy as np
import csv
import os
import matplotlib.pyplot as plt
import warnings
import math
import pandas as pd
import random

warnings.filterwarnings("ignore") # To ignore warnings produced
```

In [2]:
```
zips_df = pd.read_csv('Zipcode.csv')
# zips_df
zips_df = zips_df.loc[(zips_df['COUNTYNAME'] == 'ALLEGHENY') & (zips_df['population'] > 0)]
zips_df
```

Out[2]:

	OBJECTID	ZIP	NAME	ZIPTYPE	STATE	STATEFIPS	COUNTYFIPS	COUNTYNAME	S3DZIP	LAT	...	MFDU	SFI
0	4	15057	BAKERSTOWN	NON-UNIQUE	PA	42.0	42003.0	ALLEGHENY	150.0	40.361610	...	0.0	107
1	4	15206	BAKERSTOWN	NON-UNIQUE	PA	42.0	42003.0	ALLEGHENY	150.0	40.467710	...	0.0	107
2	4	15214	BAKERSTOWN	NON-UNIQUE	PA	42.0	42003.0	ALLEGHENY	150.0	40.483220	...	0.0	107
3	4	15229	BAKERSTOWN	NON-UNIQUE	PA	42.0	42003.0	ALLEGHENY	150.0	40.517330	...	0.0	107
4	4	15228	BAKERSTOWN	NON-UNIQUE	PA	42.0	42003.0	ALLEGHENY	150.0	40.368810	...	0.0	107
...	...	...	...	...	...	...	...	...	...	...	...	...	
118	65824	15047	GREENOCK	PO BOX	PA	42.0	42003.0	ALLEGHENY	150.0	40.609477	...	0.0	1
121	65828	15032	CURTISVILLE	PO BOX	PA	42.0	42003.0	ALLEGHENY	150.0	40.382820	...	346.0	4134
123	65837	15221	PITTSBURGH	NON-UNIQUE	PA	42.0	42003.0	ALLEGHENY	152.0	40.467750	...	3486.0	12256
124	65839	15205	PITTSBURGH	NON-UNIQUE	PA	42.0	42003.0	ALLEGHENY	152.0	40.483220	...	1718.0	8997
125	65841	15202	PITTSBURGH	NON-UNIQUE	PA	42.0	42003.0	ALLEGHENY	152.0	40.370729	...	2933.0	10866

96 rows × 23 columns

In [3]:
```
zips = zips_df['ZIP'].to_list()
# zips
all_n = np.array(zips)
# # zips
# np.in1d(POD_zips, all_n)
# 0,1,5,9,10,13,15,16,21,22,24,27,29,36
# POD_zips[[0,1,5,9,10,13,15,16,21,22,24,27,29,36]]
```

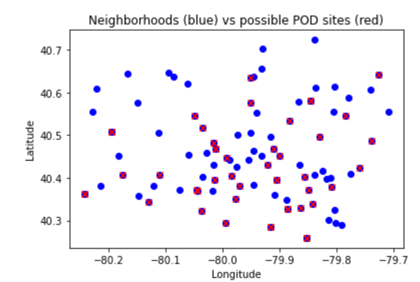
In [4]:
```
zipcodes_path = 'pittsburgh-allegheny-county.csv'
data = np.genfromtxt(zipcodes_path, dtype=str, delimiter=',', encoding='utf-8-sig')
# neighborhoods = data.astype(np.int)
neighborhoods = all_n.astype(np.int)

POD_sites_path = 'POD Sites.xlsx'
POD_df = pd.read_excel(POD_sites_path)
POD_df = POD_df[['SCHOOL/FACILITY NAME', 'STRIP MAP']]
POD_df['ZIPCODE'] = POD_df['STRIP MAP'].apply(lambda x: str(x)[-5:])
POD_df = POD_df[:47]
POD_df

POD_zips = np.array(pd.to_numeric(POD_df.ZIPCODE).values)
print(POD_zips)
print(neighborhoods)
```

```
[15237 15236 15102 15216 15227 15106 15210 15220 15025 15108 15024 15110
 15137 15037 15238 15146 15101 15065 15216 15132 15136 15108 15228 15090
 15229 15202 15235 15214 15044 15206 15217 15239 15056 15139 15209 15133
 15057 15129 15144 15120 15136 15025 15241 15126 15122 15221 15221]
[15057 15206 15214 15229 15228 15108 15101 15146 15037 15024 15106 15236
 15237 15007 15014 15015 15018 15025 15034 15030 15035 15046 15049 15064
 15065 15082 15110 15112 15129 15131 15132 15133 15135 15137 15144 15148
 15203 15211 15213 15224 15225 15232 15239 15275 15006 15051 15223 15227
 16229 15282 15083 15088 15116 15122 15204 15216 15102 15234 15120 15017
 15215 15139 15028 15145 15104 15086 15142 15241 15226 15207 15075 15076
 15209 15243 15136 16046 15056 15045 15090 15084 15147 15235 15238 15220
 15217 15233 15143 15210 15289 15044 15126 15047 15032 15221 15205 15202]
```

In [5]:
```
num_neighborhoods = len(neighborhoods)
num_sites = len(POD_zips)


random.seed(20)

zipcodes_df = pd.read_csv('Zipcode.csv')
# zipcodes_df = zipcodes_df.loc[zipcodes_df['type'].isin(['STANDARD', 'UNIQUE'])]
zipcodes_df = zipcodes_df.loc[(zipcodes_df['population'] > 0)]
zipcodes_df_filtered = zipcodes_df[['ZIP', 'NAME', 'LAT', 'LON', 'population']]


from math import sin, cos, sqrt, atan2, radians

def calcDistBetweenTwoPoints(pt1, pt2):
    # approximate radius of earth in km
    R = 6373.0
    lat1 = radians(pt1[0])
    lon1 = radians(pt1[1])
    lat2 = radians(pt2[0])
    lon2 = radians(pt2[1])
    dlon = lon2 - lon1
    dlat = lat2 - lat1
    a = sin(dlat / 2)**2 + cos(lat1) * cos(lat2) * sin(dlon / 2)**2
    c = 2 * atan2(sqrt(a), sqrt(1 - a))
    distance = R * c
    return(distance)

def getLatLongFromZip_graph(zipcode, df):
    lat = df.loc[df['ZIP'] == zipcode]['LAT'].values[0]
    long = df.loc[df['ZIP'] == zipcode]['LON'].values[0]
    return((lat, long))

def getLatLongFromZip(zipcode, df):
    lat = df.loc[df['ZIP'] == zipcode]['LAT'].values[0]
    long = df.loc[df['ZIP'] == zipcode]['LON'].values[0]
    return((lat, long))

problematic_n = []
problematic_s = []
distances = []
# nomi.query_postal_code()['latitude']
for i in range(num_neighborhoods):
    temp = []
    for j in range(num_sites):
        try:
#             print(POD_zips[j])
            pt1 = getLatLongFromZip(neighborhoods[i], zips_df)
#             pt1 = (nomi.query_postal_code(str(neighborhoods[i]))['latitude'], nomi.query_postal_code
(str(neighborhoods[i]))['longitude'])
#             print(pt1)
            pt2 = getLatLongFromZip(POD_zips[j], zips_df)
#             pt2 = (nomi.query_postal_code(str(POD_zips[j]))['latitude'], nomi.query_postal_code(str(P
OD_zips[j]))['longitude'])
            dist = calcDistBetweenTwoPoints(pt1, pt2)
            temp.append(dist)
        except:
            problematic_n.append(neighborhoods[i])
            problematic_s.append(POD_zips[j])
#             print(neighborhoods[i], POD_zips[j])
            print('-'*50)
            temp.append(1000000)
    distances.append(temp)
distances = np.array(distances)
# distances = np.delete(distances, (12), axis=0)
```

In [6]:
```
POD_lats = []
POD_longs = []

for z in POD_zips:
    lat, long = getLatLongFromZip_graph(z, zips_df)
    POD_lats.append(lat)
    POD_longs.append(long)

neighborhood_lats = []
neighborhood_longs = []

for z in neighborhoods:
    lat, long = getLatLongFromZip_graph(z, zips_df)
    neighborhood_lats.append(lat)
    neighborhood_longs.append(long)


plt.scatter(neighborhood_longs, neighborhood_lats, marker='o', c = 'blue')
plt.scatter(POD_longs, POD_lats, marker='x', c = 'red')
plt.xlabel('Longitude')
plt.ylabel('Latitude')
plt.title('Neighborhoods (blue) vs possible POD sites (red)')
```

Out[6]: Text(0.5, 1.0, 'Neighborhoods (blue) vs possible POD sites (red)')



In [7]:
```
population = []
for i in range(num_neighborhoods):
    try:
        population.append(zipcodes_df_filtered.loc[zipcodes_df_filtered['ZIP'] == neighborhoods[i]]['po
pulation'].values[0])
    except:
        pass

population = np.array(population)
pop_mean = np.mean(population)

#for i in range(len(population)):
#    if(population[i] == 0):
#        population[i] = pop_mean
print(population)
```

```
[ 6738 22090 12010 13410 17180 37850 24110 25680  9730  7970 16810 29410
 42230   360  2650  1290   750 15150  1350   850  1770  2360   860   310
  9890   386  3950  2650 10260  7280 14700  5460  4510  8540  3580  1860
  6410  8560  7230  7630   870  6760 20120    13   359   360  6180 26790
  4790    16   900   631 13950 17260  6820 19990 28950 12670 15350 14880
 11150  6080   170  5720  6020   560  1860 22040 11930  8780   207   810
 10970 13340 20030 18170   950  3500 23900  8360 13980 30560 12850 16460
 21290  2120 21220 20140   123 29140  6730   378   244 23860 20320 17220]
```

In [8]:
```
population.shape
```

Out[8]: (96,)

Find the minimum cost to vaccinate the entire population

In [9]:
```
# MODEL INITIALIZATION
m1 = Model()
sizes = range(3)
zipcodes = range(distances.shape[0])
sites = range(distances.shape[1])
days = range(25)

# CONSTANTS
D = distances
p = population
total_pop = population.sum()
e = np.array([72, 85, 100])
o = 20000
v = 50
f = 8000
r = 206.76
h = 12.5
C = 1370
```

```
Academic license - for non-commercial use only - expires 2022-08-29
Using license file C:\Users\Ben\gurobi.lic
```

In [10]:
```
population.sum()/(1370*100)
```

Out[10]: 7.333248175182482

In [11]:
```
(population.sum() - (1370*100*7))/100
```

Out[11]: 456.55

In [12]:
```
[population.sum() - 100*i*1370 for i in [1, 2, 3, 4, 5, 6, 7]]
```

Out[12]: [867655, 730655, 593655, 456655, 319655, 182655, 45655]

In [13]:
```
o+v*total_pop+r*(7*1370+835.05)+h*sum(total_pop - 100*i*1370 for i in [1, 2, 3, 4, 5, 6, 7]) + f*8
```

Out[13]: 92429545.838

The minimum total cost is $92,429,545.84, found by only utilizing one site.

Minimize distance while keeping cost at a minimum

In [14]:
```
print(np.argmin(distances.T@population[ : , np.newaxis]))
print(np.min(distances.T@population[ : , np.newaxis]))
```

```
30
13561588.025787687
```

Minimum total distance given minimum cost (achieved by only using one site): 13,561,588.03

In [15]:
```
13561588.025787687 / population.sum()
```

Out[15]: 13.498751338307864

Average total distance in this scenario: 13.50

In [16]:
```
max(distances[:,30])
```

Out[16]: 33.3645020058423

Max distance in this scenario is 33.36

Minimize Distance without respect to Cost

In [24]:
```
# DECISION VARIABLES
A = m1.addVars(zipcodes, sites, vtype = GRB.BINARY)
S = m1.addVars(sites, vtype = GRB.BINARY)
M = m1.addVars(sites, vtype = GRB.BINARY)
L = m1.addVars(sites, vtype = GRB.BINARY)
U = m1.addVars(sites, days, vtype = GRB.BINARY)
X = m1.addVars(sites, days)#, vtype = GRB.INTEGER)
I = m1.addVars(sites, days)#, vtype = GRB.INTEGER)

# OBJECTIVE
m1.setObjective(sum(p[i]*D[i,j]*A[i,j] for i in zipcodes for j in sites))
m1.modelSense = GRB.MINIMIZE

# CONSTRAINTS
for i in zipcodes:
    m1.addConstr(sum(A[i,j] for j in sites) == 1)

    for j in sites:
        m1.addConstr(A[i,j] <= S[j] + M[j] + L[j])
        m1.addConstr(A[i,j] >= 0)

for j in sites:
    assigned_pop = sum(A[i,j] * p[i] for i in zipcodes)
    m1.addConstr(S[j] + M[j] + L[j] <= 1)
    m1.addConstr(sum(e[0]*X[j,t] for t in days) >= assigned_pop - (1-S[j])*total_pop)
    m1.addConstr(sum(e[1]*X[j,t] for t in days) >= assigned_pop - (1-M[j])*total_pop)
    m1.addConstr(sum(e[2]*X[j,t] for t in days) >= assigned_pop)
    m1.addConstr(S[j] >= 0)
    m1.addConstr(M[j] >= 0)
    m1.addConstr(L[j] >= 0)

    administered = [0,0,0]
    for t in days:
        administered[0] += e[0]*X[j,t]
        administered[1] += e[1]*X[j,t]
        administered[2] += e[2]*X[j,t]
        m1.addConstr(X[j,t] >= 10*M[j] + 20*L[j] - 20*(1-U[j,t]))
        m1.addConstr(X[j,t] <= 10*S[j] + 20*M[j] + C*L[j]) #The third term is redundant
        m1.addConstr(X[j,t] <= C*U[j,t])
        m1.addConstr(U[j,t] <= S[j] + M[j] + L[j])
        m1.addConstr(I[j,t] + (1-S[j])*total_pop >= assigned_pop - administered[0])
        m1.addConstr(I[j,t] + (1-M[j])*total_pop >= assigned_pop - administered[1])
        m1.addConstr(I[j,t] + (1-L[j])*total_pop >= assigned_pop - administered[2])
        m1.addConstr(I[j,t] - (1-S[j])*total_pop <= assigned_pop - administered[0])
        m1.addConstr(I[j,t] - (1-M[j])*total_pop <= assigned_pop - administered[1])
        m1.addConstr(I[j,t] - (1-L[j])*total_pop <= assigned_pop - administered[2])
        m1.addConstr(X[j,t] >= 0)
        m1.addConstr(I[j,t] >= 0)
        m1.addConstr(U[j,t] >= 0)

for t in days:
    m1.addConstr(sum(X[j,t] for j in sites) <= C)
```

In [25]:
```
m1.Params.TimeLimit = 3*60
m1.optimize()
```

```
Parameter TimeLimit unchanged
   Value: 180.0  Min: 0.0  Max: inf  Default: inf
Gurobi Optimizer version 9.1.2 build v9.1.2rc0 (win64)
Thread count: 6 physical cores, 12 logical processors, using up to 12 threads
Optimize a model with 49498 rows, 16356 columns and 1696418 nonzeros
Model fingerprint: 0x3400a51c
Variable types: 4700 continuous, 11656 integer (11656 binary)
Coefficient statistics:
  Matrix range     [1e+00, 1e+06]
  Objective range  [5e+01, 2e+06]
  Bounds range     [1e+00, 1e+00]
  RHS range        [1e+00, 1e+06]

MIP start from previous solve produced solution with objective 1.08026e+06 (2.29s)
Loaded MIP start from previous solve with objective 1.08026e+06
Processed MIP start in 2.29 seconds

Presolve removed 32224 rows and 7475 columns
Presolve time: 3.48s
Presolved: 17274 rows, 8881 columns, 777816 nonzeros
Variable types: 3055 continuous, 5826 integer (5826 binary)

Root simplex log...

Iteration    Objective       Primal Inf.    Dual Inf.      Time
       0    0.0000000e+00   1.270800e+04   0.000000e+00      6s

Root relaxation: cutoff, 2497 iterations, 0.21 seconds

    Nodes    |    Current Node    |     Objective Bounds      |     Work
 Expl Unexpl |  Obj  Depth IntInf | Incumbent    BestBd   Gap | It/Node Time

     0     0     cutoff    0      1080260.98 1080260.98  0.00%     -    6s

Explored 0 nodes (2497 simplex iterations) in 6.26 seconds
Thread count was 12 (of 12 available processors)

Solution count 1: 1.08026e+06

Optimal solution found (tolerance 1.00e-04)
Best objective 1.080260975262e+06, best bound 1.080260975262e+06, gap 0.0000%
```

In [26]:
```
m1.objval
```

Out[26]: 1080260.9752623043

In [27]:
```
m1.objval / sum(population)
```

Out[27]: 1.0752556601642398

Minimum total distance traveled is 1,080,261

Minimum average distance traveled is 1.08

In [28]:
```
max([A[i,j].x*D[i,j] for i in zipcodes for j in sites])
```

Out[28]: 14.657912157781494

The maximum distance traveled is 14.66

In [29]:
```
o*sum((S[j].x + M[j].x + L[j].x) for j in sites) + v*sum(A[i,j].x*p[i] for i in zipcodes for j in sites
) + sum(r*X[j,t].x + h*I[j,t].x + f*U[j,t].x for j in sites for t in days)
```

Out[29]: 155903410.3714

The cost is $155,903,410.37

In [30]:
```
num_sites = 0
for j in sites:
    if (S[j].x + M[j].x + L[j].x) > 0:
        num_sites += 1
print(num_sites)
```

```
42
```

Minimizing Cost while holding total distance at minimum

In [31]:
```
m1.setObjective(o*sum((S[j] + M[j] + L[j]) for j in sites)
            + v*sum(A[i,j]*p[i] for i in zipcodes for j in sites)
            + sum(r*X[j,t] + h*I[j,t] + f*U[j,t] for j in sites for t in days))
m1.addConstr(sum(p[i]*D[i,j]*A[i,j] for i in zipcodes for j in sites) <= 1080260.9752623043)
m1.Params.TimeLimit = 8*60
m1.optimize()
```

```
Changed value of parameter TimeLimit to 480.0
   Prev: 180.0  Min: 0.0  Max: inf  Default: inf
Gurobi Optimizer version 9.1.2 build v9.1.2rc0 (win64)
Thread count: 6 physical cores, 12 logical processors, using up to 12 threads
Optimize a model with 49499 rows, 16356 columns and 1700873 nonzeros
Model fingerprint: 0xb2eede7d
Variable types: 4700 continuous, 11656 integer (11656 binary)
Coefficient statistics:
  Matrix range     [1e+00, 2e+06]
  Objective range  [1e+01, 2e+06]
  Bounds range     [1e+00, 1e+00]
  RHS range        [1e+00, 1e+06]

MIP start from previous solve produced solution with objective 1.26093e+08 (0.26s)
Loaded MIP start from previous solve with objective 1.26093e+08

Presolve removed 32289 rows and 7540 columns
Presolve time: 3.94s
Presolved: 17210 rows, 8816 columns, 771927 nonzeros
Variable types: 3055 continuous, 5761 integer (5761 binary)

Root simplex log...

Iteration    Objective       Primal Inf.    Dual Inf.      Time
    3321    5.0252757e+07   8.112436e+07   0.000000e+00      5s
    7670    5.3148641e+07   0.000000e+00   0.000000e+00      5s

Root relaxation: objective 5.314864e+07, 7670 iterations, 0.99 seconds

    Nodes    |    Current Node    |     Objective Bounds      |     Work
 Expl Unexpl |  Obj  Depth IntInf | Incumbent    BestBd   Gap | It/Node Time

     0     0 5.3149e+07    0  173 1.2609e+08 5.3149e+07  57.8%     -    5s
H    0     0                    1.260568e+08 5.3149e+07  57.8%     -    5s
H    0     0                    1.254790e+08 5.3149e+07  57.6%     -    5s
H    0     0                    1.225104e+08 5.3149e+07  56.6%     -    6s
     0     0 5.3274e+07    0  319 1.2251e+08 5.3274e+07  56.5%     -    6s
H    0     0                    1.156782e+08 5.3274e+07  53.9%     -    7s
     0     0 5.3368e+07    0  270 1.1568e+08 5.3368e+07  53.9%     -    7s
     0     0 5.3368e+07    0  268 1.1568e+08 5.3368e+07  53.9%     -    7s
     0     0 5.4977e+07    0  320 1.1568e+08 5.4977e+07  52.5%     -    8s
     0     0 5.5462e+07    0  316 1.1568e+08 5.5462e+07  52.1%     -    9s
     0     0 5.5495e+07    0  349 1.1568e+08 5.5495e+07  52.0%     -    9s
     0     0 5.5504e+07    0  352 1.1568e+08 5.5504e+07  52.0%     -    9s
     0     0 5.5504e+07    0  355 1.1568e+08 5.5504e+07  52.0%     -    9s
H    0     0                    1.145808e+08 5.5504e+07  51.6%     -   10s
     0     0 5.9895e+07    0  180 1.1458e+08 5.9895e+07  47.7%     -   10s
     0     0 5.9895e+07    0  162 1.1458e+08 5.9895e+07  47.7%     -   12s
H    0     0                    1.143197e+08 5.9895e+07  47.6%     -   13s
     0     0 6.2005e+07    0  242 1.1432e+08 6.2005e+07  45.8%     -   13s
     0     0 6.4431e+07    0  250 1.1432e+08 6.4431e+07  43.6%     -   13s
     0     0 6.5491e+07    0  257 1.1432e+08 6.5491e+07  42.7%     -   13s
     0     0 6.5667e+07    0  271 1.1432e+08 6.5667e+07  42.6%     -   13s
     0     0 6.5799e+07    0  267 1.1432e+08 6.5799e+07  42.4%     -   13s
     0     0 6.5807e+07    0  266 1.1432e+08 6.5807e+07  42.4%     -   13s
     0     0 6.5807e+07    0  276 1.1432e+08 6.5807e+07  42.4%     -   14s
     0     0 6.7775e+07    0  300 1.1432e+08 6.7775e+07  40.7%     -   14s
H    0     0                    1.138498e+08 6.7775e+07  40.5%     -   14s
H    0     0                    1.122702e+08 6.7775e+07  39.6%     -   14s
     0     0 6.8470e+07    0  259 1.1227e+08 6.8470e+07  39.0%     -   14s
     0     0 6.8713e+07    0  271 1.1227e+08 6.8713e+07  38.8%     -   14s
     0     0 6.8840e+07    0  248 1.1227e+08 6.8840e+07  38.7%     -   14s
     0     0 6.9029e+07    0  260 1.1227e+08 6.9029e+07  38.5%     -   14s
H    0     0                    1.081762e+08 6.9029e+07  36.2%     -   14s
     0     0 6.9117e+07    0  262 1.0818e+08 6.9117e+07  36.1%     -   14s
     0     0 6.9169e+07    0  262 1.0818e+08 6.9169e+07  36.1%     -   14s
     0     0 6.9174e+07    0  264 1.0818e+08 6.9174e+07  36.1%     -   15s
     0     0 6.9176e+07    0  265 1.0818e+08 6.9176e+07  36.1%     -   15s
H    0     0                    1.074147e+08 6.9176e+07  35.6%     -   15s
     0     0 6.9961e+07    0  230 1.0741e+08 6.9961e+07  34.9%     -   15s
     0     0 7.0295e+07    0  226 1.0741e+08 7.0295e+07  34.6%     -   15s
     0     0 7.0563e+07    0  248 1.0741e+08 7.0563e+07  34.3%     -   15s
     0     0 7.0571e+07    0  226 1.0741e+08 7.0571e+07  34.3%     -   15s
     0     0 7.0575e+07    0  227 1.0741e+08 7.0575e+07  34.3%     -   15s
H    0     0                    1.047586e+08 7.0575e+07  32.6%     -   16s
     0     0 7.1982e+07    0  278 1.0476e+08 7.1982e+07  31.3%     -   16s
     0     0 7.2352e+07    0  273 1.0476e+08 7.2352e+07  30.9%     -   16s
     0     0 7.2448e+07    0  254 1.0476e+08 7.2448e+07  30.8%     -   16s
H    0     0                    1.009104e+08 7.2448e+07  28.2%     -   16s
     0     0 7.2474e+07    0  263 1.0091e+08 7.2474e+07  28.2%     -   16s
     0     0 7.2485e+07    0  258 1.0091e+08 7.2485e+07  28.2%     -   16s
H    0     0                    9.814060e+07 7.2485e+07  26.1%     -   16s
     0     0 7.2491e+07    0  263 9.8141e+07 7.2491e+07  26.1%     -   16s
     0     0 7.2494e+07    0  265 9.8141e+07 7.2494e+07  26.1%     -   16s
H    0     0                    9.707829e+07 7.2494e+07  25.3%     -   16s
H    0     0                    9.676092e+07 7.2494e+07  25.1%     -   16s
     0     0 7.3165e+07    0  267 9.6761e+07 7.3165e+07  24.4%     -   17s
     0     0 7.3165e+07    0  257 9.6761e+07 7.3165e+07  24.4%     -   18s
H    0     0                    9.673392e+07 7.3165e+07  24.4%     -   18s
H    0     0                    9.621491e+07 7.3165e+07  24.0%     -   18s
H    0     0                    9.619279e+07 7.3165e+07  23.9%     -   18s
     0     0 7.4714e+07    0  269 9.6193e+07 7.4714e+07  22.3%     -   18s
H    0     0                    9.604736e+07 7.4714e+07  22.2%     -   18s
     0     0 7.5806e+07    0  246 9.6047e+07 7.5806e+07  21.1%     -   18s
     0     0 7.5985e+07    0  244 9.6047e+07 7.5985e+07  20.9%     -   18s
H    0     0                    9.588311e+07 7.5985e+07  20.8%     -   18s
     0     0 7.6056e+07    0  242 9.5883e+07 7.6056e+07  20.7%     -   18s
     0     0 7.6075e+07    0  240 9.5883e+07 7.6075e+07  20.7%     -   18s
     0     0 7.6087e+07    0  245 9.5883e+07 7.6087e+07  20.6%     -   18s
     0     0 7.6088e+07    0  247 9.5883e+07 7.6088e+07  20.6%     -   18s
H    0     0                    9.488748e+07 7.6088e+07  19.8%     -   18s
H    0     0                    9.486386e+07 7.6088e+07  19.8%     -   18s
H    0     0                    9.439507e+07 7.6088e+07  19.4%     -   19s
     0     0 7.6554e+07    0  235 9.4395e+07 7.6554e+07  18.9%     -   19s
H    0     0                    9.438607e+07 7.6554e+07  18.9%     -   19s
     0     0 7.6658e+07    0  224 9.4386e+07 7.6658e+07  18.8%     -   19s
     0     0 7.6710e+07    0  229 9.4386e+07 7.6710e+07  18.7%     -   19s
     0     0 7.6714e+07    0  228 9.4386e+07 7.6714e+07  18.7%     -   19s
     0     0 7.6715e+07    0  228 9.4386e+07 7.6715e+07  18.7%     -   19s
H    0     0                    9.418324e+07 7.6715e+07  18.5%     -   19s
     0     0 7.7629e+07    0  221 9.4183e+07 7.7629e+07  17.6%     -   19s
     0     0 7.7776e+07    0  228 9.4183e+07 7.7776e+07  17.4%     -   19s
     0     0 7.7802e+07    0  238 9.4183e+07 7.7802e+07  17.4%     -   19s
     0     0 7.7808e+07    0  238 9.4183e+07 7.7808e+07  17.4%     -   19s
H    0     0                    9.410347e+07 7.7808e+07  17.3%     -   19s
     0     0 7.7810e+07    0  239 9.4103e+07 7.7810e+07  17.3%     -   19s
H    0     0                    9.402047e+07 7.7810e+07  17.2%     -   19s
H    0     0                    9.392300e+07 7.7810e+07  17.2%     -   20s
     0     0 7.8553e+07    0  240 9.3923e+07 7.8553e+07  16.4%     -   20s
     0     0 7.8653e+07    0  216 9.3923e+07 7.8653e+07  16.3%     -   20s
     0     0 7.8697e+07    0  229 9.3923e+07 7.8697e+07  16.2%     -   20s
     0     0 7.8705e+07    0  227 9.3923e+07 7.8705e+07  16.2%     -   20s
     0     0 7.8706e+07    0  231 9.3923e+07 7.8706e+07  16.2%     -   20s
     0     0 7.8963e+07    0  225 9.3923e+07 7.8963e+07  15.9%     -   20s
     0     0 7.8998e+07    0  234 9.3923e+07 7.8998e+07  15.9%     -   20s
     0     0 7.9003e+07    0  239 9.3923e+07 7.9003e+07  15.9%     -   20s
     0     0 7.9007e+07    0  233 9.3923e+07 7.9007e+07  15.9%     -   20s
     0     0 7.9007e+07    0  236 9.3923e+07 7.9007e+07  15.9%     -   20s
H    0     0                    9.389070e+07 7.9007e+07  15.9%     -   20s
     0     0 7.9379e+07    0  239 9.3891e+07 7.9379e+07  15.5%     -   20s
     0     0 7.9416e+07    0  239 9.3891e+07 7.9416e+07  15.4%     -   20s
     0     0 7.9436e+07    0  224 9.3891e+07 7.9436e+07  15.4%     -   20s
     0     0 7.9437e+07    0  242 9.3891e+07 7.9437e+07  15.4%     -   20s
     0     0 7.9624e+07    0  214 9.3891e+07 7.9624e+07  15.2%     -   21s
H    0     0                    9.375118e+07 7.9624e+07  15.1%     -   21s
     0     0 7.9649e+07    0  227 9.3751e+07 7.9649e+07  15.0%     -   21s
     0     0 7.9683e+07    0  233 9.3751e+07 7.9683e+07  15.0%     -   21s
     0     0 7.9689e+07    0  232 9.3751e+07 7.9689e+07  15.0%     -   21s
     0     0 7.9689e+07    0  235 9.3751e+07 7.9689e+07  15.0%     -   21s
     0     0 7.9826e+07    0  227 9.3751e+07 7.9826e+07  14.9%     -   21s
     0     0 7.9864e+07    0  230 9.3751e+07 7.9864e+07  14.8%     -   21s
     0     0 7.9865e+07    0  231 9.3751e+07 7.9865e+07  14.8%     -   21s
     0     0 7.9906e+07    0  229 9.3751e+07 7.9906e+07  14.8%     -   21s
     0     0 7.9926e+07    0  232 9.3751e+07 7.9926e+07  14.7%     -   21s
     0     0 7.9926e+07    0  232 9.3751e+07 7.9926e+07  14.7%     -   21s
     0     0 8.0117e+07    0  226 9.3751e+07 8.0117e+07  14.5%     -   22s
H    0     0                    9.367582e+07 8.0117e+07  14.5%     -   22s
     0     0 8.0127e+07    0  229 9.3676e+07 8.0127e+07  14.5%     -   22s
     0     0 8.0127e+07    0  231 9.3676e+07 8.0127e+07  14.5%     -   22s
     0     0 8.0176e+07    0  224 9.3676e+07 8.0176e+07  14.4%     -   22s
     0     0 8.0192e+07    0  224 9.3676e+07 8.0192e+07  14.4%     -   22s
     0     0 8.0197e+07    0  228 9.3676e+07 8.0197e+07  14.4%     -   22s
     0     0 8.0197e+07    0  238 9.3676e+07 8.0197e+07  14.4%     -   22s
     0     0 8.0207e+07    0  247 9.3676e+07 8.0207e+07  14.4%     -   22s
     0     0 8.0209e+07    0  247 9.3676e+07 8.0209e+07  14.4%     -   22s
     0     0 8.0213e+07    0  254 9.3676e+07 8.0213e+07  14.4%     -   23s
     0     0 8.0213e+07    0  223 9.3676e+07 8.0213e+07  14.4%     -   23s
     0     2 8.0746e+07    0  215 9.3676e+07 8.0746e+07  13.8%     -   24s
    68    75 8.7558e+07   11  220 9.3676e+07 8.1573e+07  12.9%   130   25s
H  101   104                    9.365982e+07 8.1573e+07  12.9%   122   25s
H  103   104                    9.365182e+07 8.1573e+07  12.9%   121   25s
H  108   110                    9.353982e+07 8.1573e+07  12.8%   120   25s
H  148   149                    9.350382e+07 8.1573e+07  12.8%   125   26s
H  334   325                    9.347582e+07 8.1642e+07  12.7%  81.4   27s
H  335   325                    9.344573e+07 8.1642e+07  12.6%  82.2   27s
H  336   325                    9.341773e+07 8.1642e+07  12.6%  82.1   27s
  1297  1084 8.6925e+07   16  180 9.3418e+07 8.1642e+07  12.6%  60.3   32s
  1322  1101 9.2175e+07   60  212 9.3418e+07 8.2658e+07  11.5%  59.1   35s
  1382  1141 9.3378e+07   84  222 9.3418e+07 8.5912e+07  8.03%  56.6   40s
  1426  1170 9.2506e+07   67  204 9.3418e+07 8.7793e+07  6.02%  54.8   45s
H 1440  1117                    9.341056e+07 8.8033e+07  5.76%  54.3   46s
  1477  1142 9.2385e+07   25  255 9.3411e+07 8.8342e+07  5.43%  52.9   50s
H 1483  1086                    9.340951e+07 8.8379e+07  5.39%  52.7   50s
  1512  1105 9.1968e+07   32  231 9.3410e+07 8.8457e+07  5.30%  51.7   55s
  1533  1120 8.8489e+07   13  240 9.3410e+07 8.8489e+07  5.27%  72.8   60s
  1584  1154 9.1080e+07   24  192 9.3410e+07 8.9346e+07  4.35%  70.4   65s
  1637  1190 9.1517e+07   18  190 9.3410e+07 9.0025e+07  3.62%  68.2   70s
  1663  1210 9.0197e+07   25  181 9.3410e+07 9.0197e+07  3.44%  97.0   75s
  3100  1993 9.3388e+07  100   14 9.3410e+07 9.0788e+07  2.81%  73.5   80s
  6248  4043 9.3093e+07   60   60 9.3410e+07 9.1238e+07  2.32%  53.4   85s
  7016  4808 9.3400e+07  192   11 9.3410e+07 9.1238e+07  2.32%  51.6   90s
 10529  7896 9.3380e+07  109   11 9.3410e+07 9.1258e+07  2.30%  42.8   95s
 13674 10390 9.2276e+07   35  127 9.3410e+07 9.1317e+07  2.24%  40.8  100s
 14731 11328 9.2876e+07   42  107 9.3410e+07 9.1365e+07  2.19%  41.3  105s
 17342 13741 9.3116e+07   51   97 9.3410e+07 9.1488e+07  2.06%  43.6  110s
 20454 16320 9.2641e+07   40   99 9.3410e+07 9.1523e+07  2.02%  44.0  115s
 22210 17416 9.1719e+07   40  119 9.3410e+07 9.1532e+07  2.01%  42.7  121s
 24616 19550 9.3380e+07  190   11 9.3410e+07 9.1565e+07  1.97%  43.1  125s
 25774 20034 9.3408e+07  179  180 9.3410e+07 9.1572e+07  1.97%  43.2  163s
 25792 20046 9.3388e+07  186  147 9.3410e+07 9.1572e+07  1.97%  43.1  165s
 25823 20067 9.3401e+07  140  145 9.3410e+07 9.1589e+07  1.95%  43.1  170s
 25852 20086 9.2942e+07   49  150 9.3410e+07 9.1860e+07  1.66%  43.0  175s
 25887 20109 9.3406e+07   77  137 9.3410e+07 9.2137e+07  1.36%  43.0  180s
 25917 20129 9.2854e+07   48  129 9.3410e+07 9.2384e+07  1.10%  42.9  185s
 25952 20153 9.2942e+07   49  147 9.3410e+07 9.2493e+07  0.98%  42.9  190s
 25974 20167 9.3408e+07  179  133 9.3410e+07 9.2509e+07  0.96%  42.8  195s
 25997 20183 9.3408e+07  228  148 9.3410e+07 9.2535e+07  0.94%  42.8  200s
 26014 20194 9.3360e+07   64  134 9.3410e+07 9.2541e+07  0.93%  42.8  205s
 26031 20205 9.3408e+07  167  123 9.3410e+07 9.2552e+07  0.92%  42.7  210s
 26053 20220 9.3320e+07   57  134 9.3410e+07 9.2562e+07  0.91%  42.7  215s
 26069 20231 9.3384e+07  203  138 9.3410e+07 9.2567e+07  0.90%  42.7  220s
 26084 20241 9.3400e+07  110  146 9.3410e+07 9.2570e+07  0.90%  42.7  225s
 26093 20247 9.3380e+07  148  151 9.3410e+07 9.2571e+07  0.90%  42.6  230s
 26105 20255 9.2761e+07   42  159 9.3410e+07 9.2572e+07  0.90%  42.6  235s
 26116 20262 9.3381e+07  155  155 9.3410e+07 9.2573e+07  0.90%  42.6  240s
H26123 19247                    9.340238e+07 9.2573e+07  0.89%  42.6  243s
 26130 19251 9.3380e+07  150  152 9.3402e+07 9.2573e+07  0.89%  42.6  245s
 26141 19259 9.2831e+07   43  145 9.3402e+07 9.2573e+07  0.89%  42.6  250s
 26150 19265 9.3183e+07   55  134 9.3402e+07 9.2573e+07  0.89%  42.6  255s
 26157 19275 9.2635e+07   40  121 9.3402e+07 9.2635e+07  0.82%  44.2  260s
 26481 19441 9.3181e+07   69  114 9.3402e+07 9.2664e+07  0.79%  44.7  265s
 27958 20041 9.3322e+07   69   59 9.3402e+07 9.2735e+07  0.71%  46.2  270s
 29710 20602 9.2959e+07   49  130 9.3402e+07 9.2763e+07  0.68%  46.7  275s
H30294 19743                    9.340236e+07 9.2787e+07  0.66%  47.2  276s
 31564 20213 9.3352e+07   80   40 9.3402e+07 9.2840e+07  0.60%  48.4  280s
 33834 20927 9.3214e+07   52  118 9.3402e+07 9.2876e+07  0.56%  49.9  285s
 35260 21258 9.2991e+07   51  124 9.3402e+07 9.2885e+07  0.55%  50.8  292s
 36573 21634     cutoff   61       9.3402e+07 9.2896e+07  0.54%  51.5  296s
 38322 21898 9.3401e+07  172   12 9.3402e+07 9.2909e+07  0.53%  52.1  302s
 38472 23444     cutoff   72       9.3402e+07 9.2915e+07  0.52%  52.3  309s
 41088 23037 infeasible   63       9.3402e+07 9.2931e+07  0.51%  53.1  311s
 43362 23673     cutoff   73       9.3402e+07 9.2944e+07  0.49%  52.8  316s
 44959 24275 9.3329e+07   69   48 9.3402e+07 9.2959e+07  0.47%  54.0  321s
 47074 24752 9.3323e+07   54  163 9.3402e+07 9.2969e+07  0.46%  54.2  325s
 48895 24775 9.3065e+07   54  134 9.3402e+07 9.2981e+07  0.45%  55.1  330s
 49512 24882 9.3220e+07   53  112 9.3402e+07 9.2985e+07  0.45%  55.7  335s
 51407 25216 9.3174e+07   53  133 9.3402e+07 9.2997e+07  0.43%  56.8  341s
 53652 25435 9.3045e+07   52  133 9.3402e+07 9.3005e+07  0.42%  57.4  346s
 55359 25571     cutoff   68       9.3402e+07 9.3017e+07  0.41%  58.4  351s
 57488 25696 9.3385e+07   68   90 9.3402e+07 9.3028e+07  0.40%  59.0  356s
 58735 25743     cutoff   55       9.3402e+07 9.3036e+07  0.39%  59.8  360s
 60752 25918     cutoff   56       9.3402e+07 9.3048e+07  0.38%  60.7  365s
 62997 26190     cutoff   54       9.3402e+07 9.3057e+07  0.37%  61.1  371s
 64575 26798 9.3380e+07  110   11 9.3402e+07 9.3062e+07  0.36%  61.4  376s
 66727 26968 9.3400e+07  121   12 9.3402e+07 9.3070e+07  0.36%  61.7  380s
 68684 26824     cutoff   53       9.3402e+07 9.3080e+07  0.34%  62.2  385s
 70867 27054 9.3319e+07   63   78 9.3402e+07 9.3088e+07  0.34%  62.8  391s
 73217 27568 9.3401e+07   90   12 9.3402e+07 9.3094e+07  0.33%  62.9  396s
 74594 27676 9.3253e+07   55  125 9.3402e+07 9.3098e+07  0.33%  62.8  400s
 77010 28003 9.3346e+07   64   68 9.3402e+07 9.3105e+07  0.32%  62.9  405s
 78676 28402 9.3373e+07   64   66 9.3402e+07 9.3112e+07  0.31%  63.2  410s
 81146 28631 9.3187e+07   55  122 9.3402e+07 9.3117e+07  0.31%  63.2  415s
 83626 29716 9.3283e+07   54  158 9.3402e+07 9.3122e+07  0.30%  63.0  421s
 85208 30692 9.3335e+07   65   71 9.3402e+07 9.3126e+07  0.30%  63.2  426s
 86984 31953 9.3396e+07   85   17 9.3402e+07 9.3130e+07  0.29%  63.0  431s
 89001 33018 9.3389e+07  225    7 9.3402e+07 9.3131e+07  0.29%  62.4  435s
 90695 33966 9.3258e+07   57   87 9.3402e+07 9.3135e+07  0.29%  62.2  440s
 93299 35105 9.3308e+07   49  180 9.3402e+07 9.3139e+07  0.28%  61.9  446s
 94859 35915 9.3216e+07   57  106 9.3402e+07 9.3143e+07  0.28%  62.0  451s
 96531 36745 9.3380e+07  127   13 9.3402e+07 9.3147e+07  0.27%  61.9  455s
 99424 38279 9.3374e+07   61   71 9.3402e+07 9.3151e+07  0.27%  61.5  461s
 101201 38995 9.3400e+07   93   13 9.3402e+07 9.3155e+07  0.27%  61.3  472s
 101549 39327     cutoff   61       9.3402e+07 9.3155e+07  0.26%  61.2  475s
 103213 40193 9.3324e+07   63   76 9.3402e+07 9.3157e+07  0.26%  60.9  480s

Cutting planes:
  Gomory: 16
  Cover: 21
  Implied bound: 286
  Projected implied bound: 94
  MIR: 455
  Flow cover: 1037
  GUB cover: 4
  Inf proof: 2
  RLT: 428
  Relax-and-lift: 13

Explored 103872 nodes (6344300 simplex iterations) in 480.18 seconds
Thread count was 12 (of 12 available processors)

Solution count 10: 9.34024e+07 9.34024e+07 9.34095e+07 ... 9.36518e+07

Time limit reached
Best objective 9.340235880506e+07, best bound 9.315839246478e+07, gap 0.2612%
```

In [32]:
```
m1.objval
```

Out[32]: 93402358.80505842

The minimum total distance is found using a budget of $93,402,358.81

In [33]:
```
max([A[i,j].x*D[i,j] for i in zipcodes for j in sites])
```

Out[33]: 14.657912157781494

The maximum distance in this strategy is 14.66

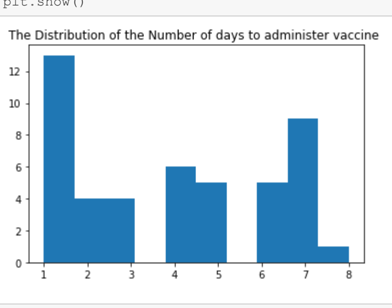
In [ ]:
```
num_sites = 0
for j in sites:
    if (L[j].x) > 0:
        num_sites += 1
print(num_sites)
```

In [36]:
```
for j in sites:
    print("Site:", j+1, '\t', int(L[j].x), '\t', int(S[j].x + M[j].x + L[j].x))
```

```
Site: 1          1       1
Site: 2          1       1
Site: 3          1       1
Site: 4          0       0
Site: 5          1       1
Site: 6          1       1
Site: 7          1       1
Site: 8          1       1
Site: 9          0       0
Site: 10         0       0
Site: 11         0       0
Site: 12         1       1
Site: 13         1       1
Site: 14         1       1
Site: 15         1       1
Site: 16         1       1
Site: 17         1       1
Site: 18         1       1
Site: 19         1       1
Site: 20         1       1
Site: 21         1       1
Site: 22         1       1
Site: 23         1       1
Site: 24         1       1
Site: 25         1       1
Site: 26         1       1
Site: 27         1       1
Site: 28         1       1
Site: 29         0       0
Site: 30         1       1
Site: 31         1       1
Site: 32         1       1
Site: 33         0       1
Site: 34         1       1
Site: 35         1       1
Site: 36         1       1
Site: 37         0       0
Site: 38         1       1
Site: 39         1       1
Site: 40         1       1
Site: 41         0       0
Site: 42         1       1
Site: 43         1       1
Site: 44         1       1
Site: 45         1       1
Site: 46         1       1
Site: 47         0       0
```

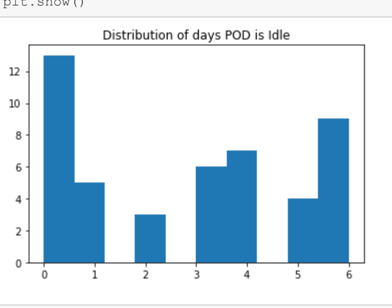
In [38]:
```
max_ts = []
for j in sites:
    max_t = 0
    for t in days:
        if np.allclose(I[j,t].x,0):
            max_t = t + 1
            max_ts.append(max_t)
            break
```

In [39]:
```python
plt.hist(max_ts)
plt.title("The Distribution of the Number of days to administer vaccine ")
plt.show()
```

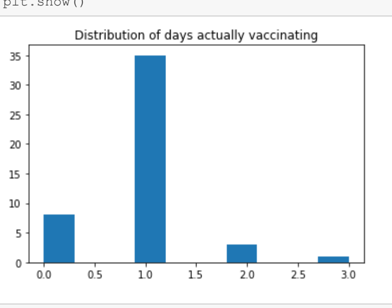


In [40]:
```python
days_not_serving = []
days_serving = []
for j in sites:
    num_days = 0
    serve = 0
    for t in days:
        if np.allclose(X[j,t].x, 0) and np.allclose(I[j,t].x, 0) is False:
            num_days += 1
        if np.allclose(X[j,t].x, 0) is False:
            serve +=1
    days_not_serving.append(num_days)
    days_serving.append(serve)
```

In [41]:
```python
plt.hist(days_not_serving)
plt.title("Distribution of days POD is Idle")
plt.show()
```



In [42]:
```python
plt.hist(days_serving)
plt.title("Distribution of days actually vaccinating")
plt.show()
```



In [43]:
```python
arr = np.genfromtxt('Aij_Ben_total_distance.csv', delimiter=",")
```

In [44]:
```python
arr = arr[:,:47]
```

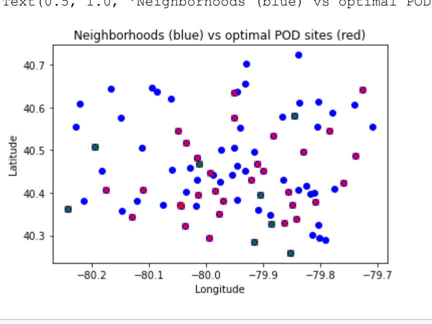
In [45]:
```python
utilized = []
not_utilized = []
for j in sites:
    for i in zipcodes:
        if arr[i,j] == 1:
            utilized.append(j)
            break
        if i == len(zipcodes)-1:
            not_utilized.append(j)
```

In [46]:
```python
utilized = []
not_utilized = []
for j in sites:
    if (S[j].x + M[j].x + L[j].x) > 0:
        utilized.append(j)
    else:
        not_utilized.append(j)
```

In [47]:
```python
print(num_sites)
plt.scatter(neighborhood_longs, neighborhood_lats, marker='o', c = 'blue')
plt.scatter(np.take(POD_longs, utilized), np.take(POD_lats, utilized), marker='x', c = 'red')
plt.scatter(np.take(POD_longs, not_utilized), np.take(POD_lats, not_utilized), marker='x', c = 'green')
plt.xlabel('Longitude')
plt.ylabel('Latitude')
plt.title('Neighborhoods (blue) vs optimal POD sites (red)')
```

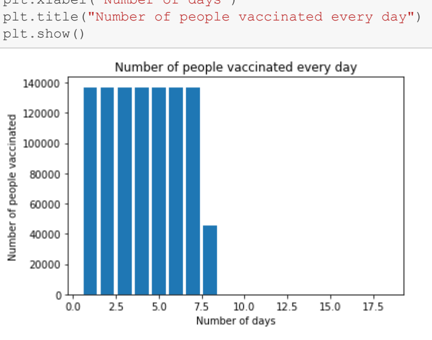
```
1
```

Out[47]:
```
Text(0.5, 1.0, 'Neighborhoods (blue) vs optimal POD sites (red)')
```

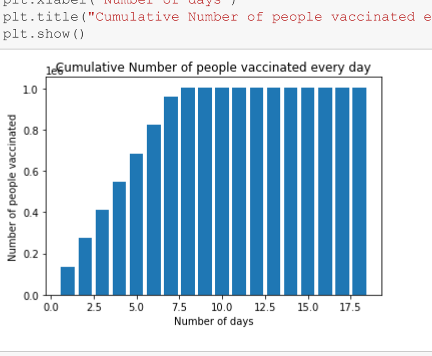


In [48]:
```python
vaccinations = []
cum_vaccinations = []
cum_vaccinated = 0
for t in days:
    vaccinated = sum(X[j,t].x*(e[0]*S[j].x + e[1]*M[j].x + e[2]*L[j].x) for j in sites)
    cum_vaccinated += vaccinated
    vaccinations.append(vaccinated)
    cum_vaccinations.append(cum_vaccinated)
```

In [49]:
```python
timeline = np.arange(100)+1
plt.bar(timeline[:18], vaccinations[:18])
plt.ylabel("Number of people vaccinated")
plt.xlabel("Number of days")
plt.title("Number of people vaccinated every day")
plt.show()
```



In [50]:
```python
timeline = np.arange(100)+1
plt.bar(timeline[:18], cum_vaccinations[:18])
plt.ylabel("Number of people vaccinated")
plt.xlabel("Number of days")
plt.title("Cumulative Number of people vaccinated every day")
plt.show()
```



In [51]:
```python
vaccinations
```

Out[51]:
```
[137000.00000000055,
 137000.00000000006,
 137000.00000000003,
 136990.0000000001,
 137000.0,
 137000.0,
 136821.99999998597,
 45842.99999999998,
 0.0,
 0.0,
 0.0,
 0.0,
 0.0,
 0.0,
 0.0,
 0.0,
 0.0,
 0.0,
 0.0,
 0.0,
 0.0,
 0.0,
 0.0,
 0.0,
 0.0]
```

In [52]:
```python
for j in sites:
    print("Site:", j+1, '\t', sum(A[i,j].x*p[i] for i in zipcodes))
```

```
Site: 1         58890.0
Site: 2         29410.0
Site: 3         29260.0
Site: 4         0.0
Site: 5         27034.0
Site: 6         22530.0
Site: 7         58570.0
Site: 8         16460.0
Site: 9         0.0
Site: 10        0.0
Site: 11        0.0
Site: 12        3950.0
Site: 13        12170.0
Site: 14        54500.0
Site: 15        13776.0
Site: 16        46900.0
Site: 17        24126.0
Site: 18        18560.0
Site: 19        20890.0
Site: 20        26490.0
Site: 21        21380.0
Site: 22        56688.0
Site: 23        24000.0
Site: 24        31130.0
Site: 25        14280.0
Site: 26        18030.0
Site: 27        36970.0
Site: 28        41110.0
Site: 29        0.0
Site: 30        55453.0
Site: 31        34640.0
Site: 32        20120.0
Site: 33        950.0
Site: 34        23380.0
Site: 35        10983.0
Site: 36        5460.0
Site: 37        0.0
Site: 38        10260.0
Site: 39        5850.0
Site: 40        15350.0
Site: 41        0.0
Site: 42        15150.0
Site: 43        35990.0
Site: 44        13854.0
Site: 45        17260.0
Site: 46        32851.0
Site: 47        0.0
```

In [53]:
```python
j = 32
for t in days:
    print(I[j,t].x, '\t', '\t', X[j,t].x)
```

```
950.0           0.0
950.0           0.0
950.0           0.0
950.0           0.0
950.0           0.0
950.0           0.0
0.0             11.176470588235292
0.0             0.0
0.0             0.0
0.0             0.0
0.0             0.0
0.0             0.0
0.0             0.0
0.0             0.0
0.0             0.0
0.0             0.0
0.0             0.0
0.0             0.0
0.0             0.0
0.0             0.0
0.0             0.0
0.0             0.0
0.0             0.0
0.0             0.0
0.0             0.0
```

In [ ]:

In [ ]:

In [ ]:

In [54]:
```python
# Writing to a csv file

with open('Aij_Ben_total_distance.csv', 'w') as f:
    for i in zipcodes:
        for j in sites:
            f.write(str(A[i,j].x) + ',')
        f.write('\n')
print('Done')

with open('Dij_Ben_total_distance.csv', 'w') as f:
    for i in zipcodes:
        for j in sites:
            f.write(str(D[i,j]) + ',')
        f.write('\n')
print('Done')
```

```
Done
Done
```

In [ ]: