shu registrlarga yozib qo‘yiladi va kengaytirilgan ma’lumotning keyingi 512 bitlik blokini qayta ishlashga o‘tiladi.

5- bosqich. *Natija.*

Ma’lumotning xesh qiymati A, B, C, D va E registrlardagi qiymatlarni birlashtirish natijasida hosil qilinadi.

Mavzu yuzasidan savollar:

1. Xesh funksiya tushunchasiga ta’rif bering.
2. Kriptografik xesh funksiyalarga misol keltiring
3. Xesh funksiyalarning yana qanday turlarini bilasiz
4. Kalit hosil qiluvchi xesh funksiyalarni keltiring

15-§. Graflarda eng kichik uzunlikdagi daraxtlarni qurish algoritmlari

Eng kichik uzunlikdagi daraxt – berilgan grafning eng kam darajaga ega bo‘lgan daraxti, bu yerda daraxtning darajasi uning qirralari daajalari yig‘indisi sifatida tushuniladi.

Misol. Minimal uzunlikdagi daraxtni topish muammosi ko'pincha xuddi shunday sharoitda uchraydi: masalan, har qanday shahardan boshqasiga (to'g‘ridan-to'g‘ri yoki boshqa shaharlar orqali) o'tish uchun ***n ta*** shaharlarni yo'llar bilan bog‘lash kerak. Berilgan juft shaharlar o'rtasida yo'llar qurishga ruxsat beriladi va har bir bunday yo'lni qurish qiymati ma'lum. Qurilishning umumiy narxini minimallashtirish uchun qaysi yo'llarni qurish kerakligini hal qilish talab qilinadi.

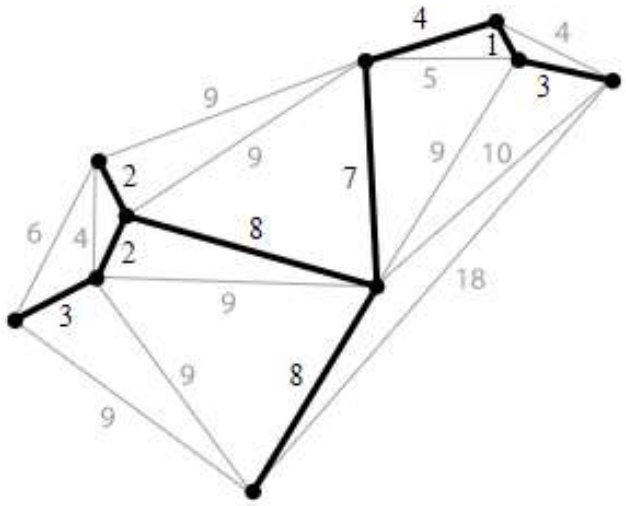
Ushbu muammoni grafika nazariyasi nuqtai nazaridan shakllantirish mumkin. Bu yerda berilgan grafnin uchlari shaharlarni, qirralari esa to'g‘ri yo'l qurilishi mumkin bo'lgan va qirralarning og‘irliklari teng bo'lgan shaharlarni ifodalaydigan minimal daraxtini topish muammosidir.

Minimal uzunlikdagi daraxtni topish uchun bir nechta algoritmlar mavjud. Eng mashhurlari quyida keltirilgan:

1) Prima algoritmi;
2) Kruskal algoritmi;

3) Boruvka algoritmi,
4) Orqadan o'chirish algoritmi

Quyida ushbu algoritmlarni ko‘rib chiqamiz.



59-rasm. Eng kichik uzunlikka ega bo‘lgan daraxt

15.1. Kruskal algoritmi

Kruskal algoritmida qirralarning butun birlashtirilgan ro'yxati kamaymaydigan uch darajalariga muvofiq tartiblangan. Bundan tashqari, qirralar darajalari kichikroq bo'lgan qirralardan yuqori tomonga siljiydi va keyingi uch ilgari tanlangan qirralar bilan sikl hosil qilmasa, karkasga qo'shiladi. Xususan, grafdagi minimal darajadagi qirralaridan biri har doim birinchi bo'lib tanlanadi.

Tanlangan qirralarning sikl hosil qilmasligini tekshirish uchun biz grafni bir nechta bog‘langan komponentlarning birlashishi sifatida namoyish etamiz. Eng boshida, grafning chekkalari tanlanmaganida, har bir uch alohida bog‘langan komponent hisoblanadi. Yangi qirralar qo'shilganda, ulanish komponentlari bitta umumiy ulanish komponenti bo'lguncha birlashadi. Barcha bog‘langan tarkibiy qismlarni raqamlaymiz va har bir uch uchun uning ulangan qismlarini sonini saqlaymiz, shuning uchun har bir uch uchun boshida uning bog‘langan komponentlari soni uchning o'zi soniga teng bo'ladi va oxirgi barcha

uchlar bir-biriga bog‘langan komponentning bir xil raqamlariga tegishli bo'ladi.

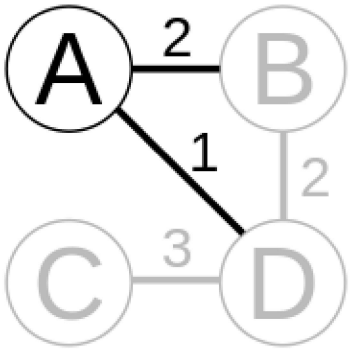
Keyingi qirrani ko'rib chiqayotganda, ushbu qirraning uchlariga to'g‘ri keladigan ulangan komponentlarning raqamlarini ko'rib chiqamiz. Agar bu raqamlar bir xil bo'lsa, unda qirra allaqachon bir xil bog‘langan komponentda joylashgan ikkita uchni birlashtiradi, shuning uchun bu qirrani qo'shish siklni tashkil qiladi. Agar qirra ikki xil bog‘langan komponentni, masalan, a va b raqamlari bilan bog‘lasa, u holda qirra asosiy daraxtning bir qismiga qo'shiladi va bu ikkita bog‘langan komponentlar birlashtiriladi. Buning uchun, masalan, ilgari b komponentida bo'lgan barcha uchlar uchun komponent raqamini a ga o'zgartirish kerak.

Kruskal algoritmini amalga oshirish bosqichlari quyidagicha:

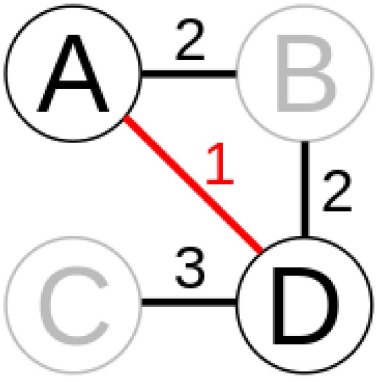
1) Barcha qirralarni quyidan yuqorigacha saralash.
2) Vazni eng kichik qirrasini oling va uni daraxtga qo'shing. Agar qirra qo‘shilganda, sikl hosil bo‘lsa, u holda bu qirrani olib tashlang.
3) Barcha uchlarga yetguncha qirralarni qo'shishni davom eting.

Quyidagi rasmda minimal uzunlikka kiritilgan qirralar qizil rang bilan, qora rang bilan esa nomzodlar ulardan eng kam darajadagi qirra tanlangan.

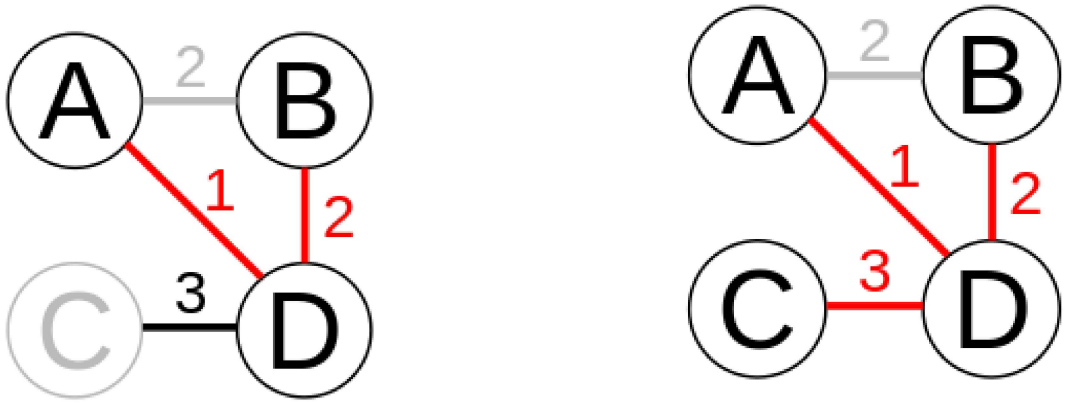
1-qadam



2-qadam



3-qadam *4-qadam. (So‘nggi natija)*



60-rasm. Kruskal algoritmining bajarilish ketma-ketligi

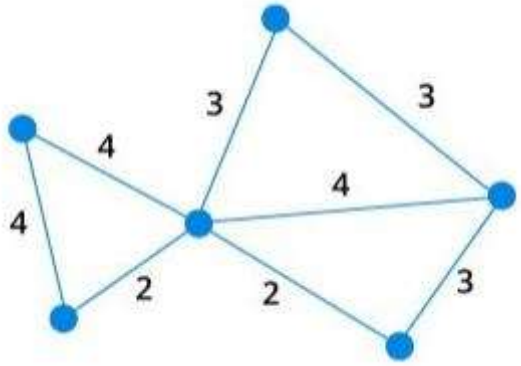
Kruskal algoritmini realizatsiya qilish (C++ kodi)

```
int n, m;
 cin >> n >> m;
 vector <vector <int> > edges(m, vector<int>(3));
 for (int i = 0; i < m; ++i)
    cin >> edges[i][1] >> edges[i][2] >> edges[i][0];
 sort(edges.begin(), edges.end());
 vector <int> comp(n);
for (int i = 0; i < n; ++i)
    comp[i] = i;
 int ans = 0;
for (auto & edge: edges)
{
    int weight = edge[0];
    int start  = edge[1];
    int end    = edge[2];
    if (comp[start] != comp[end])
    {
       ans += weight;
       int a = comp[start];
       int b = comp[end];
       for (int i = 0; i < n; ++i)
          if (comp[i] == b)
```

```
            comp[i] = a;
        }
    }
```

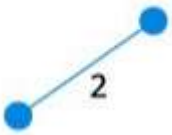
15.2. Prima algoritmi

Prima algoritmi quyidagi tartibda ishlaydi

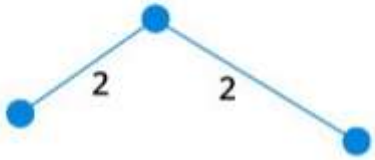


Dastlabki berilgan graf

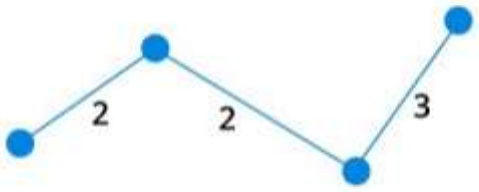
1-bosqich. Uchni tanlash



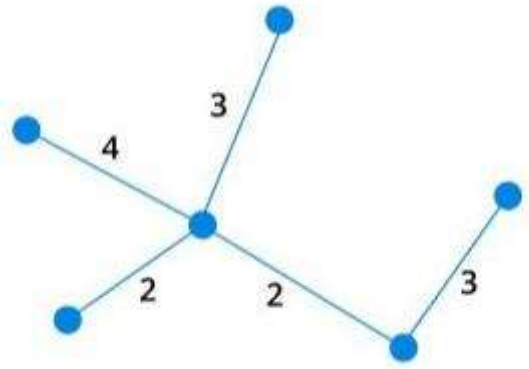
2-bosqich. Ushbu uchdan eng qisqa qirrani tanlash va uni qo'shish



3-bosqich. Grafdan hali tanlanmagan eng yaqin uchni tanlash



4-bosqich. Grafda hali topilmagan eng yaqin uchni tanlang, agar bir nechta variant, tasodifiy birini tanlash



Keyingi bosqichlar. Yuqoridagi ishlarni daraxt hosil bo‘lguncha takrorlash

Prima algoritmining C++ kodi

Quyidagi dastur Primaning algoritmini C ++ da amalga oshiradi. Grafni ko'rsatish uchun qo'shnilik matritsa ishlatilgan bo'lsa-da, ushbu algoritm samaradorligini oshirish uchun qo'shnilik ro'yxati yordamida ham amalga oshirilishi mumkin.

```
#include <iostream>
#include <cstring>
using namespace std;

#define INF 9999999

// grafdagi uchlar soni
#define V 5

//Qo'shnilik matritsasini ifodalash uchun
//5x5 o'lchamdagi ikki o'lchamli massivni yaratish

int G[V][V] = {
  {0, 9, 75, 0, 0},
  {9, 0, 95, 19, 42},
  {75, 95, 0, 51, 66},
  {0, 19, 51, 0, 31},
  {0, 42, 66, 31, 0}
```

```
};

int main () {

  int no_edge;          // qirralar soni

  // tanlangan uchni kuzatish uchun massiv yaratish
  int selected[V];

 // dastlab false qiymatini berish
  memset (selected, false, sizeof (selected));

 // qiralar soniga 0 ni berish
    no_edge = 0;
  selected[0] = true;
  int x;          //  qator raqami
  int y;          //  ustun raqami
  // qirra va og‘irlikni chop etish
  cout << "Qirra" << " : " << "Masofasi";
  cout << endl;
  while (no_edge < V - 1) {
        int min = INF;
     x = 0;
     y = 0;

     for (int i = 0; i < V; i++) {
       if (selected[i]) {
           for (int j = 0; j < V; j++) {
             if (!selected[j] && G[i][j]) {
                 if (min > G[i][j]) {
                     min = G[i][j];
                     x = i;
                     y = j;
                 }

             }
         }
       }
     }
```

```
    cout << x <<  " - " << y << " :  " << G[x][y];
    cout << endl;
    selected[y] = true;
    no_edge++;
   }

  return 0;
}
```

Mavzu yuzasidan savollar:

1. Eng kichik uzunlikdagi daraxt nima?
2. Prima algoritmining murakkabligini baholang.
3. Kruskal algoritmining murakkabligini baholang.

16-§. Minimal yo‘lni topish masalasi

Amaliyotda uchraydigan ko‘plab masalalarda marshrut uzunligi maksimallashtirilishi yoki minimallashtirilishi talab etiladi. Shunday masalalardan biriga, aniqrog‘i, kommivoyajer masalasiga Gamilton graflari bilan shug‘ullanganda duch kelamiz.

$G=(V,U)$ oriyentirlangan graf berilgan bo‘lsin, bu yerda $V=\{1,2,...,m\}$. G grafning biror $s\in V$ uchidan boshqa $t\in V$ uchiga boruvchi yo‘llar orasida uzunligi eng kichik bo‘lganini topish masalasi bilan shug‘ullanamiz. Bu masalani **minimal uzunlikka ega yo‘l haqidagi masala** deb ataymiz. Quyida bu masalaning umumlashmasi hisoblangan masalani qarab, uni ham o‘sha nom bilan ataymiz.

Grafdagi (i,j) yoyning uzunligini c_{ij} bilan belgilab, $C=(c_{ij}),\ i,j=\overline{1,m}$, matritsa berilgan deb hisoblaymiz. Yuqorida ta’kidlaganlarimizga ko‘ra, C matritsaning c_{ij} elementlari orasida manfiylari yoki nolga tenglari ham bo‘lishlari mumkin. Agar grafda biror i uchdan chiqib j uchga kirruvchi yoy mavjud bo‘lmasa, u holda bu yoyning uzunligini cheksiz katta deb qabul qilamiz ($c_{ij}=\infty$). Bundan tashqari, G grafda umumiy uzunligi