課題発表

プログラミング課題 1：経路探索問題
プログラミング課題 2：均衡配分計算

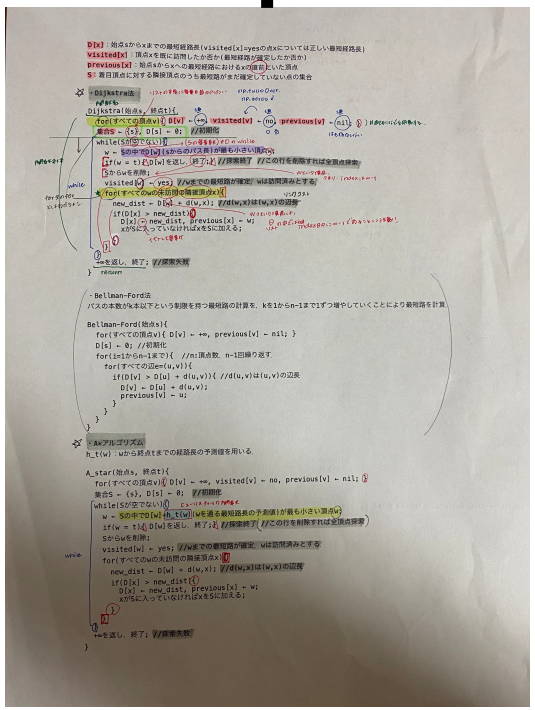
夏学期ゼミ #7 課題1・2発表

2021.4.27

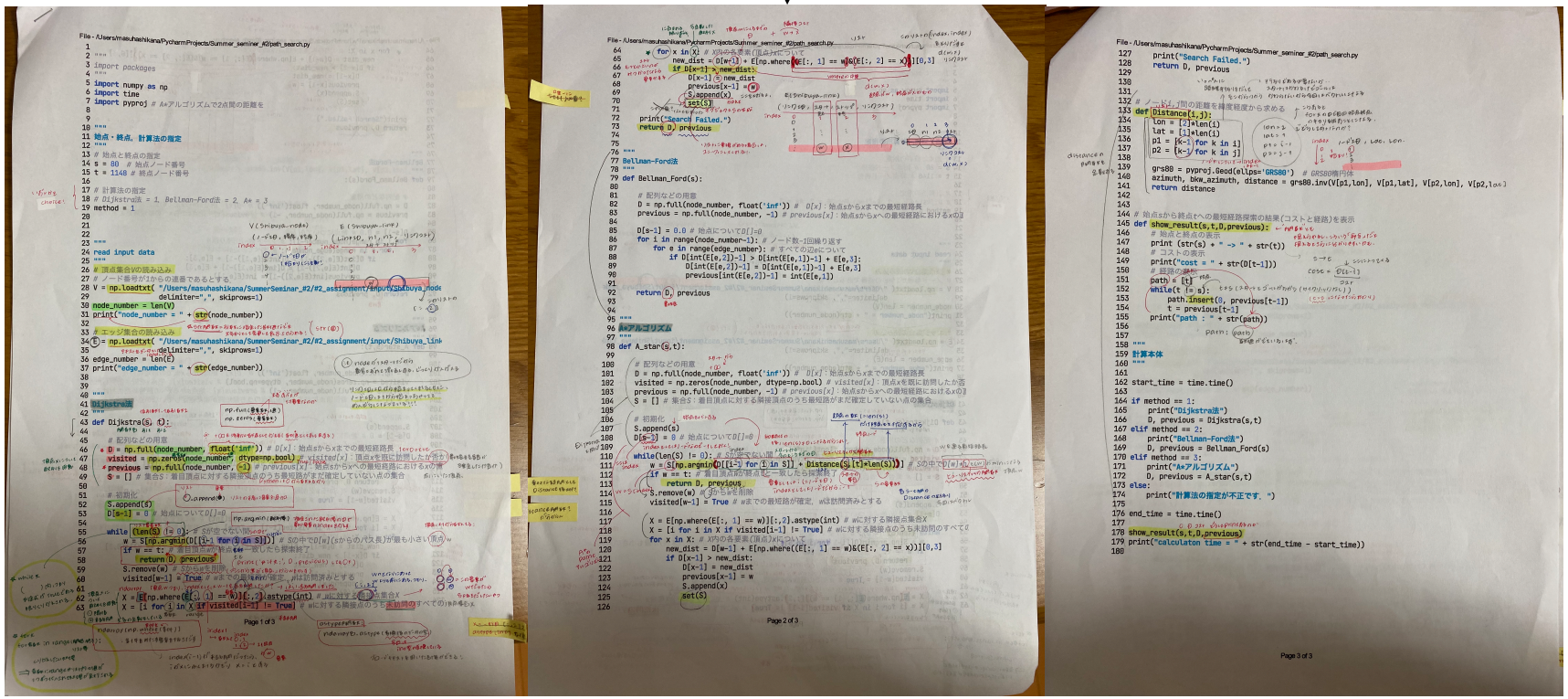
B4 増橋 佳菜

課題１：経路探索問題

[1] 擬似コードと配布解答コードを見比べながらDijkstra法とA*アルゴリズムのコードを理解、実装



擬似コード



解答コード

課題１：経路探索問題

[2] 複数のODペアに対して最短経路探索を行い、計算時間を比較

【始点Oを固定して終点Dのみを変えた】

※始点はデフォルトのs=80を採用し、終点は1~1612で乱数を発生させて選択
※各終点について5回の試行の平均を取ることで平均計算時間を算出

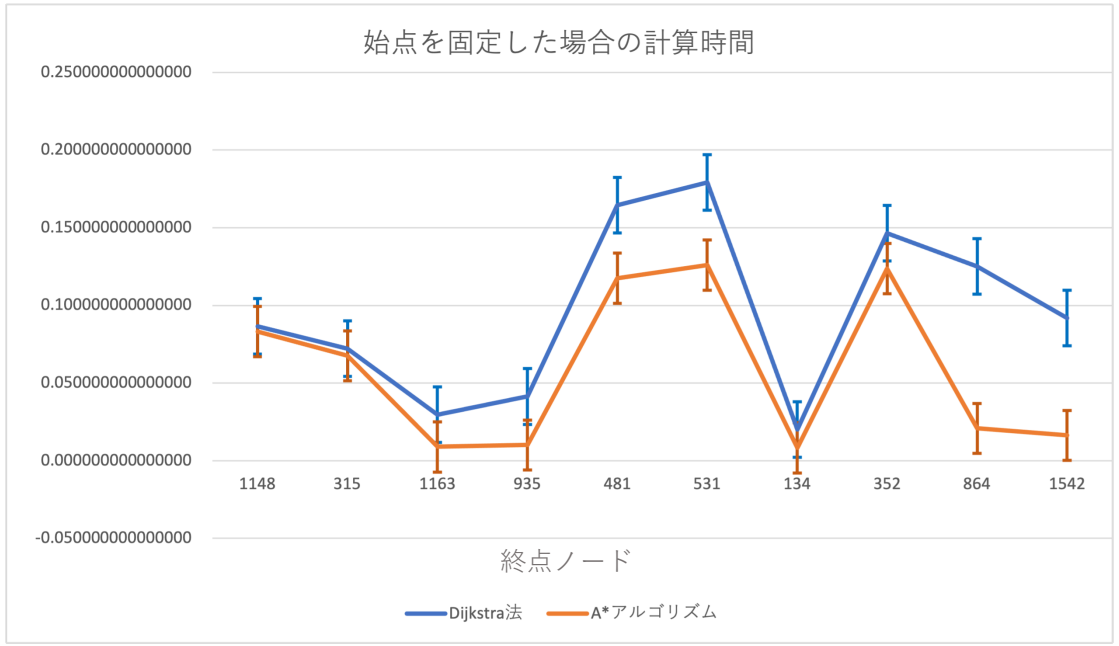
		Dijkstra法		A*アルゴリズム	
始点ノード番号	終点ノード番号	計算時間(s)	平均計算時間(s)	計算時間(s)	平均計算時間(s)
80	531	0.475268840789794	0.17924280166626	0.078425168991089	0.12596635818481
		0.100573062896728		0.134093046188354	
		0.115725755691528		0.190940856933593	
		0.103249311447143		0.140414953231811	
		0.101397037506103		0.085957765579224	
	134	0.015494823455811	0.02022352218628	0.008280038833618	0.00836782455444
		0.037010908126831		0.012944221496582	
		0.015019893646240		0.007402896881104	
		0.018942117691040		0.006479024887085	
		0.014649868011475		0.006732940673828	
	352	0.087343215942383	0.14645571708679	0.074469089508057	0.12368779182434
		0.170404195785522		0.162589788436889	
		0.202689886093139		0.082463264465332	
		0.176038265228271		0.076770782470703	
		0.095803022384644		0.222146034240722	
	864	0.130098104476928	0.12509999275208	0.014434099197388	0.02084445953369
		0.144720077514648		0.040719985961914	
		0.076200008392334		0.016781806945801	
		0.179909944534301		0.014953136444092	
		0.094571828842163		0.017333269119263	
	1542	0.042329072952271	0.09187774658203	0.020531892776489	0.01630105972290
		0.097798824310303		0.018459320068359	
		0.117641925811767		0.014917135238647	
		0.124077796936035		0.013700008392334	
		0.077541112899780		0.013896942138672	

		Dijkstra法		A*アルゴリズム	
始点ノード番号	終点ノード番号	計算時間(s)	平均計算時間(s)	計算時間(s)	平均計算時間(s)
80	1148	0.103110790252685	0.08663988113403	0.067436933517456	0.08322081565857
		0.065914869308472		0.078732013702393	
		0.081212759017944		0.159758090972900	
		0.086468935012817		0.055138111114502	
		0.096492052078247		0.055038928985596	
	315	0.071712017059326	0.07218980789185	0.065950870513916	0.06756963729858
		0.076802968978882		0.071766138076782	
		0.071573019027710		0.053704023361206	
		0.069389104843140		0.052250146865845	
		0.071471929550171		0.094177007675171	
	1163	0.031314849853516	0.02967052459717	0.009779214859009	0.00894937515259
		0.039903163909912		0.008975267410278	
		0.023914813995361		0.008886098861694	
		0.027792930603027		0.008557081222534	
		0.025426864624023		0.008549213409424	
	935	0.043256998062134	0.04136118888855	0.012072086334229	0.01009898185730
		0.042316913604736		0.008738040924072	
		0.039439201354980		0.009297847747803	
		0.037389755249023		0.008930921554565	
		0.044403076171875		0.011456012725830	
	481	0.182999134063720	0.16452107429504	0.121071815490722	0.11759247779846
		0.264191150665283		0.106375932693481	
		0.135291814804077		0.108183860778808	
		0.129895210266113		0.132543087005615	
		0.110228061676025		0.119787693023681	

課題１：経路探索問題

[2] 複数のODペアに対して最短経路探索を行い、計算時間を比較

【始点Oを固定して終点Dのみを変えた】



A*アルゴリズムの方がDijkstra法よりも計算時間が短い：

A*アルゴリズムは1つの終点までの最短経路を方向づけながら探索することでDijkstra法に比して計算時間が短くなるという利点がある。

A*アルゴリズムの方が計算時間が終点に依存する？：

A*アルゴリズムは1つの終点に対して最短経路を探索する一方で、Dijkstra法は終点がどこであっても全てのノードに対する最短経路を探索するため、A*アルゴリズムの方が計算時間の終点への依存度が高いと予想されたが…

実行する際のばらつきが大きく影響している可能性がある。

課題2：均衡配分

[1] 東京都NWで均衡配分を実行した結果をQGISで表現

本当は....

Tips：混雑を判断したければ交通量でなく**混雑度**が適切

$$(\text{混雑度}) = \frac{12\text{時間交通量}}{12\text{時間交通容量}}$$

※均衡配分のアウトプット交通量は**24**時間交通量

12時間交通量

道路交通センサスから $\frac{\mathbf{12}\text{時間交通量}}{\mathbf{24}\text{時間交通量}}$ を出して係数とする

12時間交通容量

＝ 交通容量**350**(台/$\boldsymbol{h}$・車線)×**12**($\boldsymbol{h}$)×車線数

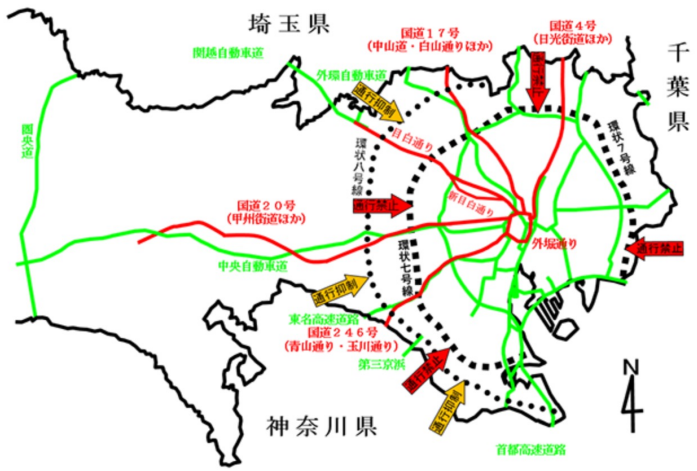
課題2：均衡配分

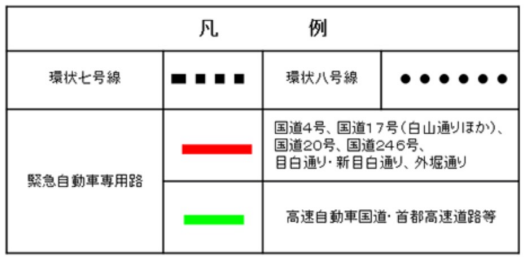
[2] 首都直下発生直後に一般車車両交通規制が実施される高速道路に該当するリンクを切断するシナリオ

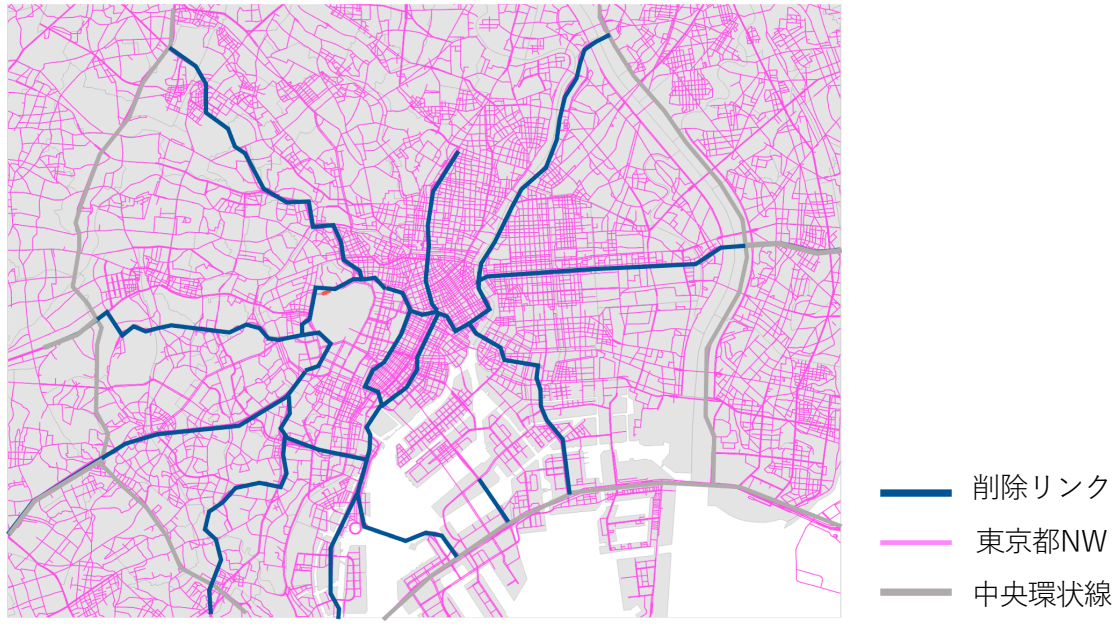
※規制対象道路のうち首都高速中央環状線以内の放射状高速道路のリンクIDをQGISから検索して切断リンクとした(作業工数の都合)

※上下線ともにリンクを削除して処理(実際は上り方面のみ規制の場合あり)

首都直下直後の第一交通規制がなされた場合、
都内の交通量がどこに流れていくのか？
どこに支障が出てくるのか？







上図 引用先リンク：
「第一次交通規制(道路交通法に基づいて実施します)　警視庁」
https://www.keishicho.metro.tokyo.jp/kurashi/saigai/shinsai_kisei/kisei/kisei_1.html

課題2：均衡配分【交通規制の表現方法】

[方法１] リンク自体を削除する手法

リンクを削除したTokyoLink.csvデータの作成

[Tips]
for文は無闇に使わない方が良い
もっと高処理のMethodがあれば使う方が良い

```python
import pandas as pd
import csv
import time
import random

start_time = time.time()

# 削除したいLinkIDのリストを用意
dellist = [...]

#削除リンク数
print('削除リンク数= ',len(dellist))
#>>>>>削除リンク数=  2751

# csvファイルを読み込む
df = pd.read_csv("/Users/masuhashikana/SummerSeminar_#3/assignment\
/equilibrium_mac_Xcode/input_edit/Tokyo/TokyoLink.csv")
print('抽出前のcsvファイルの行数 = ',len(df))
#>>>>>抽出前のcsvファイルの行数 =  431488

# もともとのDataFrameの行数
A = len(df)
print(A)
#>>>>431488

# dfから，'LinkID'列の値がdellistに含まれないものを残す
#(これは不要)for link_id in dellist:
#df_removed = df[df.LinkID != link_id]の応用ver.
df_removed = df.query('LinkID not in @dellist')
```

QGISで該当リンクIDを検索

```python
print(df_removed)
print('抽出後のcsvファイルの行数 = ',len(df_removed))
#>>>>抽出後のcsvファイルの行数 =  428843

# 処理後のDataFrameの行数
B = len(df_removed)

#削除したリンク数と一致しているか確認
REMOVE = A-B
print('処理前後の行数の差： ',REMOVE)
#>>>>>処理前後の行数の差：  2645

#csvファイルとして出力(均衡配分に使用する)
df_removed.to_csv("/Users/masuhashikana/SummerSeminar_#3/assignment\
/equilibrium_mac_Xcode/input_edit/Tokyo/TokyoLink_removed.csv")
print('ファイル作成完了')

end_time = time.time()
print("calculation time = " + str(end_time - start_time))
#>>>>>>calculation time = 2.523705244064331 (例)
```

疑問①(削除リンク数)>(削除行数)

解釈①
Dellistにありながら東京都NWにないものがあった
＝連番と判断した中に欠番があった?
手作業によるミス…

リンクが削除されたLinkのcsvファイルの作成が完了！

課題2：均衡配分 【交通規制の表現方法】

[方法１] リンク自体を削除する手法

リンク削除によって始点にも終点にもならなくなったノードを削除したTokyoNode.csvデータを作成

```python
import pandas as pd
import csv
import time
import numpy as np

##おおまかな方針
# 削除後のLinkファイルからO_id, D_idのリストSを作成
#for i  in (Node_id集合)
#Node.csvのうち、i not in S の行を削除

start_time = time.time()

# csvファイルを読み込む
NodeData = pd.read_csv('/Users/masuhashikana/SummerSeminar_#3/assignment\
/equilibrium_mac_Xcode/input_edit/Tokyo/\
TokyoNode.csv',\ sep=',', encoding='utf-8', index_col=False, header=None)

LinkData = pd.read_csv('/Users/masuhashikana/SummerSeminar_#3/assignment\
/equilibrium_mac_Xcode/input_edit/Tokyo/\
TokyoLink_removed.csv', sep=',', encoding='utf-8', index_col=False, header=None)

####保持したいNodeIDのリストを用意

#O_idとD_idのリストを別々に作成
O_id_list = list(LinkData[1])
D_id_list = list(LinkData[2])

#O_id_list = map((lambda i: int(i)), O_id_LIST)
#D_id_list = map((lambda j: int(j)), D_id_LIST)
```

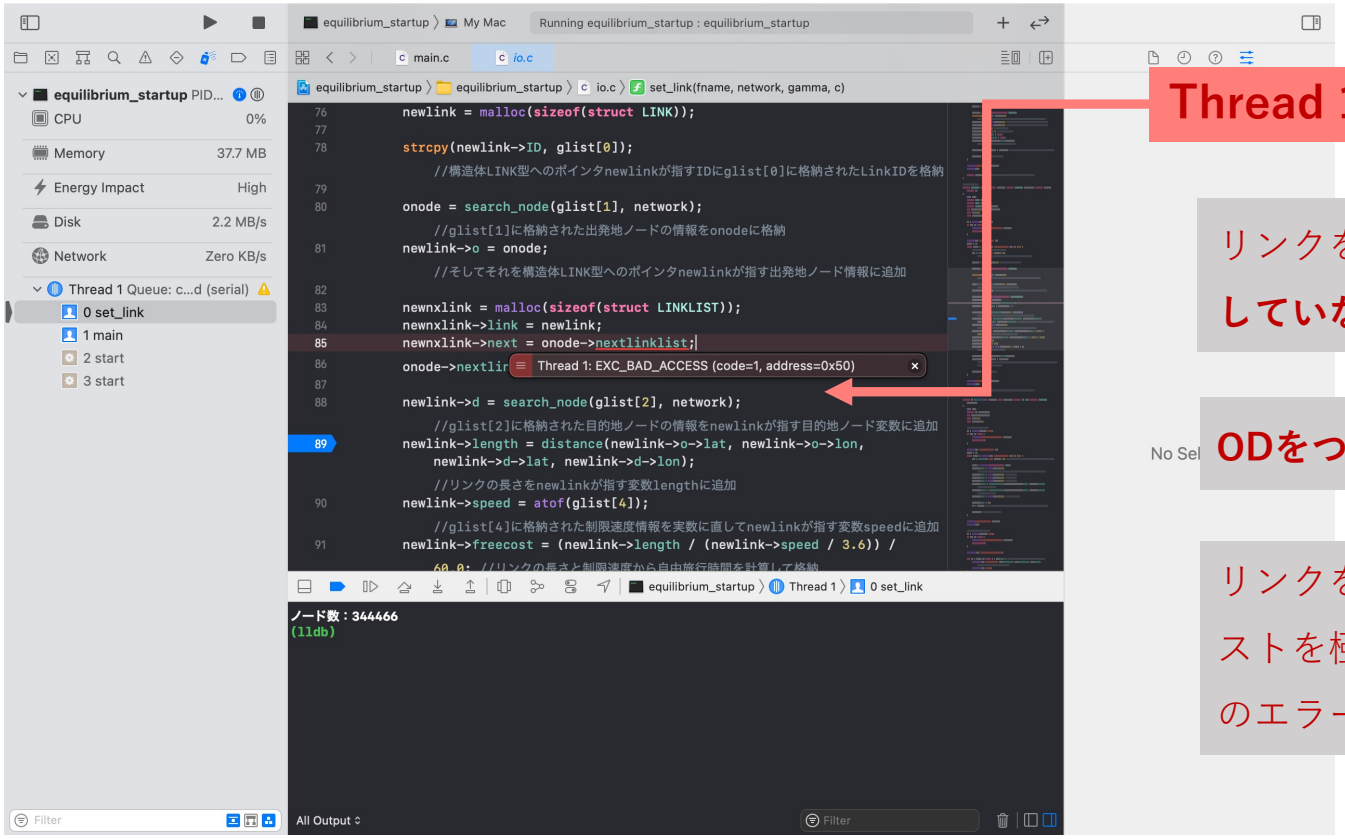
```python
print('O_idリストの要素数 = ',len(O_id_list))
#>>>>>>O_idリストの要素数 =  428844
print('D_idリストの要素数 = ',len(D_id_list))
#>>>>>>D_idリストの要素数 =  428844

#print(O_id_list)
#print(D_id_list)

# 先頭行を削除しておく(先頭行は名前なので並べ替えでエラーの原因)
del O_id_list[0]
del D_id_list[0]

#重複要素はユニークにする
set(O_id_list)
set(D_id_list)

########ここまでは問題なし###########

#各々を適切な順番に並べ替える(必須ではない)
#sorted(list(O_id_list))
#sorted(list(D_id_list))

#リストの結合・重複の削除
Nodelist = O_id_list + D_id_list
set(Nodelist)
print('Node集合の要素数：',len(Nodelist))
#>>>>>>Node集合の要素数：  857686
```

```python
# Nodeのcsvファイルを読み込む
df_Node = pd.read_csv("/Users/masuhashikana/SummerSeminar_#3/assignment\
/equilibrium_mac_Xcode/input_edit/Tokyo/TokyoNode.csv")
print('抽出前のcsvファイルの行数 = ',len(df_Node))
#>>>>>>抽出前のcsvファイルの行数 =  344466

# dfから, 'NodeID'列の値がNodelistに含まれているものを残す(抽出)
df_Node_removed = df_Node.query('NodeID in @Nodelist')

print('抽出後のcsvファイルの行数 = ', len(df_Node_removed))
#>>>>>>抽出後のcsvファイルの行数 =  342051

#Nodeのcsvファイルとして出力(均衡配分に使用する)
df_Node_removed.to_csv("/Users/masuhashikana/SummerSeminar_#3/assignment\
/equilibrium_mac_Xcode/input_edit/Tokyo/TokyoNode_removed.csv")
print('ファイル作成完了')

print('処理完了')
end_time = time.time()
print("calculation time = " + str(end_time - start_time))
#>>>>>>calculation time = 3.653381109237671(例)
```

不要ノードが削除されたNodeのcsvファイルの作成が完了！

課題2：均衡配分【交通規制の表現方法】

[方法１] リンク自体を削除する手法



Thread 1: EXC_BAD_ACCESS（code=1, address=0x50）

リンクを削除して作成したネットワーク内に**ODを繋ぐ経路が存在していない可能性大！**

ODをつなぐ経路がひとつもない場合、均衡配分は計算できない！

リンクを切断するのではなく、交通規制対象のリンクのリンクコストを極端に大きくする方法で対応すれば経路は存在するためこのエラーは回避できる！

→この方法を試してみる！

課題2：均衡配分 【交通規制の表現方法】

[方法２] 交通規制該当リンクに対して極端にリンクコストを大きくする

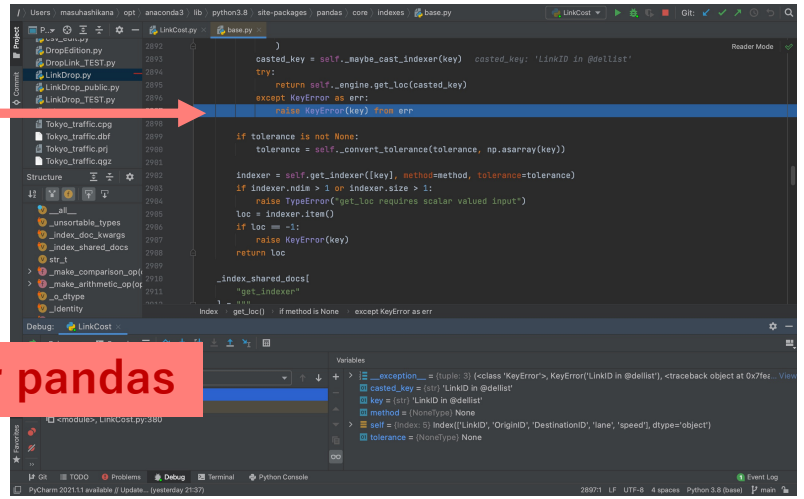
方針：該当リンクの車線数と最高速度を極端に小さくする

```python
import pandas as pd
import csv
import time


start_time = time.time()

# 削除したいLinkIDのリストを用意
dellist = [...]
print(dellist)

# csvファイルを読み込む
df = pd.read_csv("/Users/masuhashikana/SummerSeminar_#3/assignment\
/equilibrium_mac_Xcode/input_edit/Tokyo/TokyoLink.csv")
print('抽出前のcsvファイルの行数 = ',len(df))
#>>>>>抽出前のcsvファイルの行数 =  431488

#削除したい行の行indexのリスト
##python内包表記を使用した方法で
#DelLinkIndex = [n-1 for n in dellist]
#print(DelLinkIndex)

# dfから，'LinkID'列の値がdellistに含まれるもののみリンクコストを極端に大きくする
#車線数(lane)=0，最高速度(speed)=0 とする(不適だったら後から適切な値にする)
#for i in dellist:
df_cost=df[df['LinkID in @dellist']]
df_cost.loc[3] = 0
df_cost.loc[4] = 0

#df_cost = df.query('LinkID in @dellist')


#csvファイルとして出力(均衡配分に使用する)
df_cost.to_csv("/Users/masuhashikana/SummerSeminar_#3/assignment\
/equilibrium_mac_Xcode/input_edit/Tokyo/TokyoLink_cost.csv")
print('ファイル作成完了')

end_time = time.time()
print("calculation time = " + str(end_time - start_time))
```

DataFrameにおいて
LinkIDがdellistに含まれる行について
3列目(0列目,1列目,...)の要素をXに、
4列目(0列目,1列目,...)の要素をYに変更する

```python
#######その1#########
df.loc[df['LinkID' in dellist, 'lane']] = 0
df.loc[df['LinkID' in dellist, 'speed']] = 0

#######その2#########
df.loc[df['LinkID' in @dellist, 'lane']] = 0
df.loc[df['LinkID' in @dellist, 'speed']] = 0

#######その3#########
df.loc[df['LinkID in @dellist', 'lane']] = 0
df.loc[df['LinkID in @dellist', 'speed']] = 0

#######その4#########
df.loc[df['LinkID' in @dellist, 'lane']] = 0
df.loc[df['LinkID' in @dellist, 'speed']] = 0

#######その5#########
for i in dellist:
    df.loc[df['LinkID'= i, 'lane']] = 0
    df.loc[df['LinkID'= i, 'speed']] = 0
```

←このX,Yを
出来る限り小さい正数とする



課題2：均衡配分 【交通規制の表現方法】

[方法２] 交通規制該当リンクに対して極端にリンクコストを大きくする

方針：該当リンクの車線数と最高速度を極端に小さくする

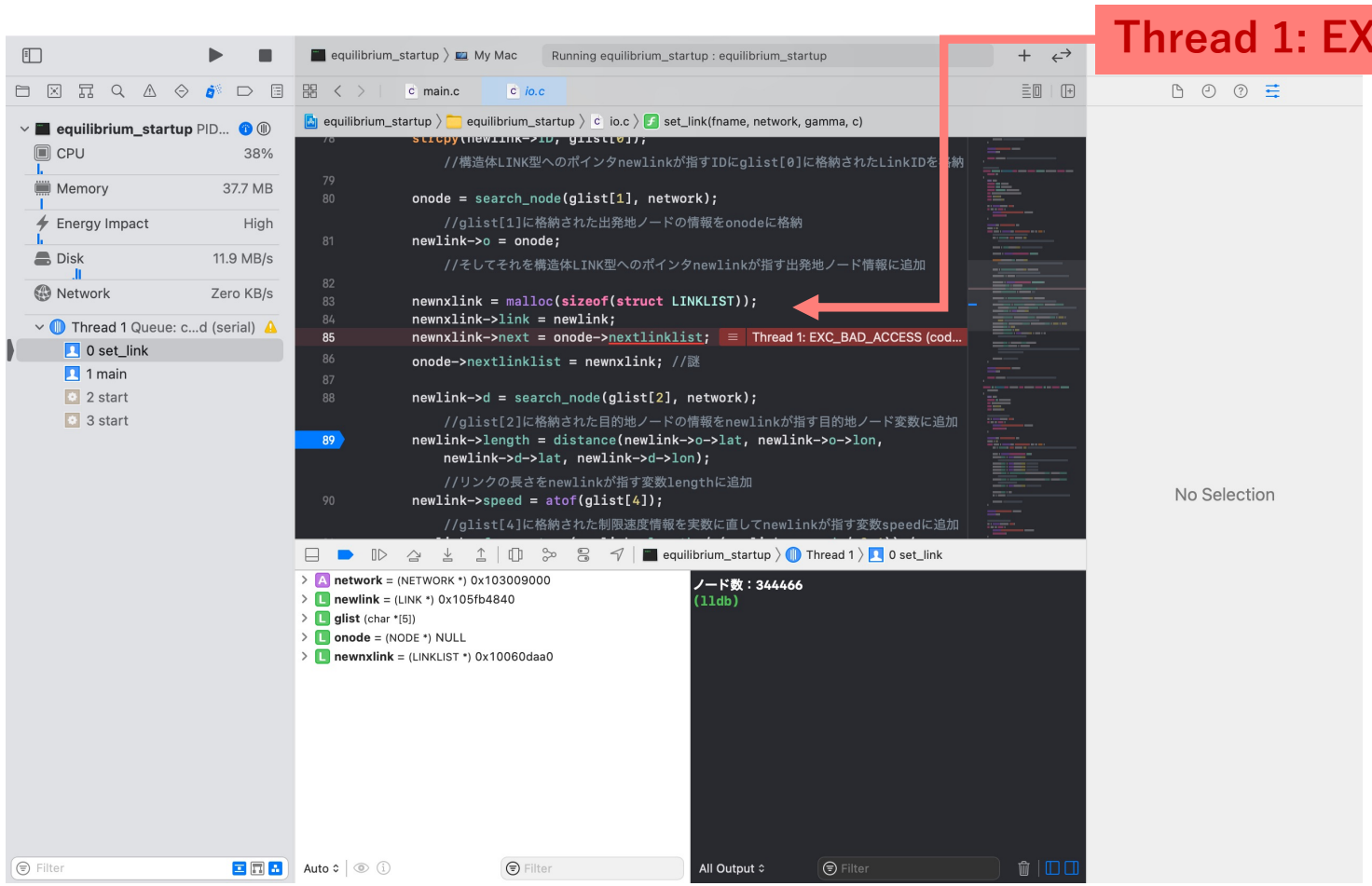
```python
import pandas as pd
import csv
import time


start_time = time.time()

# 削除したいLinkIDのリストを用意
dellist = [...]
print(dellist)

# csvファイルを読み込む
df = pd.read_csv("/Users/masuhashikana/SummerSeminar_#3/assignment\
/equilibrium_mac_Xcode/input_edit/Tokyo/TokyoLink.csv")
print('抽出前のcsvファイルの行数 = ',len(df))
#>>>>>抽出前のcsvファイルの行数 =  431488

print(df)

#削除したい行の行indexのリスト
##python内包表記を使用した方法で
#DelLinkIndex = [n-1 for n in dellist]
#print(DelLinkIndex)

# dfから, 'LinkID'列の値がdellistに含まれるもののみリンクコストを極端に大きくする
#車線数(lane)=0, 最高速度(speed)=0 とする(不適だったら後から適切な値にする)
#for i in dellist:
###df_cost=df[df['LinkID in @dellist']]
###df_cost.loc[3] = 0
###df_cost.loc[4] = 0
"""
#######その1#########
df.loc[df['LinkID' in dellist, 'lane']] = 0
df.loc[df['LinkID' in dellist, 'speed']] = 0

#######その2#########
df.loc[df['LinkID' in @dellist, 'lane']] = 0
df.loc[df['LinkID' in @dellist, 'speed']] = 0

#######その3#########
df.loc[df['LinkID in @dellist', 'lane']] = 0
df.loc[df['LinkID in @dellist', 'speed']] = 0

#######その4#########
df.loc[df['LinkID' in @dellist, 'lane']] = 0
df.loc[df['LinkID' in @dellist, 'speed']] = 0

#######その5#########
for i in dellist:
    df.loc[df['LinkID'= i, 'lane']] = 0
    df.loc[df['LinkID'= i, 'speed']] = 0

"""
#######その6##########
#df.loc[条件, ['lane', 'speed']] = 0
for i in dellist:
    df.loc[df['LinkID']== i-1, ['lane','speed']] = 0

print(df)


#df_cost = df.query('LinkID in @dellist')

#csvファイルとして出力(均衡配分に使用する)
df.to_csv("/Users/masuhashikana/SummerSeminar_#3/assignment\
/equilibrium_mac_Xcode/input_edit/Tokyo/TokyoLink_cost.csv")
print('ファイル作成完了')

end_time = time.time()
print("calculation time = " + str(end_time - start_time))
#>>>>>>>calculation time = 5.972723007202148
```

規制リンクのコストを最大化したcsvファイルの作成が完了！

課題2：均衡配分【交通規制の表現方法】

[方法２] 交通規制該当リンクに対して極端にリンクコストを大きくする

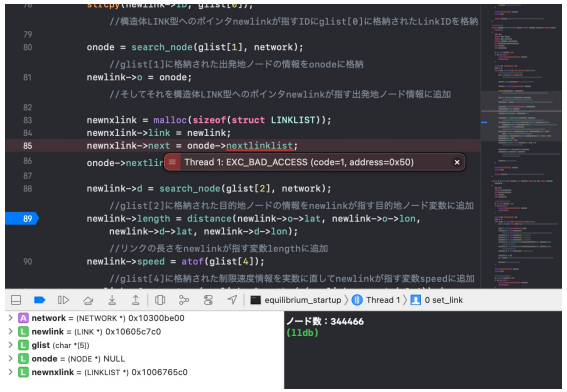
方針：該当リンクの車線数と最高速度を極端に小さくする



同じエラー

車線数、最高速度を0にしたのがいけなかった?

```
for i in dellist:
    df.loc[df['LinkID']== i-1, ['lane','speed']] = 1
```



同じエラー

課題2：均衡配分 【交通規制の表現方法】

[方法２] 交通規制該当リンクに対して極端にリンクコストを大きくする

原因は...

TokyoLink.csv

```
LinkID,起点ノードID,終点ノードID,車線数,最高速度
1,100,101,2,30
2,101,117500,2,30
3,117500,201,2,30
4,201,100,2,30
5,200,301,2,30
6,301,302,2,30
7,302,303,2,30
8,303,120500,2,30
9,120500,401,2,30
10,401,402,2,30
11,402,403,2,30
12,403,200,2,30
13,300,501,2,30
14,501,502,2,30
15,502,503,2,30
16,503,504,2,30
17,504,505,2,30
18,505,184100,2,30
19,184100,601,2,30
20,601,602,2,30
21,602,603,2,30
22,603,604,2,30
23,604,605,2,30
24,605,300,2,30
25,500,701,2,30
26,701,702,2,30
27,702,703,2,30
28,703,704,2,30
29,704,705,2,30
30,705,4100,2,30
31,4100,801,2,30
32,801,802,2,30
33,802,803,2,30
34,803,804,2,30
35,804,805,2,30
36,805,500,2,30
37,600,901,2,30
38,901,902,2,30
39,902,903,2,30
40,903,4100,2,30
41,4100,1001,2,30
42,1001,1002,2,30
43,1002,1003,2,30
44,1003,600,2,30
45,700,1101,2,30
46,1101,1102,2,30
47,1102,1103,2,30
48,1103,1104,2,30
49,1104,1105,2,30
50,1105,1106,2,30
51,1106,1107,2,30
52,1107,1108,2,30
53,1108,4700,2,30
54,4700,1201,2,30
55,1201,1202,2,30
56,1202,1203,2,30
57,1203,1204,2,30
58,1204,1205,2,30
```

もとのリンクファイル

TokyoLink_cost3.csv

```
,LinkID,起点ノードID,終点ノードID,車線数,最高速度
0,1,100,101,2,30
1,2,101,117500,2,30
2,3,117500,201,2,30
3,4,201,100,2,30
4,5,200,301,2,30
5,6,301,302,2,30
6,7,302,303,2,30
7,8,303,120500,2,30
8,9,120500,401,2,30
9,10,401,402,2,30
10,11,402,403,2,30
11,12,403,200,2,30
12,13,300,501,2,30
13,14,501,502,2,30
14,15,502,503,2,30
15,16,503,504,2,30
16,17,504,505,2,30
17,18,505,184100,2,30
18,19,184100,601,2,30
19,20,601,602,2,30
20,21,602,603,2,30
21,22,603,604,2,30
22,23,604,605,2,30
23,24,605,300,2,30
24,25,500,701,2,30
25,26,701,702,2,30
26,27,702,703,2,30
27,28,703,704,2,30
28,29,704,705,2,30
29,30,705,4100,2,30
30,31,4100,801,2,30
31,32,801,802,2,30
32,33,802,803,2,30
33,34,803,804,2,30
34,35,804,805,2,30
35,36,805,500,2,30
36,37,600,901,2,30
37,38,901,902,2,30
38,39,902,903,2,30
39,40,903,4100,2,30
40,41,4100,1001,2,30
41,42,1001,1002,2,30
42,43,1002,1003,2,30
43,44,1003,600,2,30
44,45,700,1101,2,30
45,46,1101,1102,2,30
46,47,1102,1103,2,30
47,48,1103,1104,2,30
48,49,1104,1105,2,30
49,50,1105,1106,2,30
50,51,1106,1107,2,30
51,52,1107,1108,2,30
52,53,1108,4700,2,30
53,54,4700,1201,2,30
54,55,1201,1202,2,30
55,56,1202,1203,2,30
56,57,1203,1204,2,30
57,58,1204,1205,2,30
```

新しく作成したリンクファイル

csvファイルを作成するときにindexを表す行が追加されて形式が変わったこと.....!

csvから特定の行を削除する方法で検討中...
問題：列ラベルがないので行ラベルで指定できない

解決策：勝手に列ラベルをつけて削除しよう！

課題2：均衡配分 【交通規制の表現方法】

[方法２] 交通規制該当リンクに対して極端にリンクコストを大きくする

消したい列にラベルがなかったので「Del」というラベルをつけ、
列を削除するコードで処理

LinkCost.py | DeleteEdit.py | base.py

```python
import pandas as pd
import csv
import time

start_time = time.time()

# csvファイルを読み込む
df = pd.read_csv("/Users/masuhashikana/SummerSeminar_#3/assignment\
/equilibrium_mac_Xcode/input_edit/Tokyo/TokyoLink_cost3.csv")

#0列目を削除する必要がある
df.drop('Del', axis=1)
print(df)

#csvファイルとして出力(均衡配分に使用する)
df.to_csv("/Users/masuhashikana/SummerSeminar_#3/assignment\
/equilibrium_mac_Xcode/input_edit/Tokyo/TokyoLink_cost3Del.csv")

print('ファイル作成完了')

end_time = time.time()
print("calculation time = " + str(end_time - start_time))
```

処理前

TokyoLink_cost3.csv

```
Del,LinkID,OriginID,DestinationID,lane,speed
0,1,100,101,2,30
1,2,101,117500,2,30
2,3,117500,201,2,30
3,4,201,100,2,30
4,5,200,301,2,30
5,6,301,302,2,30
6,7,302,303,2,30
7,8,303,120500,2,30
8,9,120500,401,2,30
9,10,401,402,2,30
10,11,402,403,2,30
11,12,403,200,2,30
12,13,300,501,2,30
13,14,501,502,2,30
14,15,502,503,2,30
15,16,503,504,2,30
16,17,504,505,2,30
17,18,505,184100,2,30
18,19,184100,601,2,30
19,20,601,602,2,30
20,21,602,603,2,30
21,22,603,604,2,30
22,23,604,605,2,30
23,24,605,300,2,30
24,25,500,701,2,30
25,26,701,702,2,30
26,27,702,703,2,30
27,28,703,704,2,30
28,29,704,705,2,30
29,30,705,4100,2,30
30,31,4100,801,2,30
31,32,801,802,2,30
32,33,802,803,2,30
33,34,803,804,2,30
34,35,804,805,2,30
35,36,805,500,2,30
36,37,600,901,2,30
37,38,901,902,2,30
38,39,902,903,2,30
39,40,903,4100,2,30
40,41,4100,1001,2,30
41,42,1001,1002,2,30
42,43,1002,1003,2,30
43,44,1003,600,2,30
44,45,700,1101,2,30
45,46,1101,1102,2,30
46,47,1102,1103,2,30
47,48,1103,1104,2,30
48,49,1104,1105,2,30
49,50,1105,1106,2,30
50,51,1106,1107,2,30
51,52,1107,1108,2,30
52,53,1108,4700,2,30
53,54,4700,1201,2,30
54,55,1201,1202,2,30
55,56,1202,1203,2,30
56,57,1203,1204,2,30
57,58,1204,1205,2,30
```

処理後

TokyoLink_cost3Del.csv

```
,Del,LinkID,OriginID,DestinationID,lane,speed
0,0,1,100,101,2,30
1,1,2,101,117500,2,30
2,2,3,117500,201,2,30
3,3,4,201,100,2,30
4,4,5,200,301,2,30
5,5,6,301,302,2,30
6,6,7,302,303,2,30
7,7,8,303,120500,2,30
8,8,9,120500,401,2,30
9,9,10,401,402,2,30
10,10,11,402,403,2,30
11,11,12,403,200,2,30
12,12,13,300,501,2,30
13,13,14,501,502,2,30
14,14,15,502,503,2,30
15,15,16,503,504,2,30
16,16,17,504,505,2,30
17,17,18,505,184100,2,30
18,18,19,184100,601,2,30
19,19,20,601,602,2,30
20,20,21,602,603,2,30
21,21,22,603,604,2,30
22,22,23,604,605,2,30
23,23,24,605,300,2,30
24,24,25,500,701,2,30
25,25,26,701,702,2,30
26,26,27,702,703,2,30
27,27,28,703,704,2,30
28,28,29,704,705,2,30
29,29,30,705,4100,2,30
30,30,31,4100,801,2,30
31,31,32,801,802,2,30
32,32,33,802,803,2,30
33,33,34,803,804,2,30
34,34,35,804,805,2,30
35,35,36,805,500,2,30
36,36,37,600,901,2,30
37,37,38,901,902,2,30
38,38,39,902,903,2,30
39,39,40,903,4100,2,30
40,40,41,4100,1001,2,30
41,41,42,1001,1002,2,30
42,42,43,1002,1003,2,30
43,43,44,1003,600,2,30
44,44,45,700,1101,2,30
45,45,46,1101,1102,2,30
46,46,47,1102,1103,2,30
47,47,48,1103,1104,2,30
48,48,49,1104,1105,2,30
49,49,50,1105,1106,2,30
50,50,51,1106,1107,2,30
51,51,52,1107,1108,2,30
52,52,53,1108,4700,2,30
53,53,54,4700,1201,2,30
54,54,55,1201,1202,2,30
55,55,56,1202,1203,2,30
56,56,57,1203,1204,2,30
57,57,58,1204,1205,2,30
```

削除どころか同じものがさらに増加??

課題2：均衡配分 【交通規制の表現方法】

配分計算でシナリオ実験をする際の注意点：

- ✓ **ODが経路で繋がっていない場合、交通量の配分はできない**
  （リンク自体を削除すると周辺確認が必須になるので煩雑に）

- ✓ リンク切断のシナリオはリンク自体を削除するのではなく、
  リンクコストを極端に大きくすることで事実上の切断を表現できる

- ✓ Tips：Demonに近いコードにして該当リンクに対してリンクコストを最大化させる処理を行う