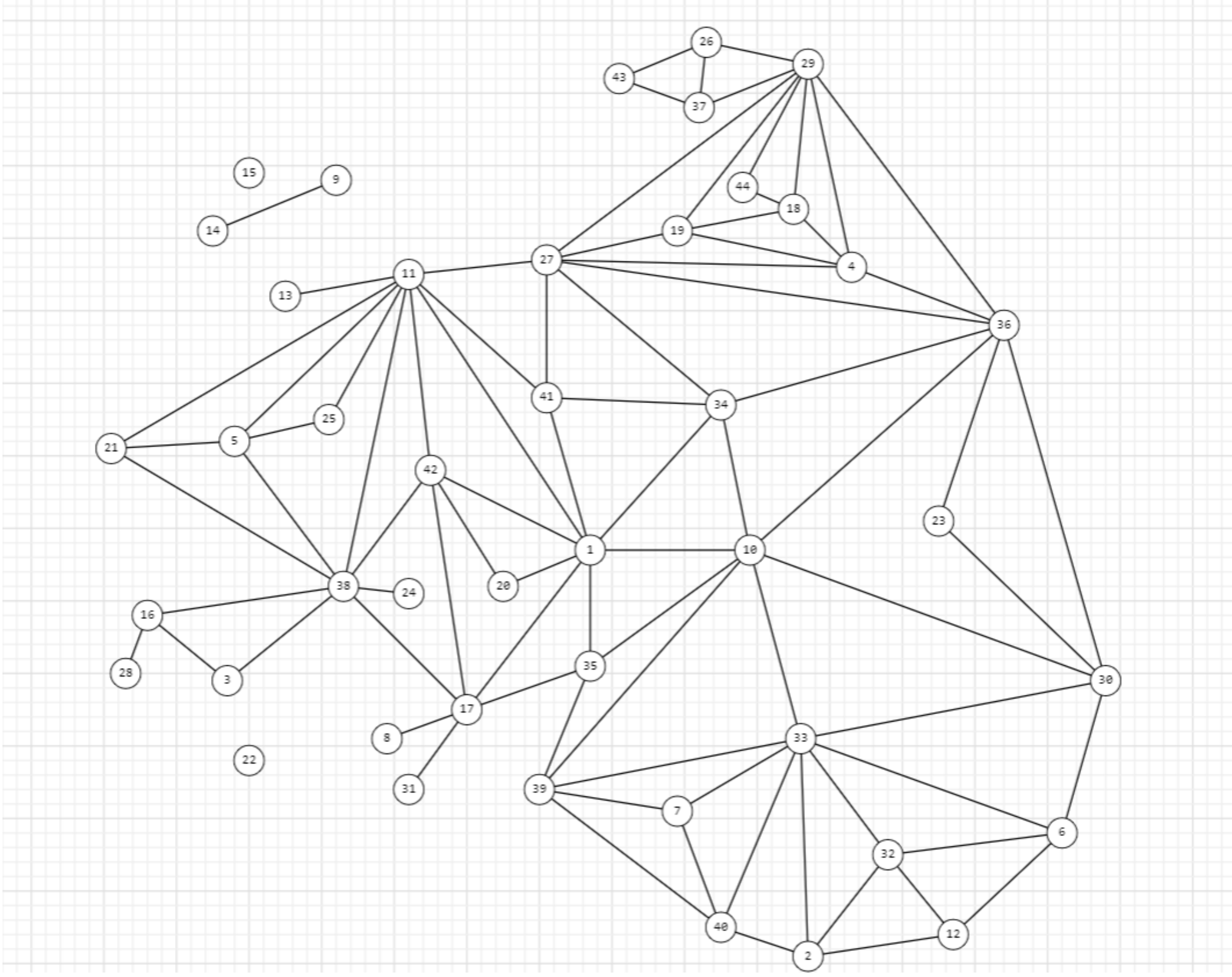
№1.

a)



b) $|V| = 44$

$|E| = 83$

$\delta(G) = 1$ (вершина 28)

$\Delta(G) = 9$ (вершина 11)

$rad(G) = 4$

$diam(G) = 8$

$girth(G) = 3$ (например, цикл из вершин 26, 37, 43)

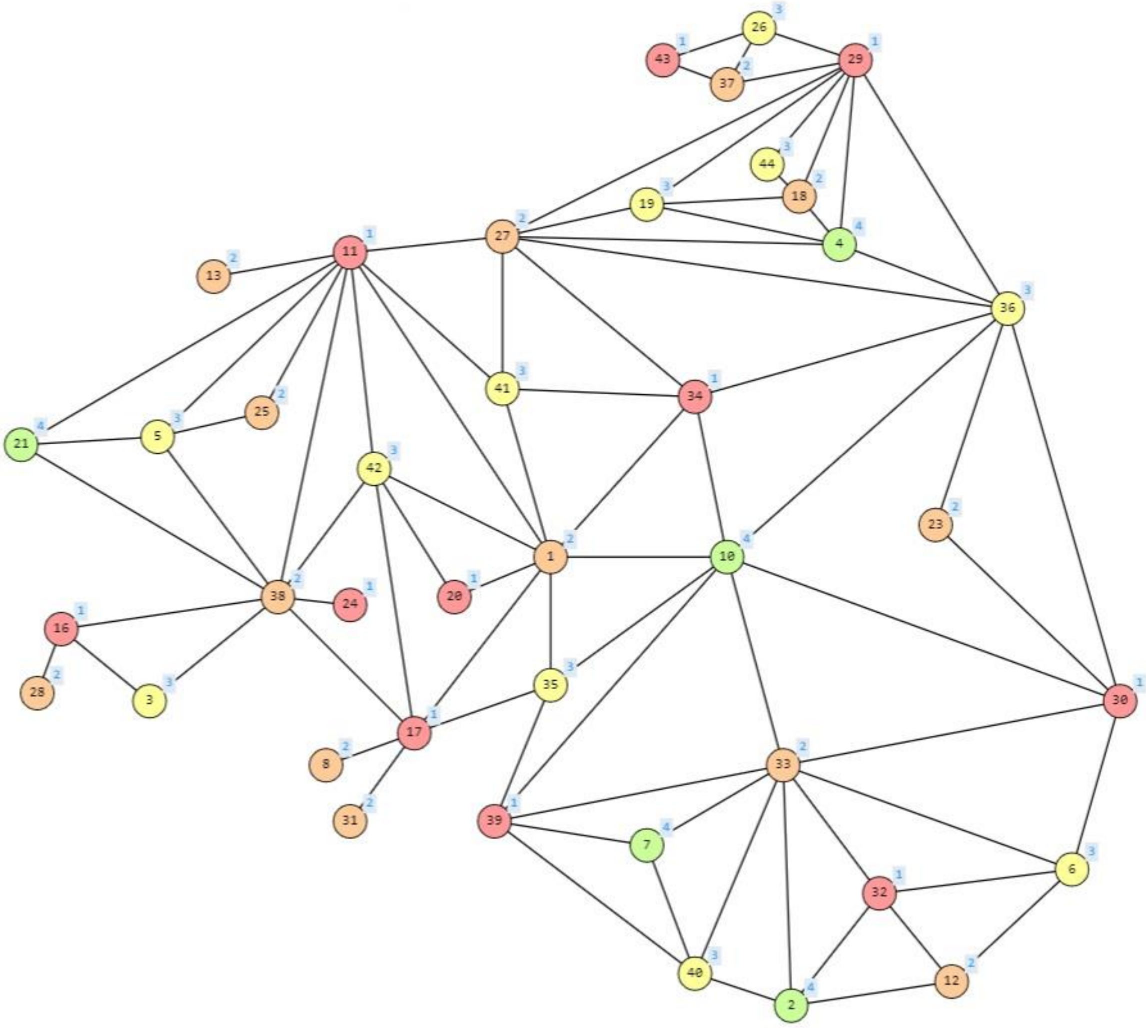
$center(G) = \{27\}$

$\kappa(G) = 1$ (есть точка сочленения 29)

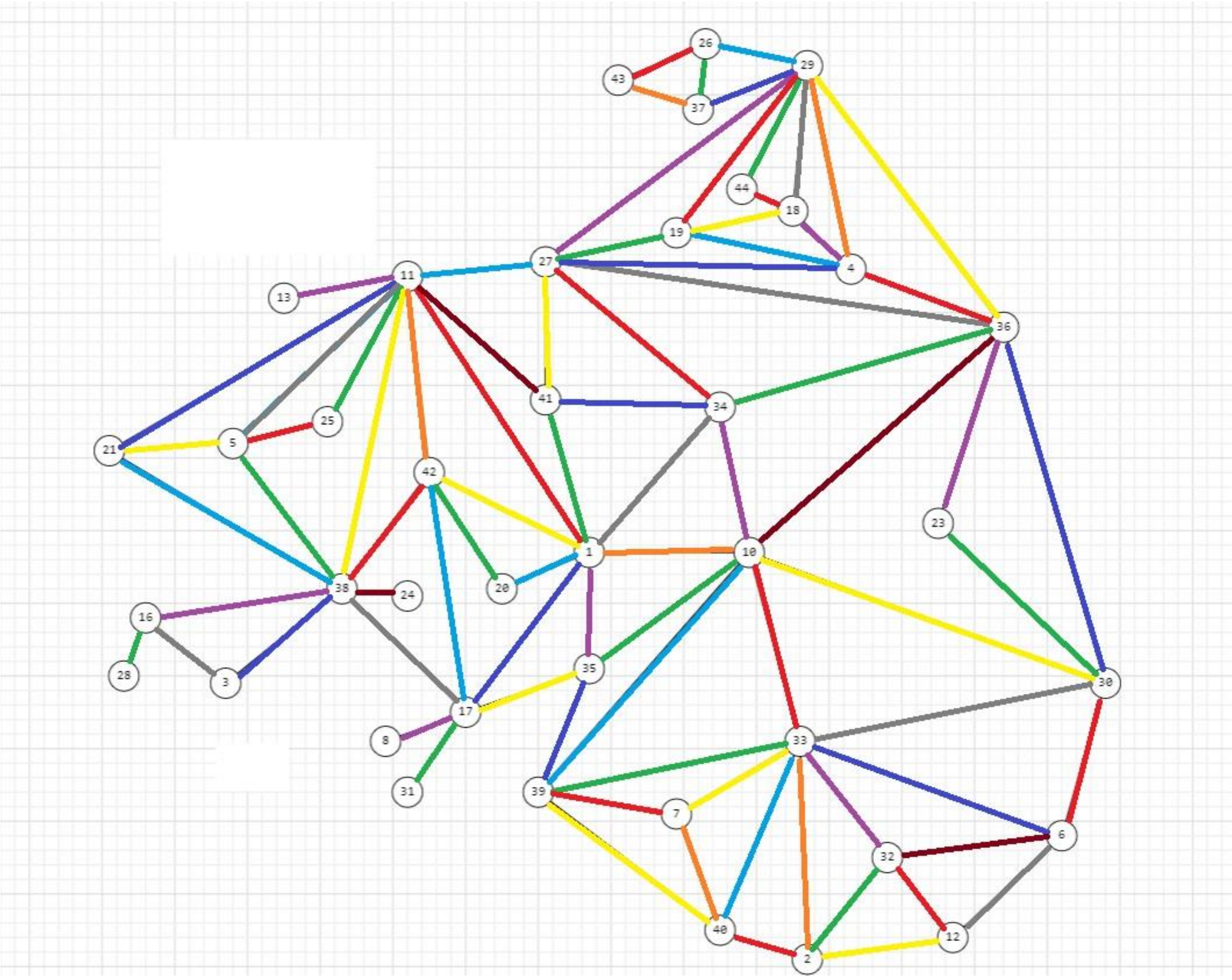
$\lambda(G) = 1$ (есть мост 11-13)

```cpp
vector<int> bfs(vector<vector<int>>& graph, int start) {
    vector<int> distances(graph.size(), -1);
    queue<int> q;
    q.push(start);
    distances[start] = 0;
    while (!q.empty()) {
        int current_node = q.front();
        q.pop();
        for (int neighbor = 0; neighbor < graph.size(); neighbor++) {
            if (graph[current_node][neighbor] == 1 && distances[neighbor] == -1) {
                distances[neighbor] = distances[current_node] + 1;
                q.push(neighbor);
            }
        }
    }
    return distances;
}

int find_radius(vector<vector<int>>& graph) {
    int n = graph.size();
    int radius = INT_MAX;
    for (int i = 0; i < n; i++) {
        vector<int> distances = bfs(graph, i);
        int max_distance = *max_element(distances.begin(), distances.end());
        radius = min(radius, max_distance);
    }
    return radius;
}

int find_diameter(vector<vector<int>>& graph) {
    int n = graph.size();
    int diameter = 0;
    for (int i = 0; i < n; i++) {
        vector<int> distances = bfs(graph, i);
        int max_distance = *max_element(distances.begin(), distances.end());
        diameter = max(diameter, max_distance);
    }
    return diameter;
}

int main() {
    int n;
    cout << "n: ";
    cin >> n;
    vector<vector<int>> graph(n, vector<int>(n));
    cout << "matrix: " << endl;
    for (int i = 0; i < n; i++) {
        char c;
        for (int j = 0; j < n; j++) {
            cin >> graph[i][j];
            if (j != n - 1) {
                cin >> c;
            }
        }
    }
    int radius = find_radius(graph);
    int diameter = find_diameter(graph);
    cout << "rad: " << radius << endl;
    cout << "diam: " << diameter << endl;
```

c) Минимальное количество цветов – 4, т.к. есть клика {39, 33, 7, 40} из 4 вершин, и каждая должна иметь свой уникальный цвет. Раскраска для 4:

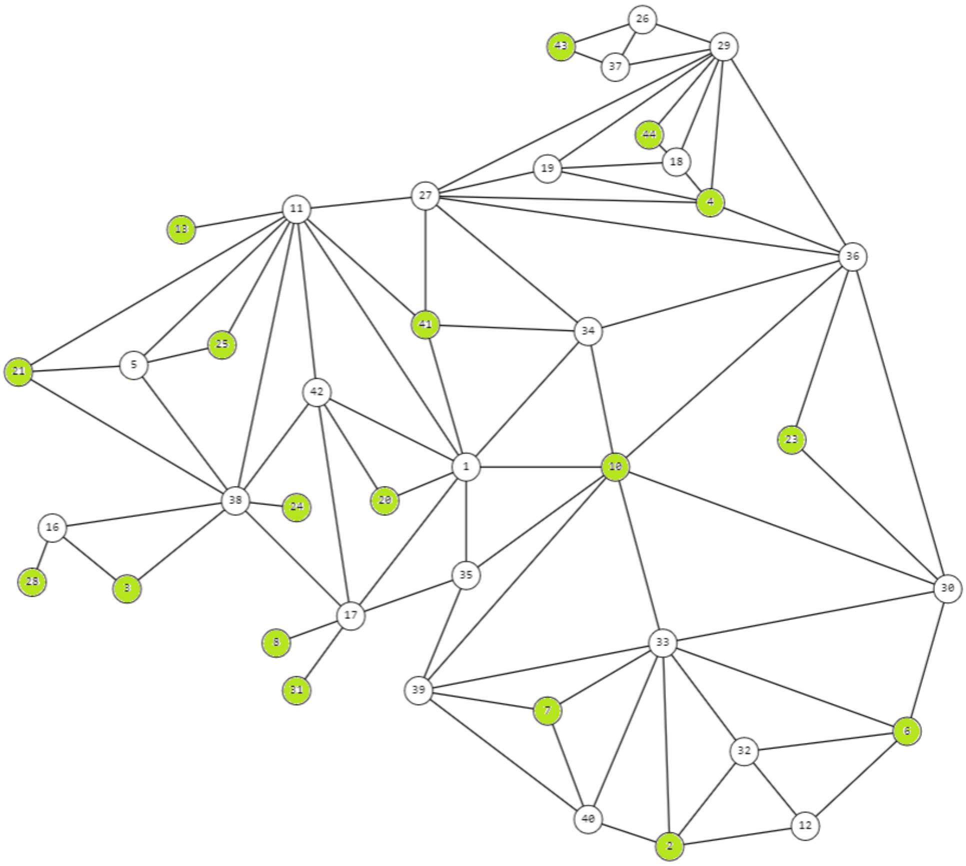


d) Минимальное количество цветов – 9, т.к. максимальная степень вершины – 9 (вершина 11). Раскраска для 9 цветов:



e) Наибольшая клика – {39, 33, 7, 40} размера 4. Большего размера быть не может, т. к. граф планарен

f) Наибольший stable set состоит из 18 вершин

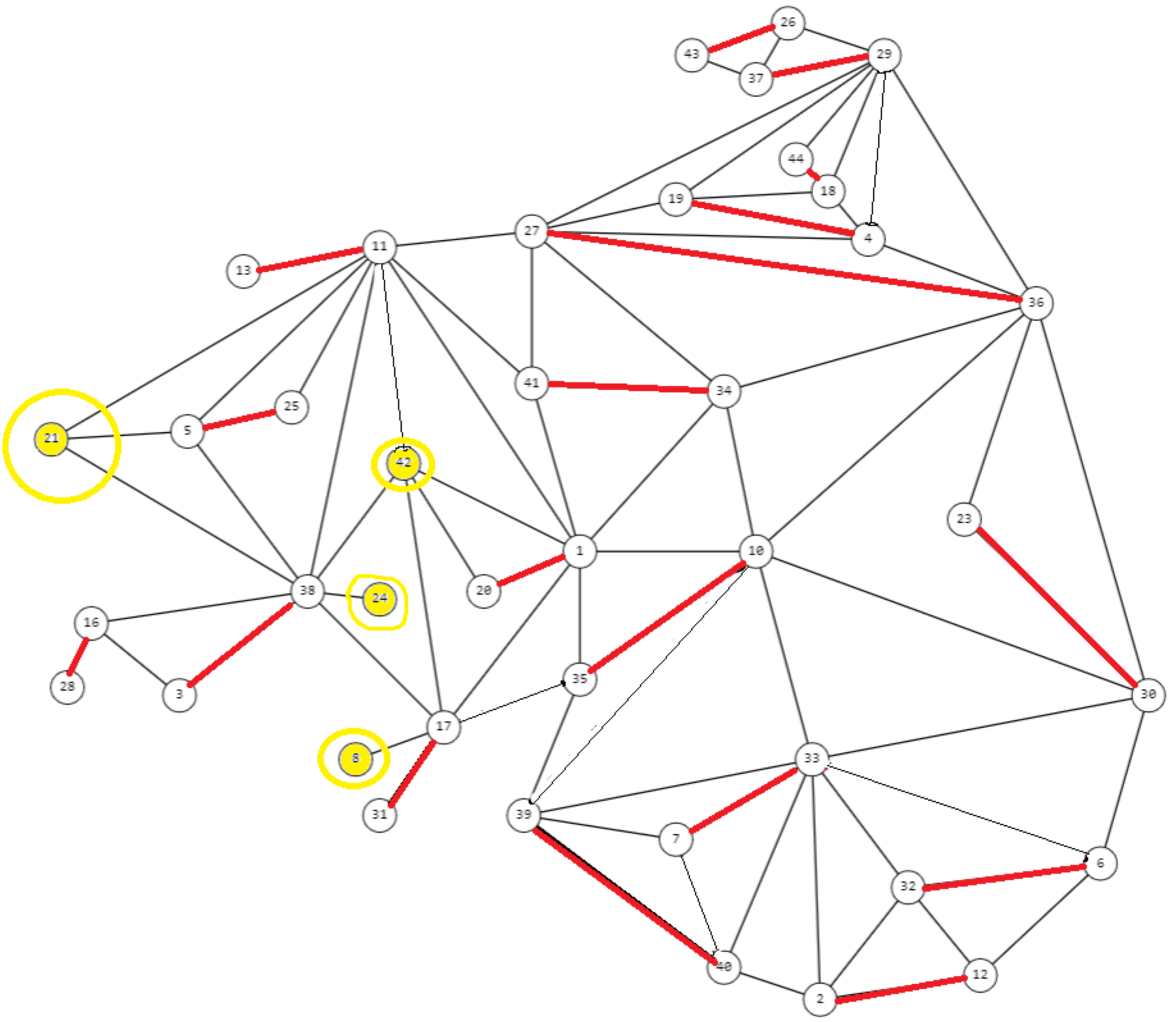


Код на C++:

```
const int n = 40;
int adjMatrix[n][n];

bool isStableSet(const vector<int>& nodes) {
    for (int i = 0; i < nodes.size(); i++) {
        for (int j = 0; j < nodes.size(); j++) {
            if (adjMatrix[nodes[i]][nodes[j]]) {
                return false;
            }
        }
    }
    return true;
}

signed main() {
    for (int i = 0; i < n; i++) {
        char c;
        for (int j = 0; j < n; j++) {
            cin >> adjMatrix[i][j];
            if (j != n-1) {
                cin >> c;
            }
        }
    }
    cout << endl;

    int maxSize = 0;
    vector<int> bestSet;
```

```
    for (int i = 1; i < (1ull << n); i++) {
        vector<int> currSet;
        for (int j = 0; j < n; j++) {
            if (i & (1ull<<j)) {
                currSet.push_back(j);
            }
        }
        if (isStableSet(currSet) && currSet.size() > maxSize) {
            maxSize = currSet.size();
            bestSet = currSet;
        }
    }

    cout << endl << bestSet.size() << endl;
    for (int i = 0; i < bestSet.size(); i++) {
        cout << bestSet[i] << " ";
    }
    cout << endl;
```

g) Наибольший matching состоит из 18 ребер



Вершина 8 либо 31 не будет включена, т. к. обе из них могут быть только с 17 в паре. Вершина 28 может быть в паре только с 16, 3 только с 38 => для 24 не остается пары. 13 может быть только с 11, 38 занято, то есть либо не включаем 21, либо 25. (т. к. каждая из

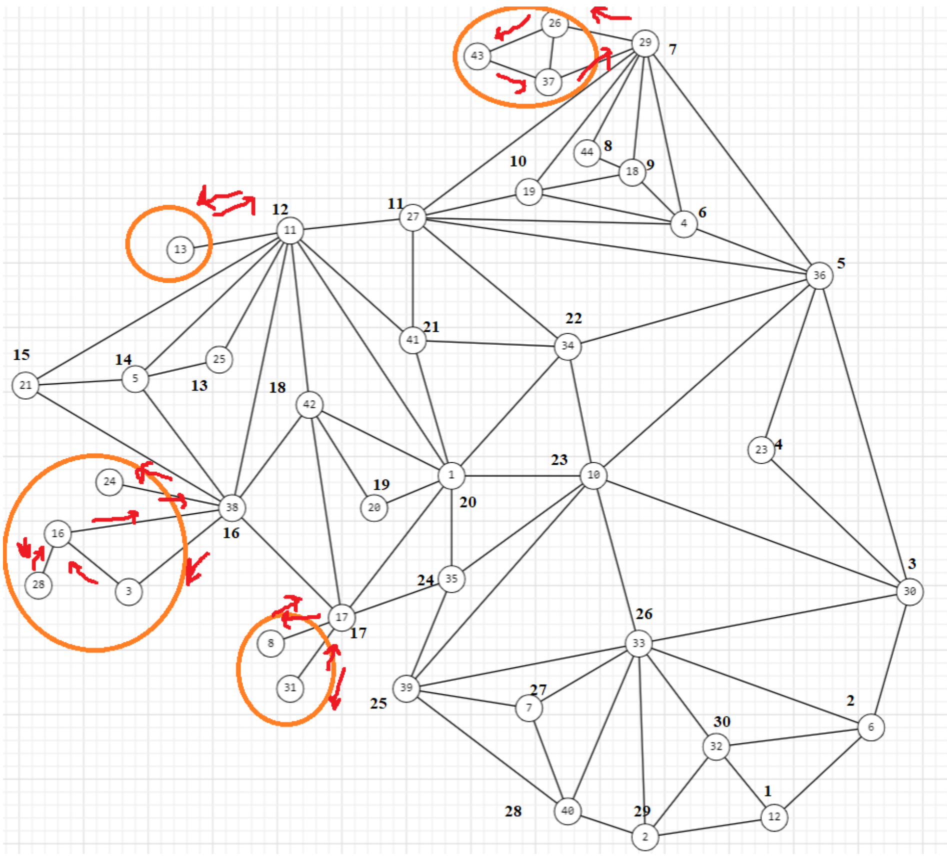
них может быть только с 5). 42 не может быть ни с какой другой вершиной, потому что они уже имеют пару. 18*2 + 4 не включенные вершины = 40 вершин всего.

h) Наименьшее вершинное покрытие: 22

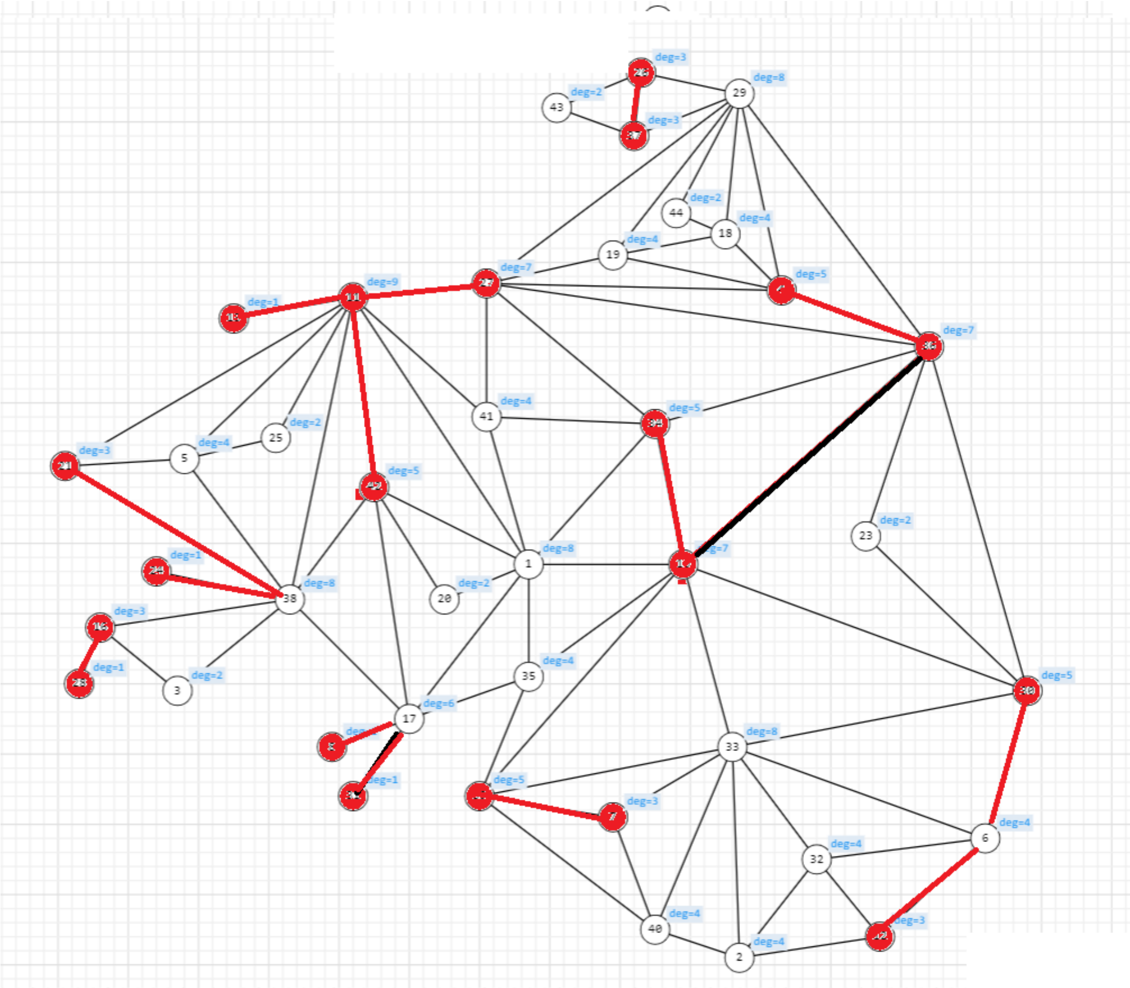
По теореме о связи вершинного покрытия и независимого множества дополнение наименьшего вершинного покрытия является наибольшим независимым множеством. Т. к. наибольшее независимое множество состоит из 18 вершин, то наименьшее вершинное покрытие будет состоять из 40–18 = 22 вершин.

g) Наименьшее реберное покрытие: 22. Возьмем каждое ребро, входящее в наибольшее паросочетание (каждое из этих ребер покроет ровно по 2 вершины) и возьмем еще 4 ребра, чтобы покрыть вершины, не покрытые ребрами паросочетания. Получится 22 ребра

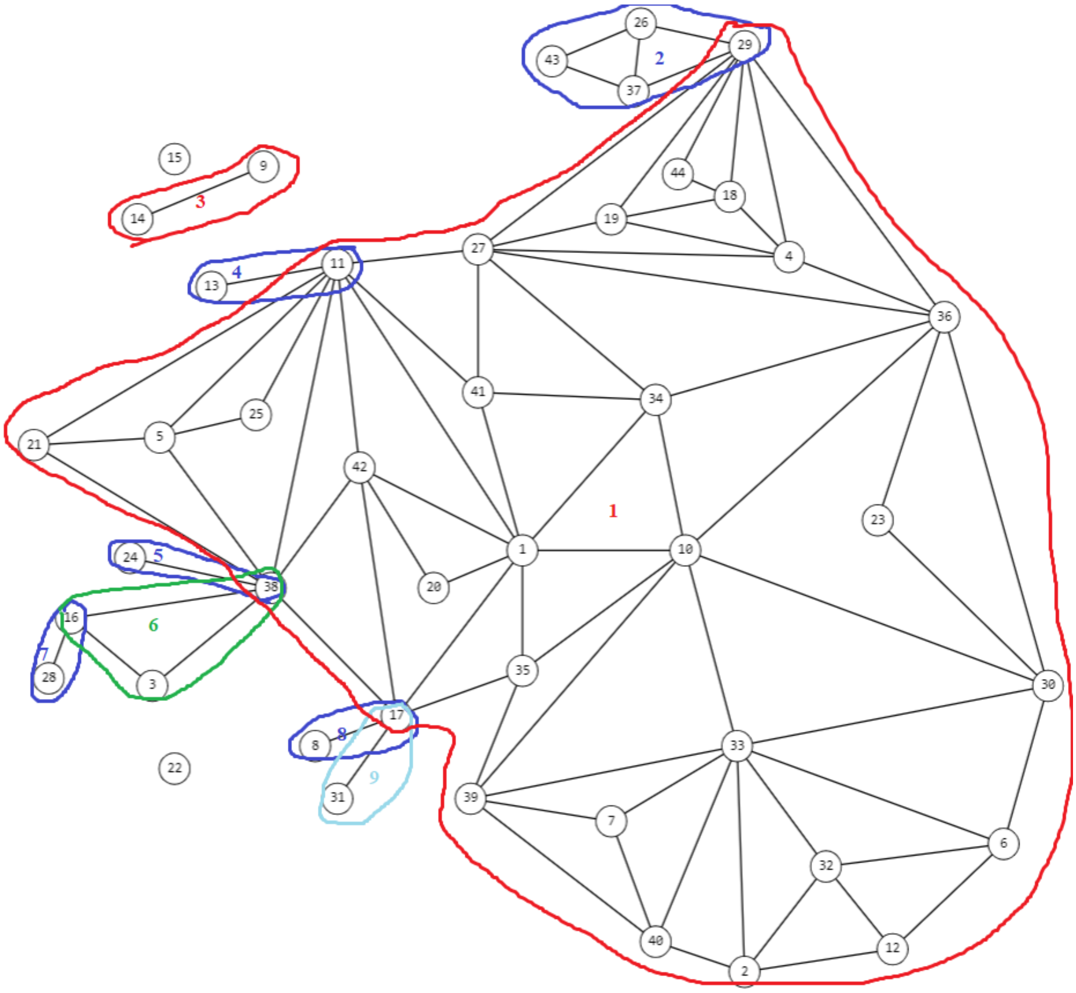
j) Путь в 47 ребер. Номера стоят у вершин, составляющих гамильтонов цикл. Обведенные части графа мы можем посетить, только дважды пройдя точку сочленения. 30 ребер в гамильтоновом цикле + 17 ребер, чтобы пройти обведенные части



k) Количество ребер в графе: 82. В графе возьмем максимальный подграф, где все степени вершин четны (значит есть эйлеров цикл). В таком подграфе 68 ребер. + 2 раза пройдем по каждому ребру, которое удалили. 68 + 2*14 = 96 ребер в пути.

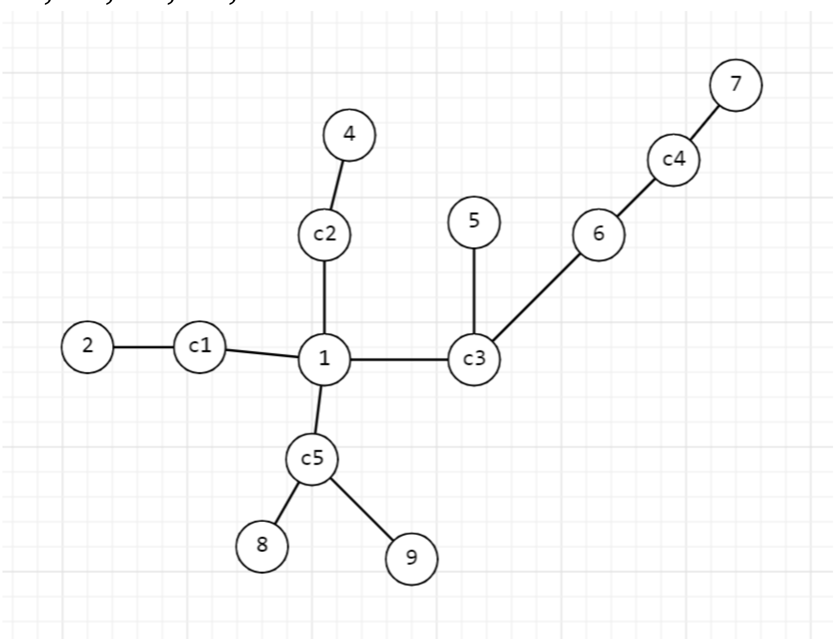


l) Блоки:

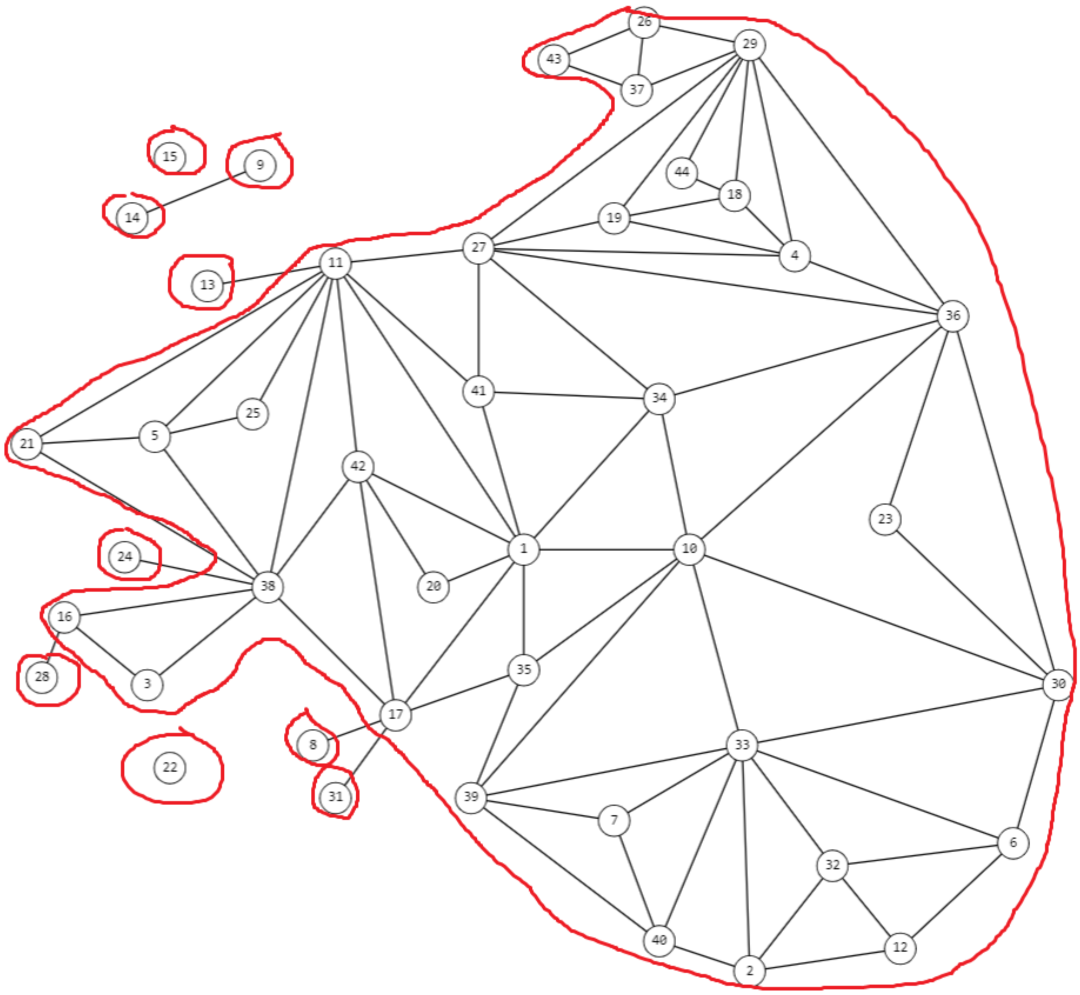


Block-cut tree:

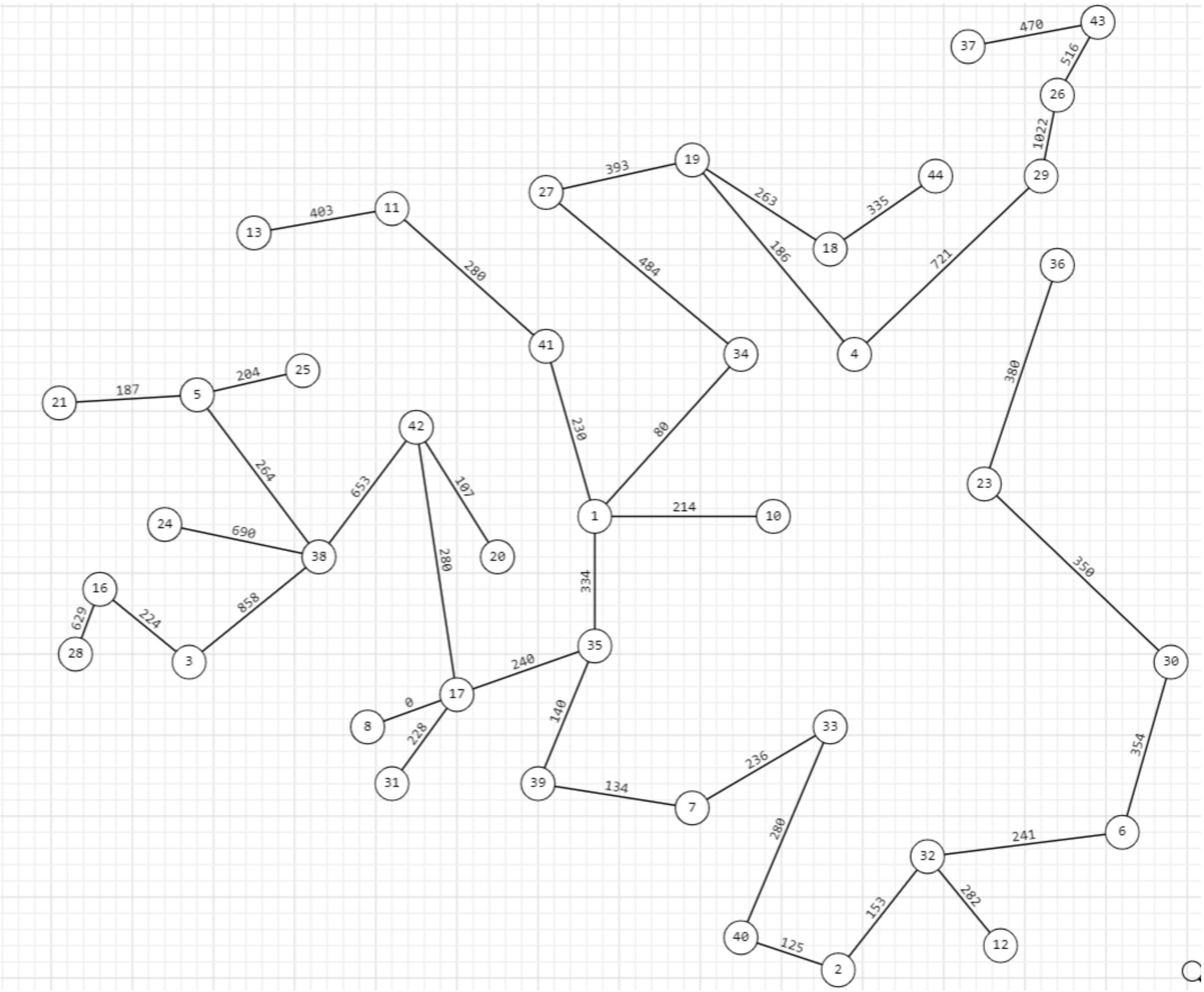
c1, c2, c3, c4, c5 – точки сочленения



m) Компоненты реберной 2-связности:



n) Наименьшее по весу ребер остовное дерево веса 13170 (используя алгоритм Краскала):



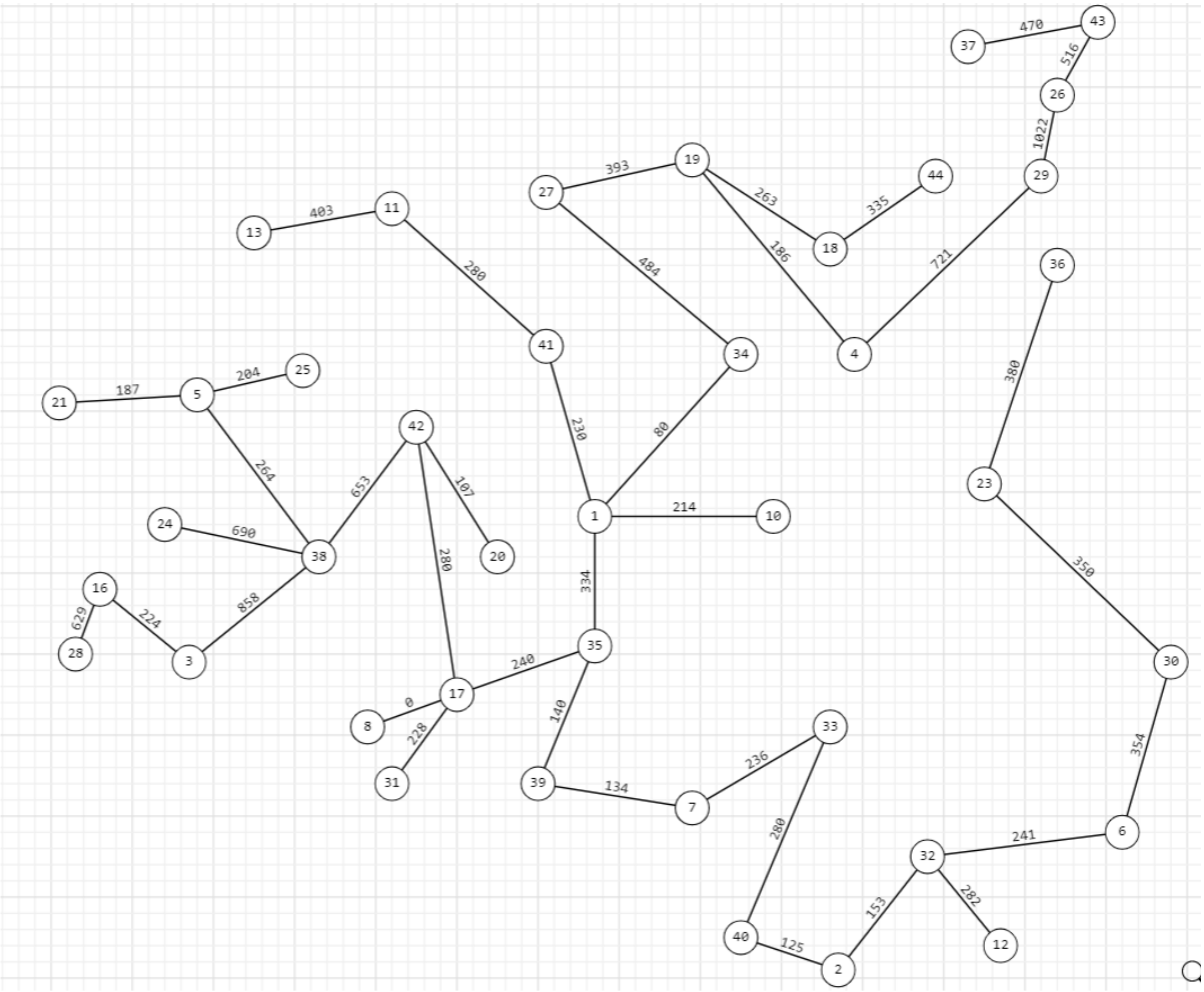
Алгоритм состоит из следующих шагов:

1) Сортируем все ребра графа по возрастанию весов.

2) Перебираем ребра по порядку и добавляем их в остовное дерево, если они не образуют цикл с уже добавленными ребрами.

Повторяем шаг 2 до тех пор, пока не будет построено дерево, содержащее все вершины графа.

(o) Центроид дерева: вершина 27.



(p) На каждой итерации ищем лист с наименьшим номером, удаляем его и записываем в последовательность номер его родителя, пока не останется дерево из 2 вершин:

1 11 5 5 16 3 26 4 6 23 17 2 6 12 1 27 19 18 4 23 7 35 1 17 2 33 7 1 11 17 8 20 38 3 5 24 26 37

№2.

Теорема 1. Для любого связного графа $G = \langle V, E\rangle$:

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

Предположим, что есть такие x, y и z, для которых сумма расстояний от x до y и от y до z меньше расстояния от x до z. Но тогда путь от x до z не будет кратчайшим, т. к. кратчайшим будет путь из x в z через y (такой путь существует, т. к. граф связный). Пришли к противоречию, т.к. по определению dist(x, z) – длина кратчайшего пути между x и z => сумма расстояний от x до y и от y до z больше либо равна расстоянию от x до z.

Теорема 2. Связный граф $G = \langle V, E\rangle$ является деревом (т. е. ациклическим графом) тогда и только тогда, когда $|E| = |V| - 1$.

1) Докажем, что если граф является деревом, то $|E| = |V| - 1$.

Докажем по индукции. База индукции: для дерева, где $|V| = 1$, $|E| = 0$ (очевидно). Предположим, что утверждение верно для n-1 вершин. Докажем, что оно верно для n вершин. Рассмотрим произвольное дерево, выберем произвольную висячую вершину v (такая вершина в дереве всегда существует, т. к. в нем нет циклов). Удалим её вместе с ребром, которое из нее выходит. Оставшийся граф также является деревом из n-1 вершин, по предположению индукции содержит n-2 ребра. Добавим в этот граф вершину, которую мы удалили, вместе с ребром, и получим n вершин и n-1 ребро. Утверждение доказано.

2) Докажем, что если в связном графе $|E| = |V| - 1$, то такой граф – дерево.

Нужно доказать, что в таком дереве нет циклов. Если в нем есть циклы, то можно удалить 1 ребро из каждого цикла, не нарушив связность. Тогда, применив эту операцию ко всем циклам, мы получим связный ациклический граф (дерево), где |E| меньше, чем $|V| - 1$, что противоречит пункту 1. Следовательно, если в связном графе $|E| = |V| - 1$, то такой граф – дерево.

Теорема 3. Для любого двудольного G = (X, Y, E) существует X-совершенное паросочетание (set of disjoint edges covering all vertices in X) тогда и только тогда, когда $|N_G(W)| \geq |W|$ для любого $W \subseteq X$.

($N_G(W)$ – множество соседей W в графе G)

1) Если существует X-полное паросочетание, то для любого $W \subseteq X$ выполнено $|N_G(W)| \geq |W|$, т. к. у любого подмножества вершин есть хотя бы столько же соседей. Иначе нам просто не хватило бы вершин, чтобы построить паросочетание.

2) Пусть G = (X, Y, E) – двудольный граф с $|X| \leq |Y|$, где $|N_G(W)| \geq |W|$ для любого $W \subseteq X$. Докажем по индукции по $k = |X|$.

База индукции: k = 1. Возможно 2 случая: |W| = 0, тогда всегда есть X-полное паросочетание, и |W| = 1, тогда $N_G(W) \geq 1$ => просто соединяем вершину из W с каким-то соседом.

Предположим, что утверждение верно |X| < k. Докажем для |X| = k > 1.

Возьмем какую-то вершину x из множества X. Т. к. $|N_G(W)| \geq |W|$ для любого $W \subseteq X$, а $\{x\} \subseteq X$, то $|N_G(W)| \geq 1$ => есть вершина y – сосед x в Y. Пусть граф H – это граф G без вершин x и y.

1) H удовлетворяет условию Холла

Заметим, что X\{x} – меньшая доля H (потому что |X|≤|Y|, и мы из каждой доли вычли по 1 вершине). |X\{x}| < |X| < k => по предположению индукции в H есть (X-x)-совершенное паросочетание. К нему добавим ребро xy и получим X-совершенное паросочетание для G.

2) H не удовлетворяет условию Холла

Тогда $\exists\, W \subseteq X \setminus \{x\}$: $|N_H(W)| < |W|$. Рассмотрим $F_1 = G[W \cup N_G(W)] = (W, N_G(W), E')$. Т. к. G удовлетворяет условию Холла, то F_1 тоже удовлетворяет условию Холла (потому что любое $A \subseteq W$ является подмножеством X, и $N_H(A) = N_G(A)$). Т. к. |W| < |X| = k, то по предположению индукции в F_1 есть W-совершенное паросочетание M.

W ⊆ X\{x} ⊆ X => |W|≤|N_G(W)|, т. к. G удовлетворяет условию Холла

|N_H (W)| < |W| ≤ |N_G(W)|, но |N_H (W)| ∈ { |N_G(W)|, |N_G(W)| - 1 (если y, который мы удалили при построении графа H, был соседом W)}. |N_H (W)| = |N_G(W)| быть не может, значит |N_H (W)| = |N_G(W)| - 1 и |W| = |N_G(W)|.

Рассмотрим F_2 = G[X\W ∪ $Y \backslash N_G$(W)]. X\W – меньшая доля, т.к. X<Y и |W| = |N_G(W)|.

0 <= |N_H (W)| < |W| => |W| > 1. |X/W|<|X|=k, т.к. |W|>1. Пусть W’ ⊆ X\W. |W∪W’| <= |N_G(W∪W’)| (т. к. G удовлетворяет условию Холла). W’ ⊆ X\W => W’ и W не пересекаются => |W| + |W’| <= |N_G(W)| + |N_{F_2}(W’)|. Т. к. |W| = |N_G(W)|, то |W’| <= |N_{F_2}(W’)|. F_2 удовлетворяет условию Холла => по предположению индукции в F_2 есть (X\W)-совершенное паросочетание M’. Тогда M∪M’ - X-совершенное паросочетание для G.

Теорема 4. Для любого графа G: κ(G) ≤ λ(G) ≤ δ (G).

1) Докажем, что λ(G) ≤ δ (G). Пусть v – вершина с минимальной степенью вершины в графе δ (G). Тогда удалим все δ (G) инцидентных ей ребер, и граф перестанет быть связным (т.к. вершина v отделится от графа). Значит, граф максимум может быть δ (G)-реберно-связным (т. к. мы точно получим несвязный граф, удалив δ (G) ребер, но возможно обойдемся и меньшим количеством удаленных ребер). => λ(G) ≤ δ (G).

2) Докажем, что κ(G) ≤ λ(G). Если G – несвязный или тривиальный граф, то κ(G) = λ(G) = 0. Если граф связный и имеет мост, то λ(G) = 1. Тогда κ(G) = 1, потому что мы можем удалить любую вершину, инцидентную этому ребру (она является точкой сочленения), либо у нас граф K2 и удалив вершину мы получим тривиальный. Если λ(G) >= 2, то если удалим из него λ – 1 ребро и получим мост (вот почему: у нас между любыми 2 вершинами есть ровно λ реберно не пересекающихся путей. Тогда уберем из каждого λ – 1 пути одно ребро, и останется только 1 путь, который будет содержать мост).

Для каждого из этих $\lambda - 1$ ребер удаляем инцидентную вершину, отличную от вершин моста. Тогда удалится минимум эти $\lambda - 1$ ребер. Если граф перестал быть связным, то $\kappa(G) < \lambda(G)$, а если связен, то остался мост, при удалении какой-то инцидентной вершины граф перестанет быть связным и $\kappa(G) = \lambda(G)$.

Теорема 5. Для связного графа $G = \langle V , E\rangle$: если $\delta (G) \geq \lfloor|V |/2\rfloor$, то $\lambda(G) =\delta (G)$.

Пусть минимальный набор ребер, необходимый для того, чтобы сделать граф не связным, делит граф при удалении на несколько компонент связности. Обозначим меньшую по количеству вершин из этих компонент через S, остальную часть графа – через D, а $k = V(S)$.

$K <= \lfloor|V |/2\rfloor$, т.к. если мы разделили как-то граф и взяли меньшую часть, то она точно не может быть больше половины (иначе мы бы просто брали за меньшую другую часть).

Каждая вершина v в S инцидентна как минимум $\delta (G)$ ребрам. (т.к. $\delta (G)$- минимальная степень вершины). Т.к. всего кроме v в S останется k-1 вершин, то максимум k-1 вершин, смежных v, находятся в S, а δ-(k-1) – в D. Т.к. $k <= \lfloor|V |/2\rfloor$, а $\delta (G) \geq \lfloor|V |/2\rfloor$ (по условию), $k-1 <= \lfloor|V |/2\rfloor - 1$, $-(k-1) >= -\lfloor|V |/2\rfloor + 1$, то

$\delta -(k-1) \geq \lfloor|V |/2\rfloor - \lfloor|V |/2\rfloor + 1 \Rightarrow \delta -(k-1) \geq 1$. А $\delta -(k-1)$ – это у нас количество вершин, смежных v, находящихся в D. Значит, для каждой вершины в S есть такое ребро, которое соединяет эту вершину из S с какой-то вершиной из D. Всего таких ребер как минимум $k*(\delta -k+1)$. То есть, чтобы разделить эти 2 части графа, надо удалить все эти $k*(\delta -k+1)$ ребер. => $\delta (G) >= \lambda(G) >= k*(\delta - k+1) >= \delta (G)$ (т.к. минимум $k*(\delta -k+1)$ достигается при $k = 1$, если исследовать функцию, и равен δ) = > $\delta (G) >= \lambda(G)$ если $\delta (G) \geq \lfloor|V |/2\rfloor$

Теорема 6. Для любой пары несмежных вершин u и v в ненаправленном графе, размер минимального вершинного разреза равен наибольшему количеству попарно внутренне вершинно не пересекающихся путей из u в v.

Докажем, что если существует минимальный вершинный разрез размера k, то в графе существуют k попарно вершинно не пересекающихся путей из u в v. Для этого рассмотрим вершинный разрез S размера k, который разделяет вершины u и v на две компоненты связности. Предположим, что не существует k попарно вершинно не пересекающихся путей из u в v. Тогда существует меньше, чем k вершин, которые могут быть удалены, чтобы разделить u и v. Это противоречит тому, что S является минимальным вершинным разрезом.

Теорема 7. Каждый блок блокового графа - клика.

Пусть H – граф блоков графа G. Предположим, что в H есть какой-то блок k, который не является кликой. Тогда в k есть пара несмежных вершин, лежащая на простом цикле, длина которого не меньше 4 (потому что цикл из 3 вершин — это клика). Но тогда подграф G, содержащий блоки, являющиеся вершинами блока k, лежащие на этом цикле, является двусвязным, т. е. находится в одном блоке => является одной вершиной графа H.