

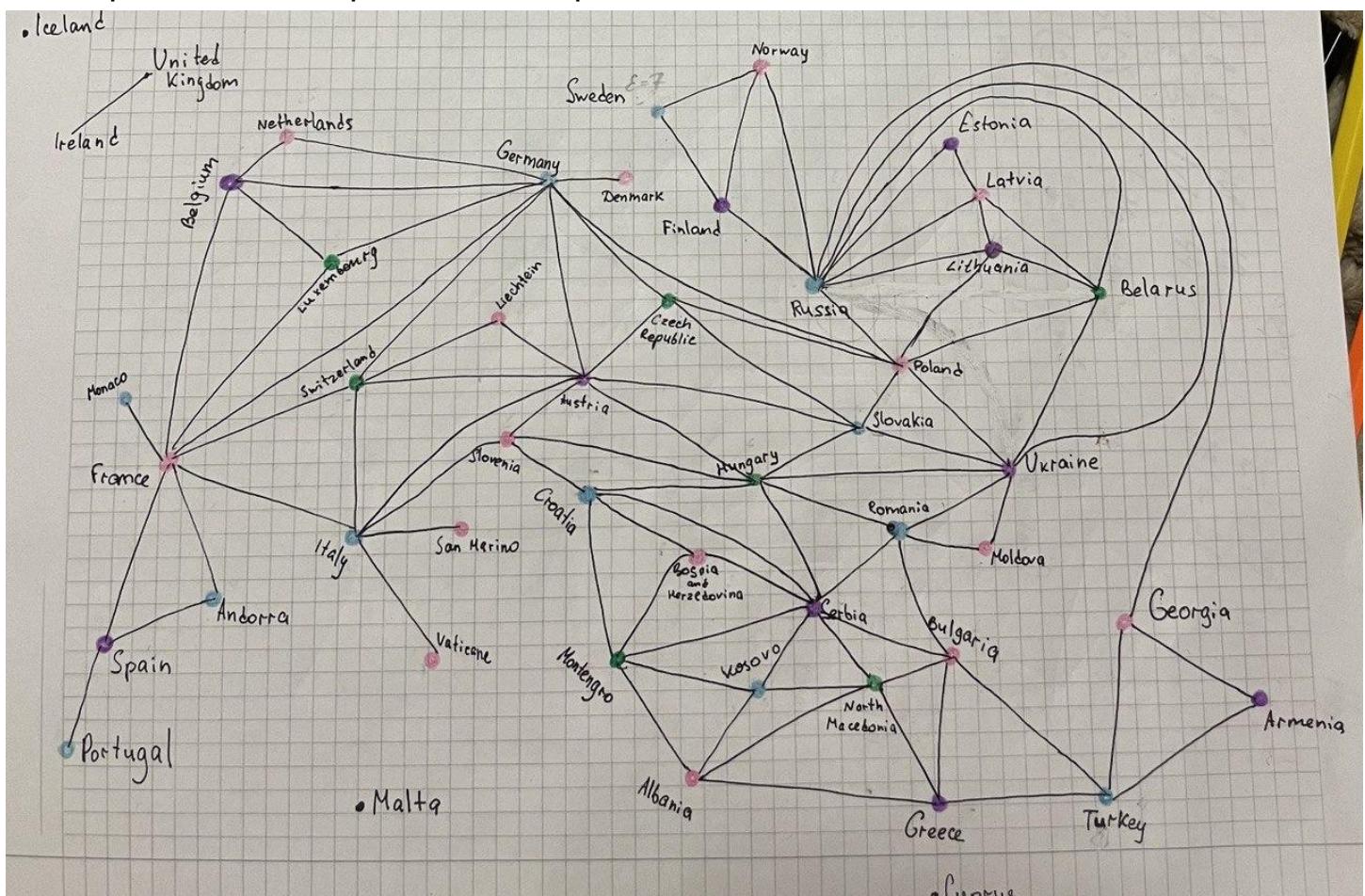
Дискретная математика
Домашнее задание 1
Кравченкова Елизавета М3103

№1

The graph of Europe $G^* = \langle V, E \rangle$ is defined as follows: each vertex $v \in V$ is a Europe country; two vertices are adjacent ($\{u, v\} \in E$) if the corresponding countries share a land border. Let G be the largest connected component of G^* .

(a) Draw G^* with minimum number of intersecting edges.

Я просто взяла карандаши и нарисовала.



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

$$\begin{array}{ll} |V| \text{ (в графе } G^*) = 49 & |E| \text{ (в графе } G^*) = 92 \\ |V| \text{ (в графе } G) = 44 & |E| \text{ (в графе } G) = 91 \end{array}$$

$\delta(G)$ - minimum degree = 1 (Vatican)

$\Delta(G)$ - maximum degree = 9 (Russia)

$\text{Girth}(G)$ - is the length of the shortest cycle in the graph = 3 (Turkey, Armenia, Georgia)

$\kappa(G)$ - is the minimum number of vertices that has to be removed in order to make the graph disconnected or trivial = 1 (Italy)

$\lambda(G)$ - is the minimum number of edges that has to be removed in order to make the graph disconnected or trivial = 1 (France - Monaco)

$\text{diam}(G)$ и $\text{center}(G)$ я подсчитала, найдя эксцентриситет для каждой вершины. Для этого я запустила BFS для каждой вершины и нашла кратчайшее расстояние до дальней его вершины. (алгоритм выводит для каждой вершины эксцентриситет и вершину до которой от нее самое большое расстояние) . Использованный код: .

https://github.com/elizabetinka/eccentricity_for_every_vertex.

$\text{diam}(G) = 8$ (от Greece до Portugal)

радиус = 5

$\text{center}(G)$: Austria, Belarus, Croatia, Czech Republic, Germany, Hungary, Lithuania, Poland, Russia, Slovakia, Slovenia, Switzerland, Ukraine (13 вершин)

(c) Find the minimum vertex coloring $Z : V \rightarrow N$ of G .

Так как существует полный подграф из 4 вершин(см пункт (e))

количество цветов для минимальной раскраски не меньше 4.

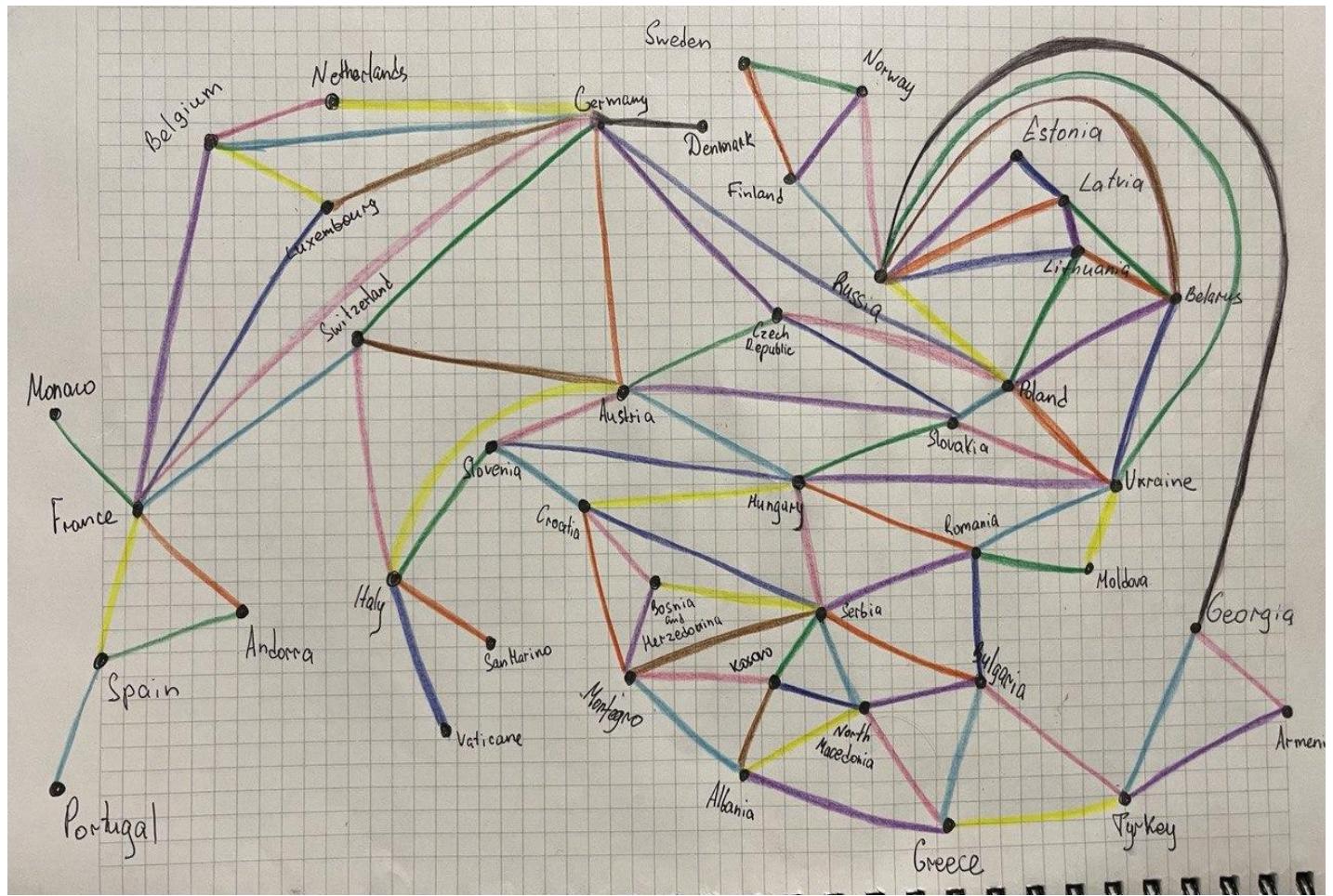
Предположим обратное, тогда в этом подграфе есть 2 вершины одного цвета - а так как он полный - то между ними существует ребро-противоречие.

Пример раскраски графа в 4 цвета можно увидеть в фотографии в пункте (a)

(d) Find the minimum edge coloring $X : E \rightarrow N$ of G .

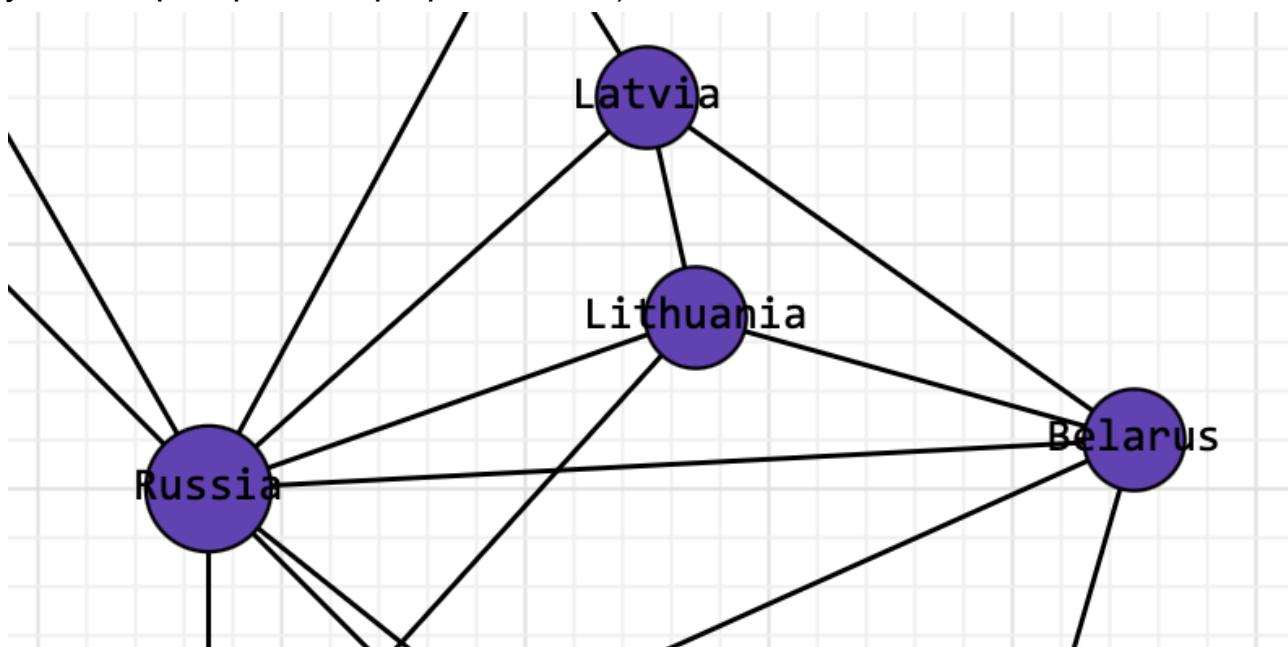
Так как максимальная степень вершины - 9, то число цветов для раскраски не меньше 9.

Пример раскраски ребер графа в 9 цветов:



(e) Find the maximum clique $Q \subseteq V$ of G .

Russia-Latvia-Lithuania-Belarus (больше точно не существует, так как удалось раскрасить граф в 4 цвета)



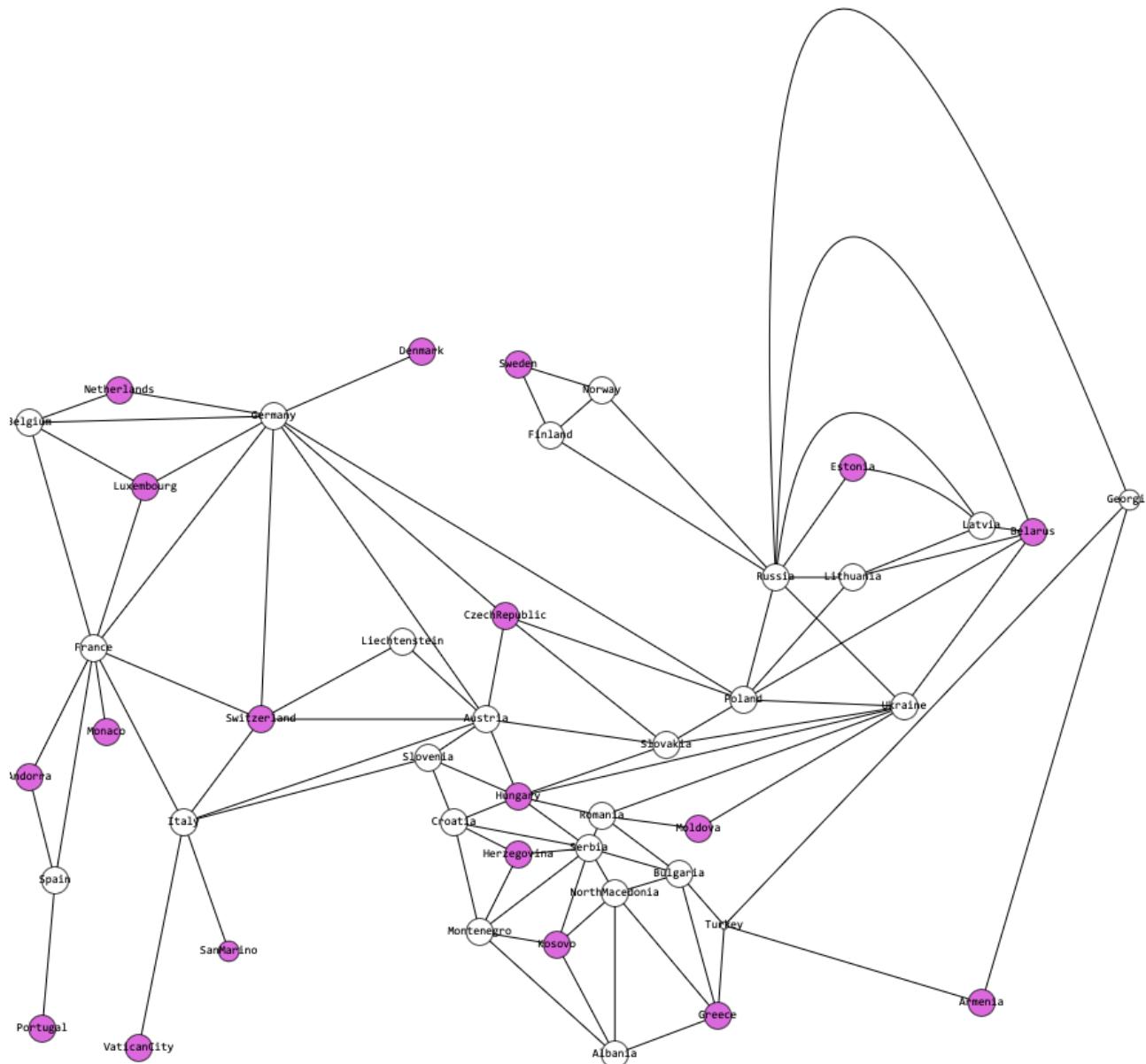
(f) Find the maximum stable set $S \subseteq V$ of G .

Используя код

https://github.com/elizabetinka/Find_max_size_of_independent_set/blob/main/README.md удалось найти мощность максимального независимого

множества вершин для нашего графа - 19. В коде используется встроенная функция библиотеки igraph для поиска числа независимости для графа, она основана на [algorithm from the paper S. Tsukiyama, M. Ide, H. Ariyoshi and I. Shirawaka](#). (там как-то сложно, я не поняла как он работает)

Пример для 19 вершин:

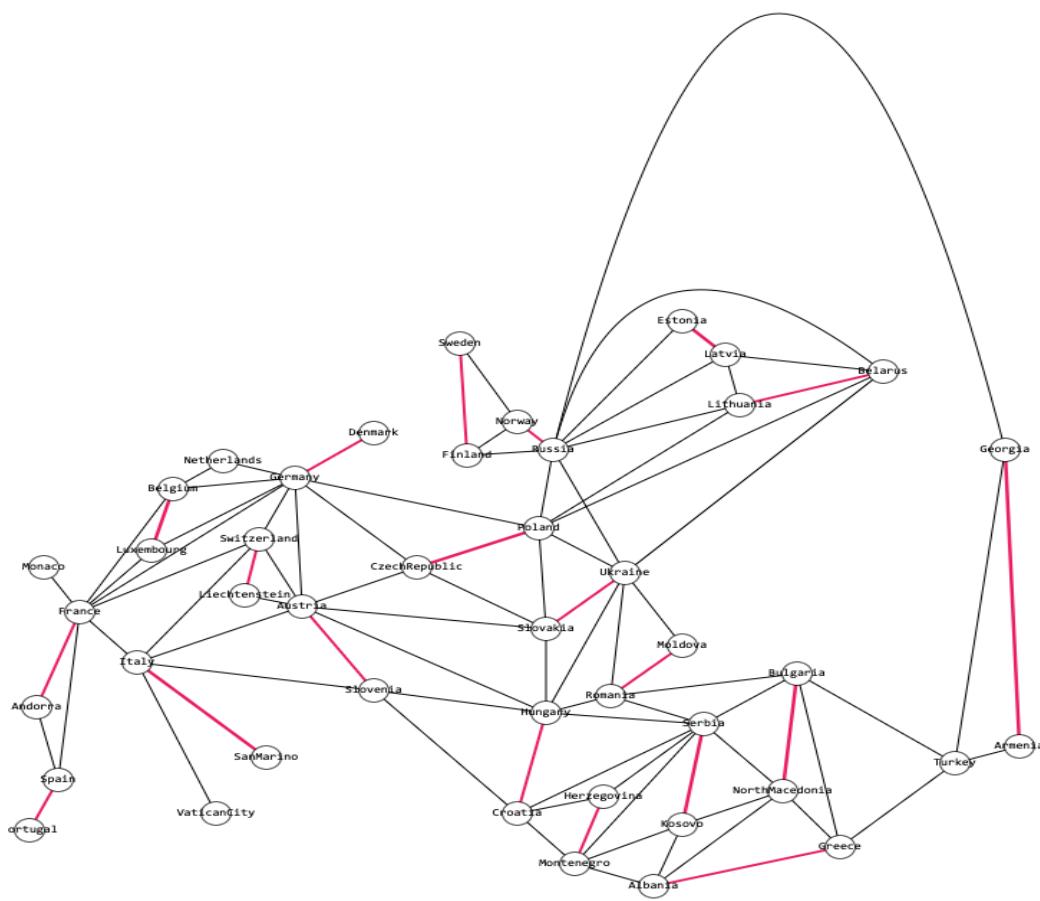


(g) Find the maximum matching $M \subseteq E$ of G .

Для решения данного пункта использовался [Edmonds' blossom algorithm](#). Идея алгоритма Эдмондса заключается в сжатии цветов. Сжатие цветка — это сжатие всего нечетного цикла в одну «вершину». Алгоритм ищет в графе все цветы, сжимает их, после чего в графе не остается «плохих» циклов нечетной длины, а на таком графе уже можно искать увеличительную цепочку, обходя в ширину. (вершина v **голая** (не покрыта паросочетанием), если нет ребра в M , инцидентного v . Путь в G является **чередующейся цепью**, если её рёбра попаременно не принадлежат M и содержатся в M . **Увеличивающий путь P** — это чередующаяся цепь, которая начинается и кончается голыми вершинами). Найдя увеличительную цепочку в поверхностном графе, необходимо «развернуть» цветы, восстановив тем самым увеличительную цепочку в исходном графе.

Полученный вывод: количество ребер, входящих в него - 20

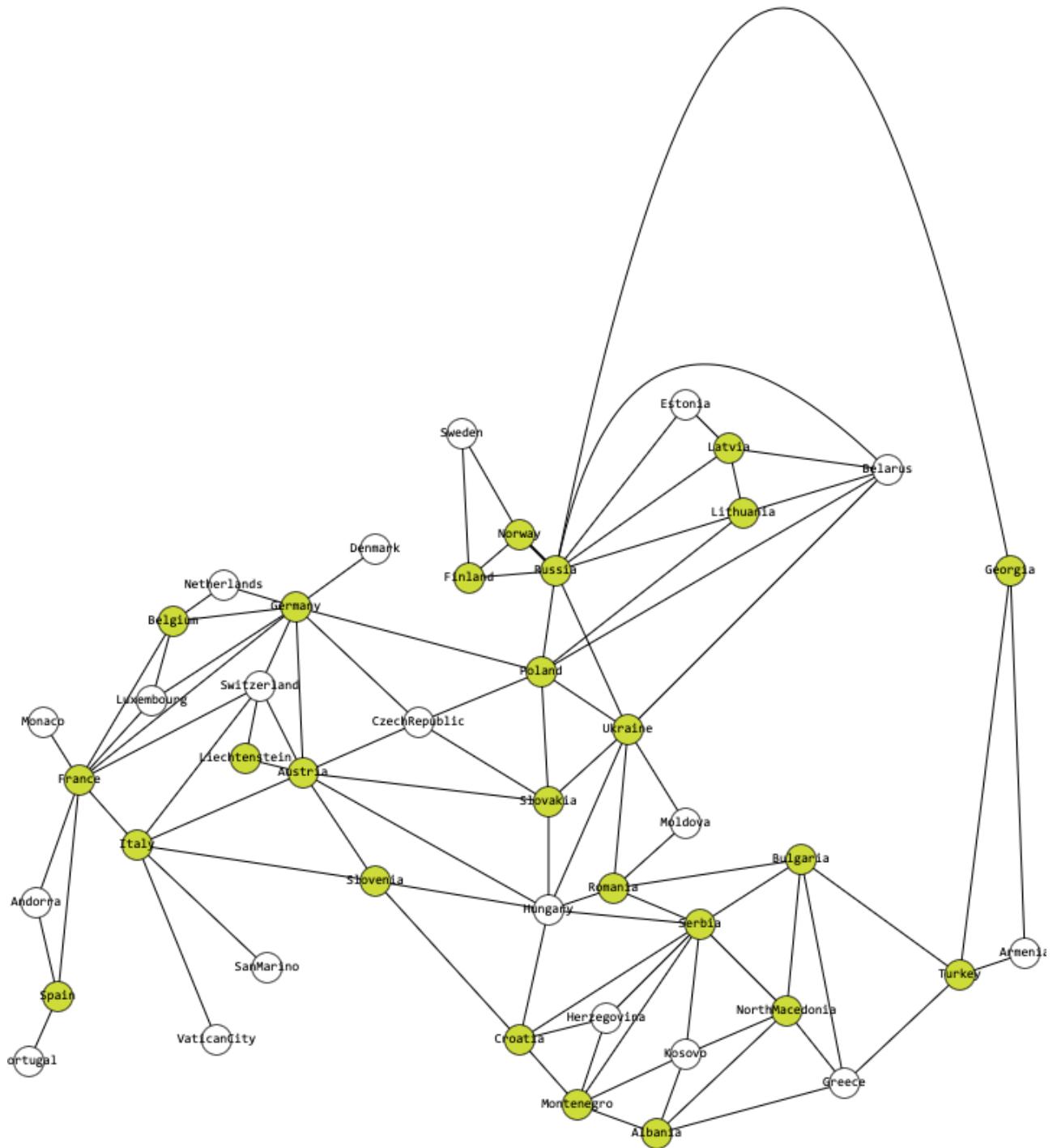
```
[indo.evri.momint@MacBook-Pro-Elizaveta EdmondsBlossoms % java -jar Edmonds.jar -] n "/Users/indo.evri.momint/Documents/untitled/dscr10.txt"
Matching:
Spain --- Portugal
Belgium --- Luxembourg
France --- Andorra
Germany --- Denmark
SanMarino --- Italy
Latvia --- Estonia
Norway --- Russia
Finland --- Sweden
Moldova --- Romania
Lithuania --- Belarus
Liechtenstein --- Switzerland
Slovenia --- Austria
Poland --- CzechRepublic
Slovakia --- Ukraine
Hungary --- Croatia
Montenegro --- Herzegovina
Serbia --- Kosovo
Bulgaria --- NorthMacedonia
Turkey --- Greece
Georgia --- Armenia
indo.evri.momint@MacBook-Pro-Elizaveta EdmondsBlossoms %
```



(h) Find the minimum vertex cover $R \subseteq V$ of G .

Из теоремы ([Связь вершинного покрытия и независимого множества — Викиконспекты](#)) понятно, что мощность минимального вершинного покрытия = $|V| - |\text{maximum stable set}| = 44 - 19 = 25$

Пример для 25:



(i) Find the minimum edge cover $F \subseteq E$ of G .

Для поиска использовался

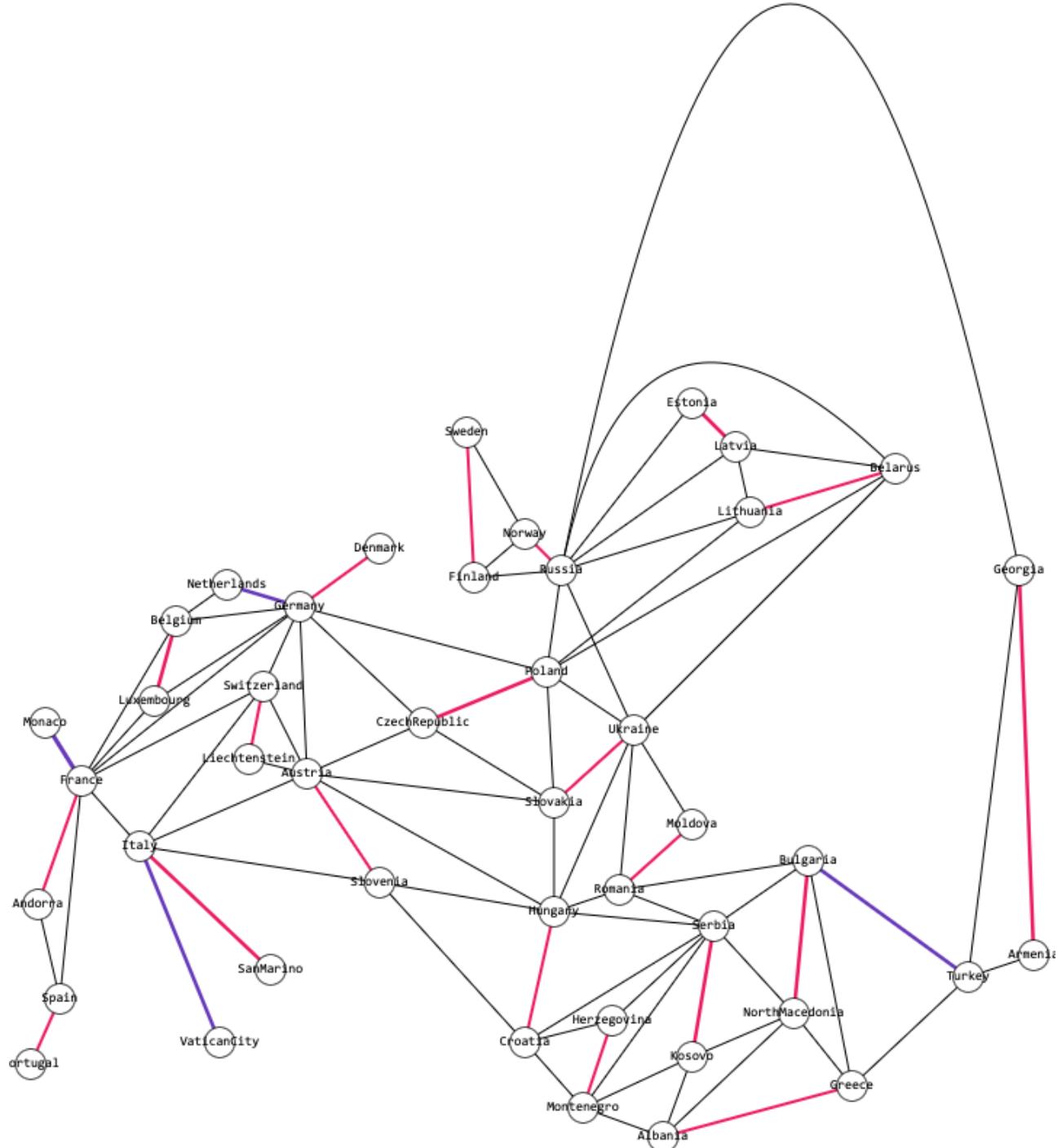
https://github.com/elizabetinka/minimum_edge_cover. Полученное

множество состоит из 24 ребер.

Тот же ответ можно получить, применив [Gallai's theorem](#) : $|\text{max stable set}| + |\text{min vertex cover}| = |\text{max matching}| +$

$|\text{min edge cover}| = |V|$. Тогда $|\text{min edge cover}| = 44 - 20 = 24$. Оно получено

минимальным добавлением к maximum matching недостающих ребер.



(j) Find the shortest closed walk W that visits every vertex of G.

Можно догадаться, что оценка снизу для количества для требуемого пути - 51.

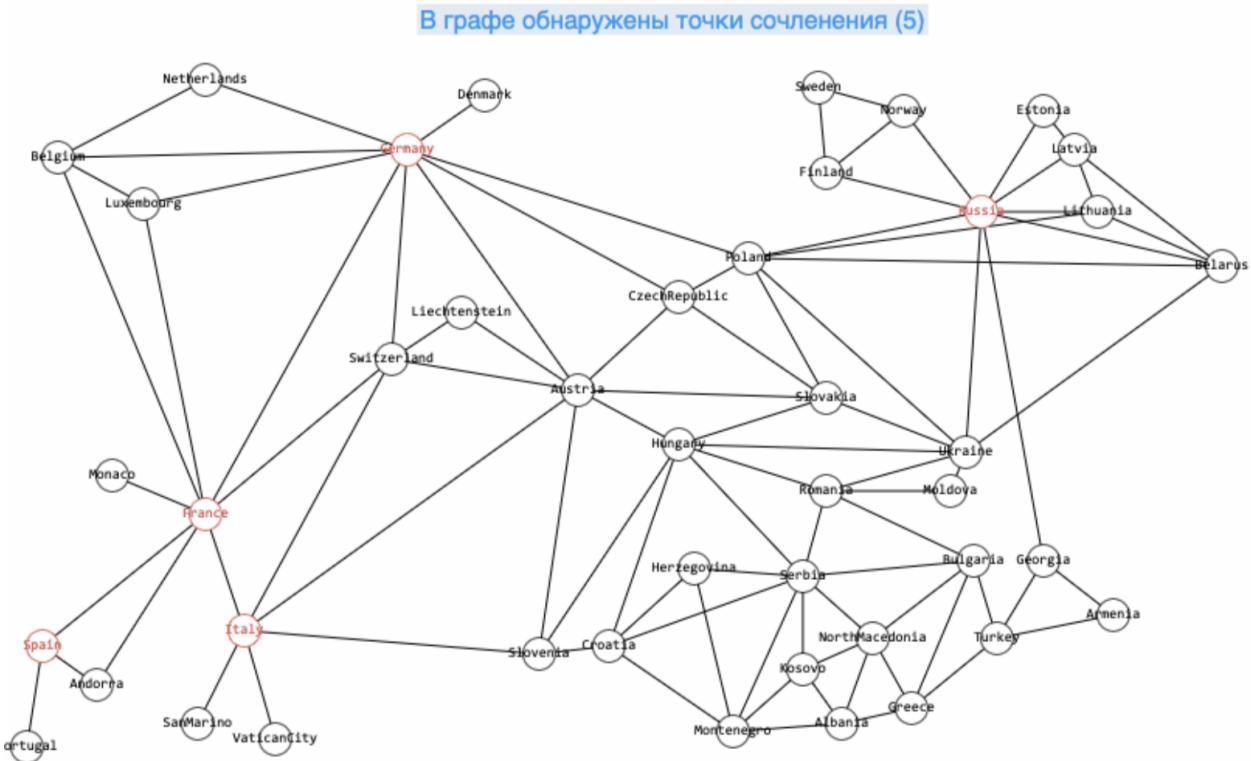
Чтобы пройти все вершины как можно с меньшим количеством повторений, надо на каждом шаге идти в непосещенную вершину, а в конце в ту, из которой начинали (минимум $|V| = 44$) шага. Но найдем такие точки, в которые невозможно зайти только один раз.

Это точки сочленения: Germany, Russia, Italy, Spain, France. Т.е в эти точки в пути мы заходим не менее 2 раз: чтобы “выйти” через них, и чтобы “зайти”. Но если рассмотреть подграф, полученный из графа G удалением Франции (или Италии) и всех инцидентных ей ребер, в нем получится 3 компоненты связности. А это значит, что в эти вершины мы посетим не менее 3 раз. А значит оценка снизу: $44+1+1+1+2+2 = 51$.

Вот пример с длиной 51:

Armenia-Turkey-Greece-Bulgaria-North
Macedonia-Albania-Kosovo-Serbia-Montenegro-Bosnia
Herzegovina-Croatia-Slovenia-Italy-Vatican
City-Italy-San Marino-Italy-France-Andorra-Spain-Portugal
-Spain-France-Monaco-France-Luxembourg-Belgium-Netherlands-Germany-Denmark-Germany-Switzerland-Liechtenstein-Austria-Czech
Republic-Slovakia-Hungary-Romania-Moldova-Ukraine-Poland
-Belarus-Lithuania-Latvia-Estonia-Russia-Norway-Sweden-Finland-Russia-Georgia-Armenia.

Поиск точек сочленения:(тут гифка работы алгоритма,но выглядит как просто картинка). Да и так как это тривиальный алгоритм, не вижу смысла его пояснять:



(k) Find the shortest closed walk U that visits every edge of G .

Я использовала этот алгоритм: [C++ solution for the chinese postman problem](#). The Chinese postman problem, postman tour or route inspection problem is to find a shortest closed path or circuit that visits every edge of an (connected) undirected graph. When the graph has an Eulerian circuit (a closed walk that covers every edge once), that circuit is an optimal solution. Идея алгоритма заключается в том, чтобы найти минимальное количество ребер которые надо дублировать(чтобы степени вершин стали четными и существовал эйлеров цикл), потом стандартными алгоритмами, такими как the Hierholzer method найти цикл. (Основная идея the Hierholzer method заключается в поэтапном построении эйлерового цикла путем соединения дизъюнктивных окружностей. Он начинается со случайного узла, а затем следует за произвольным непоседенным ребром к соседу. Этот шаг повторяется до

тех пор, пока человек не вернется к начальному узлу. Это дает первую окружность в графе. Если эта окружность охватывает все узлы, это эйлеровский цикл, и алгоритм закончен. В противном случае один выбирает другой узел среди узлов циклов с непосещенными ребрами и строит другой круг, называемый подконтуром. При выборе ребер в конструкции новая окружность не содержит ни одного ребра первой окружности, оба являются дизъюнктными. Однако обе окружности должны пересекаться по крайней мере в одном узле по выбору начального узла второй окружности. Таким образом, можно представить обе окружности как одну новую окружность. Для этого можно итерировать узлы первой окружности и заменить начальный узел подтута полной последовательностью узлов подтута. Таким образом, дополнительные круги интегрируются в первую окружность. Если расширенный цикл включает в себя все ребра, алгоритм завершается. В противном случае мы можем найти другой цикл для включения. В случае неориентированного полуэйлеровского графа алгоритм начинается с одного из двух узлов с нечетной степенью. В указанном случае с узлом с одним дополнительным исходящим ребром. Один из найденных подтутров не будет образовывать цикл, вместо этого он также будет траекторией. При интеграции этого "подтута" в круг нужно убедиться, что начальный и конечный узел этого контура также образуют начало и конец полного пути Эйлера.)

Полученное решение:

```
indo.evri.momint@MacBook-Pro-Elizaveta chinese-postman-problem % ./chinese -f "/Users/indo.evri.momint/Documents/untitled/por_chinece"
Solution cost: 106
Solution:
0 41 16 7 35 6 8 26 1 28 19 1 16 28 7 28 35 32 42 32 24 42 36 3 14 27 5 23 14 13 5 14 9 3 36 30 4 33
11 20 4 30 42 33 15 33 30 22 33 29 12 29 39 12 33 20 22 4 42 17 8 17 37 18 34 18 43 18 40 3 40 21 3
37 8 35 26 6 26 19 35 17 36 9 30 14 10 14 40 13 23 13 25 13 38 31 38 2 13 18 3 17 32 7 41 15 0
indo.evri.momint@MacBook-Pro-Elizaveta chinese-postman-problem %
```

Где вершины пронумерованы в соответствии с их расположением в лексикографической сортировке. Таким образом, путь выглядит так:

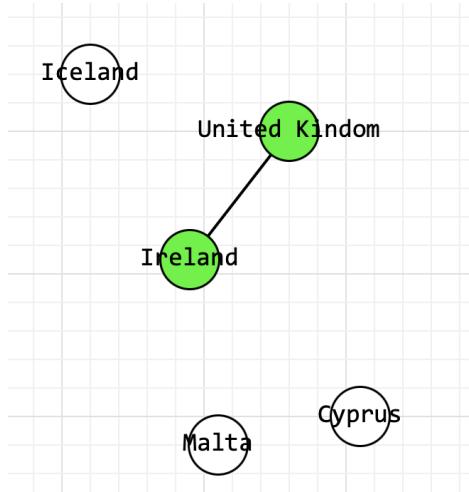
Armenia-Turkey-Greece-Bulgaria-Serbia-Herzegovina-Croatia-Montenegro-Albania-NorthMacedonia-Kosovo-Albania-Greece-NorthMacedonia-Bulgaria-NorthMacedonia-Serbia-Romania-Ukraine-Romania-Moldova-Ukraine-Slovakia-Austria-Germany-Netherlands-Belgium-Luxembourg-Germany-France-Belgium-Germany-CzechRepublic-Austria-Slovakia-Poland-Belarus-Russia

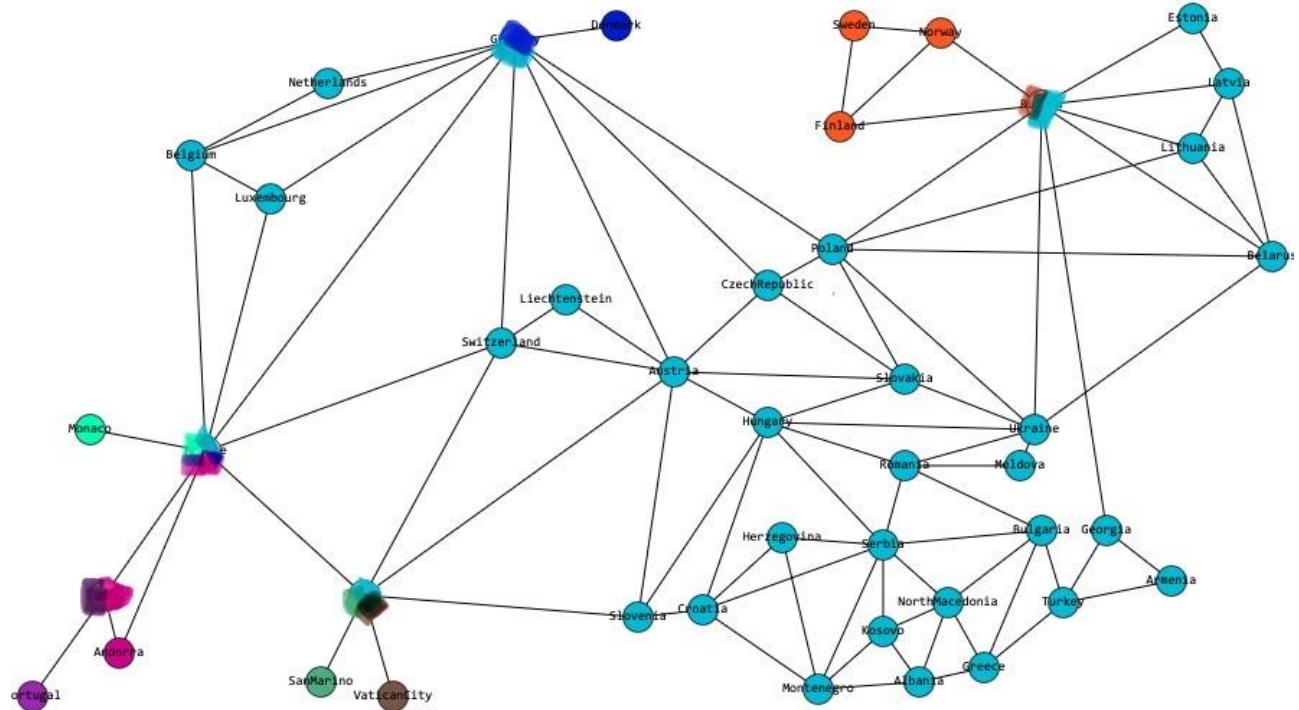
sia-Estonia-Latvia-Belarus-Poland-Ukraine-Russia-Georgia
-Russia-Poland-Lithuania-Russia-Norway-Finland-Norway-Sweden-Finland-Russia-Latvia-Lithuania-Belarus-Ukraine-Hungary-Croatia-Hungary-Slovenia-Italy-San Marino-Italy-Vatican City-Italy-Switzerland-Austria-Switzerland-Liechtenstein-Austria-Slovenia-Croatia-Serbia-Montenegro-Herzegovina-Montenegro-Kosovo-Serbia-Hungary-Slovakia-Czech Republic-Poland-Germany-Denmark-Germany-Switzerland-France-Luxembourg-France-Monaco-France-Spain-Portugal-Spain-Andorra-France-Italy-Austria-Hungary-Romania-Bulgaria-Turkey-Georgia-Armenia

(I) Find all biconnected components (blocks) and draw the block-cut tree of G*

biconnected component - такой подграф, из которого при удалении любой вершины и инцидентных ей ребер граф остается связным.

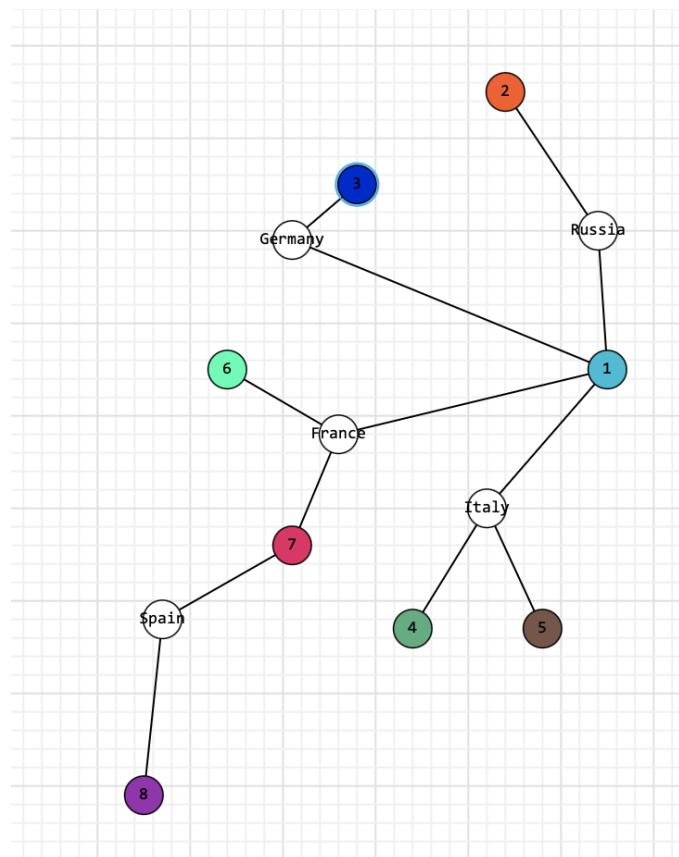
В пункте (j) были найдены точки сочленения— они разделяют biconnected components:





Вершины одного цвета принадлежат одной biconnected component вершины нескольких цветов-точки сочленения.(они принадлежат нескольким компонентам)

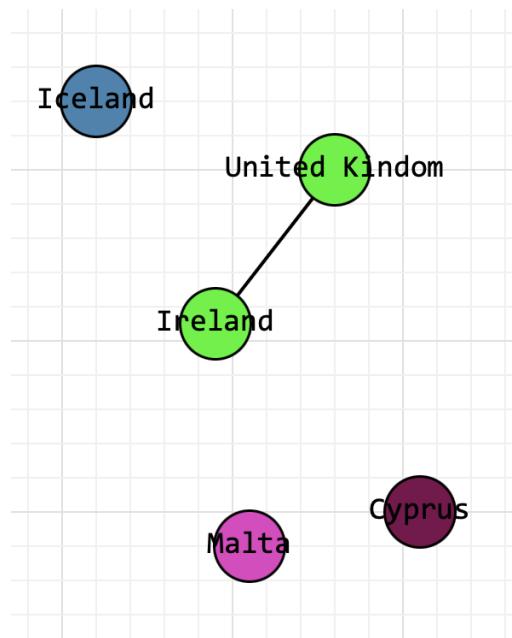
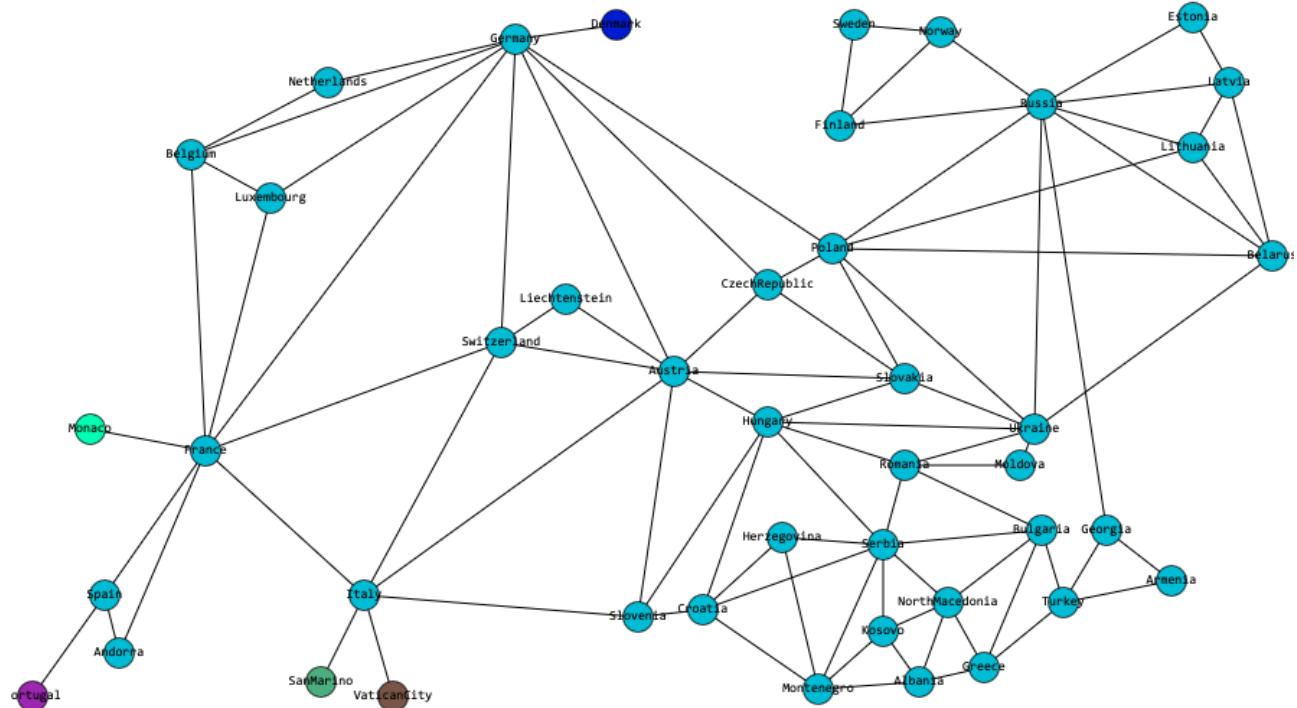
Нарисуем block-cut tree of G^* :



(m) Find all 2-edge-connected components of G*

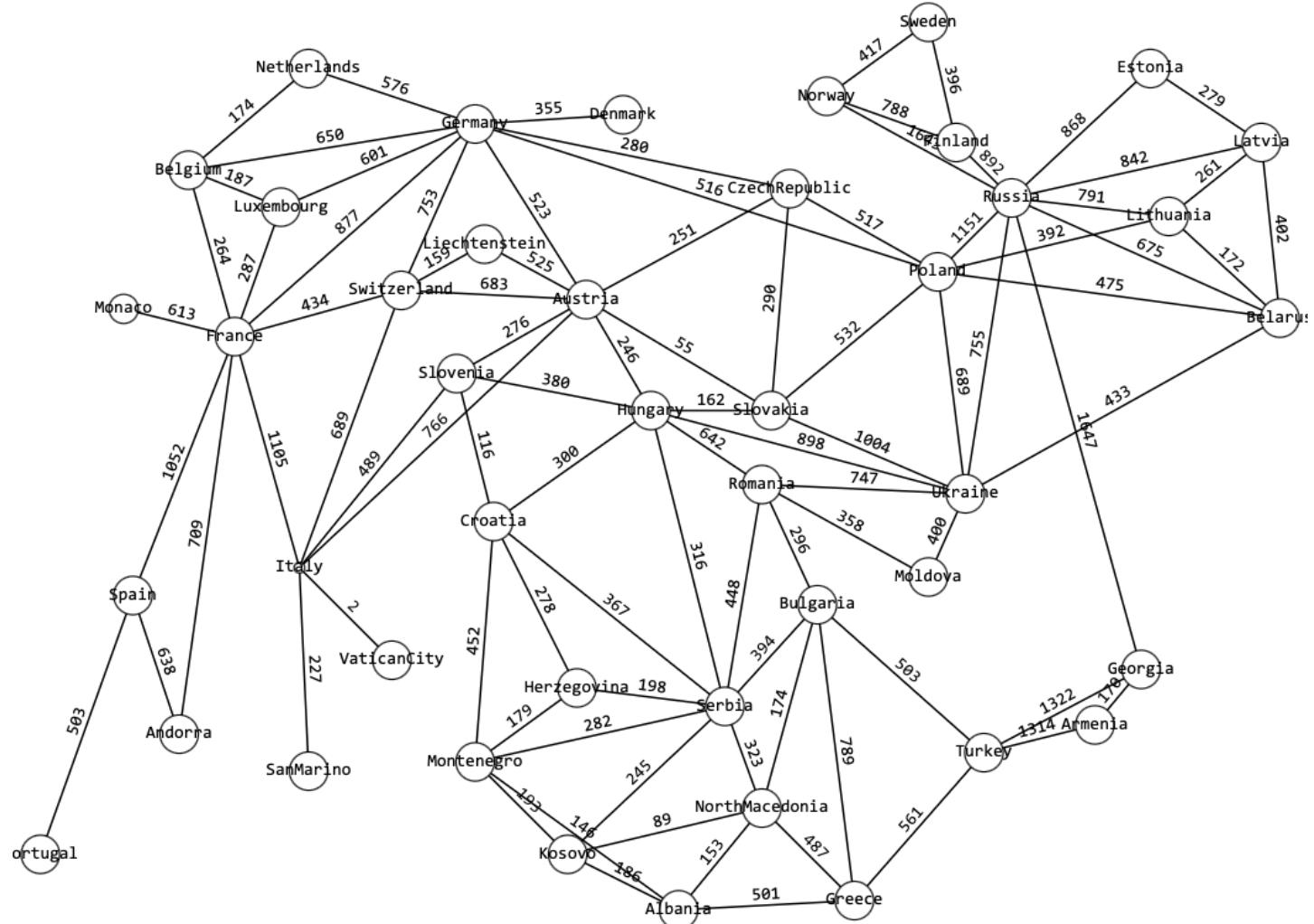
2-edge-connected component - это такой подграф, который при удалении какого либо ребра остается связным.

Разными цветами помечены разные компоненты:

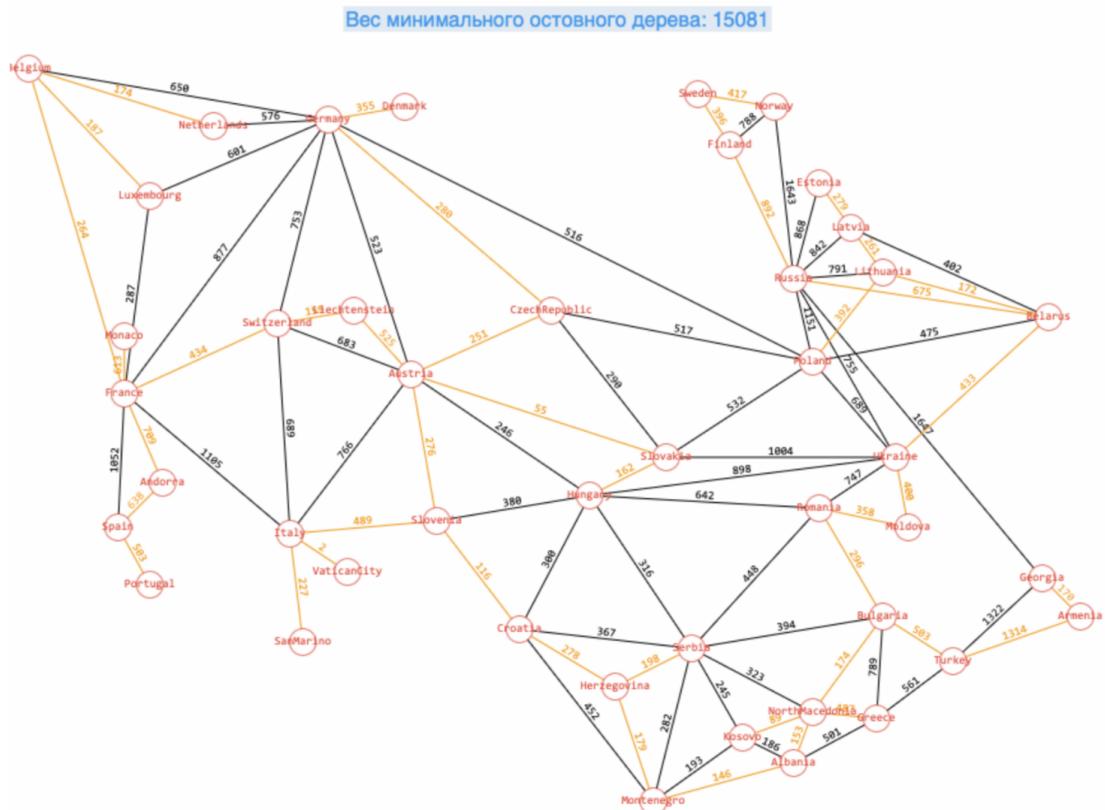


(n) Add the weight function $w : E \rightarrow \mathbb{R}$ denoting the distance between capitals. Find the minimum (w.r.t. the total weight of edges) spanning tree T for the largest connected component of the weighted Europe graph $G^* w = (V, E, w)$.

Я прописала расстояние для каждой столицы до каждой. Получился следующий граф:

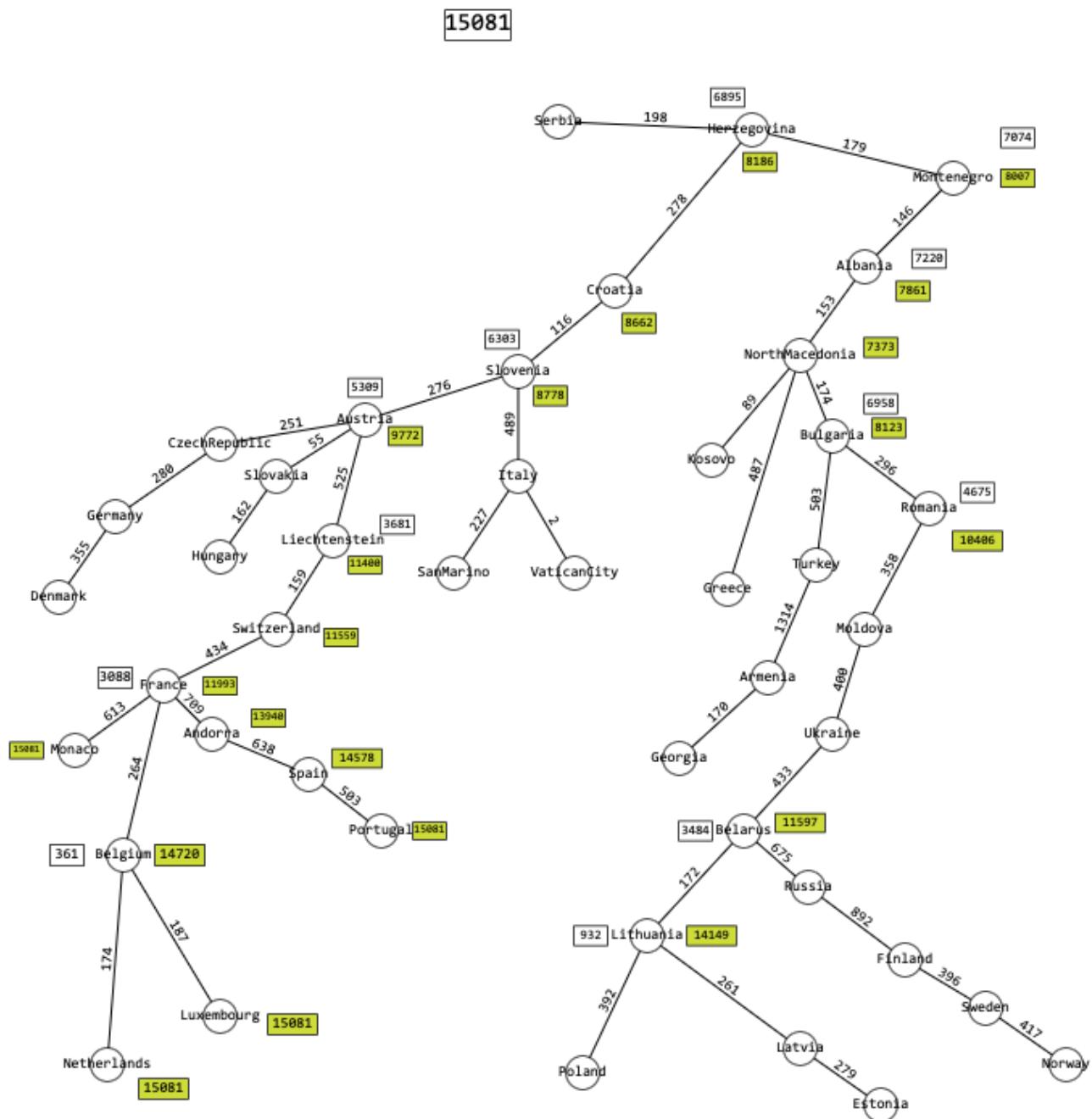


С помощью алгоритма Краскала построим минимальное остовное дерево: (тут вообще то гифка с работой алгоритма, захотите-покажу при защите) Но вообще алгоритм тривиальный, поэтому не вижу смысла его пояснять.



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

Для вершин в дереве(не для всех) подсчитан ее вес(в зеленой ячейке). Вершина с минимальным весом - центроид. В нашем случае это - North Macedonia.



(p) Construct the Prüfer code for "T .

Пронумеруем страны в алфавитном порядке:

```
map{{ x: 0, y: "Armenia"},{ x: 1, y: "Albania"},  
 { x: 2, y: "Andorra"},{ x: 3, y: "Austria"},  
 { x: 4, y: "Belarus"},{ x: 5, y: "Belgium"},  
 { x: 6, y: "Herzegovina"},{ x: 7, y: "Bulgaria"},  
 { x: 8, y: "Croatia"},{ x: 9, y: "CzechRepublic"},  
 { x: 10, y: "Denmark"},{ x: 11, y: "Estonia"},  
 { x: 12, y: "Finland"},{ x: 13, y: "France"},  
 { x: 14, y: "Germany"},{ x: 15, y: "Georgia"},  
 { x: 16, y: "Greece"},{ x: 17, y: "Hungary"},  
 { x: 18, y: "Italy"},{ x: 19, y: "Kosovo"},  
 { x: 20, y: "Latvia"},{ x: 21, y: "Liechtenstein"},  
 { x: 22, y: "Lithuania"},{ x: 23, y: "Luxembourg"},  
 { x: 24, y: "Moldova"},{ x: 25, y: "Monaco"},  
 { x: 26, y: "Montenegro"},{ x: 27, y: "Netherlands"},  
 { x: 28, y: "NorthMacedonia"},{ x: 29, y: "Norway"},  
 { x: 30, y: "Poland"},{ x: 31, y: "Portugal"},  
 { x: 32, y: "Romania"},{ x: 33, y: "Russia"},  
 { x: 34, y: "SanMarino"},{ x: 35, y: "Serbia"},  
 { x: 36, y: "Slovakia"},{ x: 37, y: "Slovenia"},  
 { x: 38, y: "Spain"},{ x: 39, y: "Sweden"},  
 { x: 40, y: "Switzerland"},{ x: 41, y: "Turkey"},  
 { x: 42, y: "Ukraine"},{ x: 43, y: "VaticanCity"}};
```

Тогда код прюфера для него:

[14, 20, 9, 3, 0, 41, 28, 36, 28, 22, 5, 13, 5, 13, 39, 22, 4, 38, 18, 6, 3, 2, 13,
 40, 12, 33, 4, 42, 21, 3, 37, 7, 24, 32, 7, 28, 1, 26, 6, 8, 37, 18]

№2

Prove rigorously the following theorems:

Theorem 1 (Triangle Inequality). For any connected graph $G = \langle V, E \rangle$:
 $\forall x, y, z \in V : \text{dist}(x, y) + \text{dist}(y, z) \geq \text{dist}(x, z)$

Proof:

Th 1 for any connected graph $G = \langle V, E \rangle$
 $\forall x, y, z \in V : \text{dist}(x, y) + \text{dist}(y, z) \geq \text{dist}(x, z)$

Доказываем обратное: $\text{dist}(x, y) + \text{dist}(y, z) < \text{dist}(x, z)$

Предположим
Т.к. $\text{dist}(x, y)$ - кратчайшее расстояние от x до y ,
а $\text{dist}(y, z)$ - кратчайшее от y , до z , то
 $(\text{dist}(x, y) + \text{dist}(y, z))$ - кратчайшее расстояние
от x до z , проходя через y .
но $\text{dist}(x, z)$ - это тоже кратчайшее расстояние от x до z
но же мы имеем противоречие, т.к. * Противоречие

Theorem 2 (Tree). A connected graph $G = \langle V, E \rangle$ is a tree (i.e. acyclic graph) iff $|E| = |V| - 1$.

Th 2 (Tree)

A connected graph $G = \langle V, E \rangle$ is a tree (i.e. acyclic graph) iff $|E| = |V| - 1$

□ Сначала докажем, что если G -дерево, то $|E| = |V| - 1$

По индукции:

1) Для дерева с 1 вершиной - орех
 $O = 1 - 1$

2) ~~Доказать~~ Есть дерево с n вершинами.

Уделим 1 место и изъедем из него ребро
Получим граф G' на $n-1$ вершине. Для
которого верно $|E'| = |V'| - 1$ (*)

Следовательно в G добавим:

$$|E| = |E'| + 1, \quad |V| = |V'| + 1$$

$$\text{Тогда } * = |E| - 1 = |V| - 1 - 1$$

$$|E| = |V| - 1 \quad \text{доказано}$$

Теперь докажем \leftarrow обратную сторону.

От противного, пусть $|E| = |V| - 1$, но
граф G - не дерево. (но узлы не рассматриваются
только связные) Значит в графе есть цикл.

Уделим какое-либо ребро в цикле. Связность
не нарушится, а цикл пропадет. Повторим это до

того момента пока не пройдут никакие
граф становят деревом. и где него станет
вполнеимес $|E'| = |V| - 1$. Но

$|E'| < |E|$, а мы знаем, что

REP

$|V| - 1 < |V| - 1$ — неверное высказывание.

Theorem 3 (Hall). For any bipartite graph $G = (X, Y, E)$, there exists X-perfect matching (set of disjoint edges covering all vertices in X) iff $|N(W)| \geq |W|$ for every $W \subseteq X$.

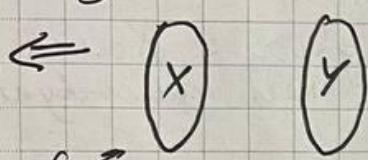
Th 3 (Hall)

for any bipartite graph $G = (X, Y, E)$, there exists X-perfect matching (set of disjoint edges covering all vertices in X) iff $|N(w)| \geq |w|$

For every $w \subseteq X$

\Rightarrow Если есть X-perfect matching, то оно покрывает вершины в w образом ребра в uniquely (таким образом нет, т.к. это matching)

А значит $|N(w)| \geq w$



Речь есть связный граф, где ребра gone - X , правая - Y . Имеет ли это выполнение:

$$\forall w \subseteq X \quad |w| \leq |N(w)| \Rightarrow |X| \leq |Y|$$

Доказательство индукции: $|X|=k$

a) при $k=1$

То $|X|=1$ и $|N(x)| \geq 1$ (т.к. выполнено $\forall w \subseteq X \quad |w| \leq |N(w)|$)

Проведен какое-либо ребро из X в Y . То и будем X-perfect matching.

b) Доказать что если, что для $< k$ выполнено

рассмотрим ребро xy ($x \in X, y \in Y$)

Пусть $H =$ граф, полученный удалением из G x, y и несуществующих ребер.

также: Для H выполнено: $\forall w \subseteq (X-x) \quad |w| \leq |N_H(w)|$

Тогда по предположению индукции в H существует X_H -perfect matching. Тогда в исходном графике G в matching добавим ребро xy . Это и будет X -perfect matching

Лемма: Условие 1 не выполняется.
 (1) Аддукт $\exists S \subseteq (X - x)$ $|N_H(S)| < |S| \Rightarrow |S| > |N_H(S)| \geq 0$

Рассмотрим граф F_1 , состоящий из $S \cup N_G(S)$ ($F_1 \leq G$)
 (2) т.к. в исходном графе G выполнено $|S| \leq |N_G(S)|$, то
 и в F_1 это выполняется.

из (1) и (2): $0 \leq |N_H(S)| < |S| \leq |N_G(S)|$ (**)

т.к. и если получено в G удаление 1 вершины, то
 $N_H(S) = N_G(S)$ или $N_H(S) = N_G(S) - 1$
 Числовая (**): $|S| = |N_G(S)|$ (***)

К тому же в F_1 : $|S| < |X|$, т.к. $S \subset (X - x)$

Аддукт $|S| \leq k$. и это не выполнение условия Хамильтона
 и по предположению индукиции в нем существует
 perfect matching

Рассмотрим граф F_1 (подграф G), такой, что
 в F_1 все вершины $(X - S)$, а во втором $(Y - N_G(S))$
 существует:

$$|X - S| < |X| = k, \text{ т.к. } S \geq 0 \text{ из (1)}.$$

Рассмотрим $S' \subseteq (X - S)$

(3) $|S \cup S'| \leq |N_G(S \cup S')|$ (т.к. это выполняется в G)

т.к. $S \cap S' = \emptyset$, то $|S \cup S'| = |S| + |S'|$

т.к. $N_G(S) \cap N_{F_1}(S') = \emptyset$, то $|N_G(S \cup S')| = |N_G(S)| + |N_{F_1}(S')|$

то переходим (3):

$$|S| + |S'| \leq |N_G(S)| + |N_{F_1}(S')|$$

$$(* *) |N_G(S)| + |S'| \leq |N_G(S)| + |N_{F_1}(S')|$$

$$|S'| \leq |N_{F_1}(S')|$$

Theorem 4 (Whitney). For any graph G : $\kappa(G) \leq \lambda(G) \leq \delta(G)$.

Th 4 (Whitney)

for any graph G : $\kappa(G) \leq \lambda(G) \leq \delta(G)$

Докажем, что $\lambda(G) \leq \delta(G)$.

т.к. в графе существует вершина степени $\delta(G)$,
значит из неё выходят $\delta(G)$ ребер, удалив их
получим несвязный граф. А значит удаление
больше ребер, чем $\delta(G)$ не обеззащищено для
того, чтобы граф стал несвязным $\Rightarrow \lambda(G) \leq \delta(G)$

Докажем, что $\kappa(G) \leq \lambda(G)$

• Для полного графа $\kappa(G) = n-1$ (тривиальный),

а $\lambda(G) = n-1$ (от одной вершины убираем все ребра)

• Если граф пасынок или тривиальный: $\kappa(G) = \lambda(G) = 0$

• Если в графе есть мост, то $\lambda(G) = 1$.

Если есть мост, то это либо K_2 , либо граф,
содержащий только соединение. В обоих случаях:

$\kappa(G) = 1$

• Для всех остальных случаев $\lambda(G) \geq 2$. (т.е.
чтобы удалить ≥ 2 ребер, чтобы нарушить связность)
Удалив $\lambda(G)-1$ ребер, получим, что оставшееся ребро —
мост между u и v .

Тогда будем доказывать, что какое ребро из $\lambda(G)-1$
удаляется в минимальной вершине синек вершину
исходящую этому ребру (это будет работать, т.к.
удалит эту вершину, так удалим также ребро).

И тогда у нас останется мост — удалим мост и либо u
либо v У нас получим $\kappa(G) \leq \lambda(G)$

Theorem 5 (Chartrand). For a connected graph $G = \langle V, E \rangle$: if $\delta(G) \geq \lfloor |V|/2 \rfloor$, then $\lambda(G) = \delta(G)$.

Th 5 (Chartrand)

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

if $\delta(G) \geq \lfloor |V|/2 \rfloor$, then $\lambda(G) = \delta(G)$

□ Докажем, что если $\delta(G) \geq \lfloor |V|/2 \rfloor$, то $\lambda(G) = \delta(G)$.
Доказательство: разделим вершины на 2 компоненты.

S -меньшая. $|S| = k$. т.к. S -меньшая, то $k \leq \lfloor n/2 \rfloor$

Чтобы сделать граф связным, надо удастить все ребра между S и $V \setminus S$. Каждая вершина в S соединена как минимум с $\delta(G)$ вершинами.
Но более $(k-1)$ (помимо S) членов в $V \setminus S$, а значит $\delta(G) - k + 1$ тоже членов в $V \setminus S$, где одной вершиной. А значит не менее $(\delta(G) - k + 1)$ рёбер идёт из S в $V \setminus S$.

$\lambda(G) \geq$ минимальное значение $k \cdot (\delta(G) - k + 1)$

на промежутке $k \in [1, n/2]$

т.к. это квадратичная функция ветвей в $V \setminus S \Rightarrow$ минимум при расщеплении на полоухах:

$$x = \frac{\delta(G) - n/2}{2}$$

$$\text{т.к. } k \cdot (\delta(G) - k + 1) = \delta(G) \geq \lfloor |V|/2 \rfloor + x$$

$$\text{т.к. } n/2 \cdot (1 + x) \geq \lfloor |V|/2 \rfloor + \lfloor |V|/2 \rfloor \cdot x$$

Значение в 1 вершина \geq значение в $n/2$, т.к.

$$\lfloor |V|/2 \rfloor + x > \lfloor |V|/2 \rfloor + \lfloor |V|/2 \rfloor \cdot x$$

$$x \geq \lfloor |V|/2 \rfloor \cdot x \text{ справедливо только если } V=1$$

но это противоречие, значит $\lambda(G) = \delta(G)$

$\lambda(G) \geq \delta(G)$ но это не верно по Th (Whitney):

$\lambda(G) \leq \delta(G)$ а значит $\lambda(G) = \delta(G)$



Theorem 6 (Menger). For any pair of non-adjacent vertices u and v in an undirected graph, the size of the minimum vertex cut is equal to the maximum number of pairwise internally vertex-disjoint paths from u to v .

Th6 (Menger)

for any pair of non-adjacent vertices u and v in an undirected graph the size of minimum vertex cut is equal to the maximum number of pairwise internally vertex-disjoint paths from u to v

□ Достаточно если к вершинам разделяющим u и v , то существует ли более k непересекающихся путей от u до v . Иначе просто пронумеруем эти пути. Рассмотрим возможный для P_{k+1} вершину x_{k+1} , получив противоречие. Остается показать, что к путям существуют.

$\partial K = 1$ — это очевидно

Предположим для некоторого числа $k \geq 1$ это неверно. Тогда пусть k — наибольшее из них, граф F — граф с наибольшим числом вершин, для которого при указанных в теореме не выполняется. Будем удалять из F вершины до тех пор, пока не получим такой граф G , что для разделяния вершин u и v требуется $k+1$ и $k+1$ для разделяния вершин u и v требуется $k+1$ и $k+1$.

У G определение графа G не меняется, так как все его ребра x существуют независимо $S(x)$, содержащее $k+1$ вершин, которое в G разделяет u и v . Далее граф $G - S(x)$ содержит хотя бы один путь от u до v , т.к. граф G не имеет $k+1$ вершин, разделяющих u и v . Этот путь должен содержать ребро $x(a-b)$, т.к. оно не удаляется в $G - S(x)$. Поэтому $a, b \notin S(x)$ и если $a \neq v, u$, то $S(x) \cup \{x\}$ разделяет u и v в G .

Если $b \in G$ есть вершина w , смежная как с u , так и с v , то в графе $G - w$ для разделяния u и v требуется $k+1$ вершин и поэтому в нем $k+1$ непересекающихся путей из u и v . Добавив w , получим в графе G $k+1$ непересекающихся путей. Значит для него все хорошо, он не удовлетворяет условию для F .

(1) Значит в графе G есть вершины, смежные с u и v .

Пусть W -произвольный набор h -вершин, разделяющих U и V в G .

P_U -набор путей из U в $W; EW$ (не содержит $w_j \in V$)
 P_V -набор путей из $W; EW$ в V (не содержит $w_j \in U$)

Тогда каждый путь из U в V начинается с элементом из P_U и заканчивается элементом из P_V , поскольку любой такой путь содержит вершину из W . Обеие вершины путей из P_U и P_V принадлежат набору W , так как по крайней мере один путь из каждого набора P_U и P_V содержит (любую) вершину W , и если бы существовала бы некоторая вершина не принадлежащая набору W , то содержалась сразу и в U и в V ; но начиная с этого пути и т.д., не имеющий вершин в W .
Наконец, выполняется либо $P_U - W = \emptyset$, либо

$P_U - W = \{v\}$, поскольку в противном случае либо P_U вместе с ребром $\{w_1, v, w_2\} \dots v$, либо P_V вместе с ребром $\{v, w_1, w_2\} \dots v$ образуют связные графы с меньшим числом вершин, чем в G , в которых U и V не имеют и, следовательно, в каждом из них имеется h непересекающихся путей из U в V . Оговаривая U и V ; $w_i \in V$ гамильтоновы пути, образуем в графе G h непересекающихся путей из U в V . Но привели к противоречию. Доказали, что

(2) любой набор W , содержащий h вершин и разделяющий U и V , имеет шесть или 7 вершин с V

Закончим доказательство. Пусть $P = \{u, a, b, \dots, v\}$ - кратчайший из U в V в G , X -ребро ($a \cdot b$). Заметим, что в G не меньше $|V| - 1$. Образует множество $S(x) = \{m_1, m_2, \dots, m_{n-1}\}$, разделяющее в $G - x$ вершину u и v .
из (1) $(u, t) \notin G$. Используя (2) берем набор $W = S(x) \cup \{u, v\}$, помножаем $Um; EG$. из (1) : $m; V \notin G$
Однако, если выбрать $W = S(x) \cup \{u_2\}$, то из (2) : $Um; EG$, это противоречит выбору P , как кратчайшему пути.

Итак, мы получим, что графа G , удовлетворяющего указанным выше требованиям не существует. Следовательно, не существует и графа F , для которого h вершин не верна

Theorem 7 (Harary). Every block of a block graph is a clique.

Th 7 (Harary)

Every block of a block graph is a clique.
 $H=B(G)$

□ Предположим, что в H есть блок H_i , который не является кляквой. Тогда в H_i есть пара несмежных вершин (блоков в G), несмежных на одной простой цепи γ , длина которой ≥ 4 . (т.к при ≤ 3 они были бы смежны)

Следовательно, обединение блоков графа G , соответствующих вершинам из H_i , не падает на γ , я видим связный график, не имеющий тогея соглашения. т.е это обединение содержит в некотором блоке, что противоречит свойству максимальности блока графа

Пришли к противоречию, а значит предположение о том, что H_i не является кляквой - неверно.

