**Задача 1.** Доказать, что граф G регулярен тогда и только тогда, когда G содержит такую вершину u, что degGu = b(G) и deg~~G~~u = b(~~G~~), где b(H) – минимальная степень вершин графа H.

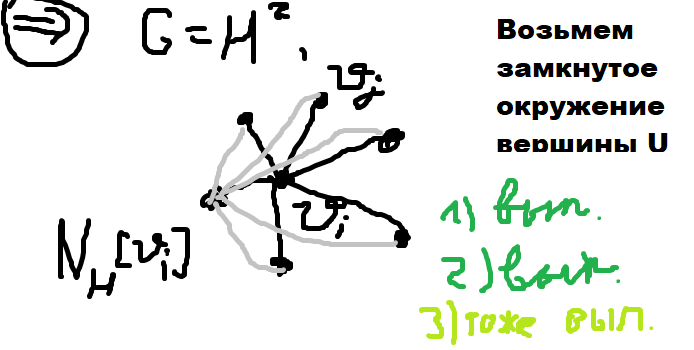
Для начала отметим один всем известный факт о том, что если G — регулярный граф порядка n степени d, то дополнительный граф ~~G~~ также регулярен, при чём deg~~G~~ = n - 1 - d. (\*)

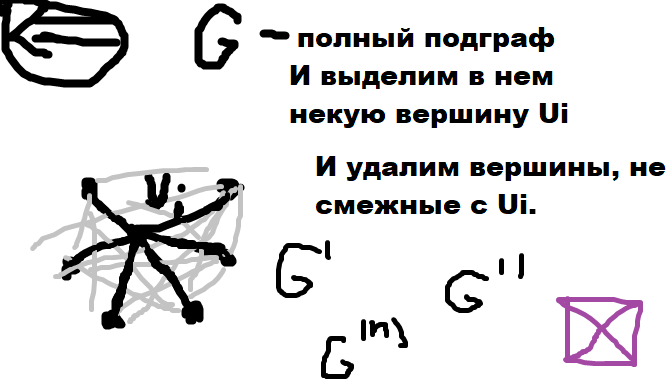
Докажем, что из первого утверждения следует второе. Т.к. граф G – регулярен, то исходя из утверждения (\*), и граф G, и его дополнение ~~G~~, являются регулярными. Тогда степени всех вершин графа G равны одному и тому числу, а в том числе и тому числу, которое соответствует минимальной степени графа G. Т.к. мы сказали, что ~~G~~ тоже регулярен, то для него аналогично степени всех вершин равны одному и тому же числа, а в том числе и тому числе, которое соответствует минимальной степени графа ~~G~~.

Теперь докажем, что из второго утверждения следует первое. Пусть G содержит такую вершину u, что degGu = b(G) и deg~~G~~u = b(~~G~~). То есть и в исходном графе G, и в его дополнении ~~G~~, найдется такая вершина u, что и для того, и для того графа, ее степень будет равна минимальной степени… Итак, G содержит такую вершину u, что degGu = b(G), тогда в его дополнении эта же вершина должна иметь максимальную степень (исходя из свойств дополнений к графам), но исходя из условия она равна минимальной степени… То, что степень этой вершины является минимальной и максимальной одновременно, может выполняться лишь в том случае, когда граф G регулярен. Следовательно, если такая вершина находится, то граф G регулярен.

**Задача 2.** Доказать, что граф G, где |G| = n, имеет корень квадратный ⬄ в G есть n полных подграфов Gi (i,…,n), таких, что:  
- Ui ∈ Gi;

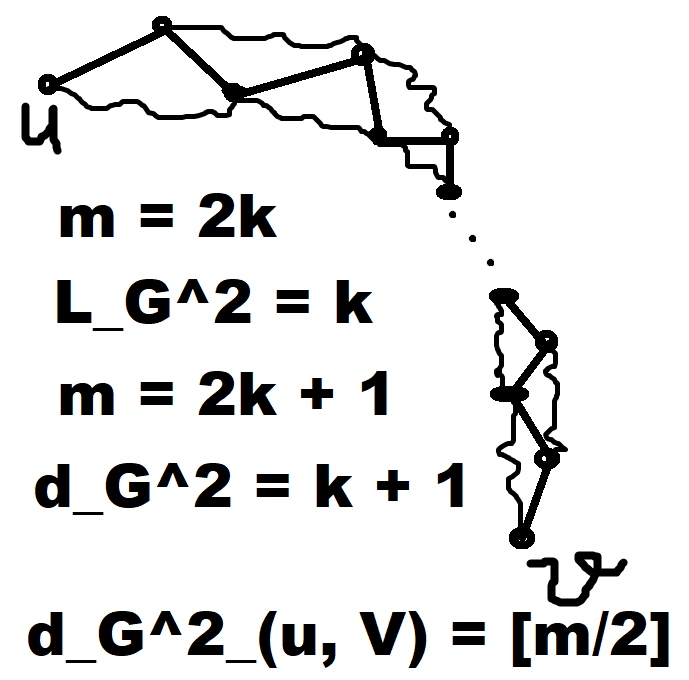
- Uj ∈ Gi ⬄ Ui ∈ Gj;

- все ребра графа покрываются {Gi}, то есть для каждого ребра e ∈ E(G) существует Gi такой, что е ∈ E(Gi) [1].  
Докажем необходимость:

Из рисунка видно, что условия 1), 2) и 3) выполняются…  
Докажем достаточность:

Из рисунка видно, что граф G такой, у которого |G| = n, будет иметь квадратный корень.

**Задача 3.** Найти dG2 (u, v), если dG (u, v) = m.

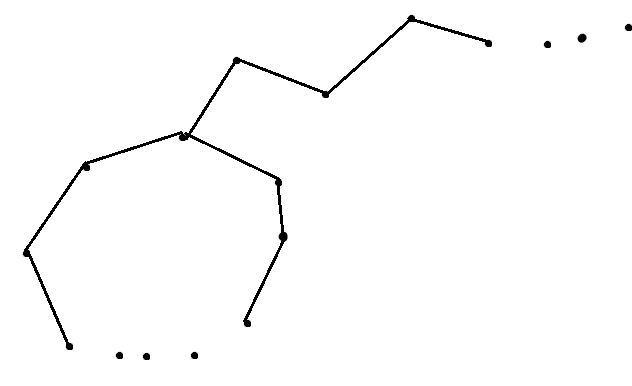


Предположим, цепь < u, v > - это цепь между u и v кратчайшей длины. По условию длина этой цепи равна m. Тогда рассмотрим два случая: 1) m – чётно. В графе G2 любые две вершины, расстояние между которыми равно 2 соединяем ребром. Из этого получаем, что dG2 (u, v) = m/2; 2) m – нечётно. Пусть x - вершина графа G, при чём x принадлежит < u, v > - цепи, и она смежна с v. Тогда dG2 (u, v) = dG2 (u, x) + dG2 (x, v) = (m - 1)/2 + 1 = (m + 1)/2.  
Ответ: если m – чётное, то m/2, иначе же (m + 1)/2.

**Задача 4.**

Доказать, что для любых положительных целых чисел r и d, таких, что r <= d <= 2r, существует связный граф G, у которого r(G) = r и d (G) = d.

Для начала построим цикл длины 2r и к произвольной точке a достроим цепь длины (d – r), получив искомый граф. Пусть b – конечная точка для построенной нами цепи. Тогда e(b) = d. При движении от точки b к точке a эксцентриситет вершин будет уменьшаться от d до r, т.к. из того, что d(G) ≤ 2r следует, что (d(G) − r) ≤ r, то есть e(z) = d(z, a) + r, для любой вершины z из <a, b> - цепи. Пусть f - вершина, принадлежащая циклу. Следовательно, r ≤ e(f) (т.к. находится в цикле длины 2r) и e(f) ≤ d (т.к. значение d достигается только в вершине, лежащей в цикле, на расстоянии r от вершины a).

Возьмем циклический G1 = G2r+1. Тогда r(G1) = d(G1) = r.  
Возьмем простую цепь Pd+1. Тогда d(Pd+1) = d.  
Тогда возьмем цикл G1 радиуса r и присоединим к нему цепь P длины d – r + 1.  
Тогда видно, что исходный граф существует.

**Задача 5.**

Доказать, что элемент матрицы (A(G))k, занимающий позицию (i, j), равен числу (i, j) – маршрутов длины k в графе G.

Возьмем матрицу смежности графа A(G) = A. Тогда каждый элемент k-ой степени этой матрицы (i, j) равен числу (i, j) – маршрутов длины k. Докажем это методом индукции.

Индукция по k. База индукции: при k =1, Ak = A.

Предположим, что утверждение верно для Ak.

Тогда докажем, что оно верно и для Ak+1 = Ak \* A.

Тогда ak+1\_(i, j) = Сумма (от L=1 до n) ak\_(i, L)\*a\_(L, j) = 0. Значит, утверждение выполняется.

**Задача 6.** Написать программу, которая находит расстояние между определенными вершинами во взвешенном графе, заданном: а) матрицей смежности; б) списками смежности.Предлагаю код на языке программирования С++.

#include *<fstream>*

#include *<vector>*

#include *<algorithm>*

**using** **namespace** std;

int **const** MAX = 2147483646;

int main()

{

ifstream fin("minpath.in");

ofstream fout("minpath.out");

int N, M, A, B, nach\_versh, kon\_versh, dlina\_dugi, i, nach, versh, inkr = 0;

fin >> N;

fin >> M;

fin >> A;

nach = A - 1;

fin >> B;

vector <int> rast(N, MAX);

vector <int> st\_versh(N);

vector <int> way;

vector <char> pomechena(N);

vector < vector < pair<int, int> > > Graf(N);

**while** (inkr < M)

{

inkr++;

fin >> nach\_versh;

nach\_versh--;

fin >> kon\_versh;

kon\_versh--;

fin >> dlina\_dugi;

Graf[nach\_versh].push\_back(make\_pair(kon\_versh, dlina\_dugi));

}

rast[nach] = 0;

**for** (i = 0; i < N; i++)

{

versh = -1;

**for** (int j = 0; j < N; j++)

{

**if** (((versh == -1) || (rast[j] < rast[versh])) && !pomechena[j])

{

versh = j;

}

}

**if** (rast[versh] == MAX)

{

**break**;

}

pomechena[versh] = true;

**for** (size\_t j = 0; j < Graf[versh].size(); j++)

{

int to = Graf[versh][j].first;

int len = Graf[versh][j].second;

**if** (rast[to] > (rast[versh] + len))

{

rast[to] = rast[versh] + len;

st\_versh[to] = versh;

}

}

}

**if** (rast[B - 1] != MAX)

{

**for** (versh = B - 1; versh != nach; versh = st\_versh[versh])

{

way.push\_back(versh);

}

way.push\_back(nach);

reverse(way.begin(), way.end());

fout << rast[B - 1] << endl;

**for** (unsigned i = 0; i < way.size(); i++)

{

fout << way[i] + 1 << " ";

}

}

**else**

{

fout << "0";

}

fin.close();

fout.close();

}

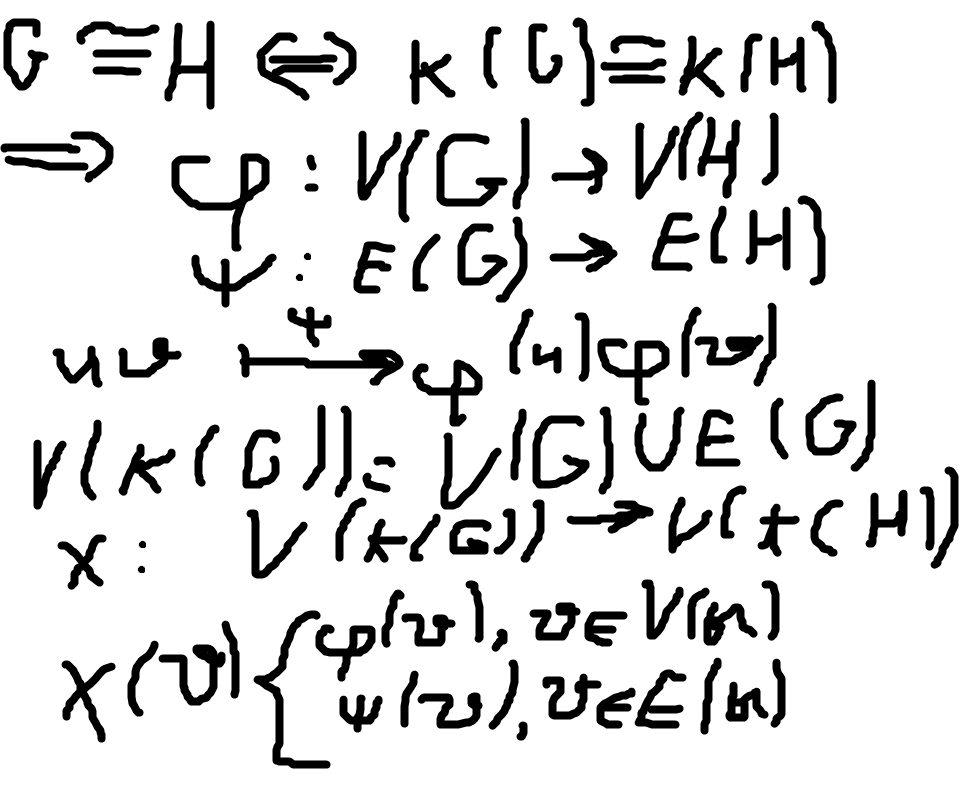
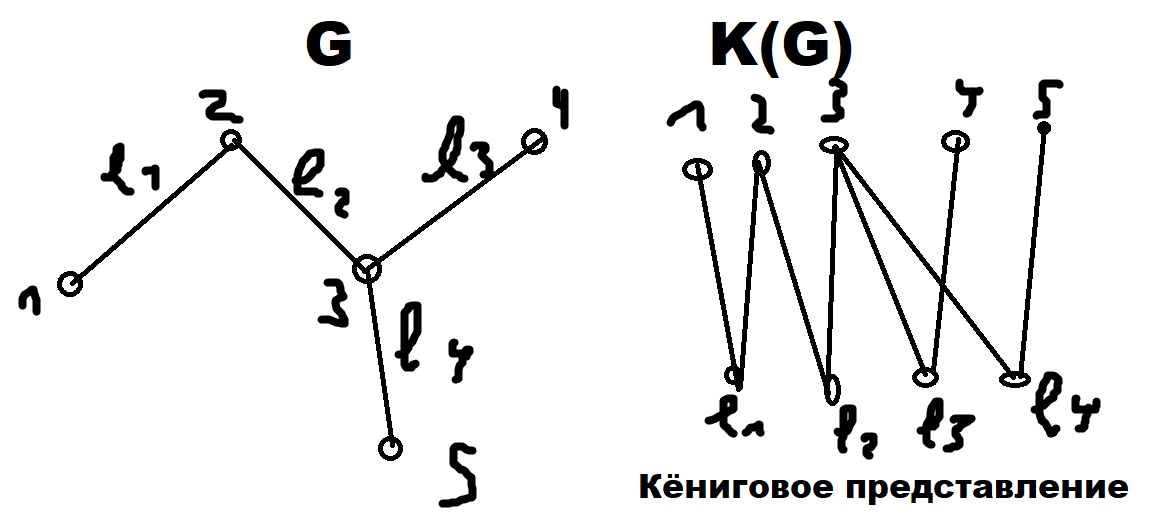
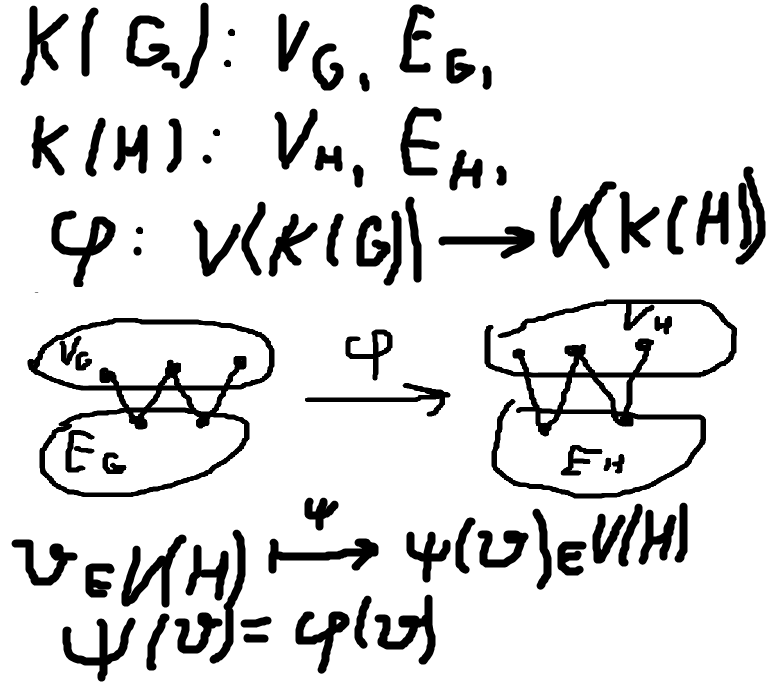
Пара тестов:  
1) Ввод:  
4 6  
1 4  
1 2 8  
2 3 6  
3 1 6  
4 1 5  
4 3 2  
4 2 4  
Вывод:  
0 (невозможно)  
2) Ввод:

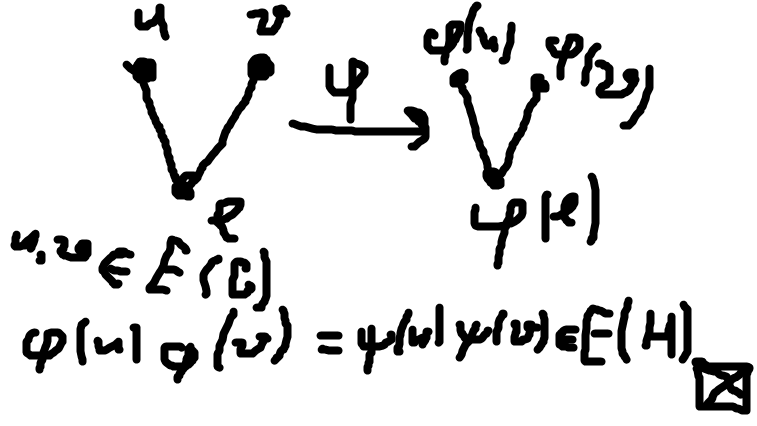
4 6  
1 4  
1 2 5  
2 3 2  
2 4 5  
3 1 4  
3 4 2  
4 1 1  
Вывод:  
9  
1 2 3 4

**Задача 7.** Доказать, что если G – граф, а K(G) – его кёнигово представление, то:  
а) Граф K(G) получается из G подразбиением всех его ребер. То есть, для каждого ребра e = (u, v) вводится новая вершина w, а ребро e заменяется двумя ребрами (u, w) и (w, v).

b) Два графа изоморфны ⬄ изоморфны их кёниговы представления.

Нарисуем несколько рисунков для представления…

  
Доказательство в обратную сторону:



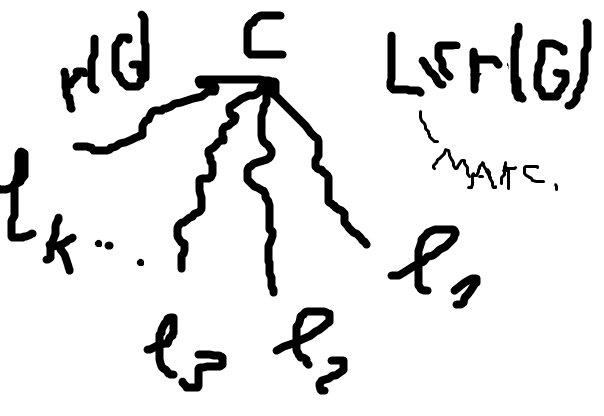
Текстовая версия доказательства:

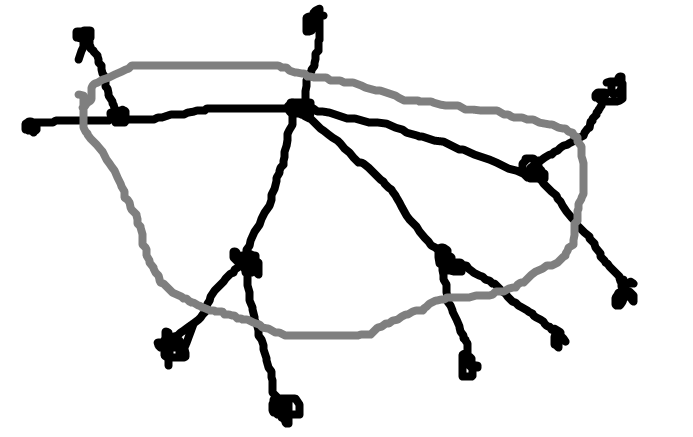
1. Пусть H – граф, полученный путем подразбиения всех рёбер графа G. Тогда разделим его на две доли: 1 - вершины, которые были изначально в графе G; 2 - вершины, полученные при разбиении рёбер. Между собой вершины в долях не смежны, т.к. если было бы ребро между вершинами в доле 1, значит, что мы разбили не все рёбра, а в доле 2 они не могут быть смежны, т.к. каждая вершина этой доли смежна только с концами того ребра, при разбиении которого она получилась. Следовательно, H - двудольный, доле 1 соответствуют вершины G, а доле 2 - рёбра G. Т.к. над любым графом можно провести эту операцию, и из любого полученного таким образом графа вернуться к исходному, то следует выражение: uv ∈ E(G) ⬄ ∃w : uw ∈ E(H) и wv ∈ E(H). Следовательно граф H - кёнигово представление графа G.  
2. Для начала проводем операцию подразбиения рёбер для G и H. Определим отображение f : V (G) -> V (K(G)) такое, что : uv ∈ E(G) ⬄ ∃w ∈ V (K(G)) и w /∈ V (G) : uw ∈ E(K(G)) и wv ∈ E(K(G)) т.к. существует биекция из V (G) в V (H) − которая сохраняет отношение смежности между вершинами, то при подразбиении всех рёбер отображение для вершин, которые были изначально, сохранится, а остальные вершины и все рёбра определяются через f, Следовательно кёниговы представления графов изоморфны.

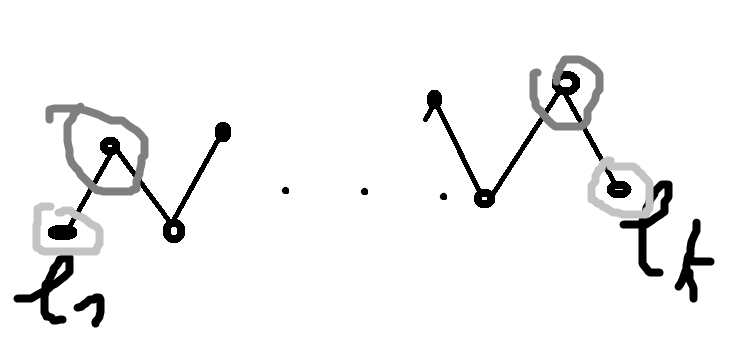
**Задача 8.**

Доказать, что центр дерева состоит из одной вершина в случае, когда его диаметр – четное число, и из двух вершин, когда диаметр – нечетное. (ссылаюсь на то, что в условии ошибка)

Центральная вершина – вершина, эксцентриситет которой совпадает с радиусом. Множество всех центральных вершин графа называется его центром. Пусть есть дерево Т, где С(T) – его центр.

Подвесим дерево за центральную вершину с. Рассмотрим любую цепочку до листьев…   
Если вдруг L\_i (1 <= i <= k) – не лист, то в любом случае цепочку можно продолжить до листа.  
Центр обязательно лежит на любой из диаметральных цепей.

Возьмем произвольное дерево и удалим из него все висячие вершины…

Если количество ребер в диаметральной цеп четное, то останется одна вершина, а иначе останется лишь одно ребро (то, что останется, и будет нашим центром):  
Тут можно сделать вывод, что центр дерева состоит из одной веришины в том случае, когда его диаметр число нечётные, и из двух вершин, когда его диаметр число четное.

**Задача 9.** Доказать, что, если S и T – два остова связного графа G, то для любого ребра e1 в S существует ребро e2 в T такое, что граф S – e1 + e2 также является остовом G.

Для начала возьмём любое ребро из графа T; пусть вершины u и v - его концы. uv = e2 Рассмотрим граф S. Т.к. он является остовом G, то он содержит все вершины G (а значит, и вершины u и v) и он является деревом. Значит, S содержит либо простую цепь, либо ребро uv. Рассмотрим два случая:  
1) Если содержит ребро uv, то e1 = uv. Граф S − e1 + e2 = S - остов G.

2) Если существует - цепь, добавим e2, образуется цикл (по теореме о свойствах дерева). Удалим любое ребро xy этого цикла (кроме e2), xy = e1. Образуется связный граф без циклов. Без циклов, т.к. при добавлении ребра был лишь один, который мы разрушили при удалении ребра. Связный, т.к. между собой все вершины цепи связны, а 1 остальные вершины, не принадлежащие ей, соединены с какой-либо вершиной цепи. Также полученный граф содержит все вершины графа G, значит он является остовом.

**Задача 10.** Доказать, что, если T – дерево порядка n со степенной последовательностью dn ≤ … ≤ d2 ≤ d1, то для каждого i = 1, 2, …, n выполняется неравенство di≤ [(n-1)/i].

Докажем по индукции:  
1. При n = 1, d1 = 0 ≤ [(1−1)/1] – верно; при n = 2, d1 = 1 ≤ [(2−1)/1] – верно; d2 = 1 ≤ [(2−1)/1] – верно.  
2. Предположим, что выражение верно для всех деревьев порядка ≤ (n − 1). Тогда di ≤ [((n−1)−1)/i], где 1 ≤ i ≤ n – 1.  
3. Теперь нужно сделать инкремент в нашем предположении и доказать, что и в этой ситуации все будет выполняться… Для этого возьмём дерево порядка (n-1), добавим ещё одну висячую вершину x, тогда di ≤ [((n−1)−1)/i] ≤ [(n−1)/i] для 1 ≤ i ≤ n – 1; dn = degGx ≤ [(n−1)/n] = 1 – Это тоже верно, т.к. соответствует добавленной висячей вершине.

**Задача 11.** Доказать или опровергнуть следующие два утверждения:

1. Диаметр графа d(G) = 2 => в графе есть остовная «звезда»;
2. В графе есть остовная «звезда» => диаметр графа d(G) = 2;

«Звезда» - это граф, в котором одна вершина соединена со всеми остальными, а каждая из остальных соединена только с одной исходной.

Докажем или опровергнем 1-ое утверждение. Диаметр графа G равен 2, а значит, он не содержит остовной звезды, т.к. в нём нет доминирующей вершины. Утверждение опровергнуто.

Теперь докажем или опровергнем 2-ое утверждение. Если в графе G есть остовная звезда, то граф G содержит по крайней мере одну доминирующую вершину, эксцентриситет которой равен 1, а расстояние между любыми двумя вершинами не превышает 2. Но, если эксцентриситет каждой вершины равен 1, то d(G) = 1, что возможно, лишь еcли граф полный. Утверждение опровергнуто.

Несколько контр-примеров:

1. Будем искать контр-пример, чтобы опровергнуть это утверждение.

Возьмем граф С5 (пятиугольник). Диаметр d(C5) = 2. Но здесь, очевидно, остовной «звезды» нету => утверждение опровергнуто.

1. Тоже будем искать контр-пример, чтобы опровергнуть это утверждение.

Возьмем полный граф K4 (ромб с соединенными диагоналями). Диаметр d(K4) = 1, что двум не равно, хотя в нем есть остовная «звезда» => утверждение опровергнуто.

**Задача 12.** Написать программу, которая находит минимальное остовное дерево во взвешенном графе с помощью алгоритма Краскала и алгоритма Прима…

Предлагаю код на языке программирования С++.

Алгоритм Краскала:  
// ---------------------------------------------

#include <stdio. h>

#include <conio. h>

#include <iostream. h>

// -------------------------------------------

typedef int\* tint; // указатель на int

void main ()

{ // int max=100; // Максимальный вес ребра

int n; // количество вершин

tint\* G; // исходный граф

tint\* H; // матрица списка ребер с весом

tint\* K; /\*матрица, отмечающая принадлежность

вершины компоненте\*/

tint\* T; // матрица остовного дерева

tint\* L; // список ребер с ценами минимального дерева

// -----ввод графа

int max;

cout<<" Maximalno dopustimoe zna4enie vesa rebra = ";

cin>> max;

cout<<"\n Vvedite 4ilo vershin: \n ";

cin>> n;

G=new tint [n];

for (int i=0; i<n; i++)

G [i] =new int [n];

cout<<" Vvedite nignij treugolnik matrici stoimosti: \n ";

for (int i=1; i<n; i++)

for (int j=0; j<i; j++) {

cin>> G [i] [j];

}

for (int i=1; i<n; i++)

for (int j=0; j<i; j++)

G [j] [i] =G [i] [j];

// ---выделение памяти для списка ребер---

int kolreb=0;

for (int i=1; i<n; i++)

for (int j=0; j<i; j++)

if (G [i] [j] <max && G [i] [j]! =0)

kolreb++;

H=new tint [kolreb];

for (int i=0; i<kolreb; i++)

H [i] =new int [3];

// -------------------------------------------

int a=0;

for (int i=1; i<n; i++)

for (int j=0; j<i; j++)

if (G [i] [j] <max && G [i] [j]! =0) {

H [a] [0] =i+1;

H [a] [1] =j+1;

H [a] [2] =G [i] [j];

a++;

}

cout<<endl;

// ----сортировка ребер по возрастанию веса

int f,d,s;

for (int i=0; i<kolreb-1; i++)

for (int j=0; j<kolreb-1; j++)

if (H [j] [2] <H [j+1] [2]) {

f=H [j] [2];

H [j] [2] =H [j+1] [2];

H [j+1] [2] =f;

d=H [j] [0];

H [j] [0] =H [j+1] [0];

H [j+1] [0] =d;

s=H [j] [1];

H [j] [1] =H [j+1] [1];

H [j+1] [1] =s;

}

// вывод ребер отсортированных по возрастанию веса

cout<<"Matrica dostigimosni otsortirovannoe po ubivaniuy: \n ";

for (int i=0; i<kolreb; i++) {

cout<<H [i] [0] <<"-->";

cout<<H [i] [1] <<" = ";

cout<<H [i] [2] <<endl;

cout<<" ";

}

for (int i=0; i<kolreb; i++) {

H [i] [0] - -;

H [i] [1] - -;

}

// матрица для определения компоненты

K=new tint [n];

for (int i=0; i<n; i++)

K [i] =new int [2];

for (int i=0; i<n; i++) {

K [i] [0] =i;

K [i] [1] =0;

}

// ----матрица остовного дерева

T=new tint [n];

for (int i=0; i<n; i++)

T [i] =new int [n];

for (int i=0; i<n; i++)

for (int j=0; j<n; j++)

T [i] [j] =0;

// -присоединение первого ребра

T [H [0] [0]] [H [0] [1]] =H [0] [2];

T [H [0] [1]] [H [0] [0]] =H [0] [2];

K [H [0] [0]] [1] =1;

K [H [0] [1]] [1] =1;

// алгорит соединения ребер без создания цикла:

int m=2; // номер компоненты

for (int i=1; i<kolreb; i++) // пройти по всем ребрам

if (K [H [i] [0]] [1]! =K [H [i] [1]] [1])

// если 2 вершины не из одной компоненты

{

if (K [H [i] [0]] [1] >0 && K [H [i] [1]] [1] >0)

// если обе вершины принадлежат разной компоненте

{

for (int j=0; j<n; j++)

if (K [H [i] [1]] [1] ==K [j] [1])

K [j] [1] =K [H [i] [0]] [1];

K [H [i] [1]] [1] =K [H [i] [0]] [1];

T [H [i] [0]] [H [i] [1]] =H [i] [2];

T [H [i] [1]] [H [i] [0]] =H [i] [2];

}

if ( (K [H [i] [0]] [1] >0 && K [H [i] [1]] [1] ==0)

|| (K [H [i] [0]] [1] ==0 && K [H [i] [1]] [1] >0))

// если одна вершина имеет компоненту а др. нет

{

if (K [H [i] [0]] [1]! =0)

K [H [i] [1]] [1] =K [H [i] [0]] [1];

if (K [H [i] [1]] [1]! =0)

K [H [i] [0]] [1] =K [H [i] [1]] [1];

T [H [i] [0]] [H [i] [1]] =H [i] [2];

T [H [i] [1]] [H [i] [0]] =H [i] [2];

}

if (K [H [i] [0]] [1] ==0 && K [H [i] [1]] [1] ==0)

// если обе вершины не имели компоненты

{

K [H [i] [0]] [1] =m;

K [H [i] [1]] [1] =m;

T [H [i] [0]] [H [i] [1]] =H [i] [2];

T [H [i] [1]] [H [i] [0]] =H [i] [2];

m++;

}

} // конец проверки всех ребер

// ---выделение памяти для списка ребер

kolreb=0;

for (int i=1; i<n; i++)

for (int j=0; j<i; j++)

if (T [i] [j] <max && T [i] [j]! =0)

kolreb++;

L=new tint [kolreb];

for (int i=0; i<kolreb; i++)

L [i] =new int [3];

// ------------------------------------------

// ---вывод ребер

cout<<endl<<" Vivod reber maximalnogo vesa: \n ";

a=0;

for (int i=1; i<n; i++)

for (int j=0; j<i; j++)

if (T [i] [j] <max && T [i] [j]! =0) {

L [a] [0] =i+1;

L [a] [1] =j+1;

L [a] [2] =T [i] [j];

a++;

}

for (int i=0; i<kolreb; i++) {

cout<<L [i] [0] <<"-->";

cout<<L [i] [1] <<" = ";

cout<<L [i] [2] <<"\n ";

}

int b=0;

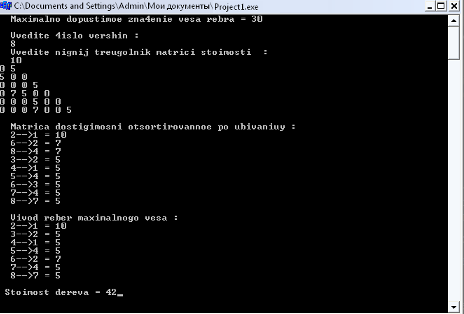
for (int i=0; i<kolreb; i++)

b+=L [i] [2];

cout<<endl <<" Stoimost dereva = "<<b; // вывод стоимости

getch ();

// return 0;

}  
Ввод и вывод:  


Алгоритм Примы (ввод и вывод аналогичны):

int n;

vector < vector<int> > graf;

const int MAX = 1000000000;

// алгоритм

vector<bool> used (n);

vector<int> min\_e (n, MAX), sel\_e (n, -1);

min\_e[0] = 0;

for (int i=0; i<n; ++i) {

int v = -1;

for (int j=0; j<n; ++j)

if (!used[j] && (v == -1 || min\_e[j] < min\_e[v]))

v = j;

if (min\_e[v] == MAX) {

cout << "No MST!";

exit(0);

}

used[v] = **true**;

if (sel\_e[v] != -1)

cout << v << " " << sel\_e[v] << endl;

for (int to=0; to<n; ++to)

if (graf[v][to] < min\_e[to]) {

min\_e[to] = graf[v][to];

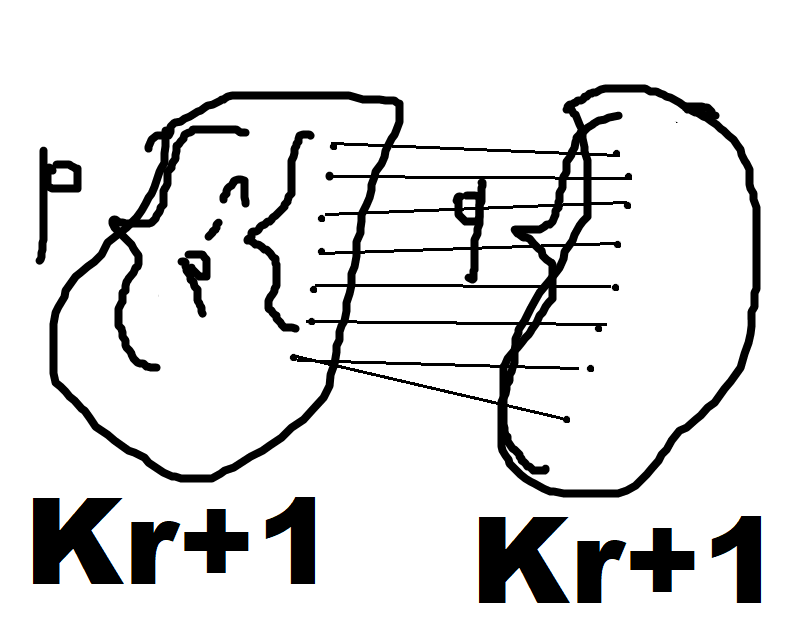
sel\_e[to] = v;

}

}

**Задача 13.** Доказать, что G связен => G2 – 2-связен.  
Из рисунка видно, что G2 действительно будет 2-связным…

**Задача 14.**

Доказать, что для любых натуральных (или равных нулю) чисел p ≤ q ≤ r, существует граф G, у которого k(G) = p, L(G) = q, б(G) = r, где соответственно указаны: вершинная связность, реберная связность, минимальная степень вершин. Возьмем два графа K(r+1).

У первого будет p вершин, а у второго q вершин. Тогда соединим все вершины из p с вершинами из q, а одну из них со всеми оставшимися вершинами из q (т.к. p < q). Тогда видно, что искомый граф можно получить.

**Задача 15.**

G – связный граф. Доказать, что k(G) = 1+ min\_u\_k(G - u), где u ∈ V (G).

Рассмотрим два случая: граф, получившийся после удаления, может быть 1) связным или 2) несвязным.  
1) Граф получился несвязным. Тогда k(G) = 0. Тогда очевидно, что ноль будет минимумом, тогда утверждение выполняется.

2) Допустим, после удаления вершины граф получится связным. Если мы будем выкидывать какую-то вершину, то мы должны взять минимальную из вершин связности (по определению). Берем минимальное число вершин. И чтобы сделать граф несвязным, мы должны удалить эти вершины. Тогда min\_u\_k(G-u) + 1 = k(G). Тогда утверждение также выполняется.