

Leksioni 9

Endri Raco

10 May, 2024



1 Python për Analizën e Rrjetit

2 Shkenca e Rrjetave



Section 1

Python për Analizën e Rrjetit



- Në Python ne përdorim një kombinim të librarive të specializuara për analizimin e rrjetave.



- **networkx** (importuar si `nx`): Një nga paketat kryesore për të punuar me të dhëna rrjeti
- **osmnx**: Një paketë për të nxjerrë të dhëna rrjeti nga OpenStreetMap dhe për t'i manipuluar ato në `networkx` për analizë komplekse të rrjeteve (<https://github.com/gboeing/osmnx>).



- Këto paketa mbështesin disa nga funksionalitetet e njëjta të implementuara në ESRI-të e famshme ArcGIS Network Analyst.



Section 2

Shkenca e Rrjetave



- Rrjetet, të njohura gjithashtu si grafe, janë një strukturë thelbësore e të dhënave që lejon përfaqësimin e sistemeve.
- Një **nyje** (node) (e quajtur gjithashtu vertex) është një element i rrjetit që përfaqëson një entitet (p.sh., një person, një mjet, një kafshë, një stacion, një organizatë).



- Një **skaj** (edge) (i quajtur gjithashtu link) është një lidhje midis dy nyjeve dhe mund të ketë attribute.
- Një skaj përfaqëson një marrëdhënie midis dy nyjeve (p.sh., një miqësi midis dy personave, një hekurudhë midis dy stacioneve të trenit, një telefonatë midis dy përdoruesve të telefonit).
- Një skaj mund të jetë **i drejtuar** (directed) ($a \rightarrow b$ dhe $b \rightarrow a$ janë dy skaje të veçanta) ose **i pa drejtuar** (undirected) ($a - b$ nuk ka drejtim).



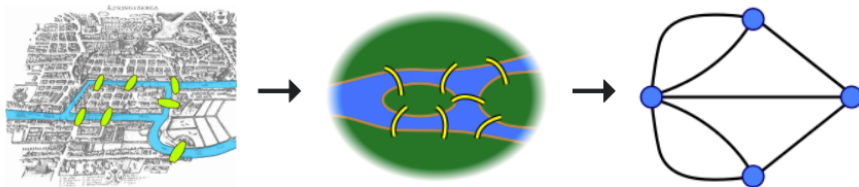
- *networkx* mbështet krijimin e llojeve të ndryshme të rrjeteve dhe ofron implementime të shumë algoritmeve dhe metrikave të grafeve.



- Një nga shembujt më të hershëm të problemeve teorike të grafeve është shtatë urat e Königsberg.
- A ka një rrugë që kalon çdo urë saktësisht një herë?



- Leonhard Euler (1735) tregoi se një shëtitje e tillë nuk ekzistonte duke formuluar problemin si një graf



- Në shkencën e të dhënave gjeografike, rrjetet gjeohapësinore janë rrjete ku vendndodhja e nyjeve dhe skajeve është e nevojshme për t'i studiuar dhe përdorur për analizë.
- Të tilla rrjete janë të kudogjendura dhe përfshijnë rrjetet e transportit, rrjetet hidrologjike, matricat origjinë-destinacion (të dhëna të rrjedhës), rrjetet e energjisë dhe rrjetet tregtare.



- Një nga problemet themelore në grafe është llogaritja e rrugës më të shkurtër midis dy nyjeve.
- Ky problem ka aplikime të panumërta në sektorin gjeohapësinor, duke përfshirë rrugëtimin nga pika A në B.



- Që nga viti 1956, algoritmi i Dijkstra ka qenë mënyra më e famshme për të llogaritur një rrugë më të shkurtër në një graf me peshë.



Algoritmi i Dijkstra

```
## Prezantimi i Grafit dhe Nyjeve
```

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

```
# Krijoni graf-in
```

```
G = nx.Graph()
```

```
edges = [
```

```
    ('A', 'B', 1),
```

```
    ('A', 'C', 4),
```

```
    ('B', 'C', 2),
```

```
    ('B', 'D', 5),
```

```
    ('C', 'D', 1),
```

```
    ('C', 'E', 3),
```

```
    ('D', 'E', 2)
```

```
]
```

```
G.add_weighted_edges_from(edges)
```

```
# Vizualizoni graf-in
```

```
pos = nx.spring_layout(G)
```

```
nx.draw(G, pos, with_labels=True, node_color='lightblue', edge_color='grey')
```

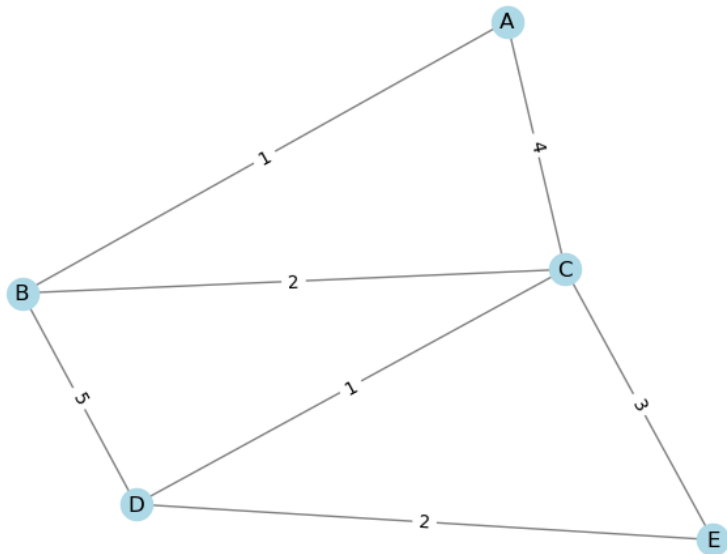
```
labels = nx.get_edge_attributes(G, 'weight')
```

```
nx.draw_networkx_edge_labels(G, pos, edge_labels=labels)
```

```
plt.show()
```



Algoritmi i Dijkstra



Algoritmi i Dijkstra - Hapi 1

Filloni me nyjen fillestare (A)

- Distanca nga A në A: 0



Filloni me nyjen fillestare (A)

- Distanca nga A te fqinjët:

B: 1

C: 4

- Rrugët më të shkurtra:

A \rightarrow A: 0

A \rightarrow B: 1

A \rightarrow C: 4



Zgjidhni nyjen më të afërt (B)

- Distanca nga B te fqinjët:

C: $3 (1 + 2)$

D: $6 (1 + 5)$

- Rrugët më të shkurtra:

- A \rightarrow B: 1

- A \rightarrow C: 3 (via B)

- A \rightarrow D: 6 (via B)



Zgjidhni nyjen më të afërt të ardhshme (C)

- Distanca nga C te fqinjët:

D: $4 (3 + 1)$

E: $6 (3 + 3)$

- Rrugët më të shkurtra:

A \rightarrow B: 1

A \rightarrow C: 3 (via B)

A \rightarrow D: 4 (via C)

A \rightarrow E: 6 (via C)



Algoritmi i Dijkstra - Hapi 5

Zgjidhni nyjen më të afërt të ardhshme (D)

- Distanca nga D te fqinjët:

E: 6 (via C)

- Rrugët më të shkurtra:

A \rightarrow B: 1

A \rightarrow C: 3 (via B)

A \rightarrow D: 4 (via C)

A \rightarrow E: 6 (via C)

Distancat më të shkurtra janë: {'A': (None, 0), 'B': ('A', 1), 'C': ('B', 3), 'D': ('C', 4), 'E': ('C', 6)} Rruga më e shkurtër nga A në E është: ['A', 'B', 'C', 'E'] me total 6.



Algoritmi i Dijkstra - Hapi 4

```
import networkx as nx

# Krijoni graf-in
G = nx.Graph()
edges = [
    ('A', 'B', 1),
    ('A', 'C', 4),
    ('B', 'C', 2),
    ('B', 'D', 5),
    ('C', 'D', 1),
    ('C', 'E', 3),
    ('D', 'E', 2)
]
G.add_weighted_edges_from(edges)

# Filloni me nyjen 'A'
shortest_paths = {'A': (None, 0)} # Nyja 'A' me kosto zero
current_node = 'A'
visited = set()

# Hapi 2 dhe 3: Zgjidhni nyjen më të afërt dhe llogaritni distancat
def calculate_shortest_paths(graph, start_node):
    visited = set()
    shortest_paths = {start_node: (None, 0)}
    current_node = start_node

    while current_node:
        visited.add(current_node)
        destinations = graph[current_node]
        weight_to_current_node = shortest_paths[current_node][1]
```

Algoritmi i Dijkstra - Hapi 4

```
for next_node, weight in destinations.items():
    weight = weight['weight']
    total_weight = weight_to_current_node + weight
    if next_node not in shortest_paths:
        shortest_paths[next_node] = (current_node, total_weight)
    else:
        current_weight = shortest_paths[next_node][1]
        if current_weight > total_weight:
            shortest_paths[next_node] = (current_node, total_weight)

next_destinations = {node: shortest_paths[node] for node in shortest_paths if node not in visited}
if not next_destinations:
    break
current_node = min(next_destinations, key=lambda k: next_destinations[k][1])

return shortest_paths
```



Algoritmi i Dijkstra - Hapi 4

```
# Llogaritni rrugët më të shkurtra nga 'A' në të gjitha nyjet
shortest_paths = calculate_shortest_paths(G, 'A')
print("Distancat më të shkurtra janë:", shortest_paths)

# Për rrugën më të shkurtër nga A në E
def get_path(shortest_paths, end_node):
    path = []
    while end_node:
        path.append(end_node)
        next_node = shortest_paths[end_node][0]
        end_node = next_node
    path = path[::-1]
    return path

# Gjeni rrugën më të shkurtër nga A në E
shortest_path = get_path(shortest_paths, 'E')
print("Rruga më e shkurtër nga A në E është:", shortest_path)
```



- Le të krijojmë një graf me skaje të peshësuara.
- Peshat mund të përfaqësojnë për shembull koston e përshkimit të një skaji (p.sh., një skaj i gjatë ka një peshë më të madhe se një i shkurtër) dhe përdoren shumë shpesh për të përfaqësuar sisteme.



Në disa skenarë, grafet mund të krijohen dhe popullohen drejtpërdrejt me networkx:

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

# krijoni një graf të thjeshtë të pa drejtuar
g = nx.Graph(name='graf i vogël')

# shtoni skaje me një atribut (peshë)
g.add_edge('a', 'b', weight=0.1)
g.add_edge('b', 'c', weight=1.5)
g.add_edge('a', 'c', weight=1.0)
g.add_edge('c', 'd', weight=2.2)

# Printoni informacionin e grafit
print("Emri:", g.graph['name'])
print("Tipi:", type(g))
print("Numri i nyjeve:", g.number_of_nodes())
print("Numri i skajeve:", g.number_of_edges())
print("Shkalla mesatare:", sum(dict(g.degree()).values()) / g.number_of_nodes())
```



- Veçanërisht për rrjete të vogla, është e mundur t'i vizatojmë ato drejtpërdrejt.
- Vini re se ky graf i pa drejtuar është jo-hapësinor, në kuptimin që nyjet nuk kanë një vendndodhje të caktuar hapësinore.



Vizualizimi i Grafit

```
# gjeneroni pozicionet e nyjeve duke përdorur një paraqitje të njohur
pos = nx.spring_layout(g)

# hapni vizatimin
nx.draw_networkx(g, pos=pos, font_color='white')

# vizatoni etiketat e skajeve
nx.draw_networkx_edge_labels(g, pos=pos)

# vizatoni skajet
nx.draw_networkx_edges(g, pos=pos)

# shfaqni graf-in
plt.show()
```



- Elementet e grafrit mund të aksesohen drejtpërdrejt

```
# Nyjet e grafrit
g.nodes
# NodeView(('a', 'b', 'c', 'd'))

# Skajet e grafrit
g.edges
# EdgeView([('a', 'b'), ('a', 'c'), ('b', 'c'), ('c', 'd')])
```



Aksesimi i Elementeve të Grafit

```
# shikoni nyjet fqinje të nyjës 'a':  
for node_nei in g.neighbors('a'):  
    print(node_nei)
```



```
weights = nx.get_edge_attributes(g, 'weight')  
weights  
# {'a', 'b'): 0.1, ('a', 'c'): 1.0, ('b', 'c'): 1.5, ('c', 'd'): 2.2}
```



Llogaritja e Rrugës më të Shkurtër

- Tani mund të përdorim **networkx** për të llogaritur rrugën më të shkurtër midis nyjeve.
- Fillimisht, mund të llogarisim rrugën më të shkurtër të peshuar (**unweighted shortest path**), duke supozuar se të gjitha skajet kanë të njëjtën peshë:



Llogaritja e Rrugës më të Shkurtër

```
unweighted_path = nx.shortest_path(g, 'b', 'd')  
unweighted_path
```



- Tani, mund të shohim rrugën më të shkurtër të peshuar (**weighted shortest path**), duke vënë re se shuma e peshave të (**bacd**) është më e lehtë se (**bcd**):



```
weighted_path = nx.shortest_path(g, 'b', 'd', weight='weight')  
weighted_path
```



- Për të përshkruar dhe kuptuar rrjetet, janë krijuar shumë metrika.
- Për shembull, ne mund të shohim se sa skaje janë të lidhura me një nyje (shkalla (**degree**) e një nyje).



- Nyja ‘a’ ka dy skaje, ndërsa ‘c’ ka 3.
- Disa nyje me shkallë të lartë janë “qendra”, duke lidhur shumë nyje: Për shembull, në rrjetet sociale, politikanët janë qendra.



- Qendra (**centrality**) i një nyje është gjithashtu i rëndësishëm: Sa “hapje” në skaje janë të nevojshme për të arritur të gjitha nyjet e tjera?



- Për shembull, Londra është një qendër e udhëtimeve ajrore më e rëndësishme se çdo qytet tjetër në Mbretërinë e Bashkuar



Metrikat e Rrjetit

```
import networkx as nx
```

```
# Krijo graf-in
```

```
g = nx.Graph(name='graf i vogël')
```

```
g.add_edge('a', 'b', weight=0.1)
```

```
g.add_edge('b', 'c', weight=1.5)
```

```
g.add_edge('a', 'c', weight=1.0)
```

```
g.add_edge('c', 'd', weight=2.2)
```

```
# Ekstraktoni të gjitha shkallët: kjo është zakonisht një pjesë e rëndësishme e informacionit për të kuptuar str  
g.degree()
```



```
# Nxirrni shkallët në një listë të thjeshtë  
degrees = [deg for node, deg in g.degree()]  
degrees
```

Metrikat e Rrjetit

```
# 'c' është nyja më qendrore, pasi mund të arrijmë të gjitha nyjet e tjera me numrin më të vogël të hapave, ndër  
nx.algorithms centrality.degree centrality(g)
```



- Një tjetër metrikë popullore e qendralitetit është **betweenness centrality** i bazuar në rrugët më të shkurtra midis nyjeve



- Peshat e skajeve **Edge weights** mund të jenë të rëndësishme për të llogaritur centralitetin dhe metrika të tjera.



Metrikat e Rrjetit

```
# Centraliteti i nyjeve  
nx.algorithms centrality.betweenness centrality(g, weight='weight')  
# 'a' dhe 'b' janë nyjet më qendrore të grafit
```



```
# Centraliteti i skajeve  
nx.algorithms centrality.edge_betweenness_centrality(g, weight='weight')  
# 'ac' është skaji më qendror i grafit
```



- Analiza e rrjetave aplikohet me sukses për të modeluar ndërveprimet në ekonomi, sociologji dhe fusha të tjera.
- Të dhënat nga kjo analizë përfaqësojnë ndërveprimet tregtare midis palëve (p.sh. tregtia e kafshëve midis Argjentinës dhe Brazilit në vitin 2017).



Rrjetet e marra nga këto të dhëna do të ofrojnë një pamje të anuar mbi tregtinë botërore.

```
import pandas as pd

# ngarkoni të dhënat nga një skedar TSV duke përdorur pandas
trade_df = pd.read_csv('data/gis4/trade_network_wits-1988_2018.tsv', sep='\t')
# shfaqni katër rreshta shembuj nga të dhënat e ngarkuara
trade_df.sample(4)
```



	partner	reporter	product_categories	indicator_type	indicator	2018	2017	2016	2015	2014	...	1995	1994	1993	1
4978	France	Antigua and Barbuda	all_products	export	trade_us_mil_top_5_export_partner	NaN	NaN	NaN	NaN	NaN	...	NaN	NaN	NaN	
11027	United States	Zimbabwe	all_products	import	partner_share_pc_top_5_import_partner	NaN	NaN	NaN	NaN	NaN	...	NaN	NaN	NaN	
15951	World	Congo, Rep.	all_products	export	exports_in_us_mil	NaN	8148.01	NaN	NaN	6550.0	...	1089.79	917.8	965.34	
401	...	Hungary	NaN	export	no_of_export_products	4006.0	3983.00	4011.0	4010.0	4011.0	...	3143.00	3024.0	2901.00	28

4 rows × 38 columns



Pasi kemi ngarkuar dhe shfaqur disa rreshta shembull nga të dhënat, le të përqendrohemi në të dhënat e vitit 2018 dhe të heqim të dhënat e paplota:



```
import pandas as pd
# përzgjedhja e kolonave të interesit për analizë
trade_df = trade_df[['reporter_country_code', 'reporter', 'partner_country_code', 'partner',
                     'product_categories', 'indicator_type', 'indicator', '2018']]

# përqendrimi në vëllimin total në USD të importit/eksportit midis vendeve
trade_df = trade_df[~trade_df['reporter_country_code'].isnull()]
trade_df = trade_df[~trade_df['partner_country_code'].isnull()]
# heqja e të dhënave të paplota
trade_df = trade_df[~trade_df['2018'].isnull()]
trade_df = trade_df[trade_df['product_categories'] == 'all_products']

# përqendrimi në tregtarët më të mëdhenj të importit/eksportit
trade_df = trade_df[trade_df['indicator'].isin(['partner_share_pc_top_5_import_partner', 'partner_share_pc_top_5
# kolonat e tjera interesante janë: 'trade_us_mil_top_5_export_partner', 'trade_us_mil_top_5_import_partner']
```



```
# numri total i rreshtave të filtruar  
len(trade_df)
```

Për të parë shembullin e rrjetit tregtar të Kinës:

```
# shembull i rrjetit tregtar të Kinës  
trade_df[trade_df.reporter_country_code=='CHN']
```



	reporter_country_code	reporter	partner_country_code	partner	product_categories	indicator_type	indicator	2018
4009	CHN	China	CHN	China	all_products	import	partner_share_pc_top_5_import_partner	6.86
5970	CHN	China	HKG	Hong Kong, China	all_products	export	partner_share_pc_top_5_export_partner	12.15
6991	CHN	China	JPN	Japan	all_products	export	partner_share_pc_top_5_export_partner	5.90
7084	CHN	China	JPN	Japan	all_products	import	partner_share_pc_top_5_import_partner	8.45
7377	CHN	China	KOR	Korea, Rep.	all_products	export	partner_share_pc_top_5_export_partner	4.37
7378	CHN	China	KOR	Korea, Rep.	all_products	import	partner_share_pc_top_5_import_partner	9.58
8437	CHN	China	OAS	Other Asia, nes	all_products	import	partner_share_pc_top_5_import_partner	8.31
11304	CHN	China	USA	United States	all_products	import	partner_share_pc_top_5_import_partner	7.31
11386	CHN	China	USA	United States	all_products	export	partner_share_pc_top_5_export_partner	19.23
12164	CHN	China	VNM	Vietnam	all_products	export	partner_share_pc_top_5_export_partner	3.37

Grafet nga dataframe

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

# Define a function to print graph information manually
def print_graph_info(g):
    num_nodes = g.number_of_nodes()
    num_edges = g.number_of_edges()
    avg_in_degree = sum(dict(g.in_degree()).values()) / num_nodes
    avg_out_degree = sum(dict(g.out_degree()).values()) / num_nodes

    print(f"Emri:")
    print(f"Tipi: {type(g).__name__}")
    print(f"Numri i nyjeve: {num_nodes}")
    print(f"Numri i degeve: {num_edges}")
    print(f"Mesatarja brenda në gradë: {avg_in_degree:.4f}")
    print(f"Mesatarja jashtë në gradë: {avg_out_degree:.4f}")

# Gjeneroni grafikun nga data frame (lista e degeve)
df = trade_df[trade_df.indicator == 'partner_share_pc_top_5_import_partner']
trade_import_g = nx.from_pandas_edgelist(df, 'reporter', 'partner', ['2018'], create_using=nx.DiGraph())
print_graph_info(trade_import_g)
```



Grafet nga dataframe

```
Emri:  
Tipi: DiGraph  
Numri i nyjeve: 140  
Numri i degeve: 663  
Mesatarja në gradë: 4.7357  
Mesatarja jashtë: 4.7357
```



Grafet nga dataframe

```
# Gjeneroni grafikun nga data frame (lista e skajeve)  
df = trade_df[trade_df.indicator=='partner_share_pc_top_5_export_partner']  
trade_export_g = nx.from_pandas_edgelist(df, 'reporter', 'partner', ['2018'], create_using=nx.DiGraph())  
print_graph_info(trade_export_g)
```



Grafet nga dataframe

```
Emri:  
Tipi: DiGraph  
Numri i nyjeve: 151  
Numri i degeve: 635  
Mesatarja në gradë: 4.2053  
Mesatarja jashtë: 4.2053
```



Shkalla e plot in- and out

Hapi i parë për të kuptuar strukturën e një rrjeti është llogaritja e shkallës së çdo nyje (numri i skajeve).

```
def plot_graph_degrees(g, title, logscale=False):
    """
    Paraqitni shkallën e hyrjes dhe daljes së grafikut g në një grafik linje.
    @ g: një grafik
    @ title: titulli i grafikës
    @ logscale: transformoni boshtet x dhe y me log10
    """
    # merrni shkallët
    indegrees = [deg for node, deg in g.in_degree()]
    outdegrees = [deg for node, deg in g.out_degree()]
    # numërimi i shkallës në hyrje
    indeg_vals = sorted(list(set(indegrees)))
    indeg_hist = [indegrees.count(x) for x in indeg_vals]
    # numërimi i shkallës në dalje
    outdeg_vals = sorted(list(set(outdegrees)))
    outdeg_hist = [outdegrees.count(x) for x in outdeg_vals]
    # shtoni informacionin në titull
    title = title + " [" + str(len(g.nodes)) + " nyje, " + str(len(g.edges)) + " skaje]"
    # paraqitni
    plt.figure()
    plt.grid()
    # paraqitni shkallët
    plt.plot(indeg_vals, indeg_hist, 'ro-')
    plt.plot(outdeg_vals, outdeg_hist, 'bv-')
```

Shkalla e plot in- and out (vazhdim)

```
if logscale:
    plt.xscale("log")
    plt.yscale("log")
    title = title + " [log]"

# shтони legjendën dhe etiketat
plt.legend(['shkalla në hyrje', 'shkalla në dalje'])
plt.xlabel('Shkalla')
plt.ylabel('Numri i nyjeve')
plt.title(title)
plt.show()

# paraqitni strukturën e rrjetit
plot_graph_degrees(trade_export_g, 'Rrjeti i eksportit tregtar', logscale=False)
plot_graph_degrees(trade_export_g, 'Rrjeti i eksportit tregtar', logscale=True)
plot_graph_degrees(trade_import_g, 'Rrjeti i importit tregtar', logscale=False)
plot_graph_degrees(trade_import_g, 'Rrjeti i importit tregtar', logscale=True)
```



Shkalla e plot in- and out (vazhdim)

```
# ndërtto strukturën e rrjetit
```

```
plot_graph_degrees(trade_export_g, 'Rrjeti i eksportit tregtar', logscale=False)
```

```
plot_graph_degrees(trade_export_g, 'Rrjeti i eksportit tregtar', logscale=True)
```

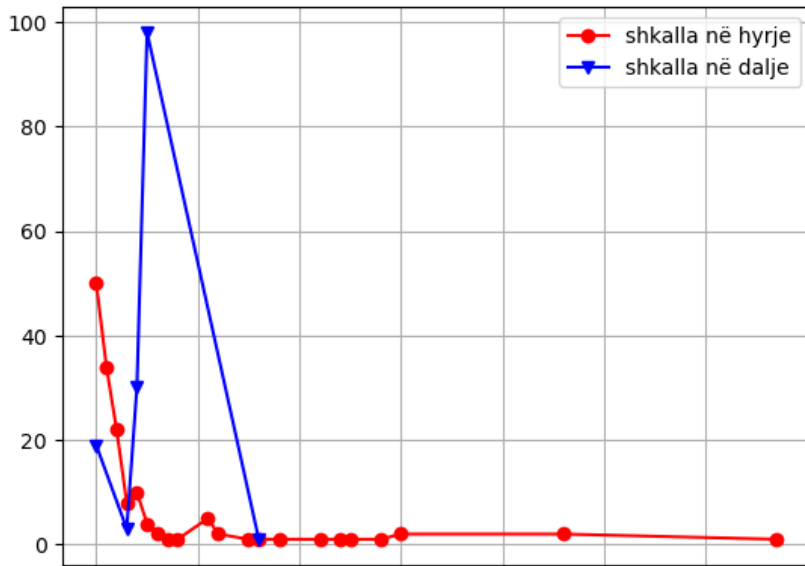
```
plot_graph_degrees(trade_import_g, 'Rrjeti i importit tregtar', logscale=False)
```

```
plot_graph_degrees(trade_import_g, 'Rrjeti i importit tregtar', logscale=True)
```



Shkalla e plot in- and out (vazhdim)

Rrjeti i eksportit tregtar [151 nyje, 635 skaje]



- Gjenerimi i vizualizimeve të mira të rrjeteve nuk është i lehtë.
- Kur një rrjet rritet në numër të nyjeve dhe shkallë, mbivendosja bëhet e pashmangshme.



- Libraria ofron funksionalitet për vizualizim të avancuar
- Me grafet hapësinore, është e nevojshme të zgjidhni një paraqitje për të rregulluar nyjet në një mënyrë të përshtatshme.



Vizualizimi i Nyjeve dhe Skajeve

```
# përkufizoni funksionin e ripërdorshëm për të vizatuar paraqitjen e rrjetit
def plot_graph(g, title, edge_label_attribute, layout='spring', fig_sz=10):
    """
    Funksioni për të vizatuar një grafik të drejtuar me vlera të përshtatshme të parazgjedhura.
    @ g: një grafik
    @ title: titulli i grafikut
    @ edge_label_attribute: emri i atributit të skajit që do të përdoret si etiketa e skajit
    @ layout: paraqitja e rrjetit që do të përdoret
    @ fig_sz: madhësia e kanavacës
    """
    # gjeneroni pozicionet e nyjeve bazuar në paraqitjen e zgjedhur
    if layout == 'spring':
        pos = nx.spring_layout(g, k=0.55, iterations=30)
    elif layout == 'circular':
        pos = nx.circular_layout(g)
    else:
        raise ValueError("Paraqitje e pavlefshme")
```



Vizualizimi i Nyjeve dhe Skajeve (vazhdim)

```
# rregulloni figurën
plt.figure(figsize=(fig_sz, fig_sz))
# hapni vizatimin
nx.draw_networkx(g, pos=pos, node_color='lightblue', alpha=.8)
# nxirrni etiketat e skajeve
edge_labels = nx.get_edge_attributes(g, edge_label_attribute)
# vizatoni etiketat e skajeve
nx.draw_networkx_edge_labels(g, pos=pos, edge_labels=edge_labels, alpha=.6)
# vizatoni skajet
nx.draw_networkx_edges(g, pos=pos, arrows=True, alpha=.6, edge_color='lightgray')
plt.title(title)
plt.show()
```

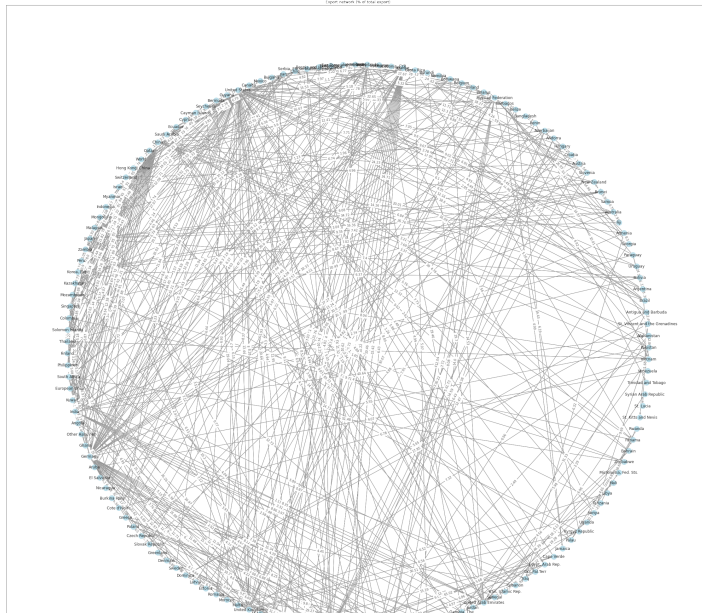


Vizualizimi i Nyjeve dhe Skajeve (vazhdim)

```
# vizatoni grafikun me dy paraqitje të ndryshme
plot_graph(trade_export_g, "Rrjeti i eksportit (% i eksportit total)", '2018', 'circular', fig_sz=40)
plot_graph(trade_import_g, "Rrjeti i importit (% i importit total)", '2018', 'spring', fig_sz=40)
```



Vizualizimi i Nyjeve dhe Skajeve (vazhdim)



- Këto vizualizime në thelb janë të padobishme pasi nuk mund të dallojmë strukturën e rrjeteve.
- Një strategji për të vizualizuar një rrjet të madh është të përfshini vetëm nyje të lidhura relativisht.
- Për shembull, le të zgjedhim nyje me shkallë të lartë nga grafiku **trade__export__g**:



Vizualizimi i Nyjeve dhe Skajeve

```
# zgjedhni nyjet me shkallë të lartë
degrees = trade_export_g.degree()
nodes_to_remove = [n for n, degree in degrees if degrees[n] < 20]

# kopjoni dhe hiqni nyjet nga grafiku
hideg_trade_export_g = trade_export_g.copy()
hideg_trade_export_g.remove_nodes_from(nodes_to_remove)

# shtypni informacionin e grafikut të shkallës së lartë
def print_graph_info(g):
    num_nodes = g.number_of_nodes()
    num_edges = g.number_of_edges()
    avg_in_degree = sum(dict(g.in_degree()).values()) / num_nodes
    avg_out_degree = sum(dict(g.out_degree()).values()) / num_nodes

    print(f"Name:")
    print(f>Type: {type(g).__name__}")
    print(f"Number of nodes: {num_nodes}")
    print(f"Number of edges: {num_edges}")
    print(f"Average in degree: {avg_in_degree:.4f}")
    print(f"Average out degree: {avg_out_degree:.4f}")

print_graph_info(hideg_trade_export_g)
```



Vizualizimi i Nyjeve dhe Skajeve

Name:

Type: DiGraph

Number of nodes: 12

Number of edges: 41

Average in degree: 3.4167

Average out degree: 3.4167



Grafiku me shkallë të lartë është më i lehtë për t'u vizatuar dhe ne mund të shohim lidhjet midis vendeve me shkallë të lartë:



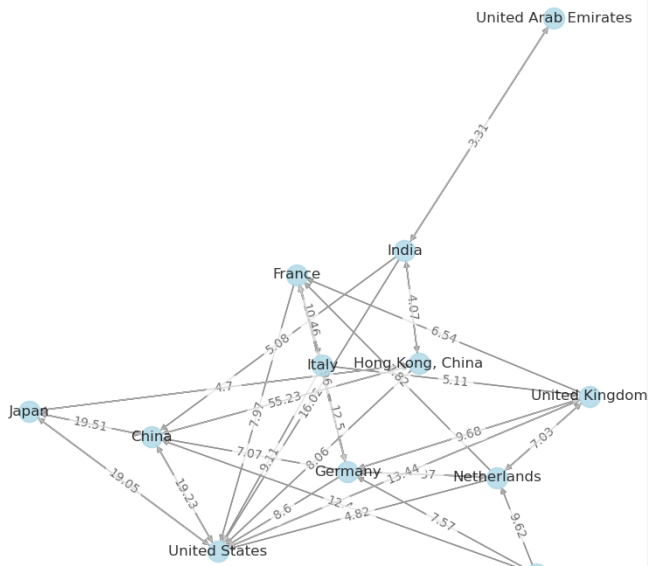
Vizualizimi i Nyjeve dhe Skajeve

```
# grafiku i shkallës së lartë  
plot_graph(hideg_trade_export_g, "Rrjeti i eksportit (% i eksportit total)", '2018', 'spring', fig_sz=10)
```



Vizualizimi i Nyjeve dhe Skajeve

Netja e Eksportit (70 Eksportues)



 Endri Raco

Vizualizimi i Nyjeve dhe Skajeve

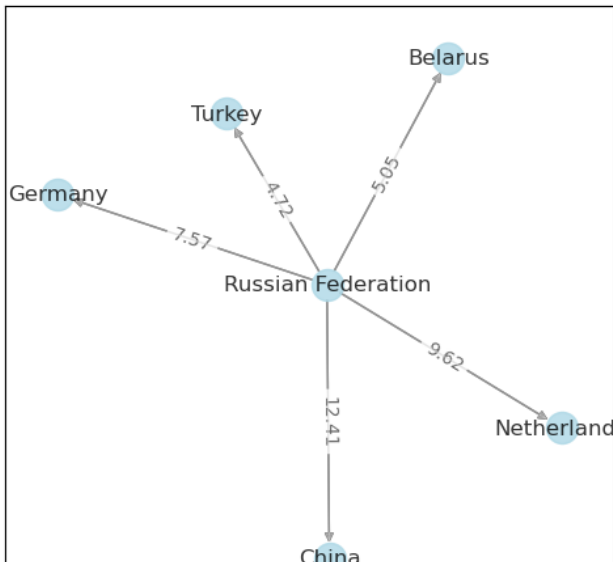
Një qasje tjetër është të zgjidhni nën-grafikë, për shembull fqinjët e një nyje të synuar:

```
# tregoni nën-grafikë për secilin vend me shkallë të lartë
for country in hideg_trade_export_g.nodes:
    # nxirrni nën-grafikun për rrjetin e eksportit të një vendi
    # skajet në dalje
    edges = trade_export_g.out_edges(country, data=True)
    g = nx.DiGraph(edges)
    plot_graph(g, "Rrjeti i eksportit nga\n" + country + ' (% i eksportit total)', '2018', fig_sz=6)
    # skajet në hyrje
    edges = trade_export_g.in_edges(country, data=True)
    g = nx.DiGraph(edges)
    plot_graph(g, "Rrjeti i eksportit për\n" + country + ' (% i eksportit total)', '2018', fig_sz=6)
```



Vizualizimi i Nyjeve dhe Skajeve

Rrjeti i eksportit nga
Russian Federation (% i eksportit total)



Endri Raco

- Shpesh është e nevojshme të eksportoni një grafik në një format standard të skedarit
- **networkX** supporton shumë formate.
- Ndër formatet më të supportuara janë **GraphML** dhe **GEXF**, të dy bazuar në XML.



Ekzaminoni skedarët e eksportuar me një redaktues teksti për të parë se si kodohen të dhënat.



Eksportimi i grafikut

```
# eksportoni grafikun si GraphML dhe GEXF  
nx.write_graphml(hideg_trade_export_g, "tmp/street_network.graphml")  
nx.write_gexf(hideg_trade_export_g, "tmp/street_network.gexf")
```



- Ne mund të përdorim **networkX** për të llogaritur metrikat e rrjetit, për shembull për të parë qendrën e ndërmjetësimit të vendeve në të dy rrjetet e ndryshme që përfaqësojnë importin dhe eksportin:



```
# llogarisni qendrën e ndërmjetësimit bazuar në dendësinë e nyjeve
country_centrality = nx.algorithms.centrality.betweenness centrality(trade_export_g)

# konvertoni të dhënat në një kuadër të dhënash
country_centrality_df = pd.DataFrame.from_dict(country_centrality,
                                              orient='index', columns=['betw centrality export'])
country_centrality_df['country'] = country_centrality_df.index

# renditni dhe tregoni rezultatet
country_centrality_df = country_centrality_df.sort_values('betw centrality export', ascending=False)
country_centrality_df.head(10)
```



	betw centrality export	country
United States	0.042676	United States
India	0.033241	India
China	0.031147	China
Hong Kong, China	0.024218	Hong Kong, China
United Arab Emirates	0.023079	United Arab Emirates
United Kingdom	0.018608	United Kingdom
Germany	0.017931	Germany
Japan	0.017918	Japan
France	0.015561	France
Singapore	0.012022	Singapore

Importo Rrjetin

	betw Centrality Import	country
China	0.040320	China
Japan	0.036756	Japan
Germany	0.029491	Germany
United States	0.026588	United States
Australia	0.019231	Australia
Saudi Arabia	0.016419	Saudi Arabia
Netherlands	0.013404	Netherlands
Thailand	0.012864	Thailand
France	0.010686	France

Endri Raco

Kombinimi i Treguesve për të Krahastuar

```
# kombinimi dy treguesit për t'i krahasuar ata
impexp_df = country_centrality_df.merge(imp_df, on='country')
impexp_df = impexp_df[['country', 'betw_centrality_import', 'betw_centrality_export']]
impexp_df = impexp_df.sort_values('betw_centrality_export', ascending=False)
impexp_df.to_csv('tmp/trade_importexport_centrality_2018.csv')
impexp_df.head(10)
```



Kombinimi i Treguesve për të Kraheluar

	country	betw_centrality_import	betw_centrality_export
0	United States	0.026588	0.042676
1	India	0.008226	0.033241
2	China	0.040320	0.031147
3	Hong Kong, China	0.000000	0.024218
4	United Arab Emirates	0.005016	0.023079
5	United Kingdom	0.003019	0.018608
6	Germany	0.029491	0.017931
7	Japan	0.036756	0.017918
8	France	0.010686	0.015561
9	Singapore	0.001280	0.012823

- Rrjetet rrugore janë një lloj rrjeti që është veçanërisht i rëndësishëm në analizën e të dhënave gjeografike.
- Libraria **osmnx** ofron funksione për të nxjerrë lehtësisht rrjetet rrugore nga **OpenStreetMap**.



- Mënyra më efikase për të ruajtur këto rrjete kur punoni në Python është formati **pickle**.
- Do të përdorim skedarë pickle të kompresuar.
- Shumë formate të tjera rrjeti ekzistojnë, duke përfshirë GeoPackages dhe GraphML, që mund të përdoren .



Merrni Rrjetet Rrugore

```
import osmnx as ox
import pickle

# Emri i vendit të synuar
place_name = "City of London, UK"

# Shkarkoni rrjetin rrugor nga OpenStreetMap bazuar në një emër vendi
graph = ox.graph_from_place(place_name, network_type='drive')

# Ruani rrjetin në disk duke përdorur gzip (.gz) për të zvogëluar madhësinë e skedarit
net_file = "data/streets_citylondon.gpik"

# Use pickle to write the graph to a file
with open(net_file, 'wb') as f:
    pickle.dump(graph, f)

print("Skedari i grafikut të rrugëve në", place_name, "shkarkuar në", net_file)
```



```
# 'del' (fshij) shkatërron një objekt. Është e dobishme për të siguruar që ky objekt  
# nuk mund të përdoret në qeliza të tjera gabimisht.  
del graph, net_file
```



Në mënyrë alternative, paketa supporton një **point and radius search** bazuar në adresë ose nga poligonet:

```
import osmnx as ox
import pickle
import gzip

# Vendndodhja e Birkbeck, rreze 1 km
radius_m = 1000
graph = ox.graph_from_address('Malet St, London, UK', network_type='drive', dist=radius_m)

# Ruani rrjetin në disk duke përdorur gzip (.gz) për të zvogëluar madhësinë e skedarit
net_file = "data/streets_birkbeck_1km.gpik.gz"

# Use pickle to write the graph to a compressed file
with gzip.open(net_file, 'wb') as f:
    pickle.dump(graph, f)

print(f"Rrjeti rrugor i ruajtur në {net_file}")

# Delete the graph object to free up memory
del graph
```



- Rrjeti rrugor shprehet si `networkx.classes.multidigraph.MultiDiGraph`.
- Le të ngarkojmë një skedar **pickle** dhe të vizatojmë rrjetin.



Vizato Rrjetin e Rrugës

```
import pickle
import gzip

# Load the graph from the compressed file
net_file = "data/streets_birkbeck_1km.gpik.gz"
with gzip.open(net_file, 'rb') as f:
    streets_g = pickle.load(f)

# Print the number of nodes and edges in the graph
print("N i nyjeve:", len(streets_g.nodes))
print("N i skajeve:", len(streets_g.edges))
```



Vizato Rrjetin e Rrugës

```
# vizatoni rrjetin  
osmnx.plot_graph(streets_g, figsize=(10,10))
```



Vizato Rrjetin e Rrugës



Skajet përmbajnë attribute, si emri i rrugës dhe lloji, të cilat mund të aksesohen si një dataframe gjeografik.

```
# Merrni vetëm skajet nga grafiku  
edges_df = osmnx.graph_to_gdfs(streets_g, nodes=False, edges=True)  
edges_df.sample(3)
```



Vizato Rrjetin e Rrugës

			osmid	name	highway	maxspeed	access	oneway	reversed	length	geometry
u	v	key									
108009	108034	0	[1186049313, 851719169, 694563843, 1065127938,...	Gower Street	primary	20 mph	NaN	False	False	199.796	LINESTRING (-0.13424 51.52396, -0.13419 51.523...
294158420	300501141	0	1067121896	Albany Street	primary	20 mph	NaN	False	False	90.621	LINESTRING (-0.14433 51.52858, -0.14435 51.529...
108901	11747638084	0	9346429	High Holborn	primary	20 mph	NaN	True	False	80.030	LINESTRING (-0.11918 51.51771, -0.11961 51.517...



```
edges_df.plot()
```

Vizato Rrjetin e Rrugës

7]: <Axes: >



2aço

Nyjet gjithashtu mund të kenë attribute, të tilla si vendndodhja dhe numri i skajeve ndërprerëse:

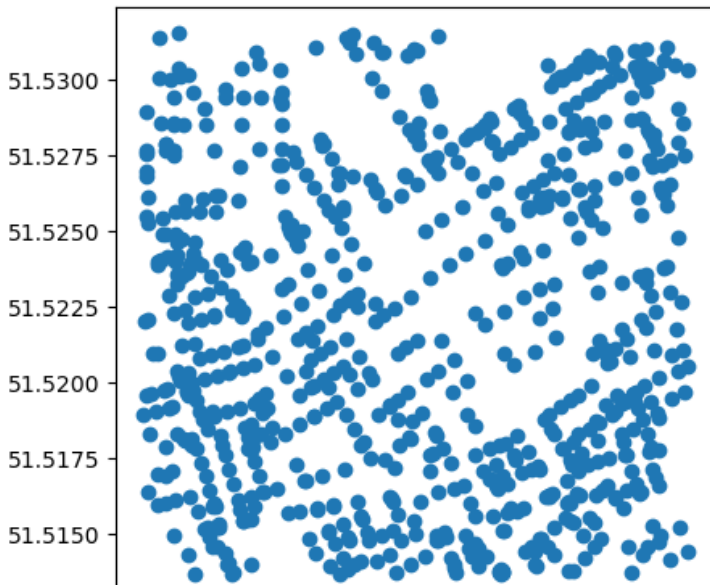
```
# Merrni nyjet nga grafiku si një kuadër të dhënash
nodes_df = osmnx.graph_to_gdfs(streets_g, nodes=True, edges=False)
nodes_df.sample(5)

# vizatoni nyjet
nodes_df.plot()
```



Vizato Rrjetin e Rugës

Fig. 10.10.1



Për të punuar në rrjetet rrugore, është e nevojshme t'i projektoni ato në një sistem të përshtatshëm të referencës koordinative.

```
print("Sistemi i koordinatave të nyjeve:", nodes_df.crs)
print("Sistemi i koordinatave të skajeve:", edges_df.crs)
```



Sistemi i koordinatave të nyjeve: epsg:4326

Sistemi i koordinatave të skajeve: epsg:4326

Projektimi i këtij rrjeti rrugor në British National Grid:

```
streets_g = osmnx.project_graph(streets_g, 27700)

# kontrolloni nëse projektimi funksionoi
nodes_proj, edges_proj = osmnx.graph_to_gdfs(streets_g, nodes=True, edges=True)
print(nodes_proj.crs)
print(edges_proj.crs)
```



EPSG:27700

EPSG:27700



Tani koordinatat janë projektuar

```
edges_proj.sample(5)
```



Projekto Rrjetin

			osmid	name	highway	maxspeed	oneway	length	geometry	lanes	ref	tunnel	access
u	v	key											
4243601481	4243601512	0	[425013568, 425013571, 425013579]	Stephen Street	unclassified	20 mph	True	45.710	LINESTRING (529679.139 181544.887, 529682.283 ...	1	NaN	NaN	NaN
11544696	7684041426	0	823004754	Dane Street	unclassified	20 mph	False	15.209	LINESTRING (530622.046 181683.648, 530628.976 ...	NaN	NaN	NaN	NaN
			Great						LINESTRING (530045.266				



Ndërtojmë përsëri

```
osmnx.plot_graph(streets_g, figsize=(10,10))
```



Ndërtojme përsëri



Rrjetet rrugore mund të analizohen në aspektin e strukturës së tyre (morfologji) duke përdorur një sërë treguesish:

```
import osmnx as ox

# Merrni statistikat bazike të rrjetit
basic_stats = ox.basic_stats(streets_g)
basic_stats
```



Llogaritja e Rrugëve

Një pjesë qendrore në udhëzime është llogaritja e rrugëve më të shkurtra në rrjetet rrugore.

```
import random
import networkx as nx
import matplotlib.pyplot as plt
from networkx.exception import NetworkXNoPath

# zgjidhni dy nyje të rastësishme
nodes = [n for n in streets_g.nodes]
orig_node = random.choice(nodes)
dest_node = random.choice(nodes)
print("Nyja e origjinës:", orig_node, ", nyja e destinacionit:", dest_node)
```



Llogaritja e Rrugëve (vazhdim)

```
# Përpiquni/gjeni se për shkak se ndonjëherë rrugët nuk mund të gjenden
try:
    # gjeni rrugën më të shkurtër midis dy nyjeve
    route = nx.shortest_path(streets_g, orig_node, dest_node, weight='length')
    route_len_m = nx.shortest_path_length(streets_g, orig_node, dest_node, weight='length')
    print("Rruga e udhëtimit e gjetur:", len(route), "segmente; gjatësi (m)", round(route_len_m))

    # vizatoni rrugën
    plt.figure(figsize=(6, 6))
    fig, ax = ox.plot_graph_route(streets_g, route, route_linewidth=6, node_size=0, bgcolor='k')
except NetworkXNoPath as e:
    print("Rruga nuk u gjet", e)
```



Llogaritja e Rrugëve



Endri Raço

- Kur udhëtoni nga pika A në pikën B, hapi i parë është të gjeni nyjet më të afërta me A dhe B.

```
from pyproj import Transformer

def project_points(points, in_epsg_code, out_epsg_code):
    """ Transformojmë pikat nga një EPSG te tjetra
    @ in_epsg_code: EPSG fillestare
    @ out_epsg_code: EPSG target
    """
    transformer = Transformer.from_crs(in_epsg_code, out_epsg_code, always_xy=True)
    proj_points = [pt for pt in transformer.itransform(points)]
    return proj_points
```



Llogaritja e Rugës së Udhëtimit

```
# origjina dhe destinacioni (lonx, laty)
birkbeck_loc = [-0.13134307, 51.522510]
british_museum_loc = [-0.12575, 51.518391]

# transformojmë lokacionin në British Grid (get first element with [0])
birkbeck_loc_bg = project_points([birkbeck_loc], 4326, 27700)[0]
britishmus_loc_bg = project_points([british_museum_loc], 4326, 27700)[0]
print("Vendndodhjet në BNG: ", birkbeck_loc_bg, britishmus_loc_bg)
```



Llogaritja e Rrugës së Udhëtimit

```
import matplotlib.pyplot as plt
import pyproj

birkbeck_node = ox.distance.nearest_nodes(streets_g, birkbeck_loc_bg[0], birkbeck_loc_bg[1])
britishmus_node = ox.distance.nearest_nodes(streets_g, britishmus_loc_bg[0], britishmus_loc_bg[1])

# Llogarisim rrugen me te shkurter
route = nx.shortest_path(streets_g, birkbeck_node, britishmus_node, weight='length')
route_len_m = nx.shortest_path_length(streets_g, birkbeck_node, britishmus_node, weight='length')

print("Rruga me e shkurter midis Birkbeck dhe British Museum u gjet:",
      len(route), "segmente; length (m):", round(route_len_m))
```



```
fig, ax = ox.plot_graph_route(streets_g, route, node_size=0)  
plt.show()
```

Këto të dhëna të rrjetit dhe mjetet mund të përdoren në një sërë aplikimesh.



