Homework Spring

Moiseev Dmitrii

Theorem 1 (Triangle Inequality)

Statement

For any connected graph $G = \langle V, E \rangle$:

$\forall x, y, z \in V : dist(x, y) + dist(y, z) \geq dist(x, z)$

Definitions

We define $dist(x, y)$ as $\min(|x \rightsquigarrow y|)$

Proof

Since G is connected, $\forall x, y, z \in V : \exists x \rightsquigarrow y, y \rightsquigarrow z, x \rightsquigarrow z$

Let's say that $dist(x, y) = |P_1|, dist(y, z) = |P_2|, dist(x, z) = |P_3|$, where P_1, P_2, P_3 are paths $x \rightsquigarrow y, y \rightsquigarrow z, x \rightsquigarrow z$ respectively.

Then we have a concatenation walk $W = P_1 + P_2$

$|W| = |P_1| + |P_2|$

If W is a path (has no repeating vertices), then by our definition of distance $|W| \geq dist(x, z)$, and thus $dist(x, y) + dist(y, z) \geq dist(x, z)$

Otherwise, we remove all edges in W between pairs of repeating vertices recursively until it's a path, which we will call W'

Since W' was obtained from W by removing edges, $|W| \geq |W'|$

At the same time, by the same logic as shown above, $|W'| \geq dist(x, z)$, and thus, once again $dist(x, y) + dist(y, z) \geq dist(x, z)$

Theorem 2

Statement

For any connected graph G: $rad(G) \leq diam(G) \leq 2rad(G)$

Definitions:

$\varepsilon(v) = \max_{u \in V} dist(v, u)$

$rad(G) = \min_{v \in V} \varepsilon(v)$ $diam(G) = \max_{v \in V} \varepsilon(v)$

Proof:

$rad(G) \leq diam(G)$ by definition.

Let's choose $v, u, w \in V$ such that $\varepsilon(v) = rad(G)$, $dist(u, w) = diam(G)$.

Per theorem 1: $dist(u, w) \leq dist(u, v) + dist(v, w)$.

Since $\varepsilon(v) = rad(G)$, $dist(u, v) \leq rad(G)$ and $dist(v, w) \leq rad(G)$.

Thus $diam(G) = dist(u, w) \leq dist(u, v) + dist(v, w) \leq 2rad(G)$.

Theorem 3

Statement

A connected graph $G = \langle V, E \rangle$ is a tree *iff* $|E| = |V| - 1$

Lemma 1

Statement

For any connected graph $G = \langle V, E \rangle$: $|E| \geq |V| - 1$

Proof

Let's start with graph $G' = \langle V, \emptyset \rangle$, and add edges from E that reduce the number of connected components one by one. G' has $|V|$ connected components, and each added edge reduces that number 1, so we will need to add $|V| - 1$ edges with this process to get a connected graph. Since G is connected, $|E| \geq |V| - 1$.

Proof

First, let's assume $|E| = |V| - 1$ and prove that G is a tree.

Suppose G contains a cycle C.

Per lemma 1, if we remove any one edge from G, it will cease to be connected.

Let's remove an edge $e \in C$, calling the resulting graph G'

Since G is connected, $\forall v, u \in V : \exists P = v \rightsquigarrow u$ in G. We can replace all occurrences of e in these paths with $C \setminus \{e\}$, getting equally valid paths.

But since the only difference between G and G' is the edge e, these new paths exist in G' as well, which means G' is connected. The contradiction means our assumption was incorrect, and G contains no cycles, making it a tree by definition.

Now, let's prove that for any tree $T\langle V, E\rangle : |E| = |V| - 1$

Let's build a minimal connected subgraph G' as in lemma 1. If we add another edge v, u from T, we create a cycle, as G' was already connected, thus having a path between v and u, and our newly added edge is another. But since T is a tree it by definition can't contain any cycles, which means there is no such edge, $T = G'$ and thus $|E| = |V| - 1$.

Theorem 4

Statement

Given a connected graph $G = \langle V, E\rangle$ with n vertices, if $\delta(G) \geq \lfloor n/2 \rfloor$, then $\lambda(G) = \delta(G)$.

Proof

Let $k = \lfloor n/2 \rfloor$

$\lambda(G) \leq \delta(G)$, since we can remove all edges incident to a vertex with minimum degree to detach it from the rest of the graph.

Let's look at a minimum edge cut, and consider the smallest connected component produced by such a cut, calling it $H\langle V', E'\rangle$.

Let $m = |V'|, l = |E'|$.

Note that $m \geq 1$

Since there have to be at least 2 components, $m <= k$.

Each vertex in V' can have at most $m-1$ incident edges in H, while in G it must have had at least $\delta(G)$ incident edges.

This means at least $m(\delta(G)-(m-1))$ edges must have been removed.

This is a quadratic function with a peak at $\frac{\delta(G)+1}{2}$, so its minimum on our interval $1 \leq m \leq k$ will lie at either $m=1$ or $m=k$ depending on which is further from the peak.

Since $\delta(G) \geq k$, $\frac{\delta(G)+1}{2} - 1 \geq k - \frac{\delta(G)+1}{2}$, so we can use the value at $m=1$, which is $\delta(G)$.

Thus we have $\delta(G) \geq \lambda(G) \geq \delta(G)$, and so $\lambda(G) = \delta(G)$.

Theorem 5

Statement

Every block of a block graph is a clique.

Proof

We'll be looking at a graph G, its block graph H and a block J of H.

Suppose J is not a clique of H, meaning $\exists v, u \in V(J)$ that are not adjacent.

Since J is a block, v and u lie on a cycle, but since they are not adjacent, this cycle must be at least of length 4. Let's call the shortest such cycle C.

Since H is the block graph of G, each vertex in $V(C)$ corresponds to a block in G, and each edge in $E(C)$ corresponds to a cut vertex in G.

For each $z \in V(C)$ we can find a path in the corresponding block of G connecting the cut vertices of G corresponding to edges incident to z in C.

Note that each pair of paths corresponding to adjacent vertices in $V(C)$ share an endpoint - the cut vertex corresponding to the edge connecting them.

Moreover, the only vertex intersections any pair of these paths can have are cut vertices, since each lies in a separate block of G.

But there are no intersections other than those already discussed, since otherwise 2 vertices in $V(C)$ are connected by an edge in $E(J)$ that is not in

$E(C)$, which would mean we can shorten C by replacing several edges with that one.

All of this means we can chain these paths to produce a cycle in G that passes through several blocks.

This is a contradiction, since any 2 vertices on an cycle must lie in the same block, which means our assumption was incorrect and every block of H is a clique.

Practical tasks

Repo

Instructions:

- install sagemath
- `sage --python -m pip install -r requirements.txt`
- if you want to reload data: `sage --python util.py`
- `sage -n`
- in this jupyter session open provided .ipynb

An html export of the notebook is also provided.

data loading

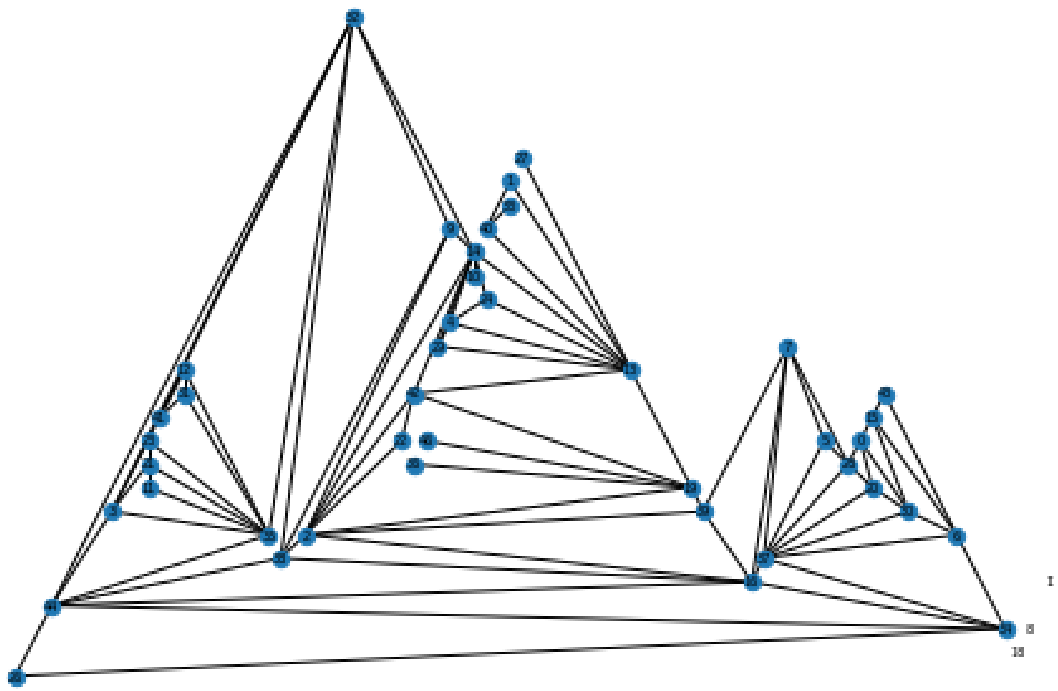
a

Statement

Prove that G' is a planar by drawing it on a plane with no intersection of edges.

```
0: Albania
1: Andorra
2: Austria
3: Belarus
4: Belgium
5: Bosnia and Herzegovina
6: Bulgaria
7: Croatia
8: Cyprus
9: Czech Republic
10: Denmark
11: Estonia
12: Finland
13: France
14: Germany
15: Greece
16: Hungary
17: Iceland
18: Ireland
19: Italy
20: Kosovo
21: Latvia
22: Liechtenstein
23: Lithuania
24: Luxembourg
25: Malta
26: Moldova
27: Monaco
28: Montenegro
29: Netherlands
30: North Macedonia
31: Norway
32: Poland
33: Portugal
34: Romania
35: Russia
36: San Marino
37: Serbia
38: Slovakia
39: Slovenia
```

```
40: Spain
41: Sweden
42: Switzerland
43: Turkey
44: Ukraine
45: United Kingdom
46: Vatican City
```



b

Statement

Find $|V|$, $|E|$, $\delta(G)$, $\Delta(G)$, $rad(G)$, $diam(G)$, $girth(G)$, $center(G)$, $\kappa(G)$, $\lambda(G)$.

Definitions

$|V|$ is the number of vertices in G

$|E|$ is the number of edges in G

$\delta(G) = \min\limits_{v \in V} deg(v)$

$\Delta(G) = \max\limits_{v \in V} deg(V)$

$\varepsilon(v) = \max_{u \in V} dist(v, u)$

$rad(G) = \min_{v\ inV} \varepsilon(v)$

$diam(G) = \max_{v \in V} \varepsilon(v)$

$girth(G)$ is the length of the shortest cycle in the G

$center(G) = \{v \in V \mid \varepsilon(v) = rad(G)\}$

$\kappa(G)$ is the minimum number of vertices, the removal of which would result in the number of connected components in G to be greater than 1

$\lambda(G)$ is the minimum number of edges, the removal of which would result in the number of connected components in G to be greater than 1

Answer

```
|V| = 42
|E| = 88
δ(G) = 1
Δ(G) = 9
rad(G) = 5
diam(G) = 8
girth(G) = 3
center(G) = ['Slovakia', 'Switzerland', 'Austria', 'Poland', 'Czech Republic',
'Russia', 'Ukraine', 'Germany', 'Hungary', 'Slovenia', 'Croatia', 'Belarus']
κ(G) = 1
λ(G) = 1
```

c

Statement

Find the minimum vertex coloring $Z : V \to \mathbb{N}$ of G

Definitions

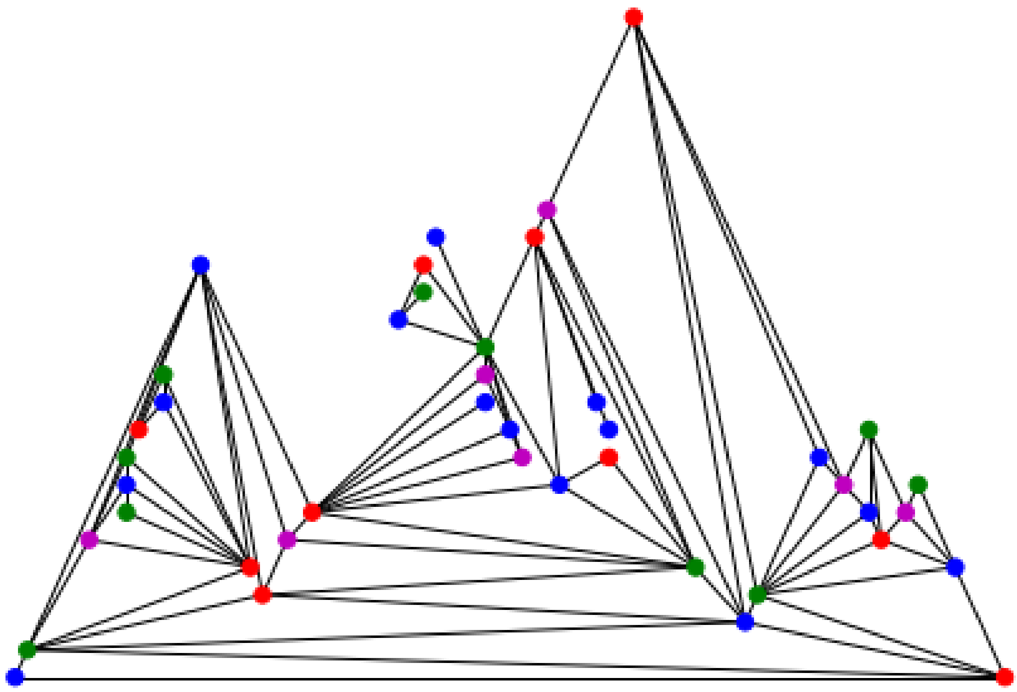
A function $Z : V \to \mathbb{N}$ is a vertex coloring of G *iff* $\forall (v, u) \in E : Z(v) \neq Z(u)$.

A vertex coloring is minimum *iff* $\max_{v \in V} Z(v)$ is minimum

Answer

```
Albania: 2
Andorra: 3
Austria: 2
Belarus: 4
Belgium: 1
Bosnia and Herzegovina: 1
```

```
Bulgaria: 1
Croatia: 3
Czech Republic: 4
Denmark: 1
Estonia: 2
Finland: 2
France: 2
Germany: 3
Greece: 4
Hungary: 1
Italy: 3
Kosovo: 1
Latvia: 1
Liechtenstein: 3
Lithuania: 2
Luxembourg: 4
Moldova: 1
Monaco: 1
Montenegro: 4
Netherlands: 4
North Macedonia: 3
Norway: 1
Poland: 1
Portugal: 2
Romania: 3
Russia: 3
San Marino: 1
Serbia: 2
Slovakia: 3
Slovenia: 4
Spain: 1
Sweden: 3
Switzerland: 1
Turkey: 2
Ukraine: 2
Vatican City: 1
4 colors
```



d

Statement

Find the minimum edge coloring $X : E \to \mathbb{N}$ of G

Definitions

A function $X : E \to \mathbb{N}$ is an edge coloring of G *iff* for any 2 adjacent edges $e_1, e_2 \in E : X(e_1) \neq X(e_2)$

An edge coloring is minimum *iff* $\max_{e \in E} X(e)$ is minimum

Answer

```
Albania - Greece: 2
Albania - Kosovo: 3
Albania - Montenegro: 9
Albania - North Macedonia: 1
Andorra - France: 8
Andorra - Spain: 9
Austria - Czech Republic: 1
Austria - Germany: 3
```

Austria - Hungary: 6
Austria - Italy: 2
Austria - Liechtenstein: 9
Austria - Slovakia: 4
Austria - Slovenia: 5
Austria - Switzerland: 7
Belarus - Latvia: 1
Belarus - Lithuania: 4
Belarus - Poland: 2
Belarus - Russia: 7
Belarus - Ukraine: 9
Belgium - France: 3
Belgium - Germany: 1
Belgium - Luxembourg: 9
Belgium - Netherlands: 2
Bosnia and Herzegovina - Croatia: 1
Bosnia and Herzegovina - Montenegro: 2
Bosnia and Herzegovina - Serbia: 4
Bulgaria - Greece: 1
Bulgaria - North Macedonia: 9
Bulgaria - Romania: 4
Bulgaria - Serbia: 3
Bulgaria - Turkey: 2
Croatia - Hungary: 3
Croatia - Montenegro: 5
Croatia - Serbia: 2
Croatia - Slovenia: 4
Czech Republic - Germany: 7
Czech Republic - Poland: 9
Czech Republic - Slovakia: 5
Denmark - Germany: 8
Estonia - Latvia: 9
Estonia - Russia: 1
Finland - Norway: 9
Finland - Russia: 6
Finland - Sweden: 2
France - Germany: 9
France - Italy: 5
France - Luxembourg: 6
France - Monaco: 7
France - Netherlands: 1
France - Spain: 2
France - Switzerland: 4
Germany - Luxembourg: 5
Germany - Netherlands: 4
Germany - Poland: 6
Germany - Switzerland: 2
Greece - North Macedonia: 3

```
Greece - Turkey: 9
Hungary - Romania: 2
Hungary - Serbia: 5
Hungary - Slovakia: 1
Hungary - Slovenia: 9
Hungary - Ukraine: 4
Italy - San Marino: 4
Italy - Slovenia: 3
Italy - Switzerland: 1
Italy - Vatican City: 9
Kosovo - Montenegro: 4
Kosovo - North Macedonia: 2
Kosovo - Serbia: 1
Latvia - Lithuania: 3
Latvia - Russia: 2
Liechtenstein - Switzerland: 3
Lithuania - Poland: 1
Lithuania - Russia: 9
Moldova - Romania: 9
Moldova - Ukraine: 1
Montenegro - Serbia: 6
North Macedonia - Serbia: 8
Norway - Russia: 5
Norway - Sweden: 1
Poland - Russia: 4
Poland - Slovakia: 3
Poland - Ukraine: 5
Portugal - Spain: 1
Romania - Serbia: 7
Romania - Ukraine: 6
Russia - Ukraine: 3
Slovakia - Ukraine: 2
9 colors
```

e

Statement

Find the maximum clique $Q \subseteq V$ of G

Definitions

$Q \subseteq V$ is a clique of G *iff* $\forall v, u \in Q : v = u \vee \{v, u\} \in E$

A clique is maximum by cardinality

Answer

```
['Lithuania', 'Belarus', 'Latvia', 'Russia']
4 vertices
```

f

Statement

Find the maximum stable set $S \subseteq V$ of G

Definitions

$S \subseteq V$ is a stable set of G *iff* $\forall v, u \in S : v = u \lor \{v, u\} \notin E$

A stable set is maximum by cardinality

Answer

```
['Moldova', 'Netherlands', 'Monaco', 'Bosnia and Herzegovina', 'Luxembourg',
'Vatican City', 'Sweden', 'Lithuania', 'Andorra', 'Albania', 'Czech Republic',
'Portugal', 'San Marino', 'Denmark', 'Slovenia', 'Estonia', 'Liechtenstein',
'Turkey']
18 vertices
```

g

Statement

Find the maximum matching $M \subseteq E$ of G

Definitions

$M \subseteq E$ is a matching of G *iff* $\forall e_1, e_2 \in M : e_1 = e_2 \lor (e_1 \cap e_2) = \emptyset$

A matching is maximum by cardinality

Answer

```
[('Slovakia', 'Ukraine'), ('Russia', 'Finland'), ('Poland', 'Czech Republic'),
('Lithuania', 'Belarus'), ('Montenegro', 'Kosovo'), ('Spain', 'Portugal'),
('Belgium', 'Luxembourg'), ('Norway', 'Sweden'), ('Andorra', 'France'),
('Albania', 'North Macedonia'), ('Latvia', 'Estonia'), ('Denmark', 'Germany'),
('Romania', 'Bulgaria'), ('Serbia', 'Bosnia and Herzegovina'), ('Croatia',
'Slovenia'), ('Liechtenstein', 'Switzerland'), ('Turkey', 'Greece'), ('Italy',
'San Marino'), ('Austria', 'Hungary')]
19 edges
```

h

Statement

Find the minimum vertex cover $R \subseteq V$ of G

Definitions

$R \subseteq V$ is a vertex cover of G *iff* $\forall e \in E : \exists v \in R : v \in e$

A vertex cover is minimum by cardinality

Answer

```
['Switzerland', 'Slovakia', 'Bulgaria', 'Norway', 'Germany', 'Montenegro',
'Belarus', 'Belgium', 'France', 'Poland', 'Kosovo', 'Italy', 'Latvia', 'Russia',
'Croatia', 'North Macedonia', 'Austria', 'Greece', 'Romania', 'Finland',
'Serbia', 'Spain', 'Ukraine', 'Hungary']
24 vertices
```

i

Statement

Find the minimum edge cover $F \subseteq E$ of G

Definitions

$F \subseteq E$ is an edge cover of G *iff* $\forall v \in V : \exists e \in F : v \in e$

An edge cover is minimum by cardinality

Explanation

We start with a maximum matching and add edges greedily to cover unmatched vertices

A proof of correctness can be found here

Answer

```
[('Slovakia', 'Ukraine'), ('Russia', 'Finland'), ('Poland', 'Czech Republic'),
('Lithuania', 'Belarus'), ('Montenegro', 'Kosovo'), ('Spain', 'Portugal'),
('Belgium', 'Luxembourg'), ('Norway', 'Sweden'), ('Andorra', 'France'),
('Albania', 'North Macedonia'), ('Latvia', 'Estonia'), ('Denmark', 'Germany'),
('Romania', 'Bulgaria'), ('Serbia', 'Bosnia and Herzegovina'), ('Croatia',
'Slovenia'), ('Liechtenstein', 'Switzerland'), ('Turkey', 'Greece'), ('Italy',
'San Marino'), ('Austria', 'Hungary'), ('Belgium', 'Netherlands'), ('France',
'Monaco'), ('Italy', 'Vatican City'), ('Moldova', 'Romania')]
23 edges
```

j

Statement

Find the shortest closed walk W that visits every vertex of G

Definitions

A hamiltonian cycle of G is a cycle that contains all vertices.

The traveling salesman problem is the problem of finding the hamiltonian cycle of G of minimum length (in the case of a weigted graph length is defined as the total weight of edges).

The distance graph of G is a complete graph with the same vertice set, in which each edge has a weight equal to the distance between its endpoints in G.

Explanation

The problem can be reduced to the traveling salesman problem on the distance graph of G.

After obtaining the minimum hamiltonian cycle of the distance graph, we replace every edge in it with the shortest path between its endpoints in G.

Here we use an approximate algorithm because the traveling salesman problem is apparently already too hard for my pc even with this relatively small graph. the commented cell contains the exact evaluation that didn't finish in an hour for me

Answer

```
['Austria', 'Liechtenstein', 'Switzerland', 'Germany', 'Denmark', 'Germany',
'Luxembourg', 'Belgium', 'Netherlands', 'France', 'Monaco', 'France', 'Andorra',
'Spain', 'Portugal', 'Spain', 'France', 'Italy', 'San Marino', 'Italy', 'Vatican
City', 'Italy', 'Slovenia', 'Croatia', 'Serbia', 'Bosnia and Herzegovina',
'Montenegro', 'Kosovo', 'North Macedonia', 'Albania', 'Greece', 'Turkey',
'Bulgaria', 'Romania', 'Moldova', 'Ukraine', 'Belarus', 'Lithuania', 'Latvia',
'Estonia', 'Russia', 'Finland', 'Sweden', 'Norway', 'Russia', 'Poland', 'Czech
Republic', 'Slovakia', 'Hungary', 'Austria']
49 edges
```

k

Statement

Find the shortest closed walk U that visits every edge of G

Definitions

A eulerian circuit is a ciruit that visits each edge exactly once

Explanation

In order solve this problem we can add parallel edges to the graph and then find a eulerian curcuit in the produced multigraph. Essentially we're reframing our question by saying that visiting the same edge several times is the same as visiting each of several parallel edges exactly once. The problem then becomes that of adding optimal parallel edges, which thankfully networkx is capable of.

Answer

```
['Moldova', 'Ukraine', 'Romania', 'Ukraine', 'Russia', 'Latvia', 'Estonia',
'Russia', 'Poland', 'Ukraine', 'Hungary', 'Serbia', 'Romania', 'Hungary',
'Croatia', 'Hungary', 'Austria', 'Slovenia', 'Hungary', 'Slovakia', 'Ukraine',
'Belarus', 'Poland', 'Belarus', 'Russia', 'Finland', 'Sweden', 'Norway',
'Finland', 'Norway', 'Russia', 'Lithuania', 'Belarus', 'Latvia', 'Lithuania',
'Poland', 'Czech Republic', 'Austria', 'Germany', 'Denmark', 'Germany',
'Poland', 'Slovakia', 'Austria', 'Italy', 'San Marino', 'Italy', 'Vatican City',
'Italy', 'France', 'Spain', 'Portugal', 'Spain', 'Andorra', 'France', 'Belgium',
'Germany', 'France', 'Monaco', 'France', 'Luxembourg', 'Germany', 'Netherlands',
'France', 'Luxembourg', 'Belgium', 'Netherlands', 'France', 'Switzerland',
'Germany', 'Czech Republic', 'Slovakia', 'Austria', 'Switzerland', 'Austria',
'Liechtenstein', 'Switzerland', 'Italy', 'Slovenia', 'Croatia', 'Serbia', 'North
Macedonia', 'Greece', 'Albania', 'Kosovo', 'Serbia', 'Montenegro', 'Kosovo',
'North Macedonia', 'Albania', 'Montenegro', 'Croatia', 'Bosnia and Herzegovina',
'Montenegro', 'Bosnia and Herzegovina', 'Serbia', 'Bulgaria', 'Greece',
'Turkey', 'Bulgaria', 'North Macedonia', 'Bulgaria', 'Romania', 'Moldova']
103 edges
```

l

Find all 2-vertex-connected components (blocks) and draw a block-cut tree of G^*

Definitions

A cut vertex is a vertex the removal of which splits the graph, increasing its number of connected components

A graph is 2-vertex-connected *iff* it contains no cut vertices.

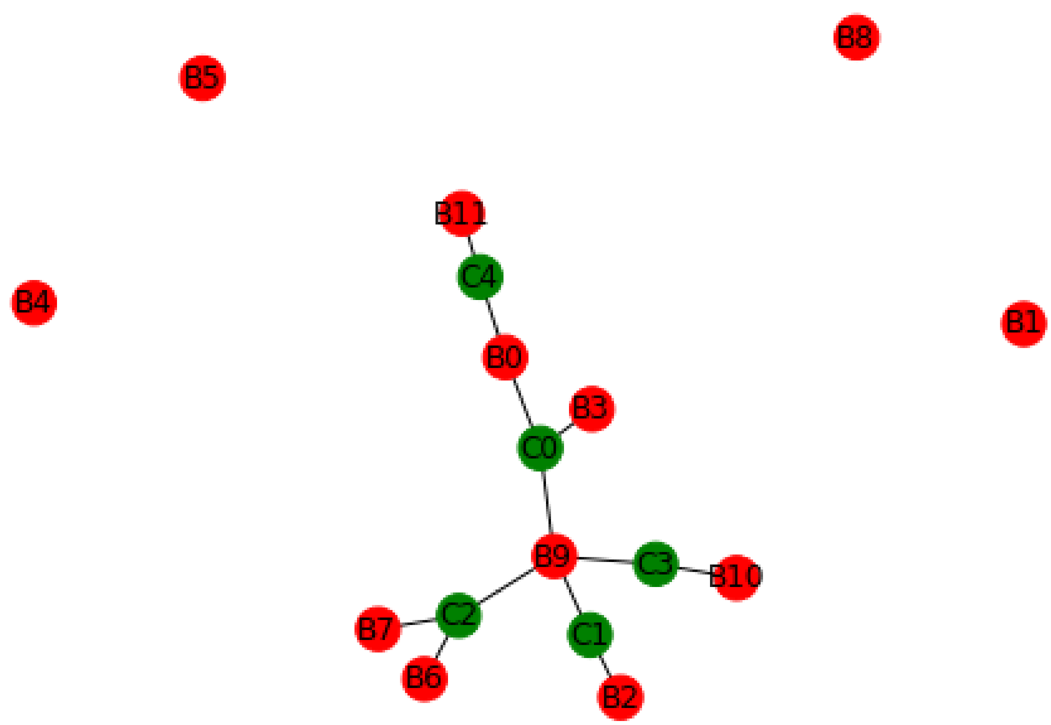
A block of G is a 2-vertex-connected subgraph of G that is maximal by inclusion.

A block-cut tree of G is a tree which has a vertex for each block and each cut vertex of G and a edge connecting each pair of vertices corresponding to a cut vertex and a block for which the vertex is in the block

Answer

```
B0: ('Andorra', 'France', 'Spain')
B1: ('Cyprus',)
B2: ('Denmark', 'Germany')
```

```
B3: ('France', 'Monaco')
B4: ('Iceland',)
B5: ('Ireland', 'United Kingdom')
B6: ('Italy', 'San Marino')
B7: ('Italy', 'Vatican City')
B8: ('Malta',)
B9: ('Moldova', 'Romania', 'Ukraine', 'Liechtenstein', 'Switzerland', 'Austria',
'France', 'Germany', 'Italy', 'Belgium', 'Netherlands', 'Bosnia and
Herzegovina', 'Montenegro', 'Croatia', 'Serbia', 'Czech Republic', 'Slovakia',
'Poland', 'Hungary', 'Bulgaria', 'North Macedonia', 'Turkey', 'Greece',
'Luxembourg', 'Latvia', 'Lithuania', 'Belarus', 'Russia', 'Albania', 'Kosovo',
'Slovenia', 'Estonia')
B10: ('Norway', 'Russia', 'Finland', 'Sweden')
B11: ('Portugal', 'Spain')
C0: France
C1: Germany
C2: Italy
C3: Russia
C4: Spain
```



m

Statement

Find all 2-edge-connected components of G^*

Definitions

A bridge is an edge the removal of which splits the graph, increasing its number of connected components

A graph is 2-edge-connected *iff* it contains no bridges

A 2-edge-connected component of G is a 2-edge-connected subgraph of G that is maximal by inclusion

Answer

```
['United Kingdom']
['Moldova', 'Switzerland', 'Netherlands', 'Slovakia', 'Bosnia and Herzegovina',
'Bulgaria', 'Norway', 'Luxembourg', 'Germany', 'Sweden', 'Lithuania', 'Andorra',
'Montenegro', 'Belarus', 'Czech Republic', 'Albania', 'Belgium', 'France',
'Poland', 'Kosovo', 'Italy', 'Slovenia', 'Estonia', 'Latvia', 'Russia',
'Croatia', 'Turkey', 'North Macedonia', 'Liechtenstein', 'Austria', 'Serbia',
'Romania', 'Finland', 'Greece', 'Spain', 'Ukraine', 'Hungary']
['Monaco']
['Vatican City']
['Malta']
['Ireland']
['Portugal']
['Cyprus']
['San Marino']
['Denmark']
['Iceland']
```

n

Statement

Construct an SPQR tree of the largest biconnected component of G

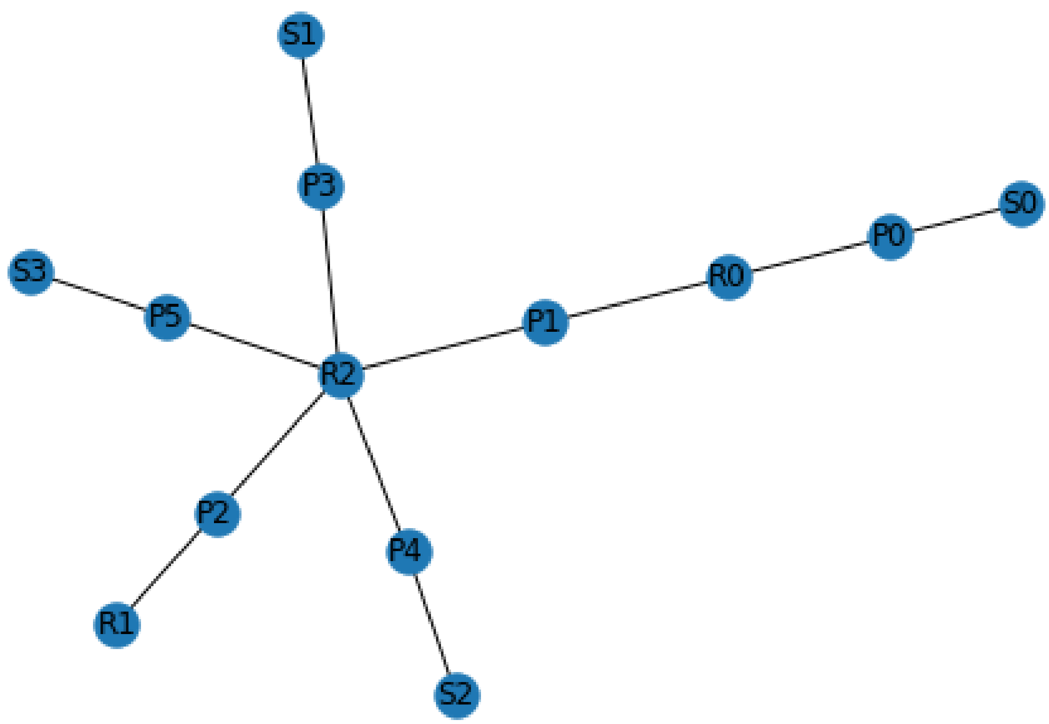
Definitions

An SPQR tree is a tree representing the triconnected components of a biconnected graph, the exact structure of which is too long to explain here and can be found on Wikipedia

Answer

```
S0: ['Estonia', 'Latvia', 'Russia']
S1: ['Austria', 'Liechtenstein', 'Switzerland']
```

```
S2: ['Bulgaria', 'Greece', 'Turkey']
S3: ['Moldova', 'Romania', 'Ukraine']
P0: ['Latvia', 'Russia']
P1: ['Poland', 'Ukraine']
P2: ['France', 'Germany']
P3: ['Austria', 'Switzerland']
P4: ['Bulgaria', 'Greece']
P5: ['Romania', 'Ukraine']
R0: ['Belarus', 'Latvia', 'Lithuania', 'Poland', 'Russia', 'Ukraine']
R1: ['Belgium', 'France', 'Germany', 'Luxembourg', 'Netherlands']
R2: ['Albania', 'Austria', 'Bosnia and Herzegovina', 'Bulgaria', 'Croatia',
'Czech Republic', 'France', 'Germany', 'Greece', 'Hungary', 'Italy', 'Kosovo',
'Montenegro', 'North Macedonia', 'Poland', 'Romania', 'Serbia', 'Slovakia',
'Slovenia', 'Switzerland', 'Ukraine']
```



o

Statement

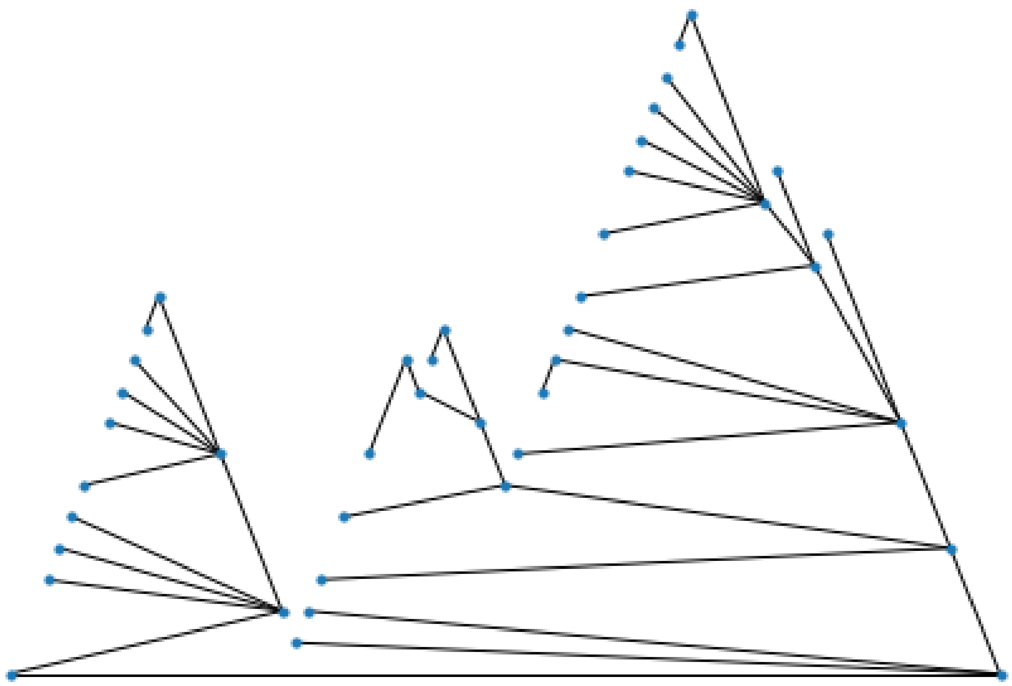
Find the minimum (w.r.t. the total weight of edges) spanning tree T for the maximum connected component of the weighted version of G

Definition

A spanning tree of G is a tree with the same vertex set as G

Answer

`22164.45282255872`



p

Statement

Find the $centroid(T)$ (w.r.t. the edge weight function).

Definitions

A branch of a tree T starting at one of its vertices v is a subtree of T in which v has exactly one neighbor.

The centroid of a tree (w.r.t. edge weight) is the set of vertices with minimum weight, where we define the weight of a vertex to be the maximum total edge weight of a branch starting at that vertex

Answer

```
['Hungary']
```

q

Statement

Construct the Prüfer code for T

Definitions

The Prüfer code of a tree labeled with integers $\{1, 2, ..., n\}$ is a sequence of $n-2$ labels that uniquely identifies that tree.

Answer

```
[12, 40, 12, 7, 30, 2, 13, 31, 2, 31, 2, 31, 12, 12, 12, 17, 24, 31, 40, 36, 16,
30, 40, 15, 12, 16, 11, 31, 40, 2, 14, 0, 24, 7, 15, 22, 30, 15, 2, 16]
```