Домашняя работа №5

Год: 2021

Предмет: Дискретная математика

Группа: М3101

Выполнил: Сухов Владимир

Принял: Чухарев Константин

Задание 1

Граф Европы $\mathbf{E}=\langle V, E\rangle$ определяется следующим образом: каждая вершина $v \in V$ —страна Европы; две вершины смежные $\{u, v\} \in E$, если соответствующие страны граничат.

Все используемые алгоритмы и файлы (с подробным и не очень описанием) будут лежать тут:

https://github.com/babtiss/ITMO_DM/tree/main/1%20course/Homework_1_GraphTheory

a) Нарисуйте граф Европы **E** и покажите, что он плоский.

Вершины были пронумерованы в порядке заданном в файле europe.txt .

Плоский граф – граф, изображенный на плоскости так, что его вершины – точки на этой плоскости, а рёбра – непересекающиеся кривые на ней.

Как видно, изображенный граф **E** удовлетворяет условиям плоского графа, а, значит, он является плоским. (Проверено на сайте https://graphonline.ru/)

b) Найдите: $|\boldsymbol{V}|$, $|\mathbf{E}|$, $\delta(\mathbf{G})$, $\Delta(\mathbf{G})$, rad($\mathbf{G}$), diam($\mathbf{G}$), girth($\mathbf{G}$), center($\mathbf{G}$), $\boldsymbol{\lambda}(\mathbf{G})$, $\boldsymbol{\kappa}(\mathbf{G})$.

Для графа $\mathbf{E} = <\mathbf{V}, \mathbf{E}>$:

$|\boldsymbol{V}| = 42$ – Общее количество вершин графа

$|\mathbf{E}| = 87$ – общее количество ребер графа

Алгоритм : с помощью алгоритма Флойда – Уоршелла была построена матрица кратчайших расстояний от каждой вершины до каждой, после чего для всех вершин был найден ее эксцентриситет, равный наибольшему расстоянию от данной вершины до любой другой, были найдены радиус и диаметр графа как минимум и максимум строчки, соответствующей данной вершин в матрице расстояний. В центр графа были добавлены номера вершин, чей эксцентриситет равен радиусу графа.

Степень вершин графа (deg (V_i)) – количество ребер графа, инцидентны данной вершине.

$\deg(V_i) = |\{e|e\ I\ \mathbf{V}_i\}|$ I – “инцидентность”

$\delta(\mathbf{G}) = \min(\deg(\vartheta))$ $\delta(\mathbf{G}) = 1$

$\vartheta \in \boldsymbol{V}$

$\Delta(\mathbf{G}) = \max(\deg(\vartheta))$ $\Delta(\mathbf{G}) = 9$

$\vartheta \in \boldsymbol{V}$

(Нахождение минимальной и максимальной степени вершины графа)

Эксцентриситет вершины графа – расстояние от данной до самой дальней вершины графа.

$$\varepsilon(\vartheta) = \max_{\upsilon \in \boldsymbol{V}} \mathrm{dist}(\vartheta, \upsilon)$$

Радиус графа – минимальный эксцентриситет среди всех вершин графа.

$$rad(\mathbf{G}) = \min_{\upsilon \in \boldsymbol{V}} \varepsilon(\upsilon)$$

Диаметр графа –максимальный эксцентриситет среди всех вершин графа.

$$diam(\mathbf{G}) = \max_{\upsilon \in \boldsymbol{V}} \varepsilon(\upsilon)$$

Центр графа – множество, состоящее из вершин, эксцентриситет которых равен радиусу графа.

$$\text{center}(\mathbf{G}) = \{v \mid \varepsilon(v) \ = \ rad(\mathbf{G}) \wedge v \in \boldsymbol{V}\}$$

Для данного графа **G:**

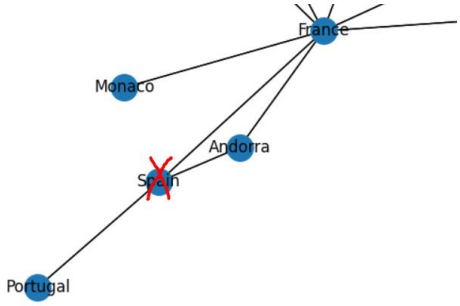
$$rad(\mathbf{G}) \ = \ 5$$

$$diam(\mathbf{G}) \ = 8$$

$\text{center}(\mathbf{G}) =$ ['Austria', 'Belarus', 'Croatia', 'CzechRepublic', 'Germany', 'Hungray', 'Poland', 'Russia', 'Slovakia', 'Slovenia', 'Switzerland', 'Ukraine']

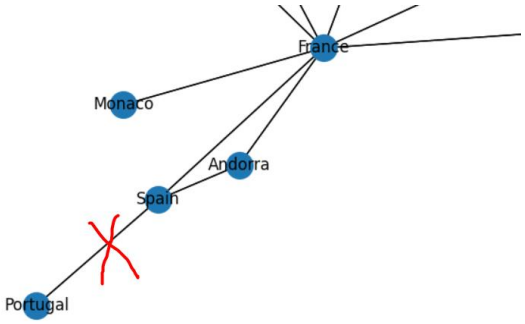
$\text{girth}(\mathbf{G}) = 3$ (2 не может быть, т.к. нет кратных рёбер; 1 не может быть, т.к. нет петель).

Вершинной связностью κ графа **G** называется наименьшее число вершин, удаление которых приводит к несвязному или тривиальному графу.



Удаляем Испанию (удалили шарнир) . Граф становится несвязным. κ**=1.**

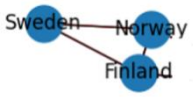
Реберной связностью λ графа **G** называется наименьшее количество ребер, удаление которых приводит к несвязному или тривиальному графу.



Отрезаем Португалию от Испании (удалили мост). Граф становится несвязным. λ =1.

c) Найдите наименьшую вершинную раскраску Z графа **G:**

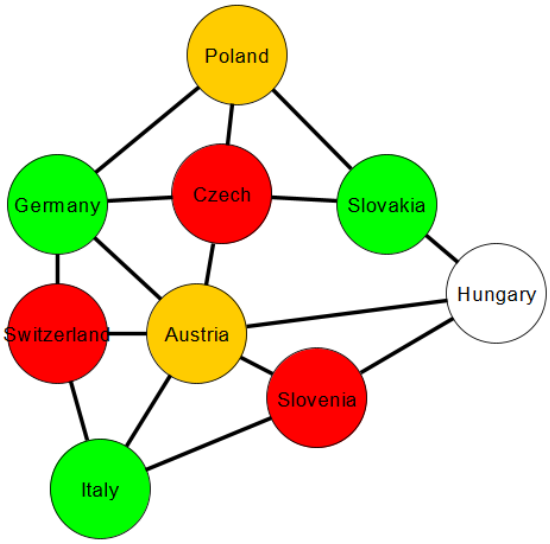
Вершинная раскраска – частный случай разметки графа, при котором каждой вершине ставится в соответствие цвет так, что любые две смежные вершины имеют разные цвета.



Данный подграф исходного графа – полный граф. А значит минимальное кол–во цветов для построения вершинной раскраски данного подграфа ≥ 3.

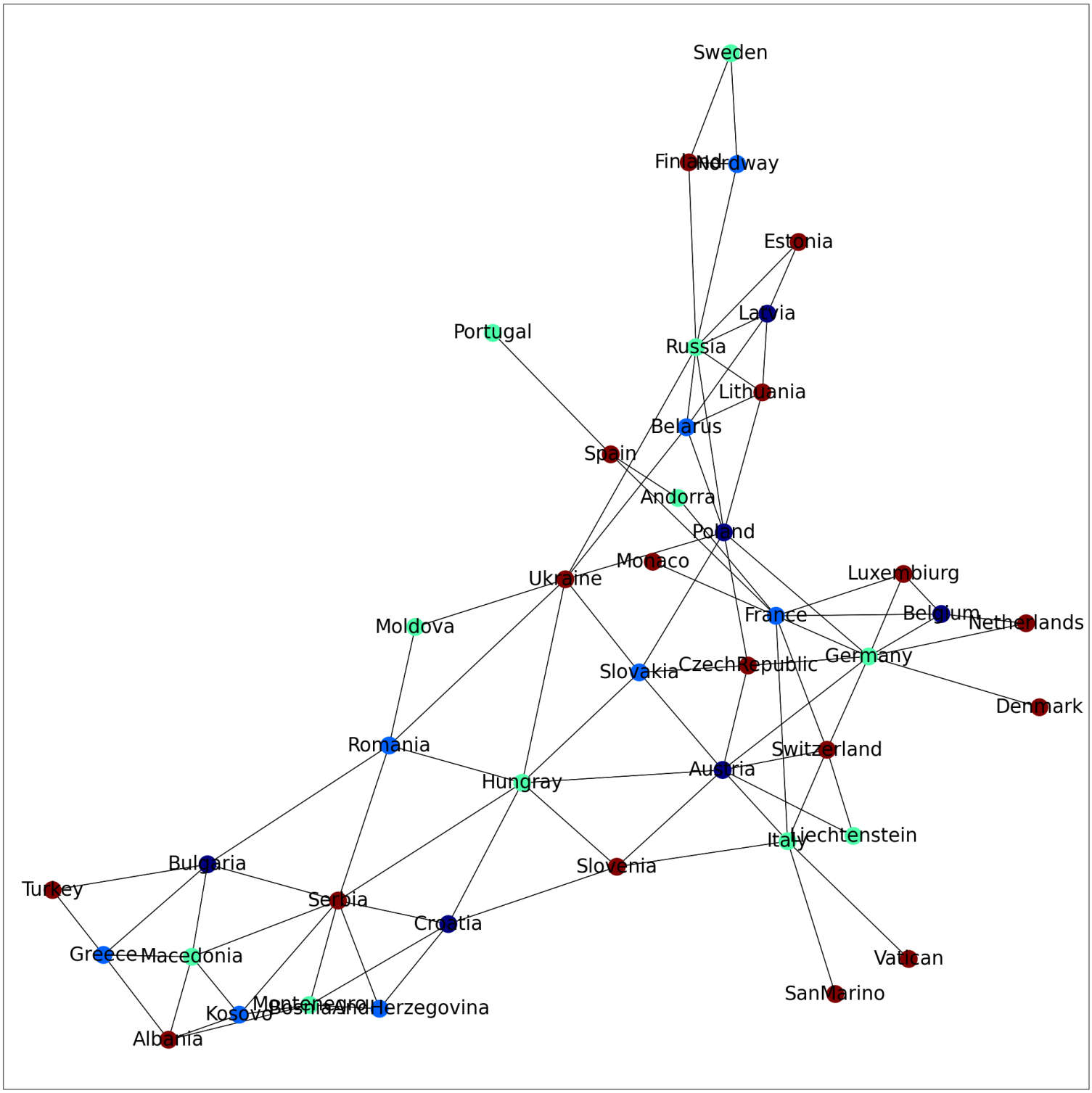
Докажем, что минимальное кол–во цветов для построение вершинной раскраски данного графа $= 4$.

1) Рассмотрим следующий подграф графа **G** и попробуем покрасить его в 3 цвета:



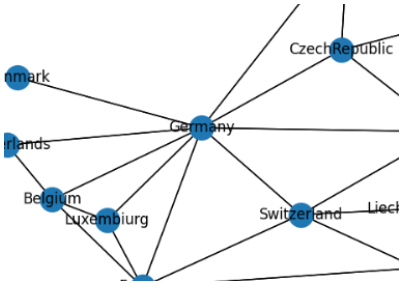
Вершина “Hunagary” осталась неокрашенной, так как она смежна одновременно с тремя вершинами различного цвета, следовательно, ей нельзя присвоить ни один из 3-ёх данных цветов. Вершины можно попробовать покрасить в любые 3 цвета, но результат будет аналогичен данному всегда, т.к удалив вершину “Hunagary”, получим подграф, в котором любые 3 попарно смежные вершины образуют полный граф, а значит для таких K_3 цвет вершин, не являющихся общими для обоих графов, определяется единственным образом.

2) Попробуем окрасить граф в 4 цвета (с помощью алгоритма).
3) Как видно на картинке ниже, это возможно. Тогда ниже представлена минимальная вершинная раскраска Z графа **G.**


Sweden
Finland
Norway
Estonia
Latvia
Portugal
Russia
Lithuania
Belarus
Spain
Andorra
Poland
Monaco
Ukraine
Luxembiurg
Belgium
Netherlands
France
Moldova
Germany
Slovakia
CzechRepublic
Denmark
Romania
Switzerland
Austria
Hungray
Liechtenstein
Italy
Bulgaria
Slovenia
Turkey
Serbia
Croatia
Greece
Macedonia
Vatican
SanMarino
Montenegro
Kosovo
BosniaandHerzegovina
Albania

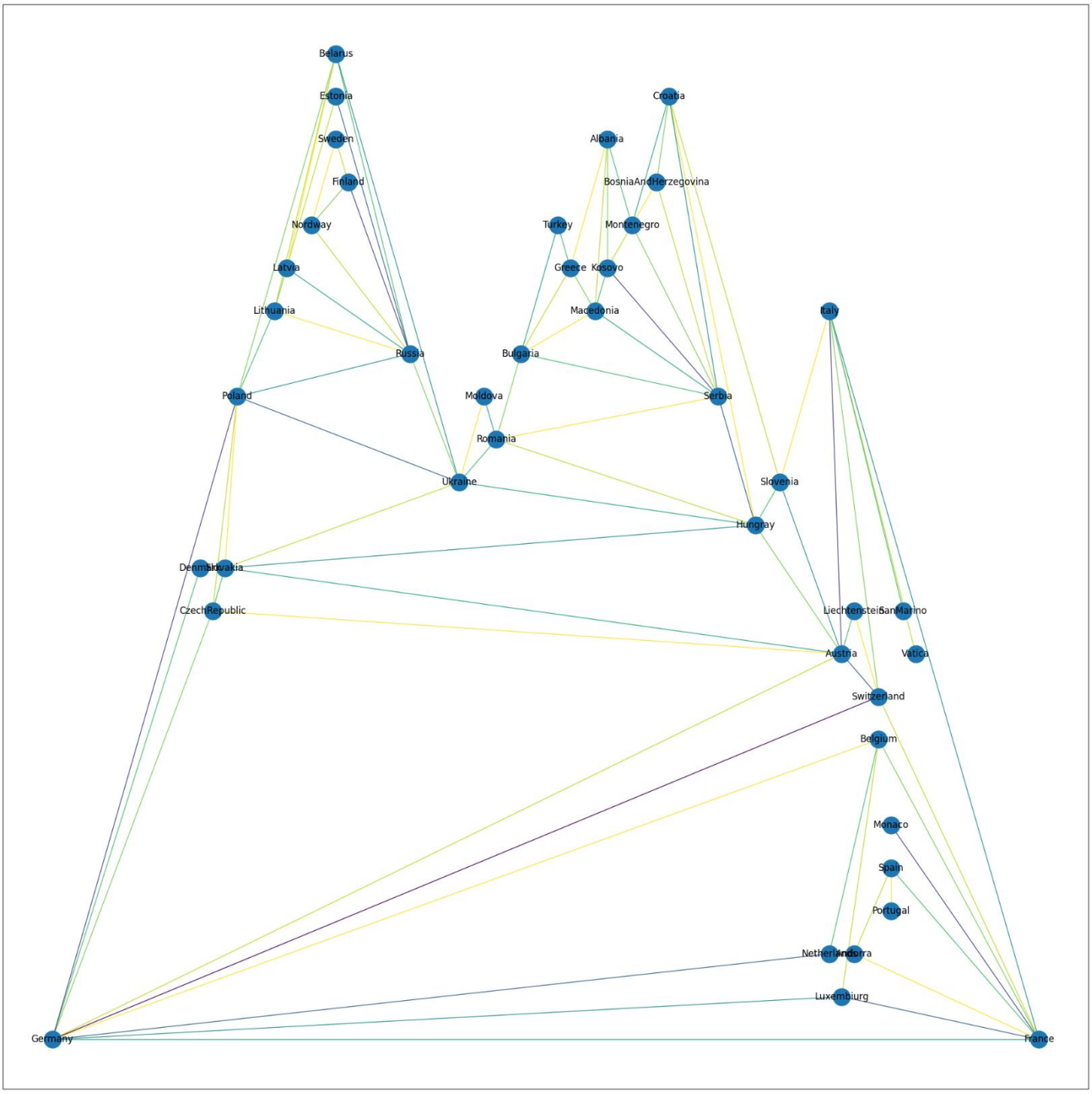
d) Найти минимальную рёберную раскраску X в графе **G.**

Рёберная раскраска — назначение цветов рёбрам графа таким образом, что никакие два смежных ребра не имеют один и тот же цвет.



Рассмотрим следующий фрагмент графа. А именно, вершину “Germany”. Степень данной вершины = 9. А значит раскраска имеет ≥9 «цветов». Покажем, что минимальная рёберная раскраска имеет 9 «цветов».

(Если плохо видно – можно проверить какой цвет у ребра в программе)



e) Найти максимальную клику Q графа **G.**

Кликой в неориентированном графе **G** = (V, E) называется подмножество вершин C ⊆ V такое что для любых двух вершин в C существует ребро, их соединяющее. Это эквивалентно утверждению, что подграф, порождённый C, является полным.

Максимальная клика — это клика, которая не может быть расширена путём включения дополнительных смежных вершин, то есть нет клики большего размера, включающей все вершины данной клики.

```
Clique  1 ['Montenegro', 'Serbia', 'BosniaAndHerzegovina', 'Croatia']
Clique  2 ['Russia', 'Belarus', 'Ukraine', 'Poland']
Clique  3 ['Russia', 'Belarus', 'Lithuania', 'Latvia']
Clique  4 ['Russia', 'Belarus', 'Lithuania', 'Poland']
Clique  5 ['Germany', 'Belgium', 'France', 'Luxembiurg']
```

В данном графе 5 клик из 4-ёх вершин.

Нет ни одной клики, состоящей из 5-ти вершин.

f) Найдите наибольшее независимое множество S графа **G**.

Множество вершин графа **G** = <V, E> называется **независимым**, если никакие две вершины этого множества не соединены ребром.

Независимое множество вершин графа ε называется **максимальным независимым множеством** вершин графа **G** = <V, E>, если не существует такого независимого множества ϑ графа **G**, что $\varepsilon \subseteq \vartheta$.

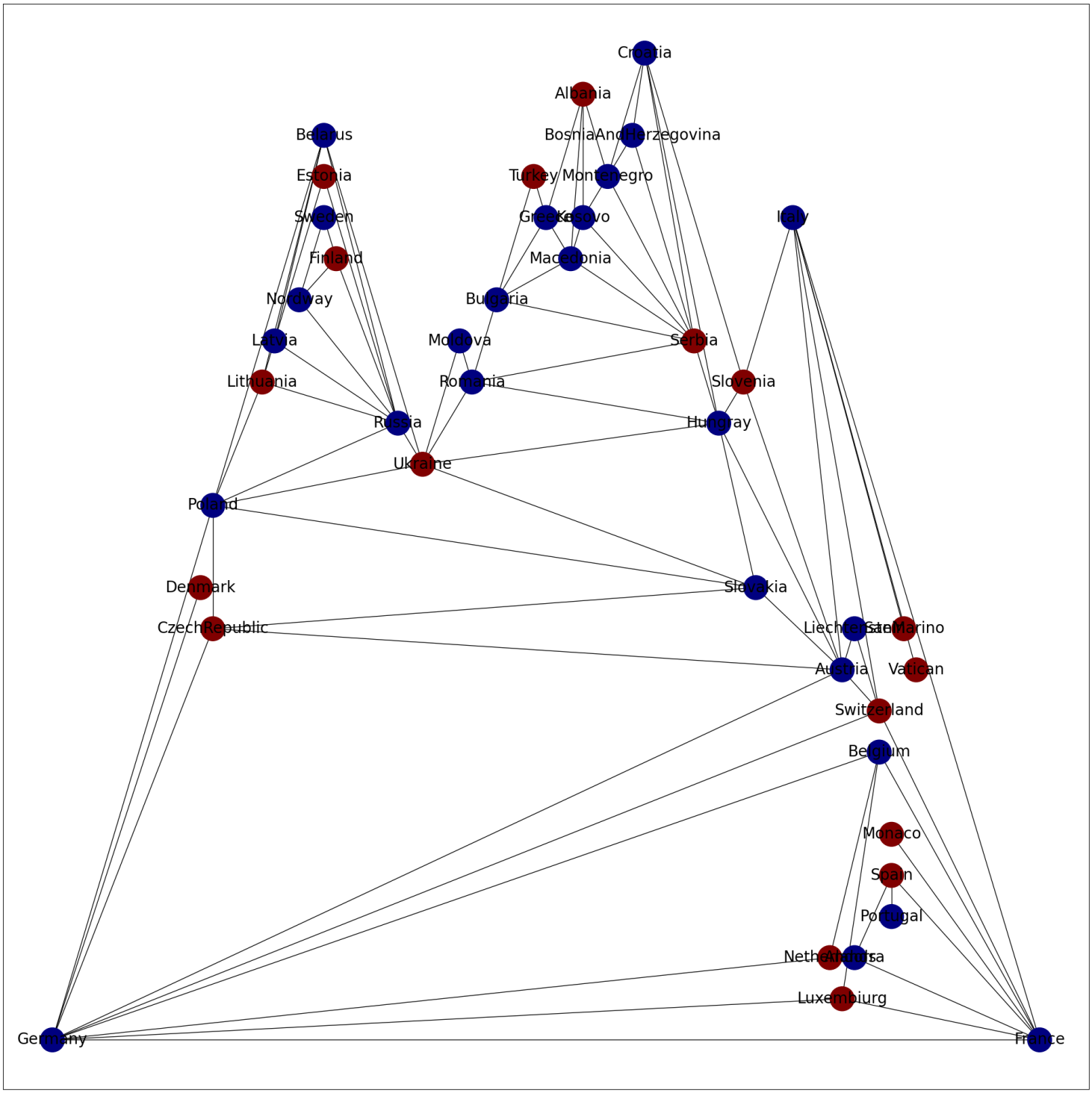
Максимальное независимое множество вершин графа **G** = <V, E> называется **наибольшим независимым множеством** этого графа, которое содержит в себе максимальное число вершин.

Для решения задачи был написан алгоритм, раскрашивающий вершины в красный цвет, если это возможно (так называемый жадный алгоритм).

S = {Albania, CzechRepublic, Denmark, Estonia, Finland, Lithuania, Luxembiurg, Monaco, Netherlands, SanMarino, Serbia, Slovenia, Spain, Switzerland, Turkey, Ukraine, Vatica}

|S| = 17

Изобразим исходный в граф, в котором все $\vartheta \subseteq S$ окрашены в красный цвет.


Croatia
Albania
Belarus
BosniaAndHerzegovina
Estonia
Turkey
Montenegro
Sweden
Greece
Kosovo
Italy
Finland
Macedonia
Norway
Bulgaria
Latvia
Moldova
Serbia
Lithuania
Romania
Slovenia
Russia
Hungray
Ukraine
Poland
Denmark
Slovakia
CzechRepublic
Liechtenstein
SanMarino
Austria
Vatican
Switzerland
Belgium
Monaco
Spain
Portugal
Netherlands
Andorra
Luxembiurg
Germany
France

g) Найдите наибольшее паросочетание M графа **G** и докажите, что оно максимально.

Паросочетанием графа **G** = <V, E> называется множество $M \subseteq E$, такое, что любые 2 ребра из этого множества не имеют общих вершин, то есть не являются смежными.

Максимальным паросочетанием M в графе **G** = <V, E> называется такое паросочетание, которое не содержится ни в каком другом паросочетании этого графа.

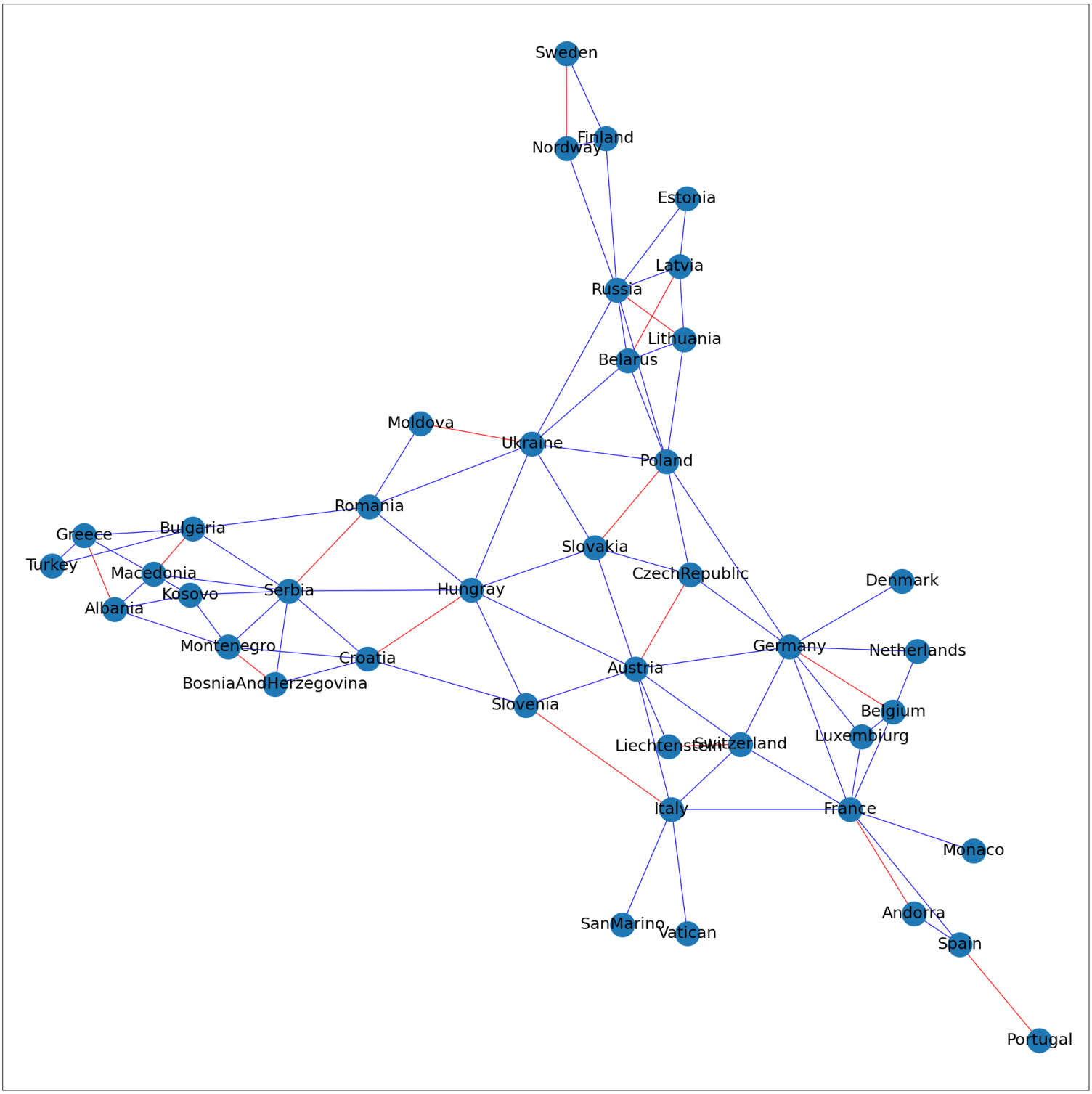
Наибольшим паросочетанием M в графе **G** = <V, E> называется такое максимальное паросочетание, которое содержит в себе максимальное количество ребер.

Ниже представлен результат работы алгоритма:

M = {Slovakia – Poland, Lithuania – Russia, Austria – CzechRepublic, Spain – Portugal, Germany – Belgium, Macedonia – Bulgaria, Romania – Serbia, BosniaAndHerzegovina - Montenegro, Hungray – Croatia, Liechtenstein - Switzerland, Albania – Greece, Slovenia – Italy, Andorra – France, Belarus – Latvia, Nordway – Sweden, Ukraine – Moldova}

|M| = 16

Изобразим исходный в граф, в котором все рёбра $e \subseteq M$ окрашены в красный цвет.


Sweden
Finland
Nordway
Estonia
Latvia
Russia
Lithuania
Belarus
Moldova
Ukraine
Poland
Romania
Greece
Bulgaria
Turkey
Macedonia
Kosovo
Albania
Serbia
Hungray
Slovakia
CzechRepublic
Denmark
Montenegro
Croatia
BosniaAndHerzegovina
Austria
Germany
Netherlands
Slovenia
Belgium
Luxembiurg
Liechtenstein
Switzerland
Italy
France
Monaco
SanMarino
Vatican
Andorra
Spain
Portugal

h) Найдите наименьшее вершинное покрытие R графа **G** и докажите, что оно минимально.

Вершинное покрытие графа **G** = <V, E> является таким множеством $R \subseteq V$, что любое ребро $\varepsilon \in E$ инцидентно хотя бы одной вершине $v \in R$.

Вершинное покрытие R графа **G** = <V, E> является **минимальных вершинным покрытием** графа **G** = <V, E>, если не существует такого множества $U \subseteq V$, что $U \subseteq R$

Минимальное вершинное покрытие R графа **G** = <V, E> является **наименьшим вершинным покрытием** графа **G** = <V, E>, если оно содержит минимальное количество вершин.

Для решения данной задачи воспользуемся следующей теоремой:

*Множество независимо тогда и только тогда, когда его дополнение является вершинным покрытием.

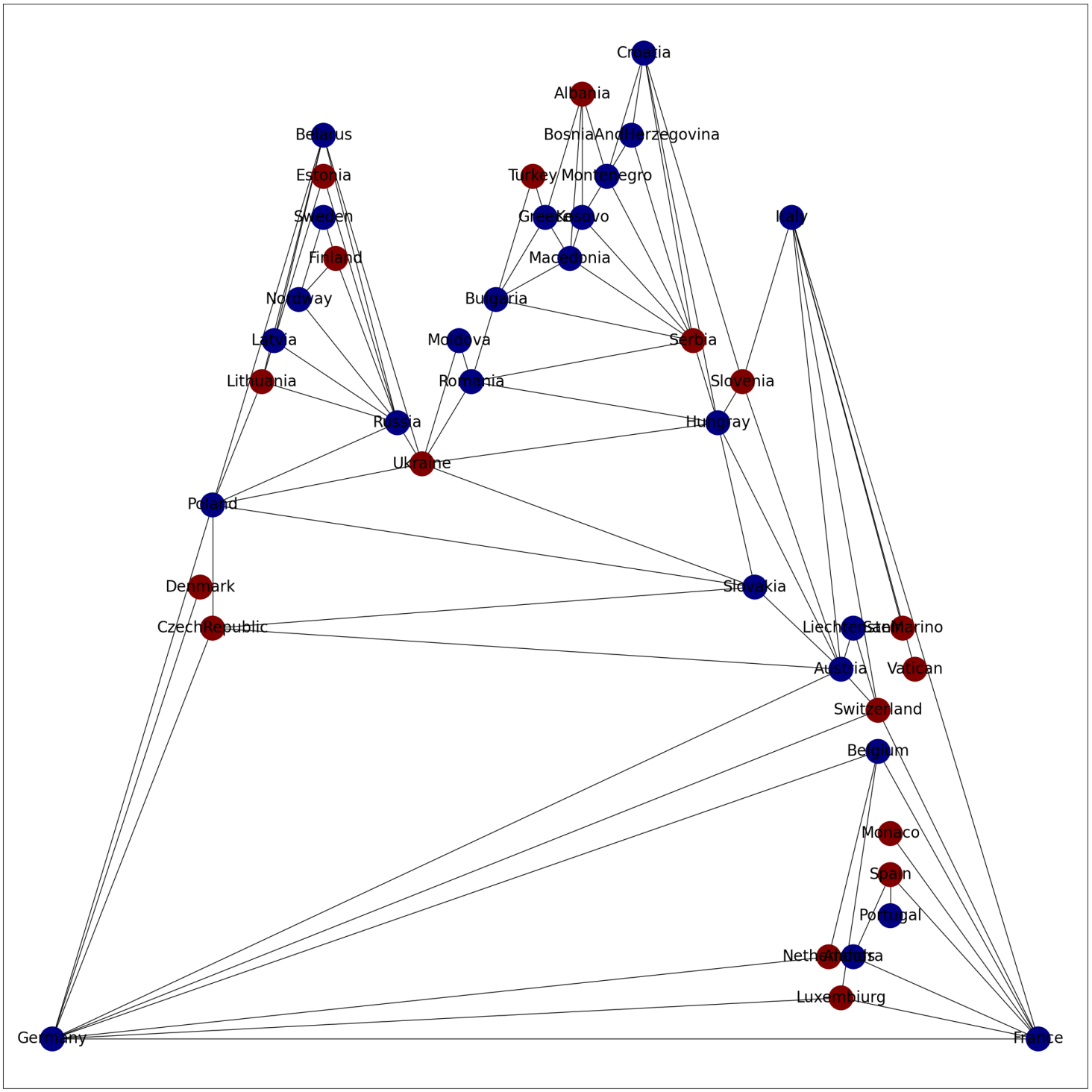
Воспользуемся следствием из этой теоремы:

*Независимое множество является наибольшим тогда и только тогда, когда его дополнение является наименьшим вершинным покрытием.

Из этого следует, что **наименьшим вершинным покрытием** R графа **E** является дополнение множества S (пункт **f**).

R = S = {CzechRepublic, Denmark, Estonia, Finland, Lithuania, Luxembiurg, Monaco, Netherlands, SanMarino, Serbia, Slovenia, Spain, Switzerland, Turkey, Ukraine, Vatica, Albania} |R| = 17.

Изобразим граф, в котором все $\vartheta \subseteq R$ окрашены в красный цвет.


Croatia
Albania
Belarus
BosniaAndHerzegovina
Estonia
Turkey
Montenegro
Sweden
Greece
Kosovo
Italy
Finland
Macedonia
Norway
Bulgaria
Latvia
Moldova
Serbia
Lithuania
Romania
Slovenia
Russia
Hungray
Ukraine
Poland
Denmark
Slovakia
CzechRepublic
Liechtenstein
SanMarino
Austria
Vatican
Switzerland
Belgium
Monaco
Spain
Portugal
Netherlands
Luxembiurg
Germany
France

i) Найдите наименьшее рёберное покрытие F графа **G** и докажите, что оно минимально.

Рёберное покрытие графа **G** = <V, E> является таким множеством $F \subseteq E$, что любая вершина $\vartheta \in V$ инцидентно хотя бы одному ребру $\varepsilon \in F$.

Рёберное покрытие F графа **G** = <V, E> является **минимальных рёберным покрытием** графа **G** = <V, E>, если не существует такого множества $K \subseteq E$, что $K \subseteq F$

Минимальное рёберное покрытие F графа **G** = <V, E> является **наименьшим рёберным покрытием** графа **G** = <V, E>, если оно содержит минимальное количество рёбер.

Для решения данной задачи следует найти наибольшее паросочетание в графе **E** и включить в это наибольшее паросочетание рёбра, необходимые для покрытия непокрытых паросочетанием вершин.

В пункте **g)** было найдено наибольшее паросочетание, но оно не покрыло вершины (Denmark, Estonia, Finland, Kosovo, Luxembiurg, Monaco, Netherlands, SanMarino, Turkey, Vatican)

Следуя алгоритму, мы должны добавить множеству F ребра:

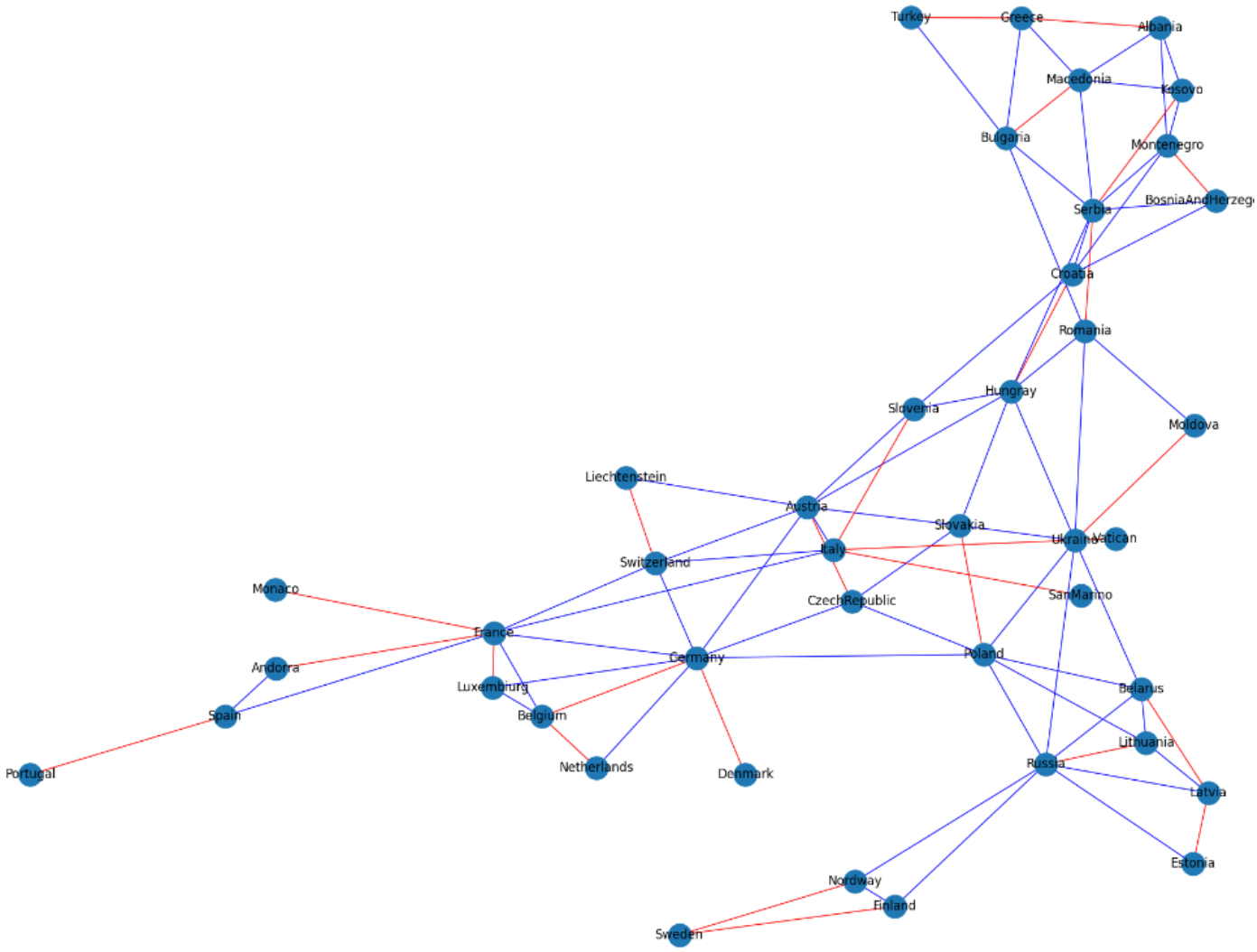
Denmark - Germany	Luxembiurg - France	Turkey - Greece
Estonia - Latvia	Monaco - France	Vatican - Italy
Finland - Sweden	Netherlands - Belgium	
Kosovo - Serbia	SanMarino - Italy	

После чего получившееся множество $F \subseteq E$ будет являться **наименьшим реберным покрытием** графа **G.**

Тогда F = {

Slovakia - Poland	Liechtenstein - Switzerland	Finland - Sweden
Lithuania - Russia	Albania - Greece	Kosovo - Serbia
Austria - CzechRepublic	Slovenia - Italy	Luxembiurg - France
Spain - Portugal	Andorra - France	Monaco - France
Germany - Belgium	Belarus - Latvia	Netherlands - Belgium
Macedonia - Bulgaria	Nordway - Sweden	SanMarino - Italy
Romania - Serbia	Ukraine - Moldova	Turkey - Greece
BosniaAndHerzegovina - Montenegro	Denmark - Germany	Vatican – Italy
Hungray - Croatia	Estonia - Latvia	}

Изобразим граф, в котором все рёбра $e \subseteq F$ окрашены в красный цвет.



j) Найдите кратчайший замкнутый путь W, содержащий все вершины графа **G.**

Перебрал ручками:

W = {Sweden – Nordway – Russia – Poland – CzechRepublic – Germany – Denmark – Germany – Netherlands – Belgium – Luxembiurg – France – Spain – Portugal – Spain – Andorra – France – Monaco – France – Switzerland – Liechtenstein – Austia – Italy- Sanmarini – Italy - Slovenia – Hungray – Croatia -Bulgaria -Turkey -Greece – Macedonia – Albania – Kosovo – Montenegro – BosniaAndHerzeg – Serbia – Romania – Moldovia – Ucraina – Vatican – Ucraina – Belarus – Lithuania – Latvia – Estonia – Russia – Finland – Sweden }.

Замкнутый путь проходит через все вершины графа. Кроме того, путь проходит через каждую вершину ровно один раз, кроме вершин – шарниров.

Очевидно, что если путь графа лежит из одной компоненты вершинной двусвязности в другую, то возвращается такой путь через точку сочленения (шарнир), значит он обязательно посетит такую вершину 2 раза. Остальные вершины мы посетили по одному разу , следовательно данный путь – кратчайший.

k) Найдите кратчайший замкнутый путь U, содержащий все рёбра графа **G.**

Эйлеров путь (эйлерова цепь) в графе — это путь, проходящий по всем рёбрам графа и притом только по одному разу.

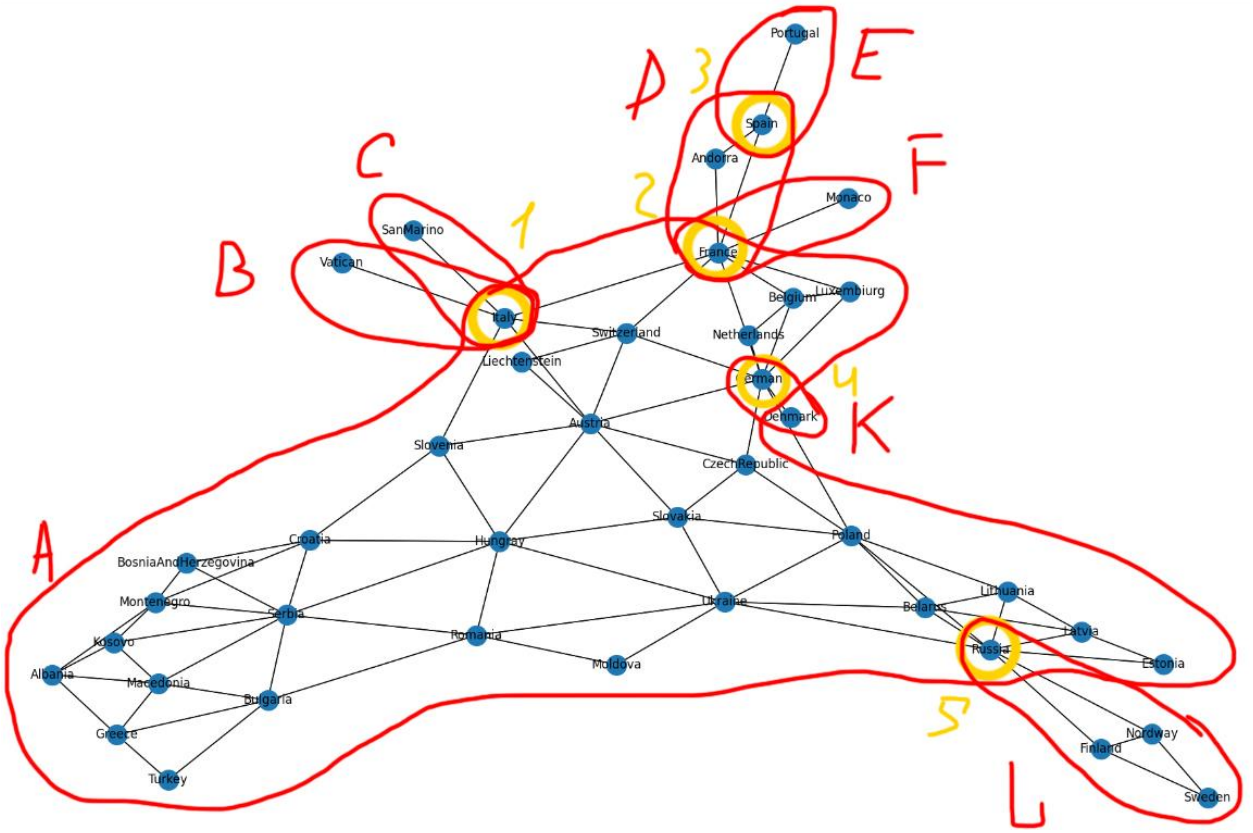
Алгоритм находит кратчайший замкнутый путь U, добавляя в исходный граф кратные рёбра, по которым можно вернуться из “тупика”. Таким образом создаётся эйлеров путь. Данный подход гарантирует нахождение КРАТЧАЙШЕГО пути U в графе.

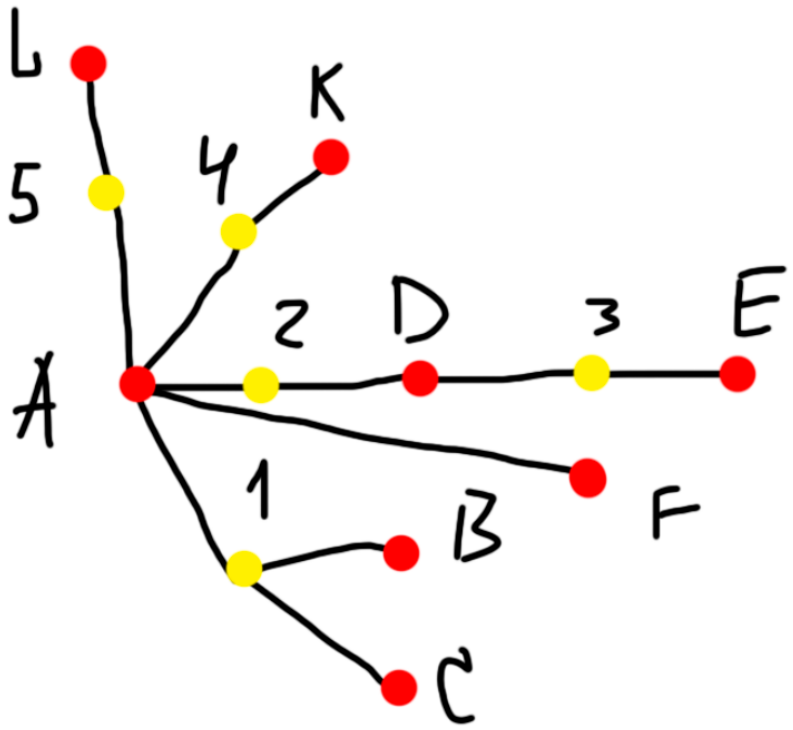
U = { (Austria --> Italy) (Italy --> SanMarino) (SanMarino --> Italy) (Italy --> Vatican) (Vatican --> Italy) (Italy --> France) (France --> Monaco) (Monaco --> France) (France --> Spain) (Spain --> Portugal) (Portugal --> Spain) (Spain --> Andorra) (Andorra --> France) (France --> Luxembiurg) (Luxembiurg --> Belgium) (Belgium --> Netherlands) (Netherlands --> Germany) (Germany --> Switzerland) (Switzerland --> Italy) (Italy --> Slovenia) (Slovenia --> Croatia) (Croatia --> Serbia) (Serbia --> Montenegro) (Montenegro --> Kosovo) (Kosovo --> Serbia) (Serbia --> Romania) (Romania --> Moldova) (Moldova --> Ukraine) (Ukraine --> Russia) (Russia --> Finland) (Finland --> Nordway) (Nordway --> Finland) (Finland --> Sweden) (Sweden --> Nordway) (Nordway --> Russia) (Russia --> Poland) (Poland --> Ukraine) (Ukraine --> Belarus) (Belarus --> Poland) (Poland --> Lithuania) (Lithuania --> Russia) (Russia --> Latvia) (Latvia --> Lithuania) (Lithuania --> Belarus) (Belarus --> Russia) (Russia --> Estonia) (Estonia --> Latvia) (Latvia --> Belarus) (Belarus --> Poland) (Poland --> Slovakia) (Slovakia --> Ukraine) (Ukraine --> Romania) (Romania --> Bulgaria) (Bulgaria --> Serbia) (Serbia --> BosniaAndHerzegovina) (BosniaAndHerzegovina --> Montenegro) (Montenegro --> BosniaAndHerzegovina) (BosniaAndHerzegovina --> Croatia) (Croatia --> Montenegro) (Montenegro --> Albania) (Albania --> Kosovo) (Kosovo --> Macedonia) (Macedonia --> Bulgaria) (Bulgaria --> Turkey) (Turkey --> Greece) (Greece --> Macedonia) (Macedonia --> Bulgaria) (Bulgaria --> Greece) (Greece --> Albania) (Albania --> Macedonia) (Macedonia --> Serbia) (Serbia --> Hungray) (Hungray --> Slovenia) (Slovenia --> Austria) (Austria --> Switzerland) (Switzerland --> France) (France --> Luxembiurg) (Luxembiurg --> Germany) (Germany --> Denmark) (Denmark --> Germany) (Germany --> France) (France --> Belgium) (Belgium --> Germany) (Germany --> Poland) (Poland --> CzechRepublic) (CzechRepublic --> Slovakia) (Slovakia --> Austria) (Austria --> Switzerland) (Switzerland --> Liechtenstein) (Liechtenstein --> Austria) (Austria --> Slovakia) (Slovakia --> Hungray) (Hungray --> Ukraine) (Ukraine --> Romania) (Romania --> Hungray) (Hungray --> Croatia) (Croatia --> Hungray) (Hungray --> Austria) (Austria --> Germany) (Germany --> CzechRepublic) (CzechRepublic --> Austria) }

l) Найдите все компоненты (блоки) вершинной двусвязности и постройте дерево блоков-точек сочленения графа **G.**

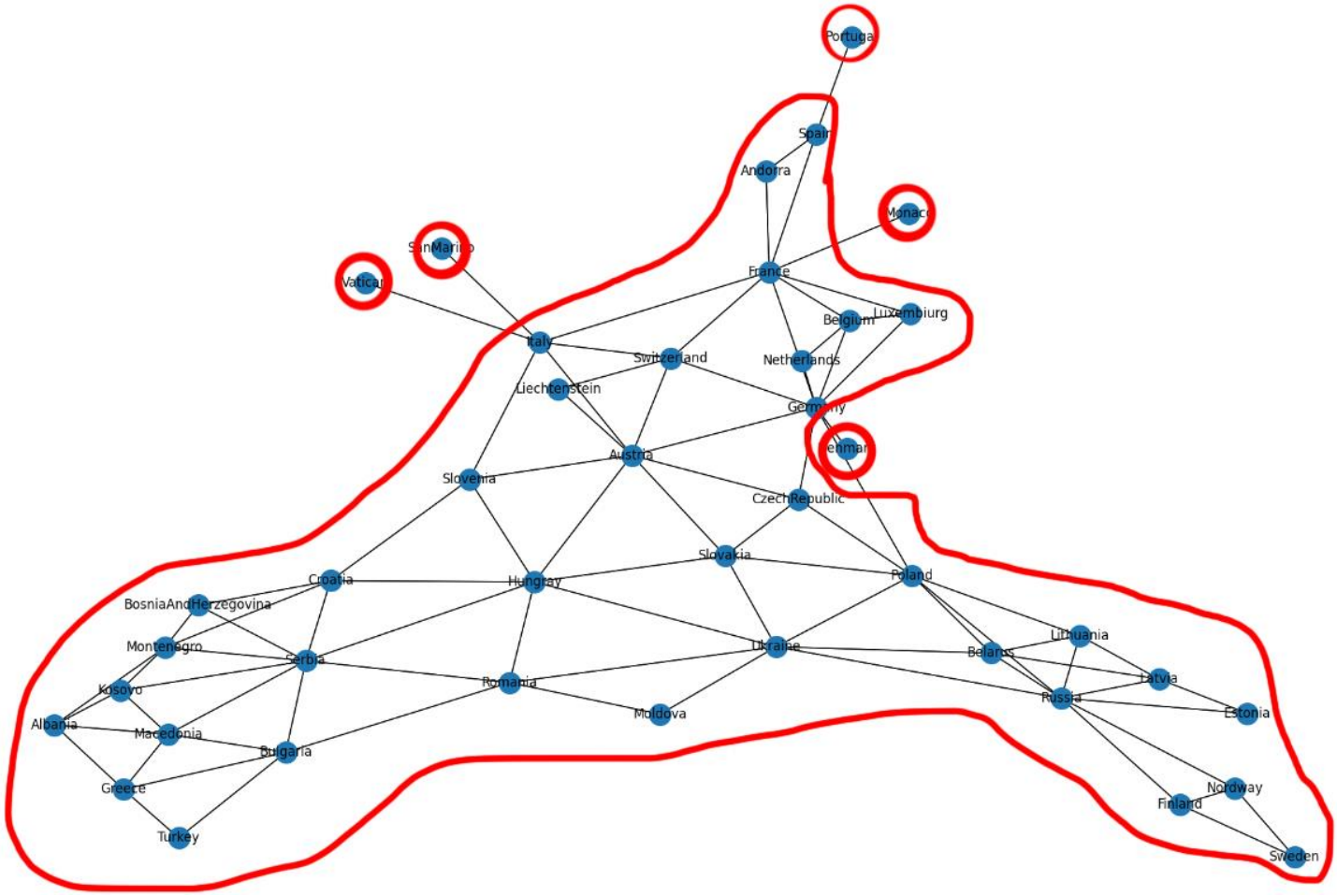
Обведём вершины – шарниры жёлтым, а блоки красным.

Построим дерево блоков-точек сочленения.




L
K
5
4
2
D
3
E
A
1
B
F
C

m) Найдите все компоненты рёберной двусвязности графа **G.**



Компонентами рёберной двусвязности графа называют его подграфы, множества вершин которых - классы эквивалентности рёберной двусвязности, а множества рёбер - множества ребер из соответствующих классов эквивалентности.

Обведём все компоненты рёберной двусвязности красным (в задании не просят найти мосты – поэтому их не трогаем).

n) Постройте **SPQR-дерево** самого большого двусвязного компонента графа **G**.

Трёхсвязные компоненты двусвязного графа — это система более мелких графов, описывающих все 2-вершинные сечения графа.

Документация:

https://doc.sagemath.org/html/en/reference/graphs/sage/graphs/connectivity.html#sage.graphs.connectivity.spqr_tree

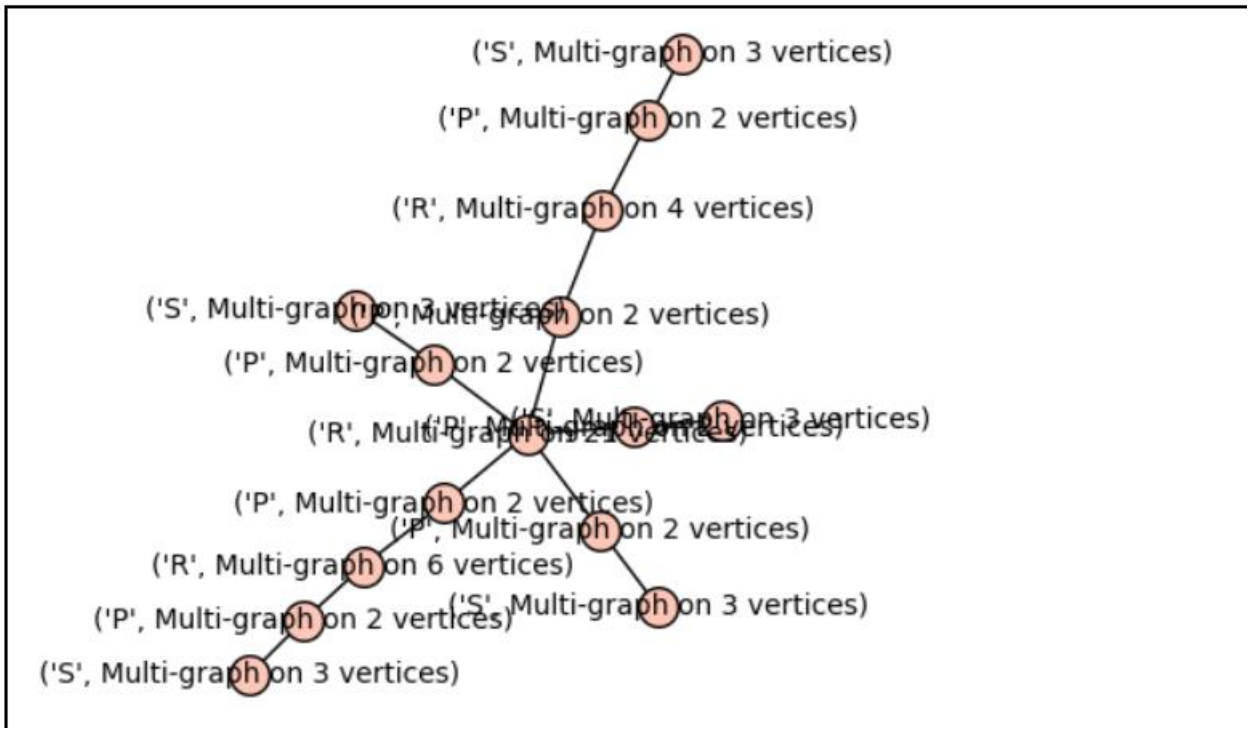
Компилятор:

https://sagecell.sagemath.org/?q=lnawao

Код:

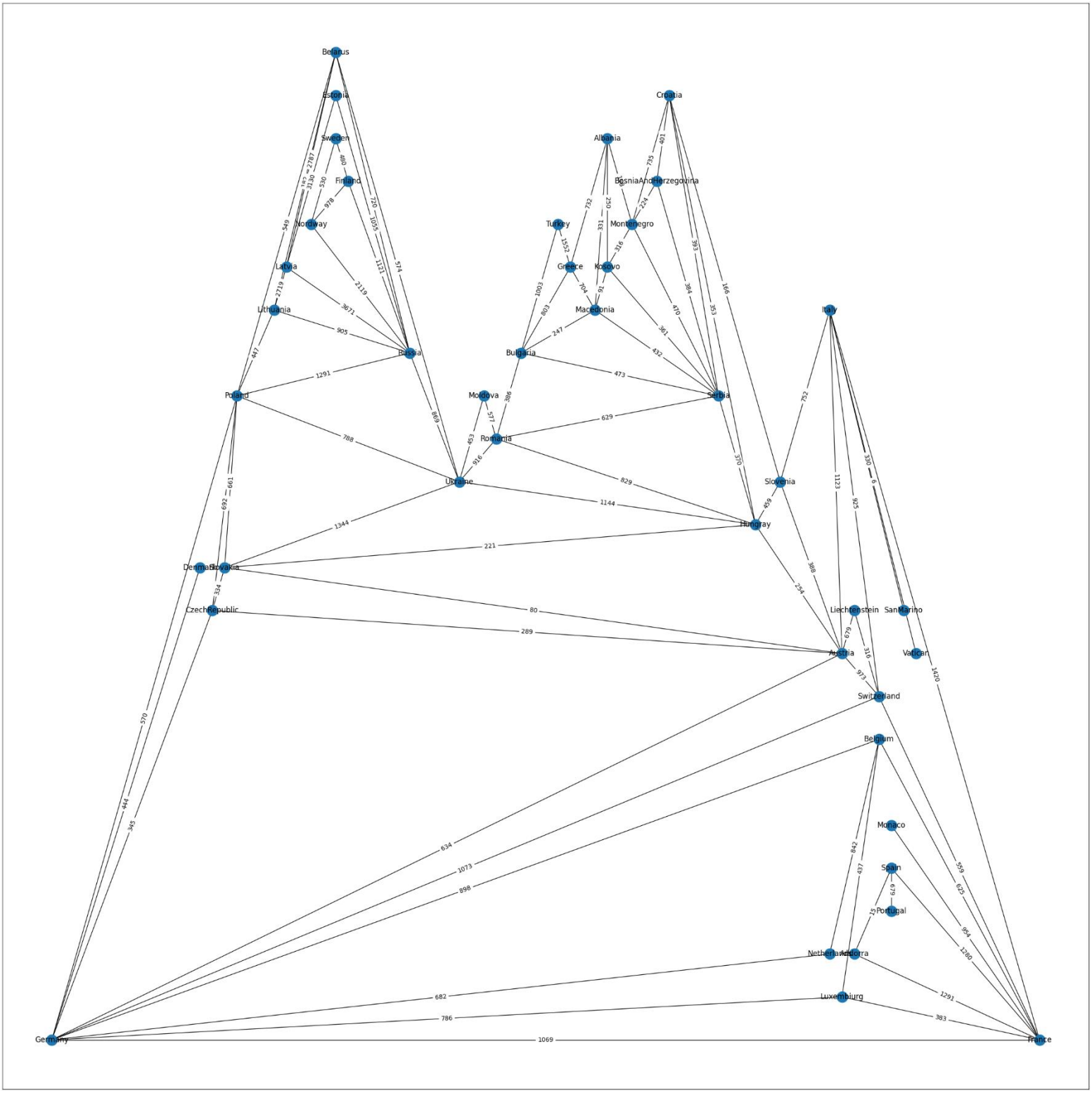
```
from sage.graphs.connectivity import TriconnectivitySPQR
from sage.graphs.connectivity import spqr_tree_to_graph
G = Graph ([[2, 8], [2, 13], [2, 15], [2, 19], [2, 34], [2, 35],
            [2, 38], [0, 14], [0, 22], [3, 18], [3, 20], [3, 28],
            [3, 31], [3, 40], [4, 12], [4, 13], [4, 21], [4, 26],
            [6, 14], [6, 22], [6, 30], [6, 39], [22, 14], [15, 7],
            [15, 30], [15, 34], [15, 35], [15, 40], [13, 8], [13, 12],
            [13, 21], [13, 26], [13, 28], [13, 38], [14, 39], [18, 10],
            [18, 20], [18, 31], [20, 28], [20, 31], [19, 38], [21, 12],
            [23, 30], [23, 40], [28, 8], [28, 31], [28, 34], [28, 40],
            [30, 40], [34, 40], [34, 8], [35, 7], [40, 31], [12, 38], [7, 5],
            [10, 31], [17, 25], [17, 0], [17, 22], [17, 33], [25, 7], [25, 5],
            [25, 33], [25, 0], [33, 15], [33, 30], [33, 6], [33, 22], [33, 5],
            [33, 7], [16, 12], [16, 35], [16, 38], [2, 16]])
tric = TriconnectivitySPQR(G)
T = tric.get_spqr_tree()

T.show()
```

SPQR - граф:

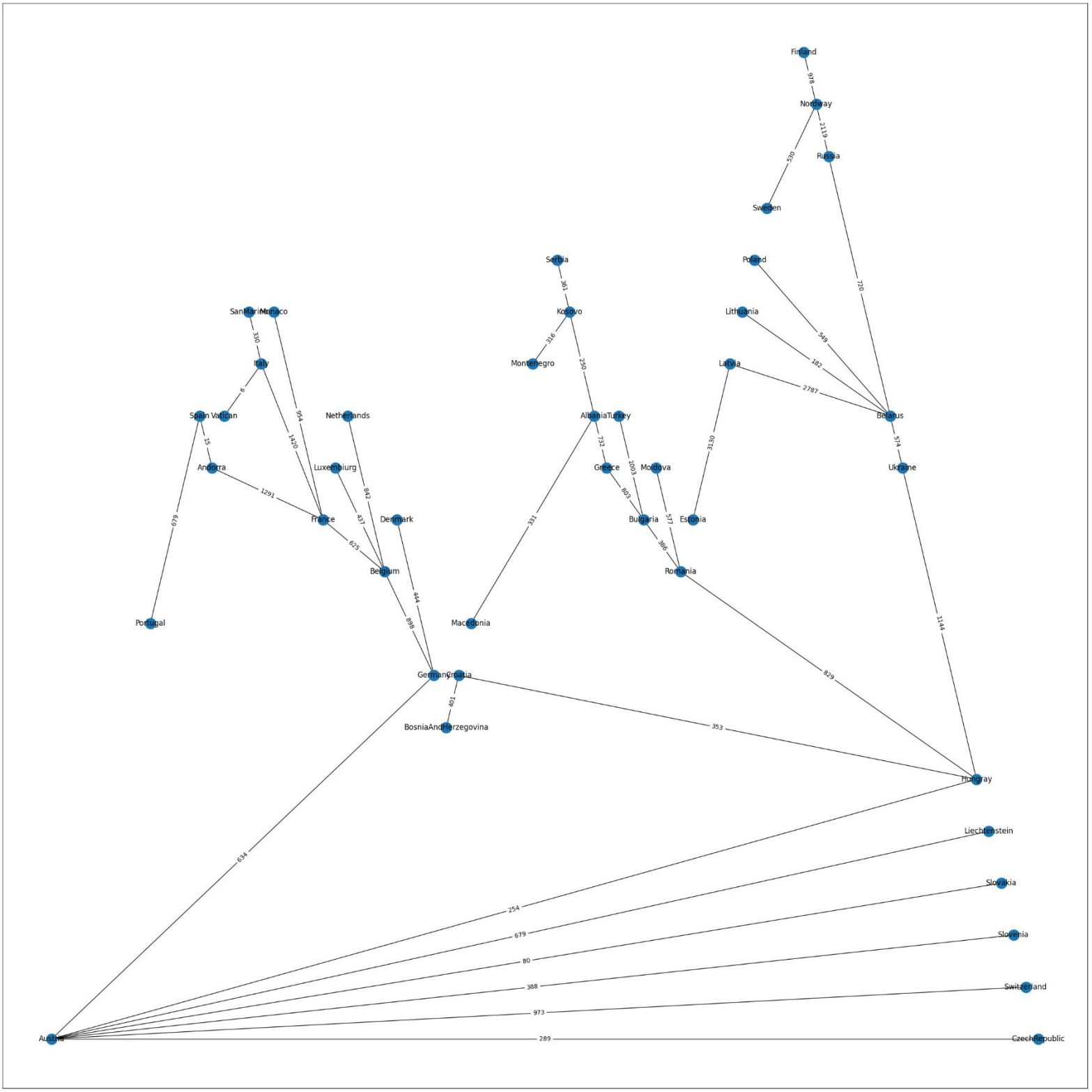


o) Добавьте весовую функцию w: $\boldsymbol{E}\rightarrow$R, обозначающую расстояние между столицами. Найдите минимальное остовное дерево $\boldsymbol{T}$ для взвешенного графа $\mathbf{E}\boldsymbol{\omega} = \langle V, E, w\rangle$.

Ниже изображен граф $\mathbf{E}\boldsymbol{\omega} = \langle V, E, w\rangle$.



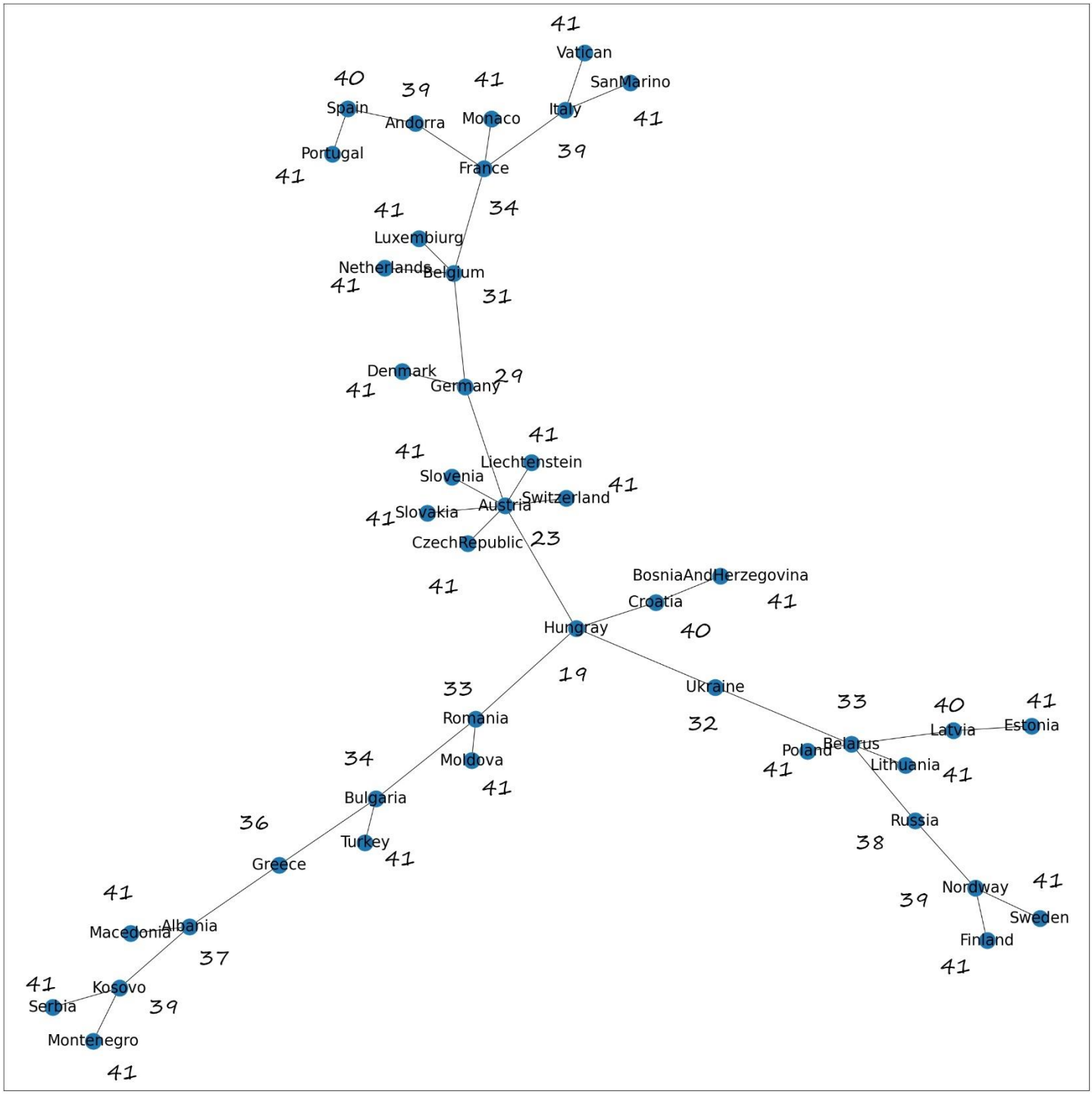
Теперь посмотрим на минимальное остовное дерево **T**, полученное с помощью алгоритма Прима:



p) Найдите **центроид T**.

Центроидом дерева называется такая вершина v дерева t, после удаления которой дерево разбивается на несколько (k) поддеревьев $t_1, t_2, \ldots, t_k$, таких что для каждого i: $|t_i| \leqslant n/2$, то есть размер каждого поддерева не превосходит половины размера исходного дерева.

Для каждой вершины найдём значение равное размеру наибольшей компоненты связности после удаления данной вершины. Как видно из рисунка **центроидом T** будет являться “Hungray”.



q) Построить код Прюфера для **T**.

Код Прюфера – это способ взаимно однозначного кодирования помеченных деревьев с n вершинами с помощью последовательности n-2 целых чисел в отрезке [1, n]. То есть, можно сказать, что **код Прюфера** – это биекция между всеми остовными деревьями полного графа и числовыми последовательностями.

Получим следующий порядок (номера вершин представлены в виде названий вершин):

Croatia, Hungray, Austria, Germany, Latvia, Nordway, Belarus, Austria, Belarus, Belgium, Albania, Romania, France, Kosovo, Belgium, Belarus, Spain, Italy, Kosovo, Albania, Greece, Bulgaria, Austria, Austria, Andorra, France, Nordway, Russia, Belarus, Ukraine, Austria, Bulgaria, Romania, Hungray, Hungray, Austria, Germany, Belgium, France, Italy.

Задание 2

1) Введем понятие между двумя вершинами dist(x, y) в графе $G = \langle V, E \rangle$:

$$dist(x, y) = min \mid x \rightsquigarrow_i y \mid,$$
$$x, y \in \boldsymbol{V}$$

i ∈ {1, … ,|множество всех цепей между x, y вершинами|}

2) Введем понятие для вершины $\varepsilon(u)$ в графе $G = \langle V, E \rangle$:

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

3) Введем понятие $rad\ G$ в графе $G = \langle V, E \rangle$:

$$rad\ G = \min \varepsilon(u)$$
$$u \in \boldsymbol{V}$$

4) Введем понятие $diam\ G$ в графе $G = \langle V, E \rangle$:

$$diam\ G = \max \varepsilon(u)$$
$$u \in \boldsymbol{V}$$

Theorem 1. Докажите «неравенство треугольника» для связного графа $G = \langle V, E \rangle$:

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

Proof:

Предположим, что $\forall x, y, z \in V : dist(x, y) + dist(y, z) < dist(x, z)$.

Из этого следует, что сумма длин кратчайших путей между вершинами x, y и между вершинами y, z меньше, чем длина кратчайшего пути между вершинами x, z. Однако, $dist(x, y)$ является путем наименьшей длины между вершинами (по опр. 1)).

Так, получаем противоречие (не может быть пути между двумя вершинами, длина которого меньше длины кратчайшего пути между ними), из чего следует, что “неравенство треугольника” для связного графа справедливо.

Theorem 2. Доказать, что для любого связного графа $rad\ G \leq diam\ G \leq 2rad\ G$

Proof:

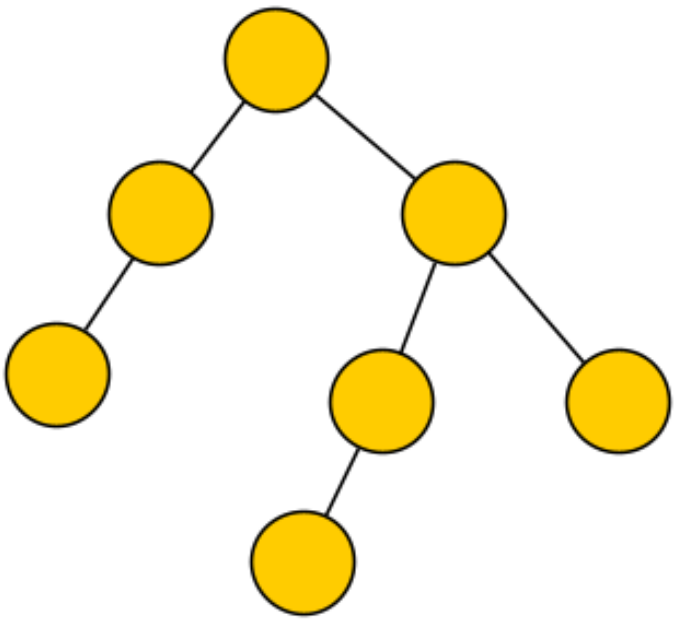
A) Первое неравенство следует из определений радиуса и диаметра.
B) Чтобы установить второе неравенство, предположим, что u и w вершины такие, что $dist(u,w) = diam\ G$, и пусть v - вершина такая что $\varepsilon(v) = rad\ G$. Тогда имеем:
$diam\ G = dist(u,w) \leq d(u,v) + d(v,w) \leq \varepsilon(v) + \varepsilon(v) = 2rad\ G$

Theorem 3. Докажите, что связный граф $G = \langle V, E \rangle$—дерево тогда и только тогда, когда $|E| = |V| - 1$.

Proof:

A) Докажем, что связного графа $G = \langle V, E \rangle$, являющегося деревом, справедливо $|E| = |V| - 1$.

Рассмотрим произвольное дерево:

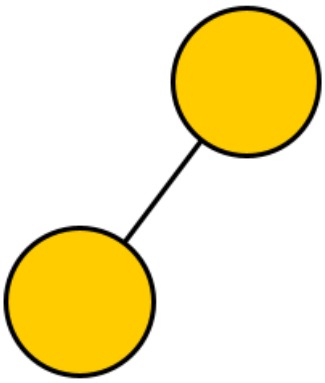


Произведем над данным графом следующую операцию: удалим из него случайный лист и ребро, инцидентное ему (так как лист – вершин в графе, чья степень равна единице, иначе лист – вершина, инцидентная только одному ребру [по определению

листа, иначе висячей вершины в дереве], из графа будет удалено только одно ребро).

Полученный граф так же будет являться деревом, следовательно можно заново произвести данное действие. Произведя данную операцию n - 2 раз граф будет состоять из двух вершин и ребра, инцидентного им.

Имеем:



Произведем эту операцию еще 1 раз, после чего оставшийся граф будет состоять из одной вершины и не будет рёбер вовсе.

В итоге после (N – 2 + 1) совершенных действий было удалено N – 1 ребер и N – 1 вершин, после чего был получен граф, состоящий из одной вершины и не имеющий ребер $\Longrightarrow$ начальный граф имел $|V| = N - 1 + 1 = N$ вершин и $|E| = N - 1$ рёбер.

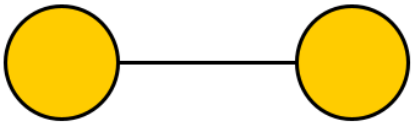
Так как $|V| = N$: $|E| = |V| - 1$

Б) Докажем, что связный граф $\boldsymbol{G} = \langle V, E \rangle$, в котором |E| = |V| - 1, является деревом

Возьмем граф, состоящий из одной вершины и не имеющий петель.

Добавим к данному графу вершину и ребро, инцидентное данной вершине и добавленной. Заметим, что при совершении такого действия любого количества раз, для полученного графа всегда будет справедливо |E| = |V| - 1 (так как на каждой итерации |V| и |E| увеличивается на 1, а изначально |V| = 1, |E| = 0). Так же данный граф будет являться связным, так как на первой итерации мы добавляем ему вершину (имеем две смежные вершины, которые образуют компоненту связности), а на последующих, добавляем в текущую компоненту новую вершину (так как новая вершина является смежной вершине из компоненты), из – за чего связность не нарушается. Из вышесказанного следует, что граф $\boldsymbol{G} = \langle V, E \rangle$ может быть получен, следую данному алгоритму.

Докажем, что такой граф является ацикличным. После первой итерации ранее описанного алгоритма будем следующий граф:



Очевидно, что в этом графе между любыми двумя вершинами существует лишь одна простая цепь, следовательно он является ацикличным. При добавлении вершины, та будет смежна вершине из данного ацикличного графа, а, следовательно, получившийся граф так же будет ацикличным. Тогда граф $G = \langle V, E \rangle$ является ацикличным связным графом $\Rightarrow$ $\underline{\boldsymbol{G} = \langle V, E \rangle \text{ - дерево.}}$

Как видно из А) и Б), что связного графа $\boldsymbol{G} = \langle V, E \rangle$, являющегося деревом, справедливо $|E| = |V| - 1$ и связный граф $\boldsymbol{G} = \langle V, E \rangle$, в котором |E| = |V| - 1, является

деревом. Следовательно, связный граф $G = \langle V, E \rangle$—дерево тогда и только тогда, когда $|E| = |V| - 1$.

Theorem 4. Докажите, что для связного графа G с n вершинами справедливо утверждение:

Если $\delta(G) \geq \lfloor n / 2 \rfloor$, то $\lambda(G) = \delta(G)$.

Proof:

a) Предположим, что $\lambda(G) < \delta(G)$. Тогда пусть удалив $\lambda(G)$ рёбер для каждой вершины $u \in V$ мы получим, что у каждой вершины осталось как минимум $\delta(G) - \lambda(G) > 0$ соседей. Пришли к противоречию.
b) Предположим, что $\lambda(G) < \delta(G)$. Тогда пусть удалив $\lambda(G)$ рёбер для каждой вершины $u \in V$ мы получим, что у каждой вершины осталось как минимум $\delta(G) - \lambda(G) < 0$ соседей. Пришли к противоречию.

Значит $\lambda(G) = \delta(G)$.

Theorem 5. Доказать, что каждый блок блочного графа - это клика.

Proof:

По определению на wiki:

Блоковый граф (кликовое дерево) — вид неориентированного графа, в котором каждая компонента двусвязности (блок) является кликой.

Блоковые графы можно описать графами пересечений блоков произвольных неориентированных графов.

https://ru.wikipedia.org/wiki/%D0%91%D0%BB%D0%BE%D0%BA%D0%BE%D0%B2%D1%8B%D0%B9_%D0%B3%D1%80%D0%B0%D1%84