```
import pandas as pd

df = pd.read_csv('/content/route_detail.csv')

df.head()
```

	Unnamed: 0	route_id	stop_id	stop_name
0	0	1	1	THIRUVOTRIYUR
1	1	1	2	THIRUVOTRIYUR TEMPLE
2	2	1	3	THANGAL
3	3	1	4	ANNA NAGAR
4	4	1	5	ROYAPURAM P.S

```
# Data cleaning
```

Data cleaning

```
df = df.iloc[:,1:]
df.head()
```

	route_id	stop_id	stop_name
0	1	1	THIRUVOTRIYUR
1	1	2	THIRUVOTRIYUR TEMPLE
2	1	3	THANGAL
3	1	4	ANNA NAGAR
4	1	5	ROYAPURAM P.S

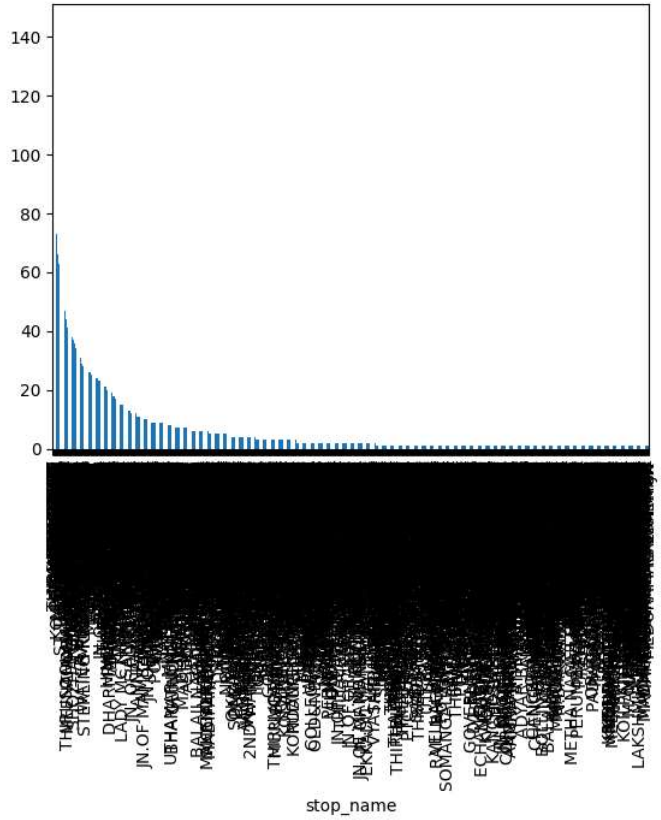
```
stat = df.groupby(by=["stop_name"])["route_id"].count().sort_values(ascending=False)
display(stat)
```

stop_name	route_id
BROADWAY	144
M.G.R.CENTRAL	117
CONCORDE	110
SAIDAPET	102
TAMBARAM	86
...	...
MAHARANI	1
MAHENDRA CITY MAIN GATE	1
MAHINDRA IND.PARK	1
MALANTHUR	1
MEDURAMALAICHERYJN	1

1413 rows × 1 columns

dtype: int64

```
import matplotlib.pyplot as plt
stat.plot(kind = 'bar')
plt.show()
```

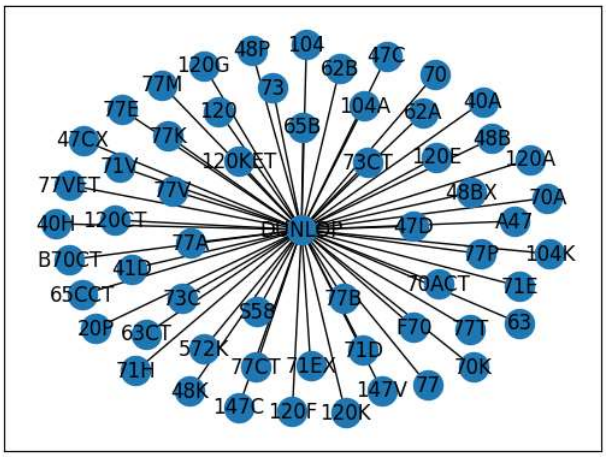


Visualize data

after reseaching found network graph will best suit the usecase

```
import matplotlib.pyplot as plt
import networkx as nx
```

```
sample = df[df["stop_name"]=="DUNLOP"]
G = nx.from_pandas_edgelist(sample, "route_id", "stop_name")
nx.draw_networkx(G)
plt.show()
```



Modifing data to be represented as graph

```
df.head()
df["next_stop"] = df.groupby(by=["route_id"])["stop_name"].shift(-1)
df.loc[:50]
```

	route_id	stop_id	stop_name	next_stop
0	1	1	THIRUVOTRIYUR	THIRUVOTRIYUR TEMPLE
1	1	2	THIRUVOTRIYUR TEMPLE	THANGAL
2	1	3	THANGAL	ANNA NAGAR
3	1	4	ANNA NAGAR	ROYAPURAM P.S
4	1	5	ROYAPURAM P.S	CLIVE BATTERY
5	1	6	CLIVE BATTERY	PARRYS
6	1	7	PARRYS	M.G.R.CENTRAL
7	1	8	M.G.R.CENTRAL	P.OR & SONS
8	1	9	P.OR & SONS	WESLEY H.S
9	1	10	WESLEY H.S	Y.M.I.A
10	1	11	Y.M.I.A	MANDAVELI
11	1	12	MANDAVELI	A.M.S.HOSPITAL
12	1	13	A.M.S.HOSPITAL	ADYAR O.T.
13	1	14	ADYAR O.T.	ADYAR DEPOT
14	1	15	ADYAR DEPOT	THIRUVANMIYUR
15	1	16	THIRUVANMIYUR	NaN
16	101	1	THIRUVOTRIYUR	THRUVOTRIYUR TEMPLE
17	101	2	THRUVOTRIYUR TEMPLE	THANGAL
18	101	3	THANGAL	ANNA NAGAR
19	101	4	ANNA NAGAR	ROYAPURAM P.S
20	101	5	ROYAPURAM P.S	CLIVE BATTERY
21	101	6	CLIVE BATTERY	PARRYS
22	101	7	PARRYS	M.G.R.CENTRAL
23	101	8	M.G.R.CENTRAL	DASAPRAKASH
24	101	9	DASAPRAKASH	TAYLORS ROAD
25	101	10	TAYLORS ROAD	AMINJIKARAI
26	101	11	AMINJIKARAI	NADUVANKARAI
27	101	12	NADUVANKARAI	ARUMBAKKAM
28	101	13	ARUMBAKKAM	NERKUNDRAM
29	101	14	NERKUNDRAM	MADURAVOYAL
30	101	15	MADURAVOYAL	VAANAGARAM

Note: last stop for every route can be consider as route_id for easy visualization purpose

by updating nan with route id

```
df.loc[df["next_stop"].isna(), "next_stop"] = df.loc[df["next_stop"].isna(), "route_id"]
df.loc[:50]
```

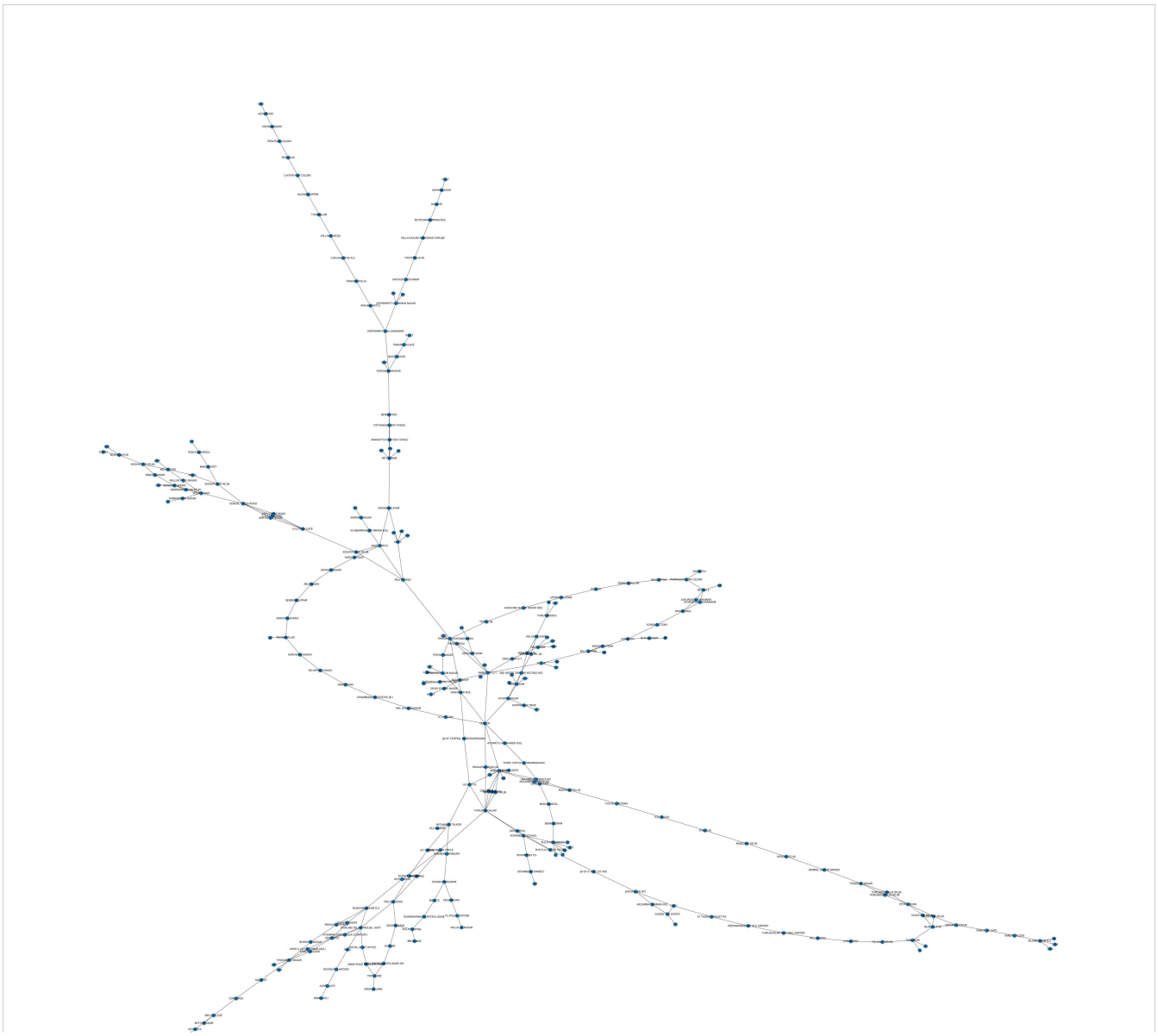
	route_id	stop_id	stop_name	next_stop
0	1	1	THIRUVOTRIYUR	THIRUVOTRIYUR TEMPLE
1	1	2	THIRUVOTRIYUR TEMPLE	THANGAL
2	1	3	THANGAL	ANNA NAGAR
3	1	4	ANNA NAGAR	ROYAPURAM P.S
4	1	5	ROYAPURAM P.S	CLIVE BATTERY
5	1	6	CLIVE BATTERY	PARRYS
6	1	7	PARRYS	M.G.R.CENTRAL
7	1	8	M.G.R.CENTRAL	P.OR & SONS
8	1	9	P.OR & SONS	WESLEY H.S
9	1	10	WESLEY H.S	Y.M.I.A
10	1	11	Y.M.I.A	MANDAVELI
11	1	12	MANDAVELI	A.M.S.HOSPITAL
12	1	13	A.M.S.HOSPITAL	ADYAR O.T.
13	1	14	ADYAR O.T.	ADYAR DEPOT
14	1	15	ADYAR DEPOT	THIRUVANMIYUR
15	1	16	THIRUVANMIYUR	1
16	101	1	THİRUVOTRİYUR	THRUVOTRİYUR TEMPLE
17	101	2	THRUVOTRIYUR TEMPLE	THANGAL
18	101	3	THANGAL	ANNA NAGAR
19	101	4	ANNA NAGAR	ROYAPURAM P.S
20	101	5	ROYAPURAM P.S	CLIVE BATTERY
21	101	6	CLIVE BATTERY	PARRYS
22	101	7	PARRYS	M.G.R.CENTRAL
23	101	8	M.G.R.CENTRAL	DASAPRAKASH
24	101	9	DASAPRAKASH	TAYLORS ROAD
25	101	10	TAYLORS ROAD	AMINJIKARAI
26	101	11	AMINJIKARAI	NADUVANKARAI
27	101	12	NADUVANKARAI	ARUMBAKKAM
28	101	13	ARUMBAKKAM	NERKUNDRAM
29	101	14	NERKUNDRAM	MADURAVOYAL
30	101	15	MADURAVOYAL	VAANAGARAM
32	101	17	VELAPPANCHAVADI	KUMUNANCHAVADI

Sample data creation - Dunlop stop

```
sample_routes = df[df["stop_name"]=="DUNLOP"]["route_id"]
sample = df[df["route_id"].isin(sample_routes)]
sample.head()
```

	route_id	stop_id	stop_name	next_stop
204	104	1	REDHİLLS	AYURVEDHA ASHARAMAM
205	104	2	AYURVEDHA ASHARAMAM	KAVANGARAI
206	104	3	KAVANGARAI	SCREW FACTORY
207	104	4	SCREW FACTORY	SURAPEDU
208	104	5	SURAPEDU	KALLIKUPPAM

```
plt.figure(figsize=(100, 100))
G = nx.from_pandas_edgelist(sample, "stop_name", "next_stop")
nx.draw_networkx(G, pos=nx.spring_layout(G, iterations=1000))
plt.savefig('dunlop_route.png')
```



Except for base stop all other stop should be unique to route. so intersection wont happen between routes

```
# sample["derived_next_stop"] = sample.apply(lambda row: row["next_stop"] if(row["next_stop"]=="DUNLOP") else row["route_id"]+row["next_stop"], axis=1)
# sample = sample.drop("derived_next_stop",axis=1)
sample.loc[~(sample["next_stop"]=="DUNLOP") & ~(sample["next_stop"]==sample["route_id"]),"derived_next_stop"] =  sample["route_id"]+ "-" +sample["next_
sample.loc[(sample["next_stop"]=="DUNLOP") | (sample["next_stop"]==sample["route_id"]),"derived_next_stop"] = sample["next_stop"]
sample.loc[~(sample["stop_name"]=="DUNLOP"),"derived_stop_name"] = sample["route_id"]+ "-" + sample["stop_name"]
sample.loc[sample["stop_name"]=="DUNLOP","derived_stop_name"] = sample["stop_name"]
sample.head()
```

```
<ipython-input-17-5519beb56ca2>:3: SettingWithCopyWarning:
A value is trying to be set on a copy of a slice from a DataFrame.
Try using .loc[row_indexer,col_indexer] = value instead

See the caveats in the documentation: https://pandas.pydata.org/pandas-docs/stable/user_guide/indexing.html#returning-a-view-versus-a-copy
  sample.loc[~(sample["next_stop"]=="DUNLOP") & ~(sample["next_stop"]==sample["route_id"]),"derived_next_stop"] =  sample["route_id"]+ "-" +sample[
<ipython-input-17-5519beb56ca2>:5: SettingWithCopyWarning:
A value is trying to be set on a copy of a slice from a DataFrame.
Try using .loc[row_indexer,col_indexer] = value instead

See the caveats in the documentation: https://pandas.pydata.org/pandas-docs/stable/user_guide/indexing.html#returning-a-view-versus-a-copy
  sample.loc[~(sample["stop_name"]=="DUNLOP"),"derived_stop_name"] = sample["route_id"]+ "-" + sample["stop_name"]
```

	route_id	stop_id	stop_name	next_stop	derived_next_stop	derived_stop_name
204	104	1	REDHILLS	AYURVEDHA ASHARAMAM	104-AYURVEDHA ASHARAMAM	104-REDHILLS
205	104	2	AYURVEDHA ASHARAMAM	KAVANGARAI	104-KAVANGARAI	104-AYURVEDHA ASHARAMAM
206	104	3	KAVANGARAI	SCREW FACTORY	104-SCREW FACTORY	104-KAVANGARAI
207	104	4	SCREW FACTORY	SURAPEDU	104-SURAPEDU	104-SCREW FACTORY
208	104	5	SURAPEDU	KALLIKUPPAM	104-KALLIKUPPAM	104-SURAPEDU

```
plt.figure(figsize=(100, 100))
G = nx.from_pandas_edgelist(sample, "derived_stop_name", "derived_next_stop")
nx.draw_networkx(G, pos=nx.spring_layout(G, iterations=1000))
plt.savefig('routewise_map_from_dunlop.png')
```

Add more detail

- coluring base stop differently
- colouring edges
- trying to find best layout for representing data - looks like only spring_layout is best for representing data
- have to find ways to give more space between node so stop name is visible
- add direction
- add different color for different route

```
G = nx.from_pandas_edgelist(sample, "derived_stop_name", "derived_next_stop")
color =[]
for node in G:
    if node == "DUNLOP":
        # current stop
        color.append((0,1,0)) #green
    elif node in list(sample["route_id"]):
        color.append((1,0,0)) #red
    else:
        color.append((0,0,1)) #blue
```

```
plt.figure(figsize=(100,100))
nx.draw_networkx(G, pos=nx.spring_layout(G, iterations=100),edge_color=(0.8,0.6,0.3), node_color=color, arrows=True, arrowstyle= '-|>',
    arrowsize= 12)
# plt.legend()
plt.savefig('routewise_map_from_dunlop_with_color.png')
```

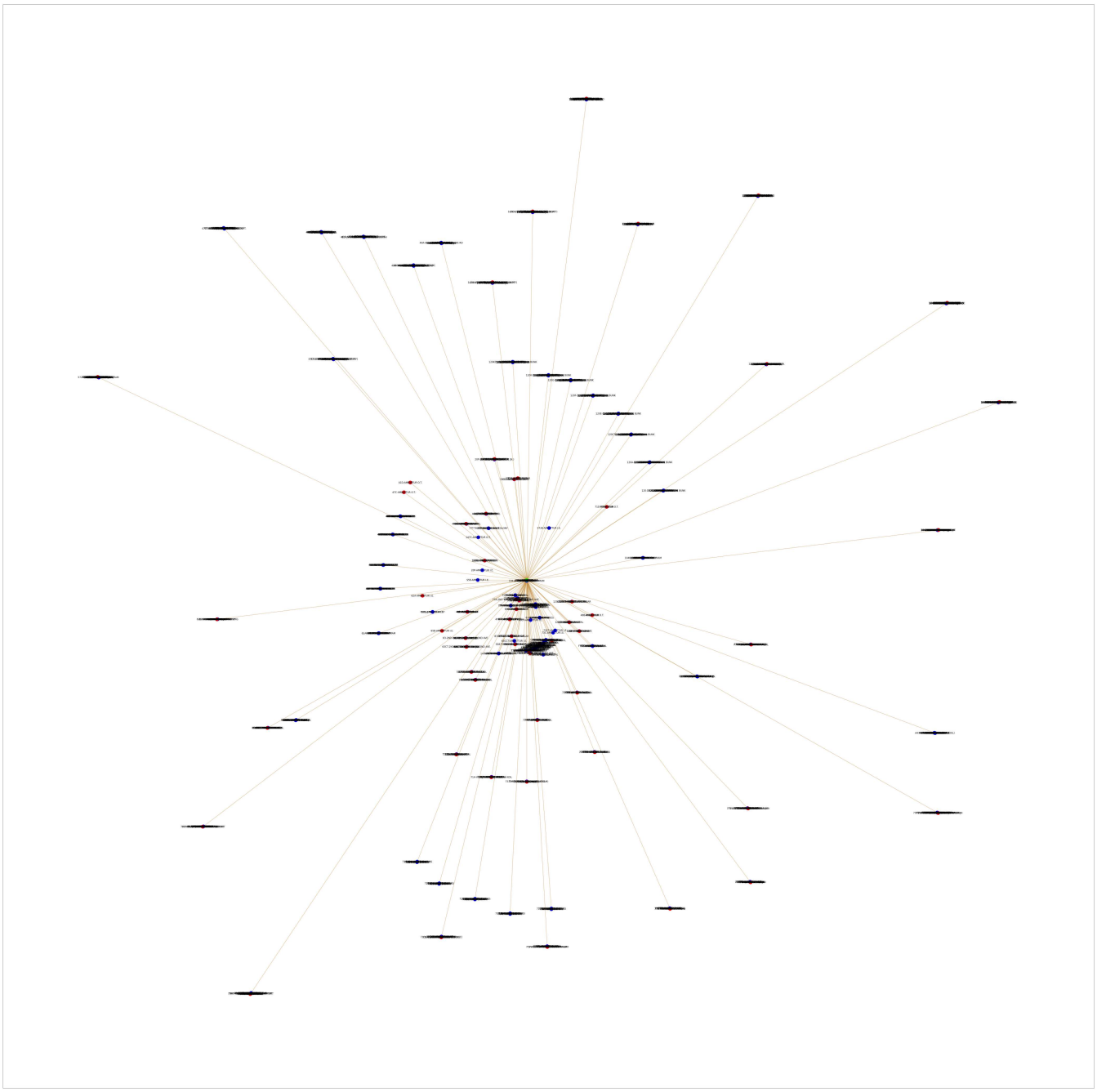
```
plt.figure(figsize=(100,100))
nx.draw_networkx(G, pos=nx.kamada_kawai_layout(G),edge_color=(0.8,0.6,0.3), node_color=color)
plt.savefig('kamada_kawai_layout.png')
```

- set same distance between all nodes

```
distance = {}
for u,v in G.edges():
    if u not in distance.keys():
        distance[u] = {}
    if u == "DUNLOP":
        distance[u][v] = 500
    else:
        distance[u][v] = nx.shortest_path_length(G, source=u, target=v)
print(distance["DUNLOP"])
```

```
{'104-AMBATTUR I.E.': 500, '104A-AMBATTUR O.T.': 500, '104A-AMBATTUR I.E.': 500, '104K-AMBATTUR O.T.': 500, '104K-AMBATTUR I.E.': 500, '120-AMBATTU
```

```
plt.figure(figsize=(100,100))
nx.draw_networkx(G, pos=nx.kamada_kawai_layout(G, dist=distance),edge_color=(0.8,0.6,0.3), node_color=color)
plt.savefig('kamada_kawai_layout.png')
```



Adding Legend

```
from matplotlib.patches import Patch
from matplotlib.lines import Line2D

legend_elements = [
    Line2D([0], [0], marker='o', color='w', label='Current Stop',markerfacecolor='g', markersize=15),
    Line2D([0], [0], marker='o', color='w', label='Bus number',markerfacecolor='r', markersize=15),
    Line2D([0], [0], marker='o', color='w', label='Stop',markerfacecolor='b', markersize=15),
]

# Create the figure
# fig, ax = plt.subplots()
# ax.legend(handles=legend_elements, loc='upper right')

# print(ax)

plt.figure(figsize=(100,100))
nx.draw_networkx(G, pos=nx.spring_layout(G, iterations=100, scale=2),edge_color=(0.8,0.6,0.3), node_color=color, arrows=True, arrowstyle= '-|>',
    arrowsize= 12)
# , ax=ax
# Setting it to how it was looking before.
# plt.axis('off')
plt.legend(handles=legend_elements, loc='upper right')
plt.savefig('routewise_map_from_dunlop_with_legend.png')
```

```
plt.figure(figsize=(100,100))
nx.draw_networkx(G, pos=nx.shell_layout(G),edge_color=(0.8,0.6,0.3), node_color=color)
plt.savefig('shell_layout.png')
```

```
plt.figure(figsize=(100,100))
nx.draw_networkx(G, pos=nx.fruchterman_reingold_layout(G),edge_color=(0.8,0.6,0.3), node_color=color)
plt.savefig('shell_layout.png')
```

recoloring based on bus number

```
!pip install distinctipy
```

```
Requirement already satisfied: distinctipy in /usr/local/lib/python3.10/dist-packages (1.3.4)
Requirement already satisfied: numpy>=1.16.3 in /usr/local/lib/python3.10/dist-packages (from distinctipy) (1.26.4)
```

```
from distinctipy import distinctipy

routes = sample["route_id"].unique()
n=len(list(routes))
# print(list(route))

reserved_color = [(1,0,0),(0,1,0),(0,0,1),(0.8,0.6,0.3)]

# generate N visually distinct colours
colors = distinctipy.get_colors(n,reserved_color)
# print(colors)
route_color_map = { route: color for route, color in zip(routes,colors)}
print(route_color_map)
```

```
{'104': (0.0, 1.0, 1.0), '104A': (0.0, 0.0, 0.0), '104K': (1.0, 0.0, 1.0), '120': (1.0, 1.0, 1.0), '120A': (0.0, 0.5, 0.5), '120CT': (0.5, 0.0, 0.5
```

```
G = nx.from_pandas_edgelist(sample, "derived_stop_name", "derived_next_stop")
color =[]
for node in G:
    if node == "DUNLOP":
        # current stop
        color.append((0,1,0)) #green
    elif node in list(sample["route_id"]):
        color.append((1,0,0)) #red
    else:
        color.append(route_color_map[node.split("-")[0]]) #route wise color
```

```
from matplotlib.lines import Line2D

legend_elements = [
```

```
    Line2D([0], [0], marker='o', color='w', label='Current Stop',markerfacecolor='g', markersize=15),
    Line2D([0], [0], marker='o', color='w', label='Bus number',markerfacecolor='r', markersize=15),
]
route_legend = [
    Line2D([0], [0], marker='o', color='w', label= bus_num,markerfacecolor=route_color_map[bus_num], markersize=15) for bus_num in route_color_map
]
legend_elements = legend_elements + route_legend

# Create the figure
# fig, ax = plt.subplots()
# ax.legend(handles=legend_elements, loc='upper right')

# print(ax)

plt.figure(figsize=(100,100))
nx.draw_networkx(G, pos=nx.spring_layout(G, iterations=30),edge_color=(0.8,0.6,0.3), node_color=color, arrows=True,
    arrowsize= 25)
# , ax=ax q
# Setting it to how it was looking before.
# plt.axis('off')
plt.legend(handles=legend_elements, loc='upper right')
plt.savefig('routewise_map_from_dunlop_with_legend.png')
```

relabeling to avoid overlapping for labels

```
G = nx.from_pandas_edgelist(sample, "derived_stop_name", "derived_next_stop", create_using=nx.DiGraph())
print(nx.is_directed(G))
color =[]
root_node = None
for node in G:
    if node == "DUNLOP":
        # current stop
        root_node = node
        color.append((0,1,0)) #green
    elif node in list(sample["route_id"]):
        color.append((1,0,0)) #red
    else:
        color.append(route_color_map[node.split("-")[0]]) #route wise color
print(root_node)
```

```
True
DUNLOP
```

```
from matplotlib.lines import Line2D

legend_elements = [
    Line2D([0], [0], marker='o', color='w', label='Current Stop',markerfacecolor='g', markersize=15),
    Line2D([0], [0], marker='o', color='w', label='Bus number',markerfacecolor='r', markersize=15),
]
route_legend = [
    Line2D([0], [0], marker='o', color='w', label= bus_num,markerfacecolor=route_color_map[bus_num], markersize=15) for bus_num in route_color_map
]
legend_elements = legend_elements + route_legend

# Create the figure
# fig, ax = plt.subplots()
# ax.legend(handles=legend_elements, loc='upper right')

# print(ax)

plt.figure(figsize=(100,100))
# pos=nx.spring_layout(G, iterations=100)
# pos=nx.kamada_kawai_layout(G)
# pos=nx.shell_layout(G)
pos=nx.nx_pydot.graphviz_layout(G,prog="twopi",root=root_node)
# print(pos)
nx.draw_networkx(G, pos=pos,edge_color=(0.8,0.6,0.3), node_color=color, arrows=True,
    arrowsize= 25)
# , ax=ax q
# Setting it to how it was looking before.
# plt.axis('off')
plt.legend(handles=legend_elements, loc='upper right')
plt.savefig('routewise_map_from_dunlop_with_legend.png')
```

Alternative Representation

- use different color edges to represent

```
G = nx.from_pandas_edgelist(sample, "derived_stop_name", "derived_next_stop", create_using=nx.DiGraph())
print(nx.is_directed(G))
```

```
True
```

```
edge_color = []
for u,v in G.edges():
    route_id = None
    u_list = u.split("-")
    v_list = v.split("-")
    if len(u_list) > 1:
        route_id = u_list[0]
    elif len(v_list) > 1:
        route_id = v_list[0]

    edge_color.append(route_color_map[route_id])
# print(edge_color)
```

- use different color node to represent stop

```
stops = sample["stop_name"].unique()
n=len(list(stops))

print(n)

reserved_color = [(1,0,0),(0,1,0),(0,0,1)] + edge_color

# generate N visually distinct colours
colors = distinctipy.get_colors(n,reserved_color)
# print(colors)
stop_color_map = { stop: color for stop, color in zip(stops,colors)}
# print(stop_color_map)
```

```
200
```

- display only route id
- set root node

```
node_color =[]
root_node = None
labels = {}
for node in G:
    if node == "DUNLOP":
        # current stop
        root_node = node
        node_color.append((0,1,0)) #green
        labels[node] = node
    elif node in list(sample["route_id"]):
        node_color.append((1,0,0)) #red
        labels[node] = node
    else:
        labels[node] = sample.loc[sample["derived_stop_name"]== node,"stop_id"].values[0]
        node_color.append(stop_color_map[node.split("-")[1]]) #route wise color
# print(root_node)
# print(labels)
```

- use graphviz radical layout
- add these details in legend
- increse edge width

```
from matplotlib.lines import Line2D

legend_elements = [
    Line2D([0], [0], marker='o', color='w', label='Current Stop (DUNLOP)',markerfacecolor='g', markersize=15),
    Line2D([0], [0], marker='o', color='w', label='Bus number',markerfacecolor='r', markersize=15),
]
route_legend = [
    Line2D([0], [0], marker='d', color='w', label= bus_num,markerfacecolor=route_color_map[bus_num], markersize=15) for bus_num in route_color_ma
]
stop_legend = [
    Line2D([0], [0], marker='o', color='w', label= stop,markerfacecolor=stop_color_map[stop], markersize=15) for stop in stop_color_map
]
legend_elements = legend_elements + route_legend + stop_legend

# Create the figure
# fig, ax = plt.subplots()
# ax.legend(handles=legend_elements, loc='upper right')

# print(ax)

plt.figure(figsize=(100,100))
# pos=nx.spring_layout(G, iterations=100)
# pos=nx.kamada_kawai_layout(G)
```

```
# pos=nx.shell_layout(G)
pos=nx.nx_pydot.graphviz_layout(G,prog="twopi",root=root_node)
# print(pos)
nx.draw_networkx(G, pos=pos,edge_color=edge_color, node_color=node_color, arrows=True,
    arrowsize= 25, width=5, labels=labels)
#  with_labels=False,
# , ax=ax q
# Setting it to how it was looking before.
# plt.axis('off')
plt.legend(handles=legend_elements, loc='upper right')
```

Tired to use external lable

- use graphviz external label layout - not possible since node is too close and tidious

```
!sudo apt-get install graphviz graphviz-dev -y
!pip install pygraphviz
```

```
Reading package lists... Done
Building dependency tree... Done
Reading state information... Done
Note, selecting 'libgraphviz-dev' instead of 'graphviz-dev'
graphviz is already the newest version (2.42.2-6ubuntu0.1).
The following additional packages will be installed:
  libgail-common libgail18 libgtk2.0-0 libgtk2.0-bin libgtk2.0-common
  libgvc6-plugins-gtk librsvg2-common libxdot4
Suggested packages:
  gvfs
The following NEW packages will be installed:
  libgail-common libgail18 libgraphviz-dev libgtk2.0-0 libgtk2.0-bin
  libgtk2.0-common libgvc6-plugins-gtk librsvg2-common libxdot4
0 upgraded, 9 newly installed, 0 to remove and 49 not upgraded.
Need to get 2,434 kB of archives.
After this operation, 7,681 kB of additional disk space will be used.
Get:1 http://archive.ubuntu.com/ubuntu jammy-updates/main amd64 libgtk2.0-common all 2.24.33-2ubuntu2.1 [125 kB]
Get:2 http://archive.ubuntu.com/ubuntu jammy-updates/main amd64 libgtk2.0-0 amd64 2.24.33-2ubuntu2.1 [2,038 kB]
Get:3 http://archive.ubuntu.com/ubuntu jammy-updates/main amd64 libgail18 amd64 2.24.33-2ubuntu2.1 [15.9 kB]
Get:4 http://archive.ubuntu.com/ubuntu jammy-updates/main amd64 libgail-common amd64 2.24.33-2ubuntu2.1 [132 kB]
Get:5 http://archive.ubuntu.com/ubuntu jammy-updates/universe amd64 libxdot4 amd64 2.42.2-6ubuntu0.1 [16.4 kB]
Get:6 http://archive.ubuntu.com/ubuntu jammy-updates/universe amd64 libgvc6-plugins-gtk amd64 2.42.2-6ubuntu0.1 [22.5 kB]
Get:7 http://archive.ubuntu.com/ubuntu jammy-updates/universe amd64 libgraphviz-dev amd64 2.42.2-6ubuntu0.1 [58.5 kB]
Get:8 http://archive.ubuntu.com/ubuntu jammy-updates/main amd64 libgtk2.0-bin amd64 2.24.33-2ubuntu2.1 [7,936 B]
Get:9 http://archive.ubuntu.com/ubuntu jammy-updates/main amd64 librsvg2-common amd64 2.52.5+dfsg-3ubuntu0.2 [17.7 kB]
Fetched 2,434 kB in 0s (8,092 kB/s)
debconf: unable to initialize frontend: Dialog
debconf: (No usable dialog-like program is installed, so the dialog based frontend cannot be used. at /usr/share/perl5/Debconf/FrontEnd/Dialog.pm
debconf: falling back to frontend: Readline
debconf: unable to initialize frontend: Readline
debconf: (This frontend requires a controlling tty.)
debconf: falling back to frontend: Teletype
dpkg-preconfigure: unable to re-open stdin:
Selecting previously unselected package libgtk2.0-common.
(Reading database ... 123623 files and directories currently installed.)
Preparing to unpack .../0-libgtk2.0-common_2.24.33-2ubuntu2.1_all.deb ...
```