IE

Visualization & Pagerank Analysis in Metropolitan Subway Network

20201036 Mulkyeol Kim
20201181 Jihwan Oh

CONTENTS

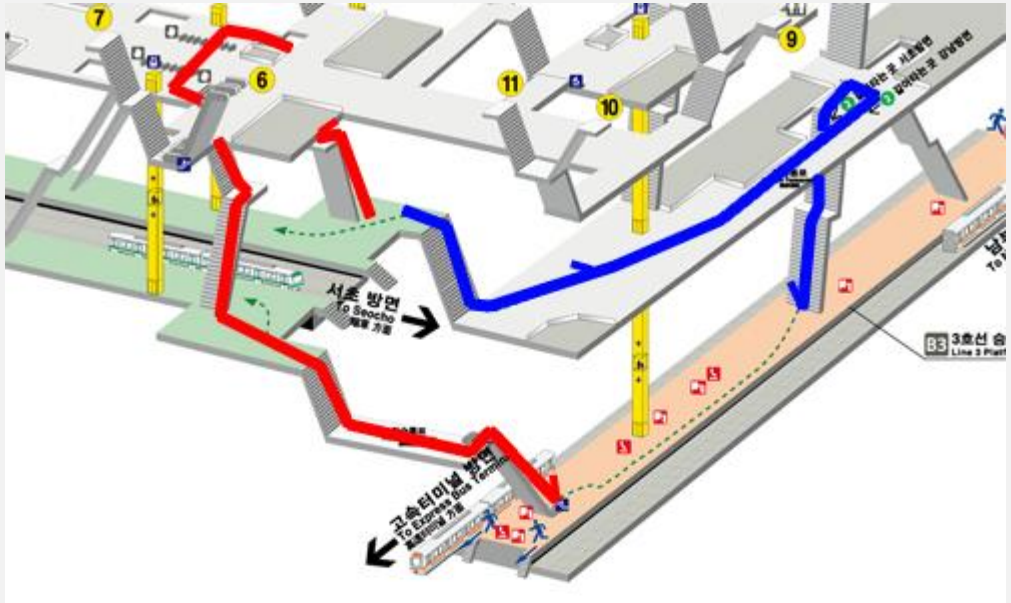
1 Visualization

2 Pagerank Calculation

3 Weighted Pagerank Calculation

4 Conclusion & Application

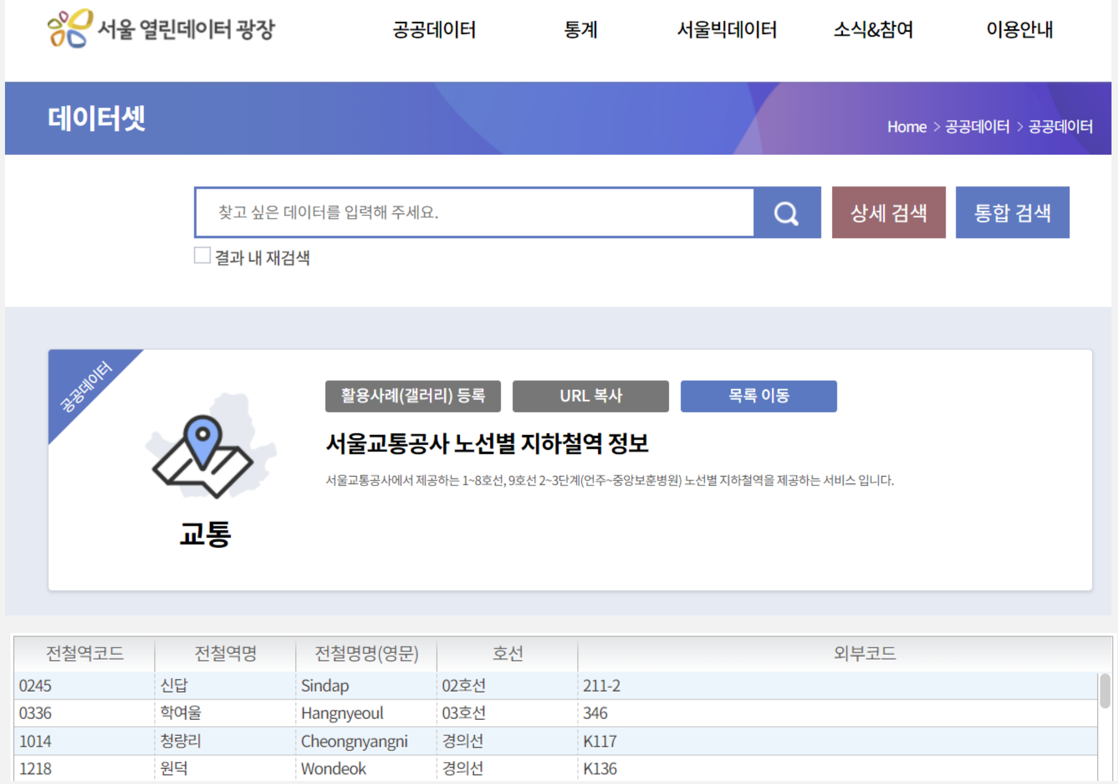
0 Motivation



How about **finding stations** with complicated transfer or a lot of people moving
and **take related measures** such as increasing the number of subways or remodeling the station?

=> Pagerank !!

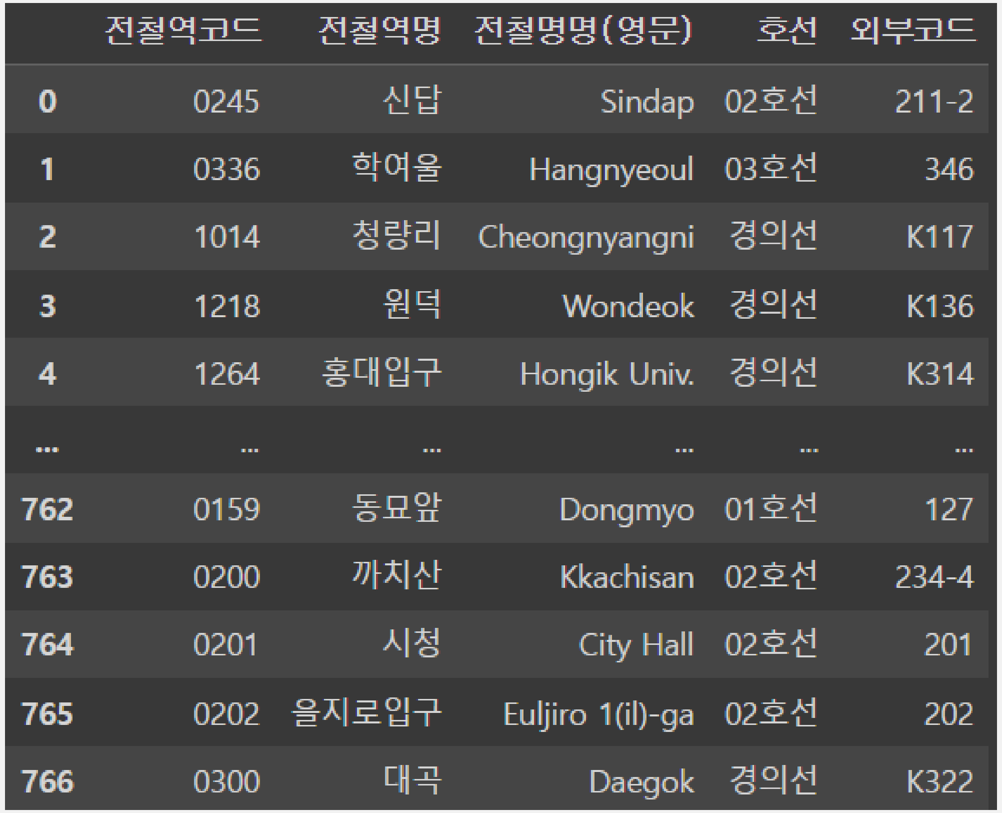
1 Data



서울교통공사 노선별 지하철역 정보

서울교통공사에서 제공하는 1~8호선, 9호선 2~3단계(언주~중앙보훈병원) 노선별 지하철역을 제공하는 서비스 입니다.

전철역코드	전철역명	전철명명(영문)	호선	외부코드
0245	신답	Sindap	02호선	211-2
0336	학여울	Hangnyeoul	03호선	346
1014	청량리	Cheongnyangni	경의선	K117
1218	원덕	Wondeok	경의선	K136

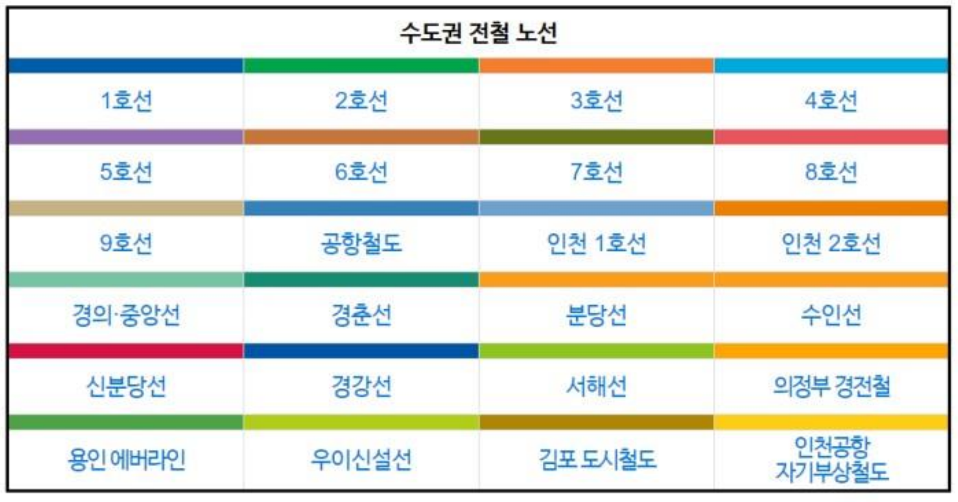


	Station code	Station name (Korean)	Station name (English)	Station line	External station code
	전철역코드	전철역명	전철명명(영문)	호선	외부코드
0	0245	신답	Sindap	02호선	211-2
1	0336	학여울	Hangnyeoul	03호선	346
2	1014	청량리	Cheongnyangni	경의선	K117
3	1218	원덕	Wondeok	경의선	K136
4	1264	홍대입구	Hongik Univ.	경의선	K314
...	...	...	...	...	...
762	0159	동묘앞	Dongmyo	01호선	127
763	0200	까치산	Kkachisan	02호선	234-4
764	0201	시청	City Hall	02호선	201
765	0202	을지로입구	Euljiro 1(il)-ga	02호선	202
766	0300	대곡	Daegok	경의선	K322

Total 767 subway stations include duplication.

2 Preprocessing

1 Only use Line 1 ~ 9

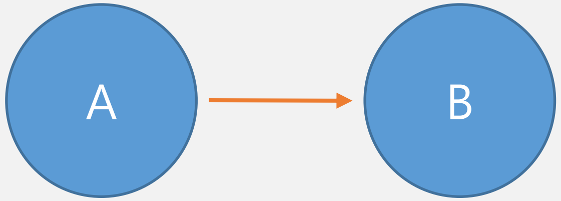


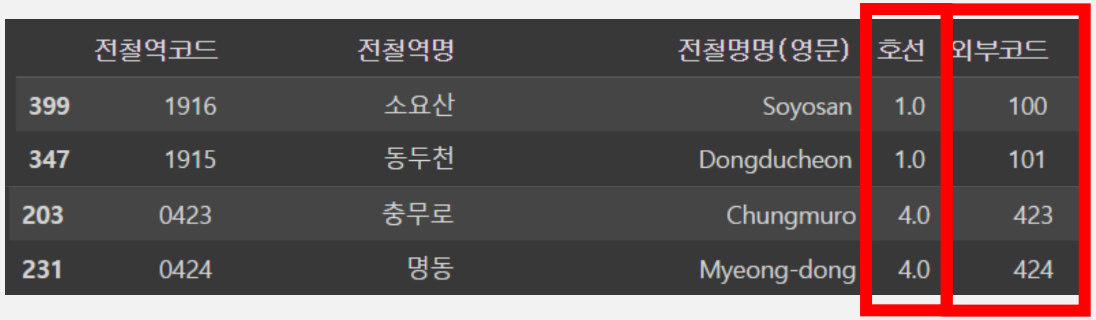
수도권 전철 노선

1호선	2호선	3호선	4호선
5호선	6호선	7호선	8호선
9호선	공항철도	인천 1호선	인천 2호선
경의·중앙선	경춘선	분당선	수인선
신분당선	경강선	서해선	의정부 경전철
용인 에버라인	우이신설선	김포 도시철도	인천공항 자기부상철도

2 Preprocessing

2 Information of **pair** between **previous station** and **the next station** is required

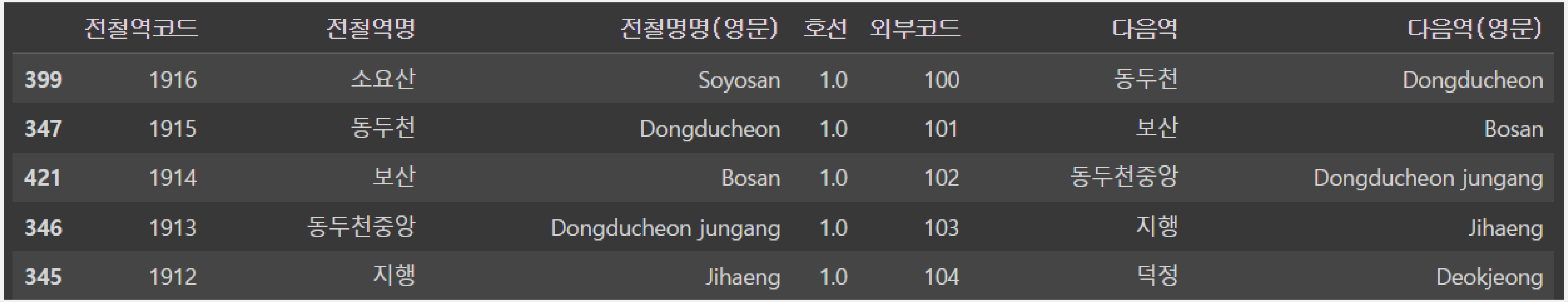




	전철역코드	전철역명	전철명명(영문)	호선	외부코드
399	1916	소요산	Soyosan	1.0	100
347	1915	동두천	Dongducheon	1.0	101
203	0423	충무로	Chungmuro	4.0	423
231	0424	명동	Myeong-dong	4.0	424

2 Preprocessing

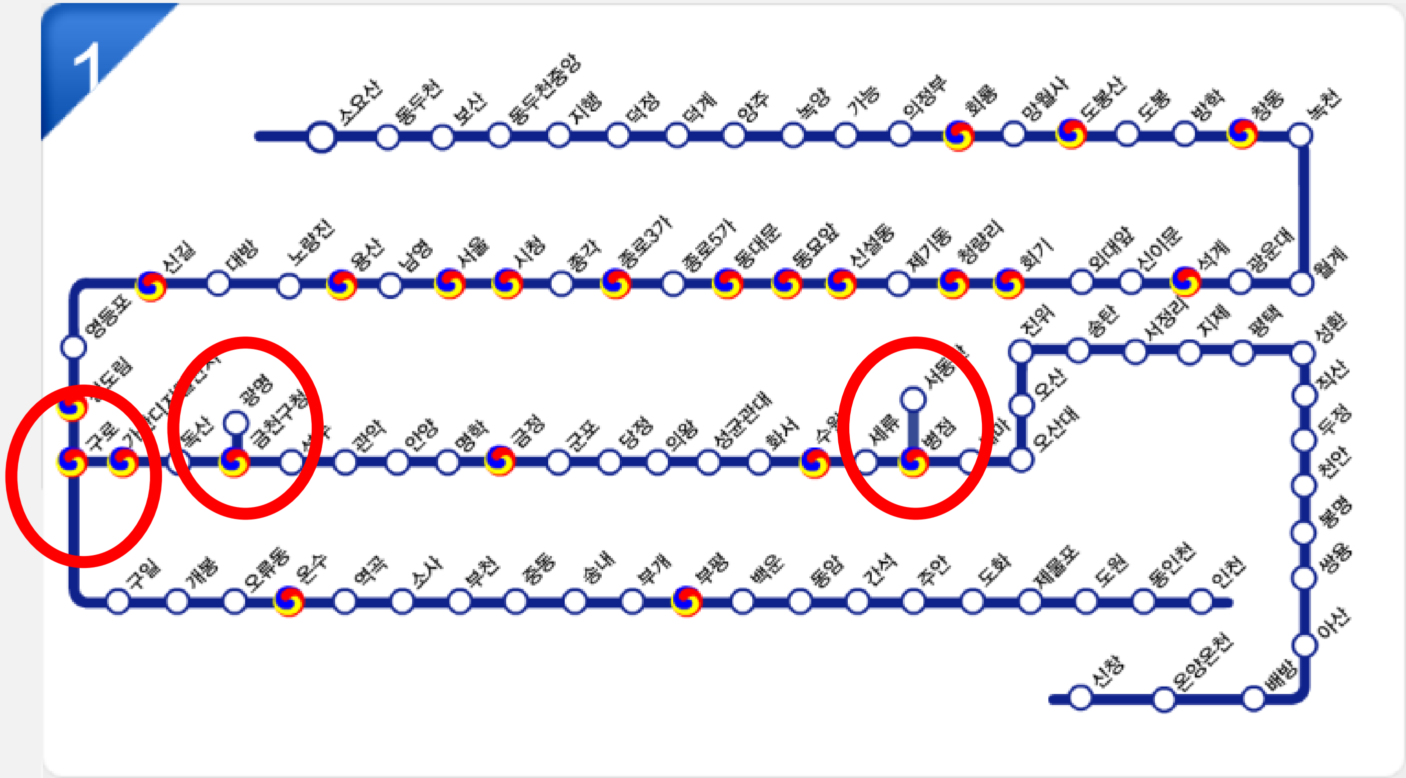
2 Information of **pair** between **previous station** and **the next station** is required



	전철역코드	전철역명	전철명명(영문)	호선	외부코드	다음역	다음역(영문)
399	1916	소요산	Soyosan	1.0	100	동두천	Dongducheon
347	1915	동두천	Dongducheon	1.0	101	보산	Bosan
421	1914	보산	Bosan	1.0	102	동두천중앙	Dongducheon jungang
346	1913	동두천중앙	Dongducheon jungang	1.0	103	지행	Jihaeng
345	1912	지행	Jihaeng	1.0	104	덕정	Deokjeong

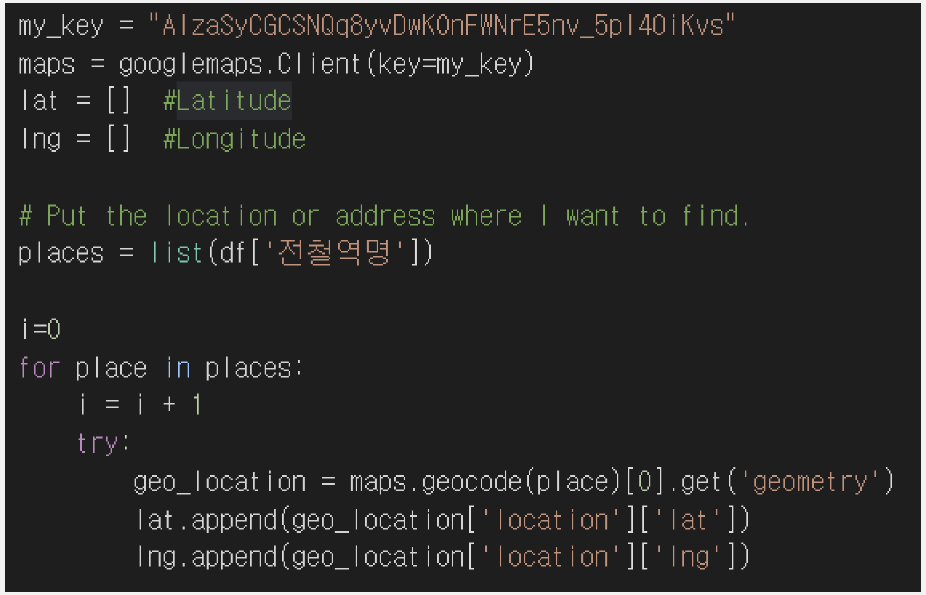
2 Preprocessing

3 **Branches** coming out from a single line



2 Preprocessing

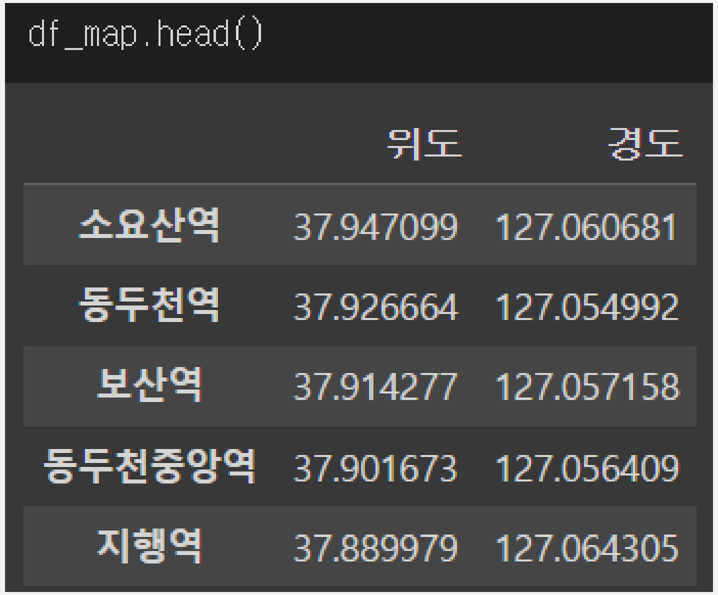
4 Specify **latitude** and **longitude** using GoogleMap



```
my_key = "AIzaSyCGCSNQq8yvDwKOnFWNrE5nv_5pl40iKvs"
maps = googlemaps.Client(key=my_key)
lat = []  #Latitude
lng = []  #Longitude

# Put the location or address where I want to find.
places = list(df['전철역명'])

i=0
for place in places:
    i = i + 1
    try:
        geo_location = maps.geocode(place)[0].get('geometry')
        lat.append(geo_location['location']['lat'])
        lng.append(geo_location['location']['lng'])
```

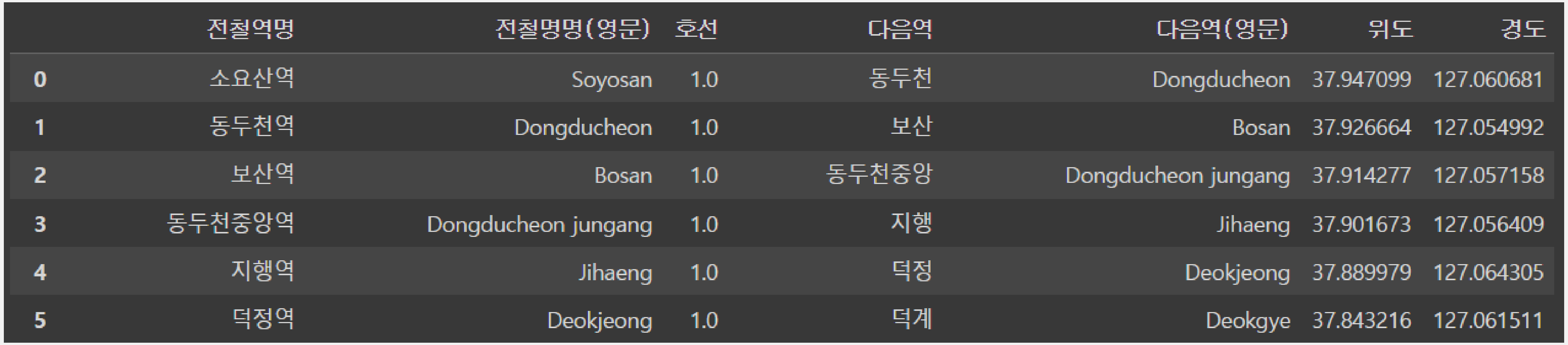


```
df_map.head()
```

	위도	경도
소요산역	37.947099	127.060681
동두천역	37.926664	127.054992
보산역	37.914277	127.057158
동두천중앙역	37.901673	127.056409
지행역	37.889979	127.064305

2 Preprocessing

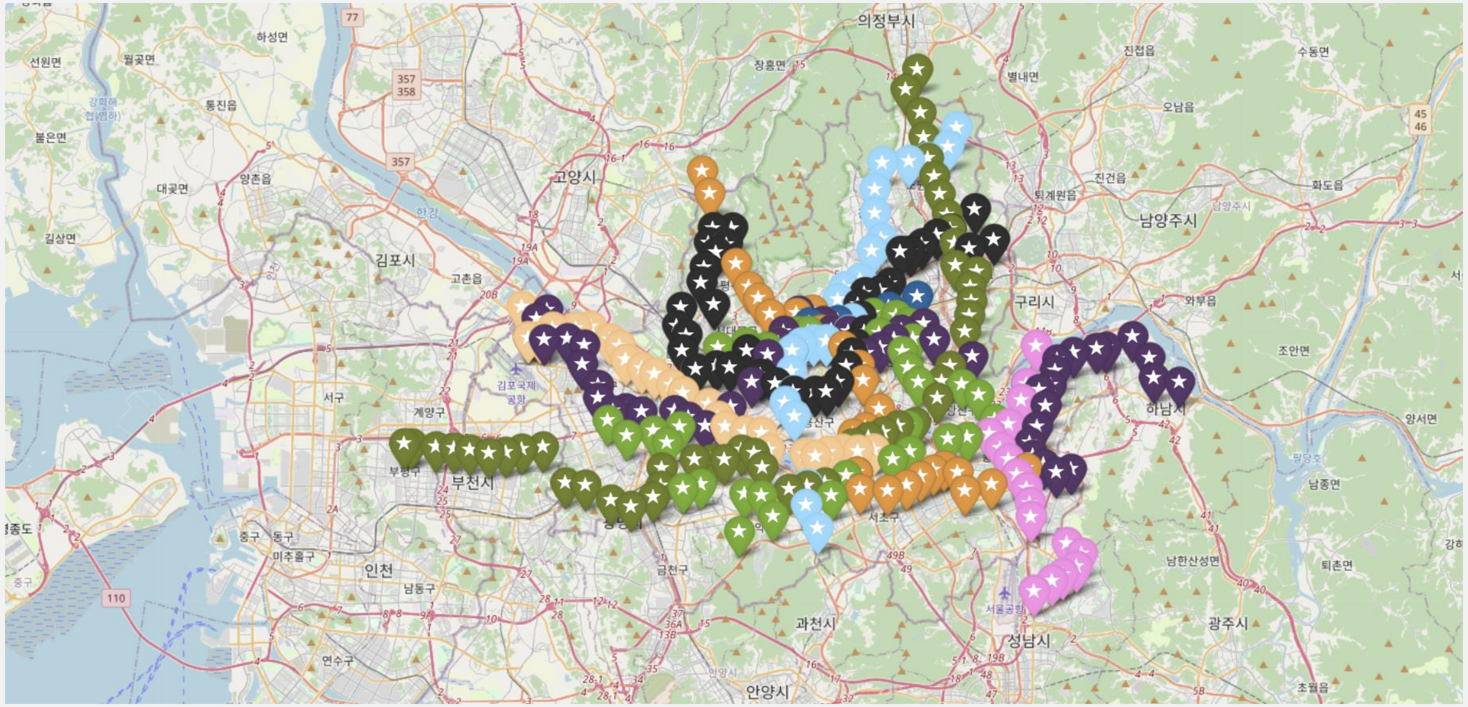
4 Specify **latitude** and **longitude** using GoogleMap



	전철역명	전철명명(영문)	호선	다음역	다음역(영문)	위도	경도
0	소요산역	Soyosan	1.0	동두천	Dongducheon	37.947099	127.060681
1	동두천역	Dongducheon	1.0	보산	Bosan	37.926664	127.054992
2	보산역	Bosan	1.0	동두천중앙	Dongducheon jungang	37.914277	127.057158
3	동두천중앙역	Dongducheon jungang	1.0	지행	Jihaeng	37.901673	127.056409
4	지행역	Jihaeng	1.0	덕정	Deokjeong	37.889979	127.064305
5	덕정역	Deokjeong	1.0	덕계	Deokgye	37.843216	127.061511

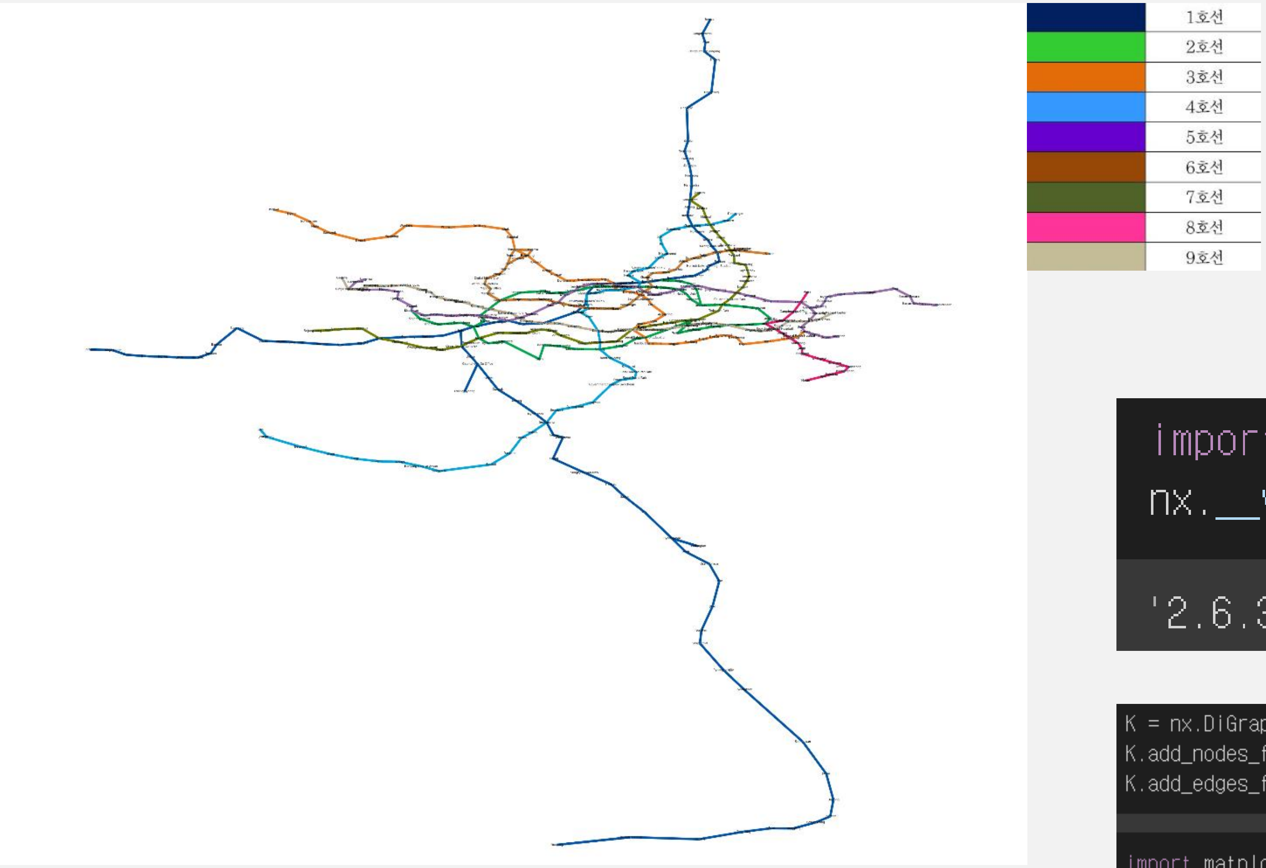
2 Preprocessing

4 Specify **latitude** and **longitude** using GoogleMap



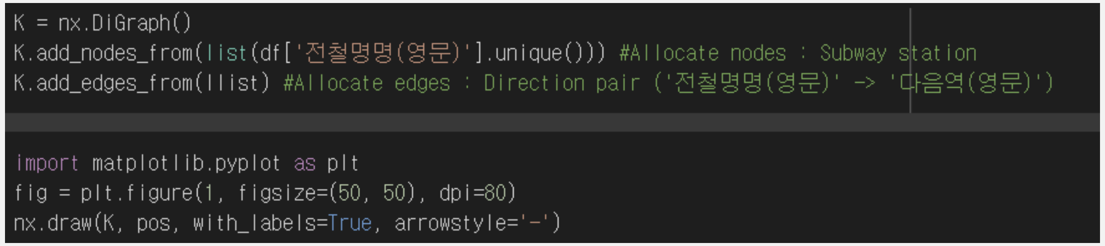
(package ‘folium’)

3 Visualization



```
import networkx as nx
nx.__version__
```

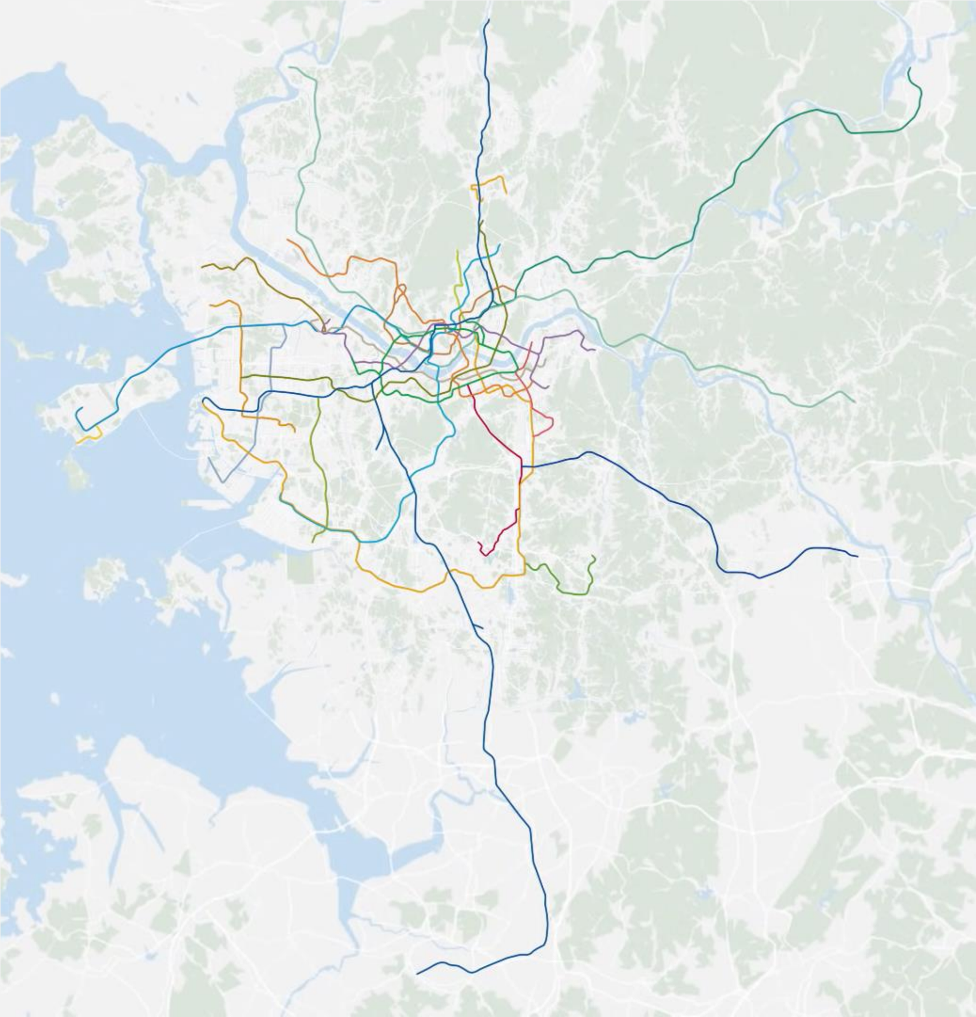
```
'2.6.3'
```



```
K = nx.DiGraph()
K.add_nodes_from(list(df['전철명명(영문)'].unique())) #Allocate nodes : Subway station
K.add_edges_from(llist) #Allocate edges : Direction pair ('전철명명(영문)' -> '다음역(영문)')
```

```
import matplotlib.pyplot as plt
fig = plt.figure(1, figsize=(50, 50), dpi=80)
nx.draw(K, pos, with_labels=True, arrowstyle='-')
```

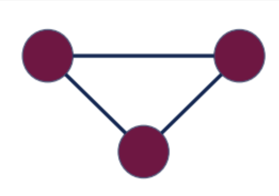
3 Visualization



4 Graph Analysis

Global Clustering Coefficient

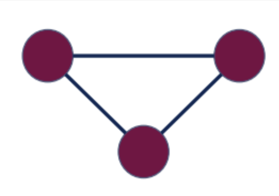
$$\mathrm{C} = \frac{3 \times \text{number of triangels}}{\text{number of all triplets}}$$

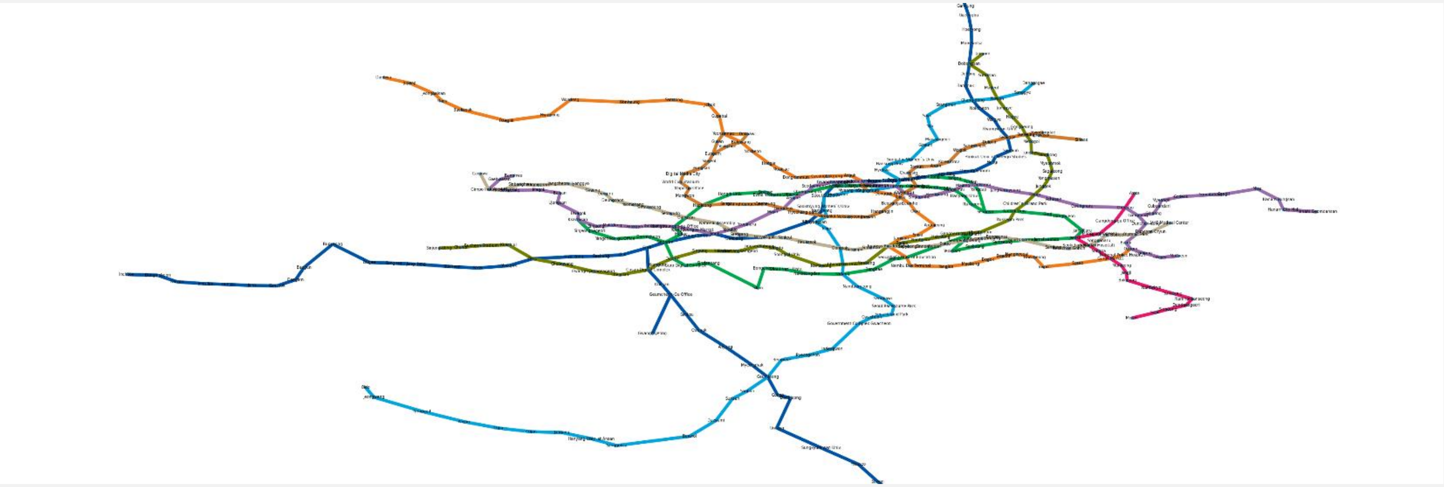


4 Graph Analysis

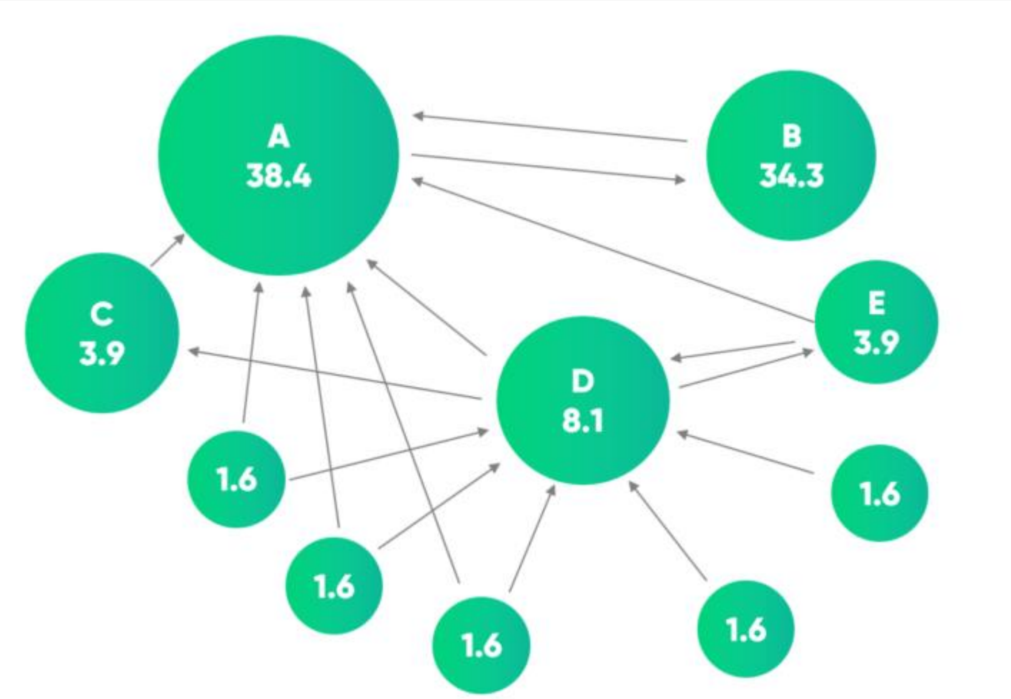
Global Clustering Coefficient

$$\mathrm{C} = \frac{3 \times \cancelto{0}{\text{number of triangels}}}{\text{number of all triplets}} = \mathbf{0}$$





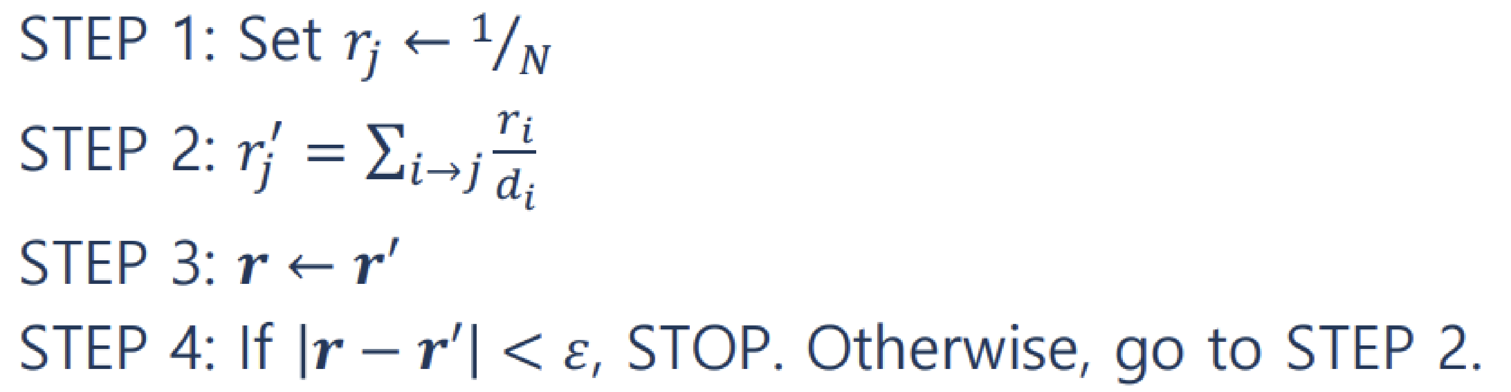
5 Pagerank Calculation



Rating the importance of web pages objectively
Mechanically using the link structure of the web

5 Pagerank Calculation

Power Iteration

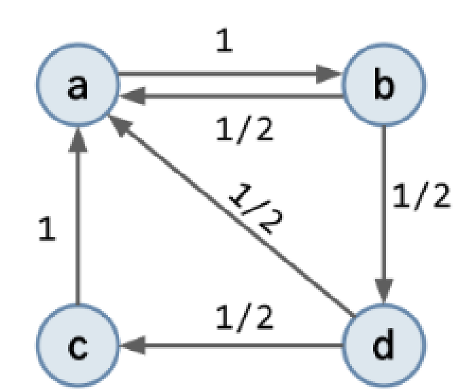


STEP 1: Set $r_j \leftarrow {}^1/_N$

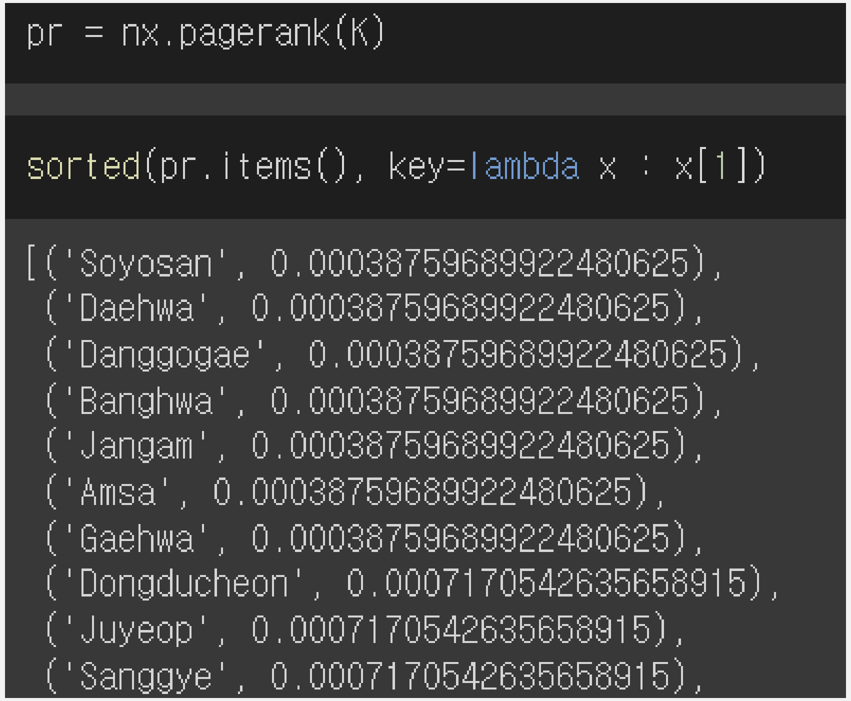
STEP 2: $r_j' = \sum_{i \to j} \frac{r_i}{d_i}$

STEP 3: $\boldsymbol{r} \leftarrow \boldsymbol{r}'$

STEP 4: If $|\boldsymbol{r} - \boldsymbol{r}'| < \varepsilon$, STOP. Otherwise, go to STEP 2.



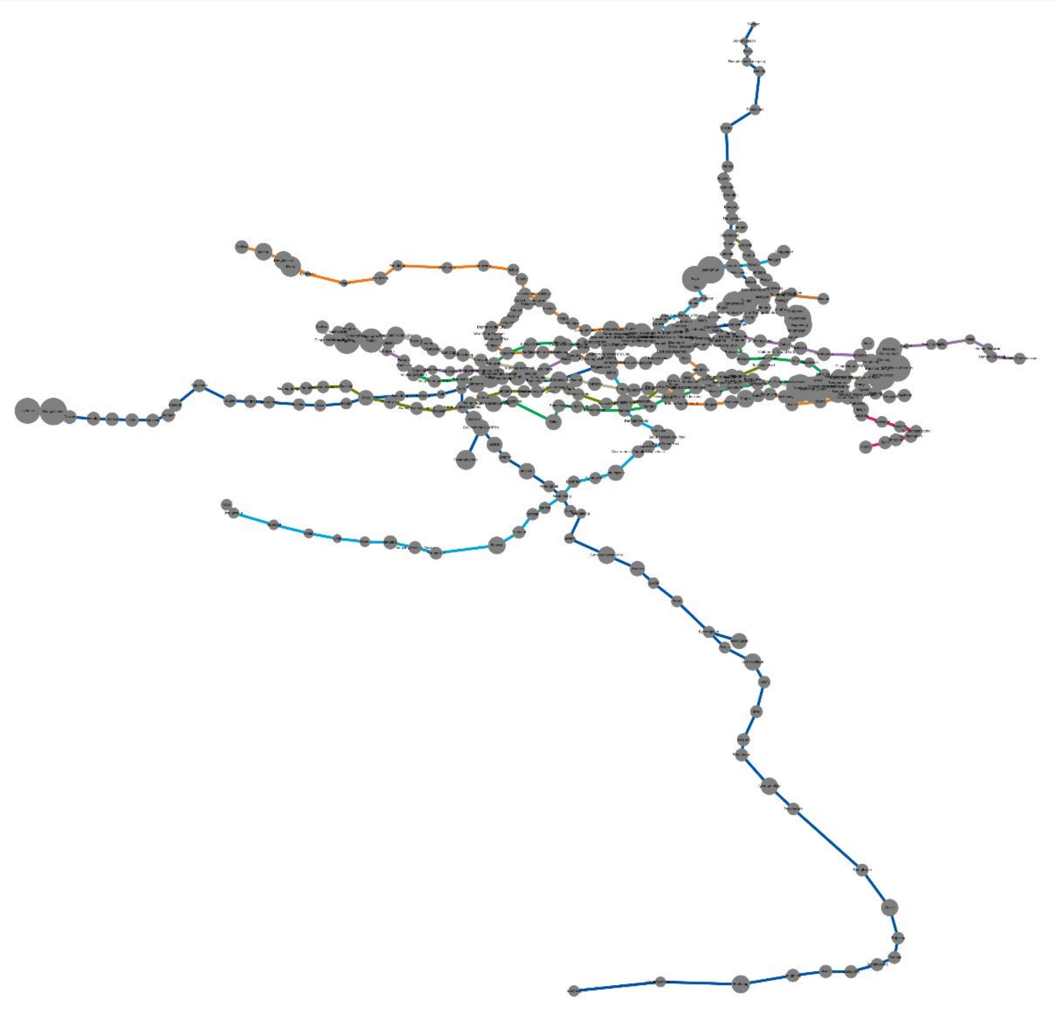
5 Pagerank Calculation



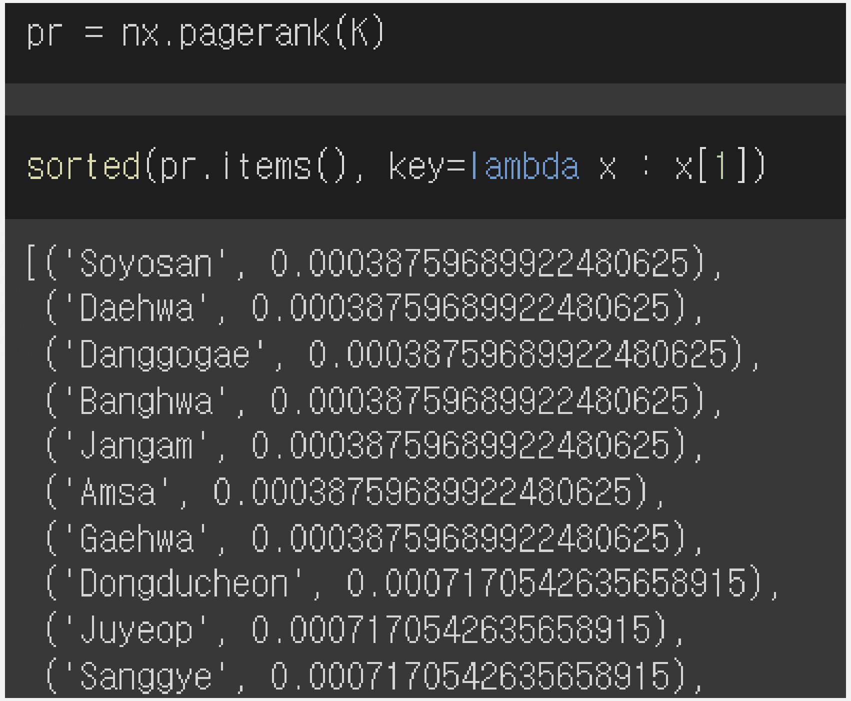
```
pr = nx.pagerank(K)
```

```
sorted(pr.items(), key=lambda x : x[1])
```

```
[('Soyosan', 0.00038759689922480625),
 ('Daehwa', 0.00038759689922480625),
 ('Danggogae', 0.00038759689922480625),
 ('Banghwa', 0.00038759689922480625),
 ('Jangam', 0.00038759689922480625),
 ('Amsa', 0.00038759689922480625),
 ('Gaehwa', 0.00038759689922480625),
 ('Dongducheon', 0.0007170542635658915),
 ('Juyeop', 0.0007170542635658915),
 ('Sanggye', 0.0007170542635658915),
```



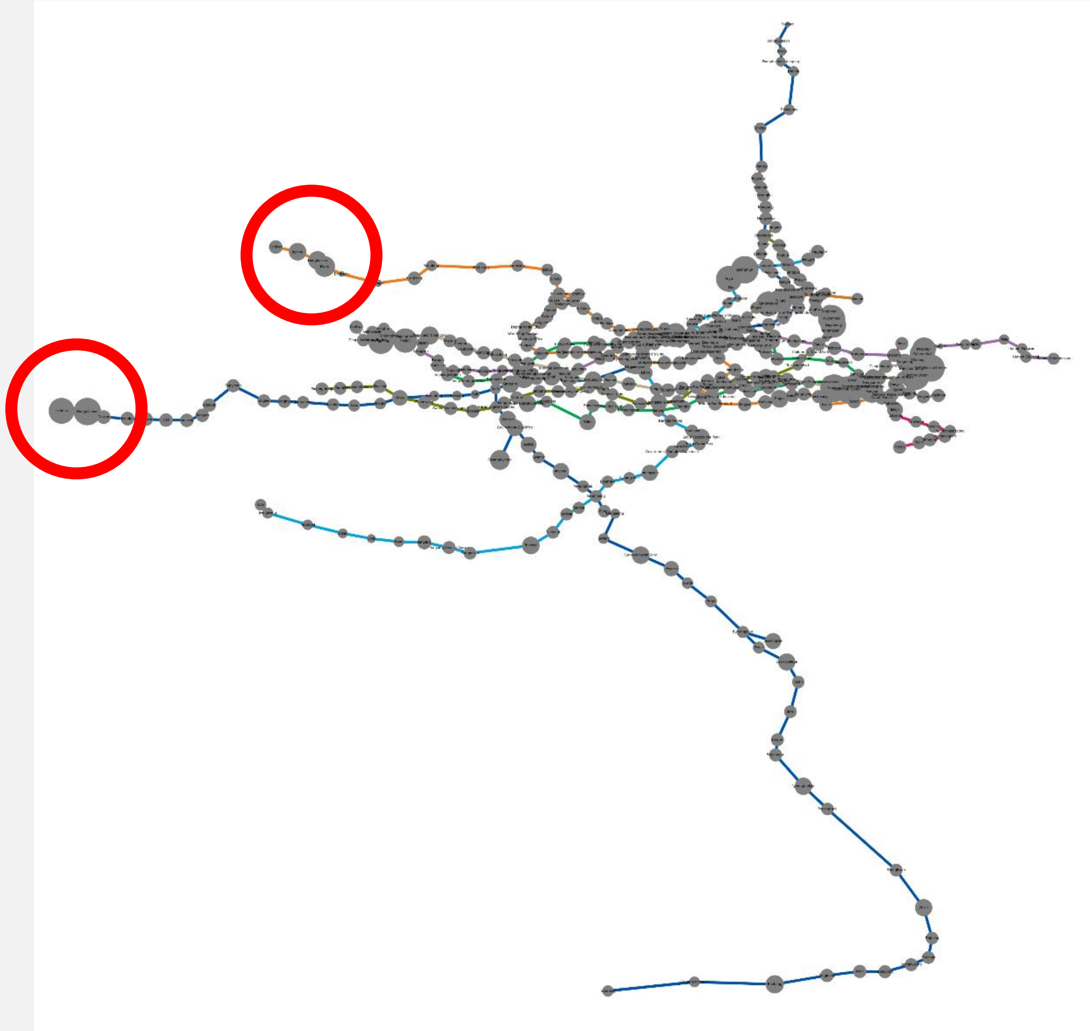
5 Pagerank Calculation



```
pr = nx.pagerank(K)
```

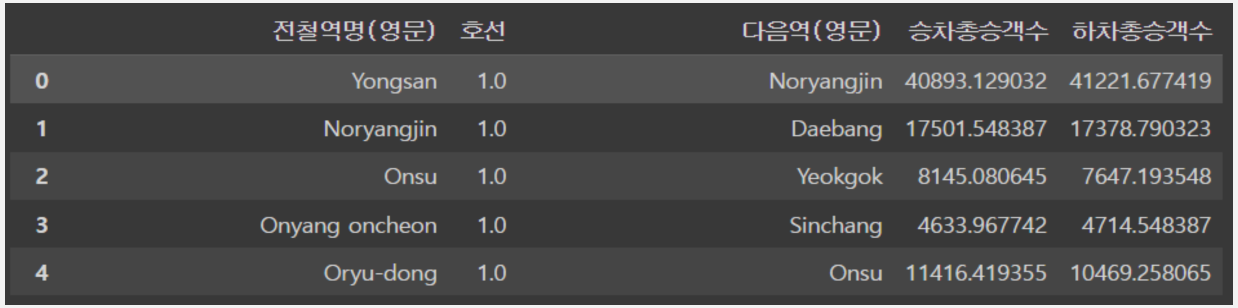
```
sorted(pr.items(), key=lambda x : x[1])
```

```
[('Soyosan', 0.00038759689922480625),
 ('Daehwa', 0.00038759689922480625),
 ('Danggogae', 0.00038759689922480625),
 ('Banghwa', 0.00038759689922480625),
 ('Jangam', 0.00038759689922480625),
 ('Amsa', 0.00038759689922480625),
 ('Gaehwa', 0.00038759689922480625),
 ('Dongducheon', 0.0007170542635658915),
 ('Juyeop', 0.0007170542635658915),
 ('Sanggye', 0.0007170542635658915),
```

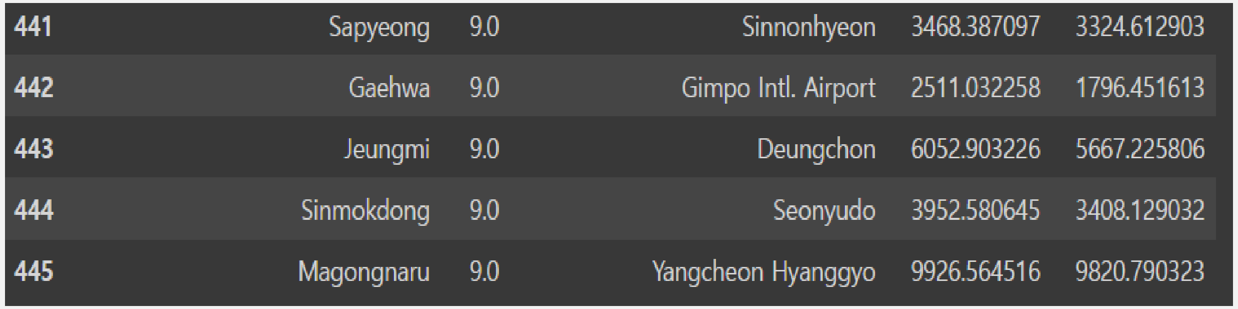


6 New Data

Station(stop) name Station line Next stop name # of getting on # of getting off



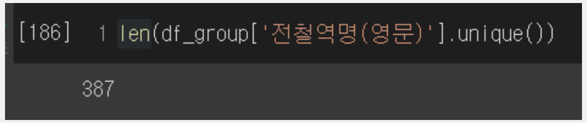
	전철역명(영문)	호선	다음역(영문)	승차총승객수	하차총승객수
0	Yongsan	1.0	Noryangjin	40893.129032	41221.677419
1	Noryangjin	1.0	Daebang	17501.548387	17378.790323
2	Onsu	1.0	Yeokgok	8145.080645	7647.193548
3	Onyang oncheon	1.0	Sinchang	4633.967742	4714.548387
4	Oryu-dong	1.0	Onsu	11416.419355	10469.258065



441	Sapyeong	9.0	Sinnonhyeon	3468.387097	3324.612903
442	Gaehwa	9.0	Gimpo Intl. Airport	2511.032258	1796.451613
443	Jeungmi	9.0	Deungchon	6052.903226	5667.225806
444	Sinmokdong	9.0	Seonyudo	3952.580645	3408.129032
445	Magongnaru	9.0	Yangcheon Hyanggyo	9926.564516	9820.790323

Total 446 subway station with next station data.

There are 387 unique station.



```
[186]   len(df_group['전철역명(영문)'].unique())

        387
```

It means there are 446-387 = 59 numbers of the transfer station.

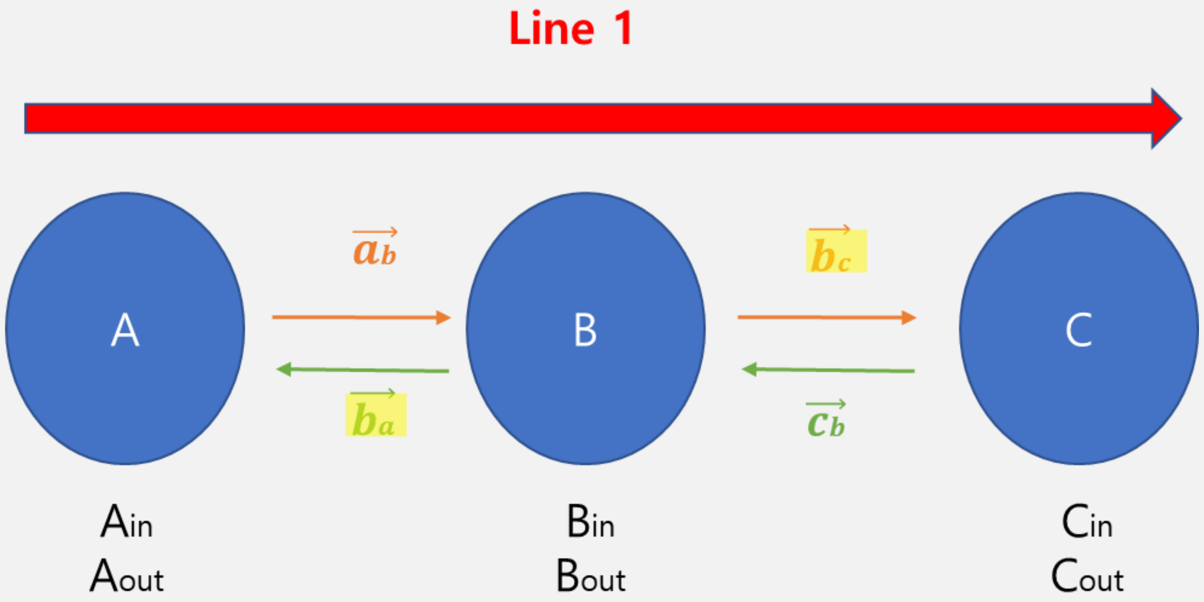
7 Weighted directed graph (1)

We want to find **weighted** directed graph in subway.
But, we don't have data about passengers' movement.

We have just number of people getting on/off data at each station.

We should **estimate** the passengers' movement at each station .

Case 1: Normal station



A_{in} : # of passengers getting on at station A
A_{out} : # of passengers getting out at station A

Passengers' movement at station B

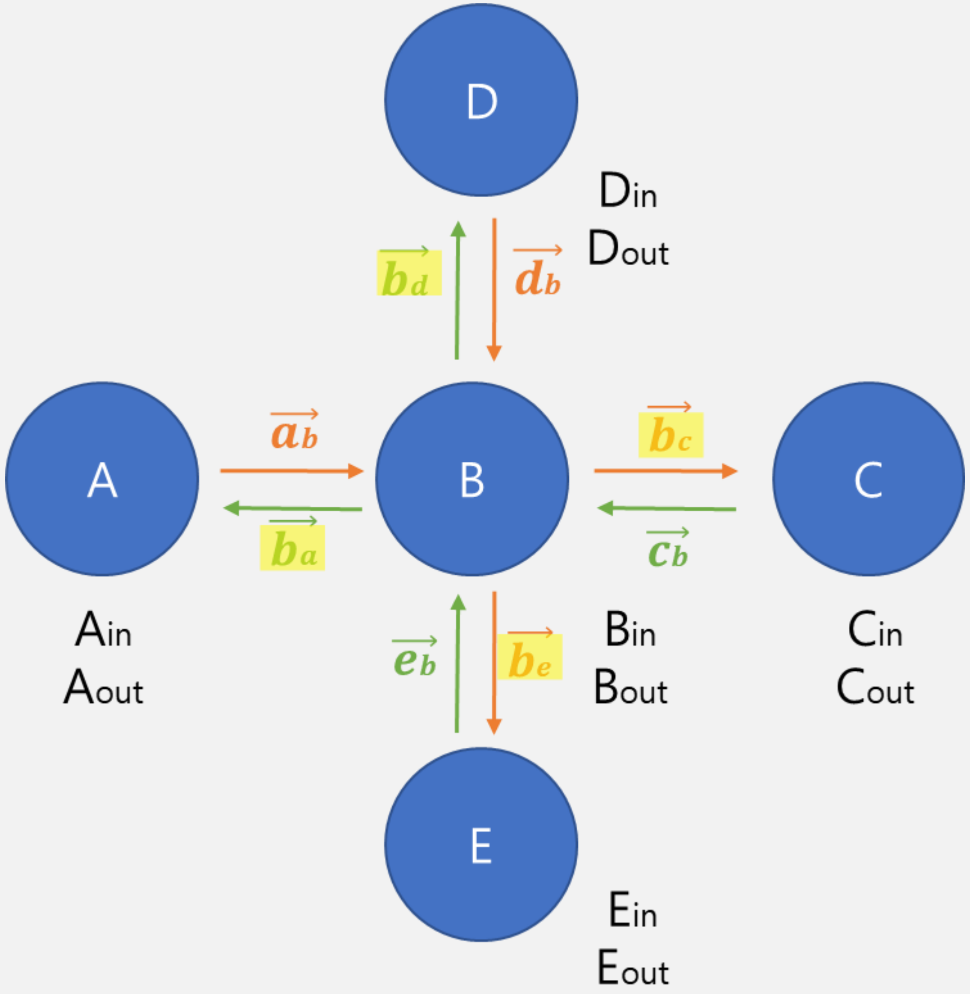
$$|\overrightarrow{b_c}| = B_{in} \times \frac{C_{out}}{A_{out} + C_{out}}$$

$$|\overrightarrow{b_a}| = B_{in} \times \frac{A_{out}}{A_{out} + C_{out}}$$

$$|\overrightarrow{b_{out}}| = |\overrightarrow{b_c}| + |\overrightarrow{b_a}| = B_{in}$$

7 Weighted directed graph (2)

Case 2 : Transfer station



A_{in} : # of passengers getting on at station A
A_{out} : # of passengers getting out at station A

Passengers' movement at station B

$$\left|\vec{b_c}\right| = B_{in} \times \frac{C_{out}}{A_{out} + C_{out} + D_{out} + E_{out}}$$

$$\left|\vec{b_e}\right| = B_{in} \times \frac{E_{out}}{A_{out} + C_{out} + D_{out} + E_{out}}$$

$$\left|\vec{b_a}\right| = B_{in} \times \frac{A_{out}}{A_{out} + C_{out} + D_{out} + E_{out}}$$

$$\left|\vec{b_d}\right| = B_{in} \times \frac{D_{out}}{A_{out} + C_{out} + D_{out} + E_{out}}$$

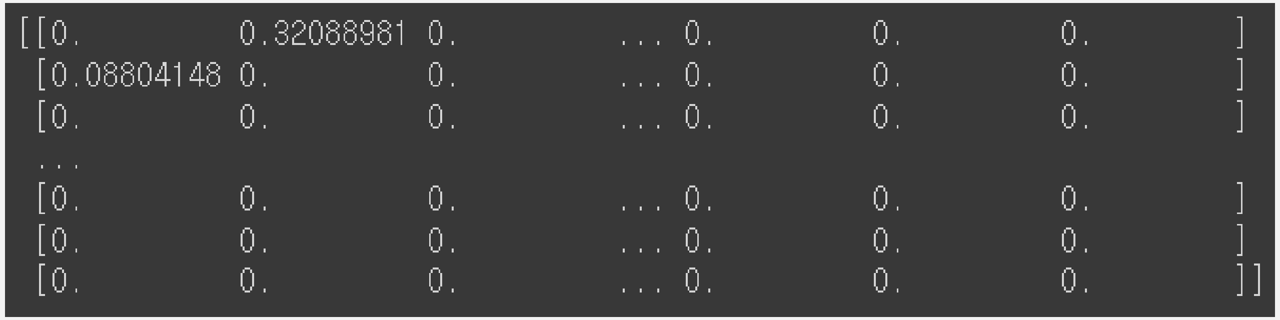
$$\left|\overrightarrow{b_{out}}\right| = \left|\vec{b_c}\right| + \left|\vec{b_e}\right| + \left|\vec{b_a}\right| + \left|\vec{b_d}\right| = B_{in}$$

8 Adjacency Matrix of weight

Now, we can calculate all weights in each node (station).

Then we can represent weights into adjacency matrix.

$$W =$$



```
[[0.         0.32088981 0.         ... 0.         0.         0.        ]
 [0.08804148 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.        ]]
```

$$\in R^{387X387}$$

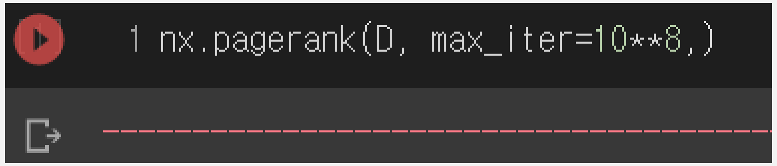
$$\# \, of \; unique \; station = 387$$

9 Weighted pagerank

There are two method for derive pagerank.

1. Power iteration

   But we have many edges and nodes in this task.



```
nx.pagerank(D, max_iter=10**8,)
```

```
------------------------------------------------------
```

It does not converge!

2. Alternative: **Find pagerank using Eigenvector and eigenvalues of W**

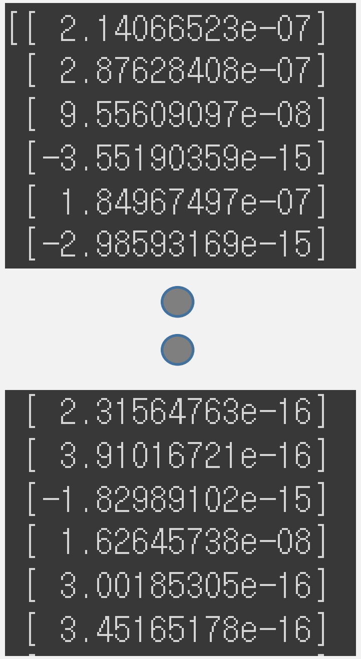
   We have weight matrix W.
   We can derive the pagerank of each node using eigenvector and eigenvalue of W
   The **pagerank** is **eigenvector that maximum eigenvalue of W.**

Reference: https://dl.acm.org/doi/abs/10.1145/775152.775190

9 Weighted pagerank

$$\vec{pr} = \vec{v} \in \{ W\vec{v} = max(\lambda)\vec{v} \}$$ (λ : $eigenvalue, \vec{v}$: $eigenvector\ of\ W$)

$\vec{pr} =$



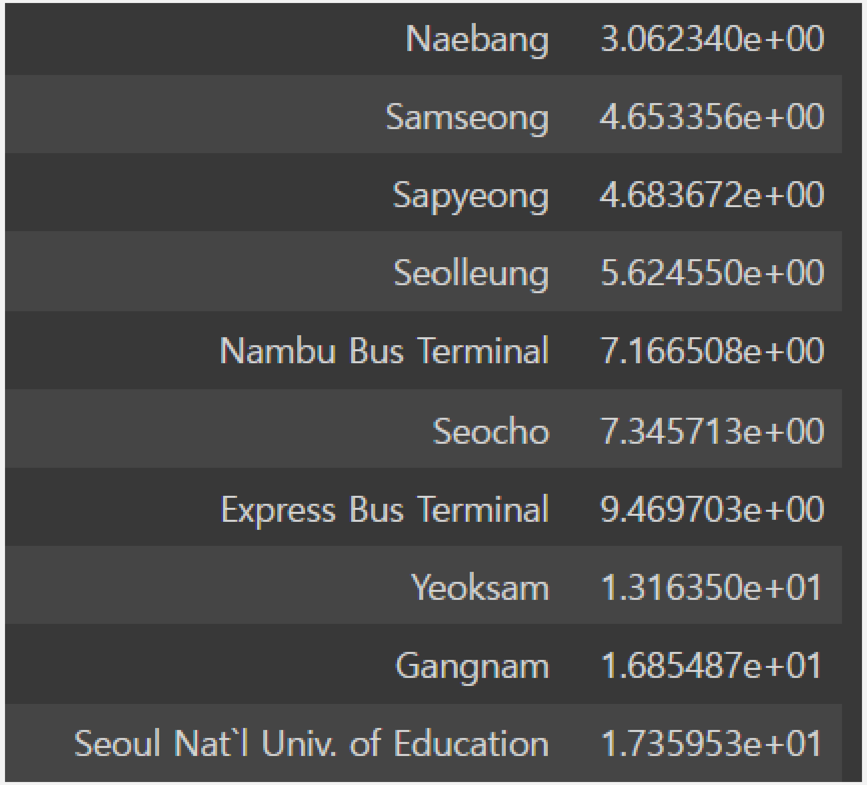
```
[[ 2.14066523e-07]
 [ 2.87628408e-07]
 [ 9.55609097e-08]
 [-3.55190359e-15]
 [ 1.84967497e-07]
 [-2.98593169e-15]
```

```
 [ 2.31564763e-16]
 [ 3.91016721e-16]
 [-1.82989102e-15]
 [ 1.62645738e-08]
 [ 3.00185305e-16]
 [ 3.45165178e-16]
```

$\in R^{387X1}$

Pagerank

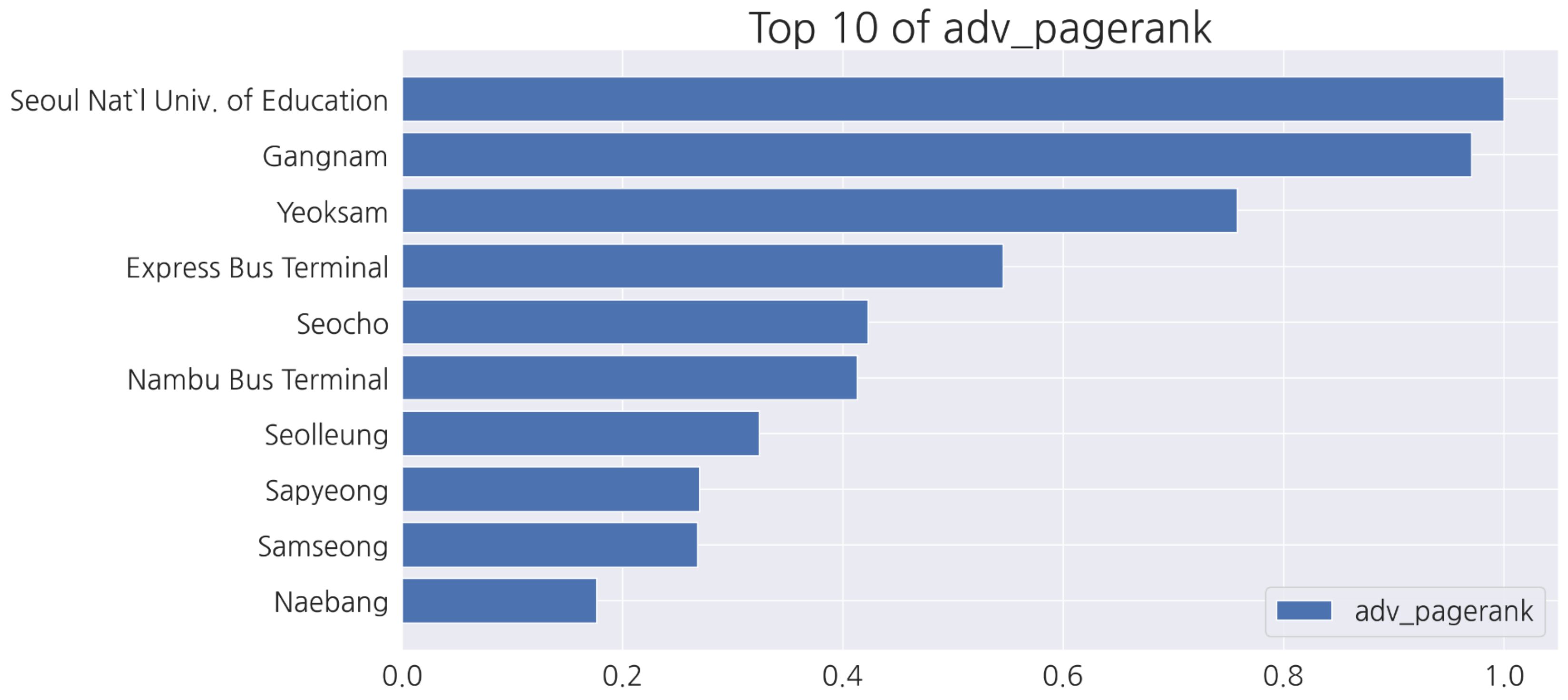
Indexing and sorting →



Naebang	3.062340e+00
Samseong	4.653356e+00
Sapyeong	4.683672e+00
Seolleung	5.624550e+00
Nambu Bus Terminal	7.166508e+00
Seocho	7.345713e+00
Express Bus Terminal	9.469703e+00
Yeoksam	1.316350e+01
Gangnam	1.685487e+01
Seoul Nat`l Univ. of Education	1.735953e+01

Top 10 station

10 Conclusion

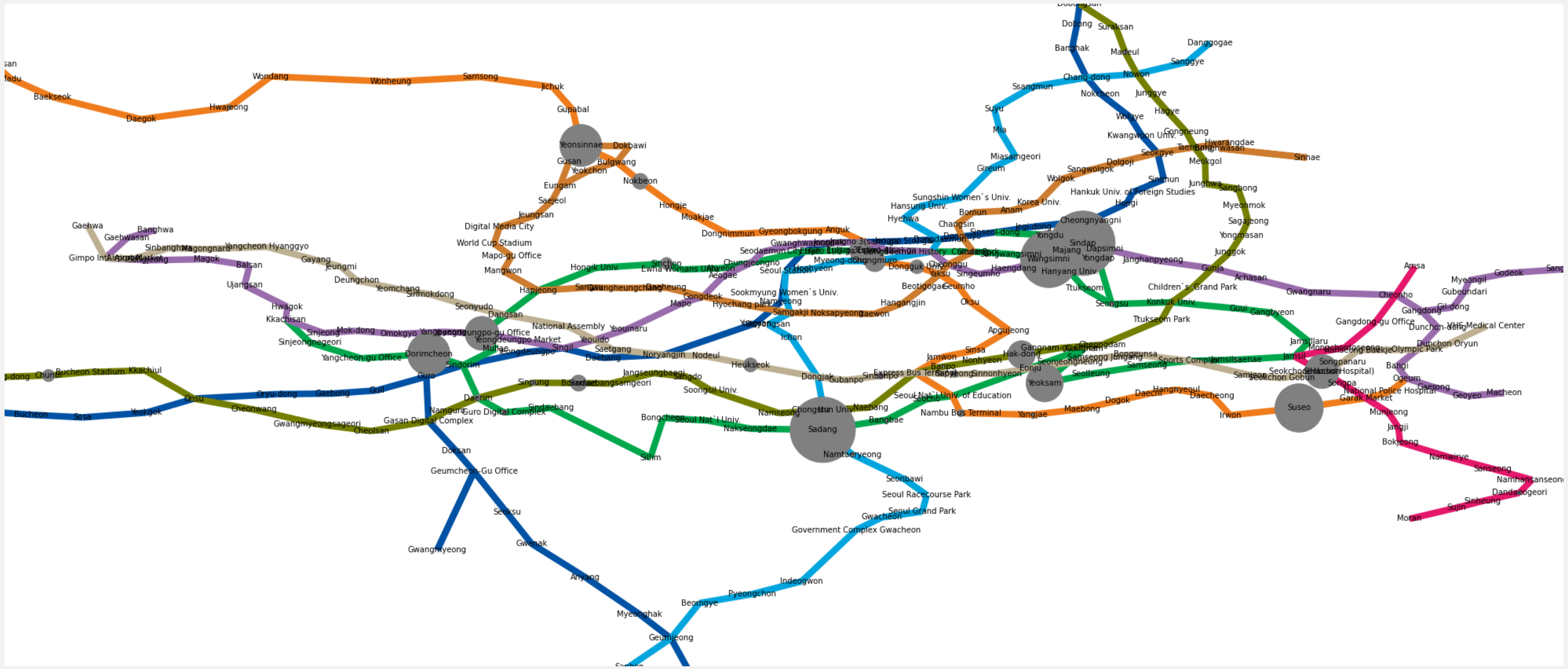


adv_pagerank = weighted pagerank

It suits our common sense!

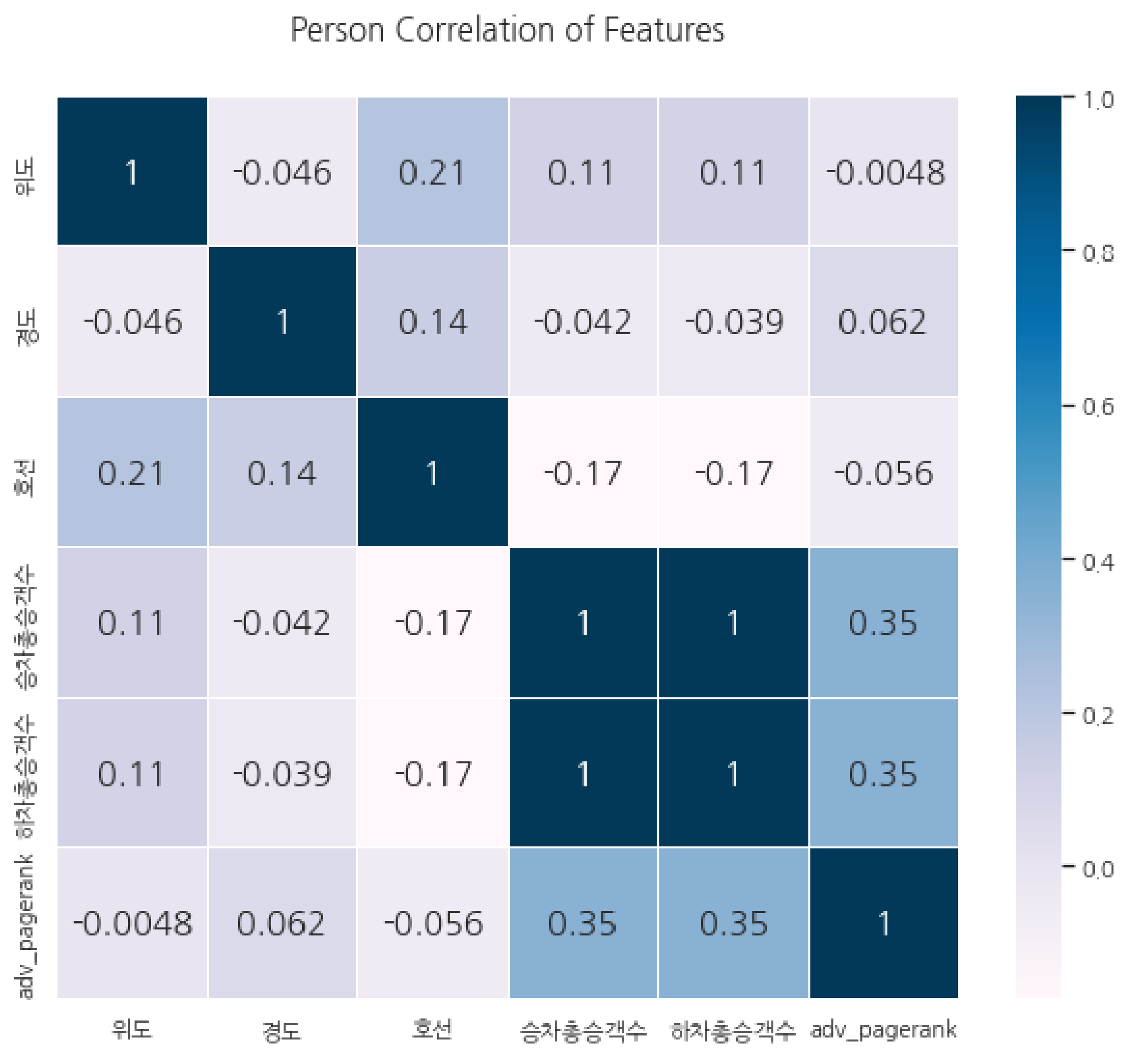
10 Conclusion

Usually in line2



Mostly located in the center of Seoul

11 Correlation Matrix



Person Correlation of Features

	위도	경도	호선	승차총승객수	하차총승객수	adv_pagerank
위도	1	-0.046	0.21	0.11	0.11	-0.0048
경도	-0.046	1	0.14	-0.042	-0.039	0.062
호선	0.21	0.14	1	-0.17	-0.17	-0.056
승차총승객수	0.11	-0.042	-0.17	1	1	0.35
하차총승객수	0.11	-0.039	-0.17	1	1	0.35
adv_pagerank	-0.0048	0.062	-0.056	0.35	0.35	1

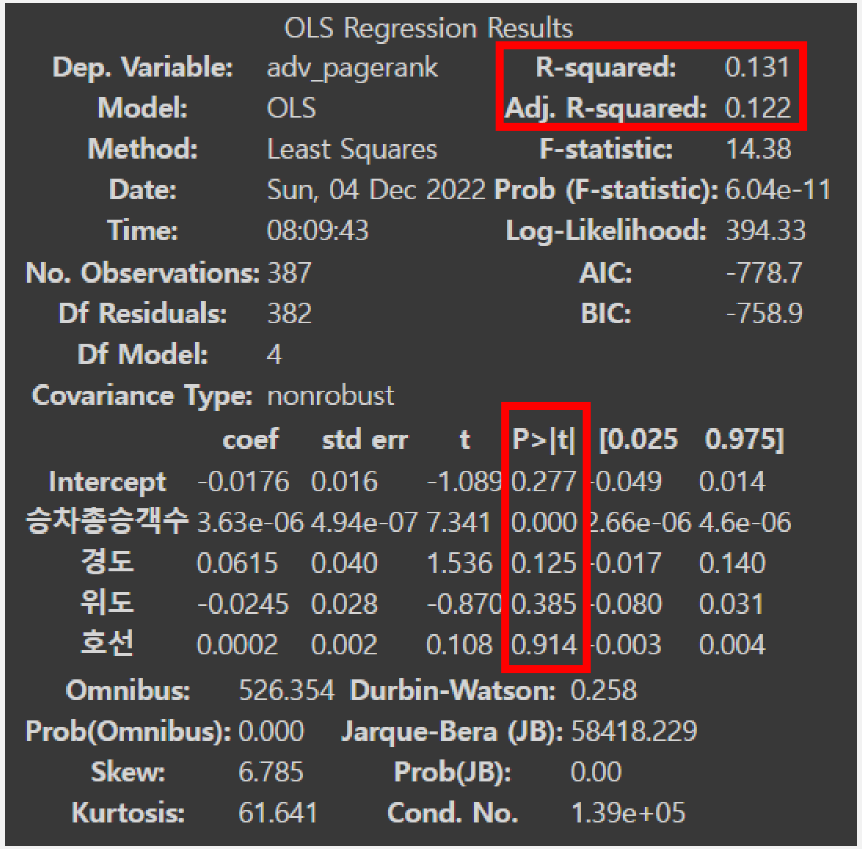
Feature: Longitude,
Latitude
Line
Total # of passengers getting on
Total # of passengers getting off
Weighted pagerank

There are very strong correlation between Total # of passengers getting on and off

There are slightly strong correlation between Total # of passengers getting on/off and weighted pagerank.

12 Linear Regression (OLS)

$$model : \boldsymbol{pagerank} = \alpha(Longitude) + \beta(Latitude) + \gamma(Line) + \delta(\#\_of\_passengers) + \varepsilon$$



```
                       OLS Regression Results
 Dep. Variable:    adv_pagerank          R-squared:           0.131
     Model:        OLS               Adj. R-squared:          0.122
    Method:        Least Squares          F-statistic:        14.38
     Date:         Sun, 04 Dec 2022 Prob (F-statistic): 6.04e-11
     Time:         08:09:43           Log-Likelihood:    394.33
No. Observations: 387                          AIC:           -778.7
  Df Residuals:    382                          BIC:           -758.9
   Df Model:       4
Covariance Type:  nonrobust
                coef     std err     t      P>|t|  [0.025   0.975]
  Intercept   -0.0176   0.016    -1.089  0.277  -0.049   0.014
승차총승객수  3.63e-06  4.94e-07  7.341  0.000  2.66e-06 4.6e-06
    경도       0.0615   0.040     1.536  0.125  -0.017   0.140
    위도      -0.0245   0.028    -0.870  0.385  -0.080   0.031
    호선       0.0002   0.002     0.108  0.914  -0.003   0.004
   Omnibus:      526.354  Durbin-Watson:    0.258
Prob(Omnibus): 0.000    Jarque-Bera (JB): 58418.229
    Skew:        6.785        Prob(JB):      0.00
  Kurtosis:      61.641      Cond. No.      1.39e+05
```

R^2 is close to 0.

> Our regression model is bad.

Only **p-value** of # of passengers is less than 0,05

It means that **# of passengers** is only **significant.**

13 Application

14 Reference

1. 서울교통공사 노선별 지하철역 정보. 서울 열린데이터 광장. 2022.12.05. https://data.seoul.go.kr/dataList/OA-15442/S/1/datasetView.do
2. 서울교통공사 지하철 환승역 환승인원 정보. 서울 열린데이터 광장. 2022.10.14. http://115.84.165.39/dataList/OA-12033/S/1/datasetView.do
3. 지도로 보는 서울 수도권 전철 1974-2021. MetroLiner. 2021. 6. 29.
4. Extrapolation methods for accelerating PageRank computations. Sepandar D. Kamvar. 2003.5.20. Extrapolation methods for accelerating PageRank computations | Proceedings of the 12th international conference on World Wide Web (acm.org)

IE

Thank you