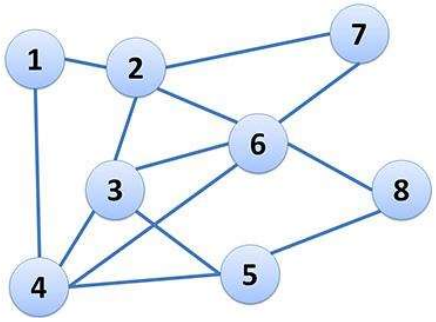
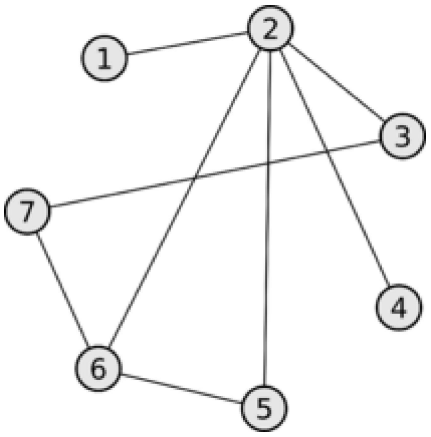
f)



g)



7-§. Grafda o‘tish algoritmlari

Graflar bilan ishlashda barcha asosiy amallar (masalan, grafni bitta ko‘rinishda ikkinchisiga o‘tkazish, bosib chiqarish yoki grafni chizish) uning tizimli o‘tishini, ya’ni grafning har bir uchiga va har bir qirrasiga tashrif buyurishni nazarda tutadi. Agar labirintni graf ko‘rinishida namoyish etsak, u yerda qirralar o‘tish joylari, uchlar esa qirralarning kesishish nuqtalari bo‘lsa, u holda grafni bosib o‘tish uchun har qanday to‘g‘ri algoritm ixtiyoriy ravishda labirintidan chiqish yo‘lini topishi kerak. Ushbu algoritmlarning eng mashhurlari grafda o‘tish bo‘yi bo‘yicha qidiruv (DFS) va o‘tish eni bo‘yicha qidiruv (BFS) algoritmlari bo‘lib, ular qo‘llaniladigan muammolarni hal qilish uchun ko‘plab boshqa algoritmlar uchun asos bo‘lib xizmat qiladi.

Graflar bo‘ylab harakatlanishning asosiy g‘oyasi shundaki, har bir uchni birinchi marta tashrif buyurganingizda belgilang va barcha qirralari ko‘rib chiqilmagan uchlar haqidagi ma’lumotlarni saqlang. Qadimgi Yunoniston afsonalarida Tesey labirint atrofida yurish uchun Ariadna bergan ipdan foydalangan, qarichlab tosh yoki maydalangan toshlar bilan bosib o‘tgan yo‘lini belgilab qo‘ygan, sanab o‘tilgan turlar grafni bosib o‘tish uchun ishlatilgan. Grafni bosib o‘tish jarayonida har bir uch uchta holatdan birida bo‘ladi:

1) ochilmagan - uchning dastlabki holati;

2) ochiq - uch topilgan, ammo unga tushgan qirralar ko‘rib chiqilmagan;

3) ishlov berilgan (belgilangan) - ushbu uchga tushgan barcha qirralarga tashrif buyuriladi.

Grafning har bir uchi ketma-ket ushbu holatlarning barchasini qabul qilishi aniq. Dastlab, faqat bitta uch ochiq bo‘ladi, ya’ni grafni bosib o‘tish ushbu uchdan boshlanadi.

7.1. Grafda o‘tish eni bo‘yicha qidiruv- BFS algoritmi

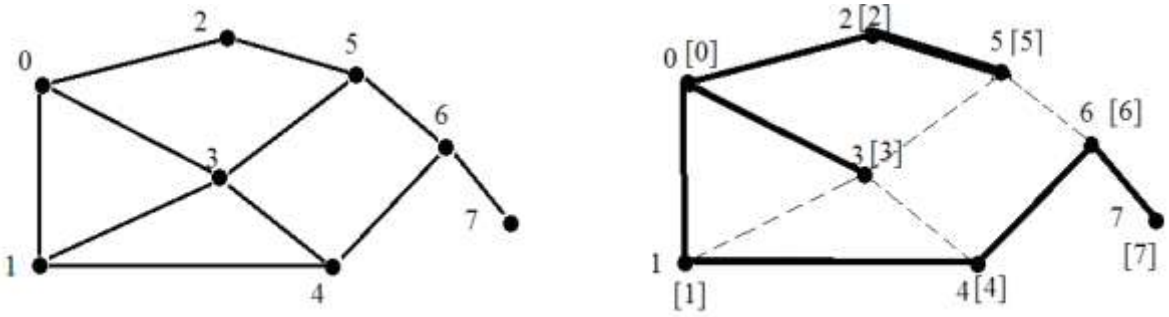
G = (V, E) grafi berilgan bo‘lsin va boshlang‘ich uchi v tanlansin. Birinchi kenglik bo‘yicha qidirish algoritmi v uchga yetib boruvchi barcha uchlarni "ochish" uchun G grafning barcha qirralarini muntazam ravishda kesib o‘tadi. O‘tish jarayonida barcha uchlarni o‘z ichiga olgan dastlabki uchda joylashgan kenglik bo‘yicha qidiruv daraxtini yaratadi. E’tibor bering, ildiz uchidan ushbu daraxtning istalgan uchiga masofa (qirralarning soni) eng qisqa bo‘ladi.

Kenglik bo‘yicha birinchi qidiruv ushbu nomga ega, chunki grafni bosib o‘tish jarayonida k+1 masofadagi har qanday uchni qayta ishlashdan oldin k masofadagi barcha uchlar belgilanadi.

Algoritm ham yo‘naltirilgan, ham yo‘naltirilmagan graflar uchun ishlaydi. Algoritm g‘oyasi: birinchisiga tutash bo‘lgan barcha uchlar ochiladi, ya’ni ular ro‘yxatga joylashtiriladi va bitta belgini oladi. Shundan so‘ng, dastlabki uch to‘liq qayta ishlanadi va 2 bilan belgilanadi.

Ro‘yxatning birinchi yuqori qismi keyingi uchga aylanadi. Amaldagi bilan qo‘shni bo‘lgan avval belgilanmagan barcha uchning ro‘yxatning oxiriga qo‘shiladi (ochiladi). Joriy uch ro‘yxatdan o‘chirilib, 2 raqami bilan belgilanadi. Jarayon uchlar ro‘yxati bo‘sh bo‘lguncha davom etadi. Ma’lumotlar ro‘yxatining ushbu ko‘rinishi navbat deyiladi.

Quyidagi misolni ko‘rib chiqamiz (27-rasm). 27-a) rasmda grafning dastlabki ko‘rinishi berilgan. 27-b) rasmda, uchlar yonida, graf uchlarini ko‘rish tartibi qavsda ko‘rsatilgan. Breadth First Search (BFS) daraxtini hosil qiladigan qirralar qalin rangda berilgan.



a) **b)**

27-rasm. BFS algoritmi jarayonida graf uchlarini ko‘rish

BFS algoritmi. Tasvirdan ko‘rinib turibdiki, algoritmning o‘zi juda ahamiyatsiz. Tashrif uchun uchlar navbati saqlanib qoladi. Keyingi uchga tashrif buyurganida, hali tashrif buyurmagan va hali navbatda bo‘lmagan barcha qo‘shnilari navbatga qo‘shiladi. Uchga allaqachon tashrif buyurilganligini tekshirish uchun bir qator yorliqlardan foydalaniladi. Dastlab, boshlang‘ich uchdan tashqari barcha i uchun visited[i] = false qiymatini qabul qiladi. i uch visited[i] navbatiga qo‘shilganda, true qiymati tayinlanadi.

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

vector<int> graph[100000];
bool used[100000];

int main() {
   //Grafni kiritish
   // ... Bu qismda graf matritsa ko‘rinishida kiritiladi
    queue<int> q;
    q.push(0);
    used[0] = true;

    while (!q.empty()) {
       int cur = q.front();
       q.pop();

       cout << "BFS : " << cur + 1 << endl;
```

```
        for (int k: graph[cur]) {
            if (!used[k]) {
                q.push(k);
                used[k] = true;
            }
        }
    }
}
```

Ushbu algoritmning murakkabligi $O(n^2)$, bu yerda n - grafdagi uchlar soni. Darhaqiqat, har bir uch ochilib, navbatga bir martagina joylashtirilgan, shuning uchun navbatning uchlari orasidagi sikl n martadan ko‘p bo‘lmagan vaqtda bajariladi. while sikli grafning barcha uchlari siklni o‘z ichiga oladi va u n marta bajariladi. Agar biz graf tasvirni qo‘shnilik ro‘yxati shaklida ishlatsak, unda murakkablik O(n+m) bo‘ladi, bu yerda m - qirralarning soni.

7.2. Grafda o‘tish bo‘yi bo‘yicha qidiruv (DFS) algoritmi

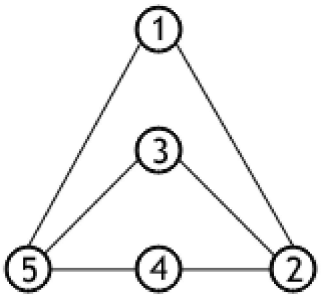
Grafda o‘tish bo‘yi bo‘yicha qidiruv (DFS) - bu graf uchlaridan o‘tishning rekursiv algoritmi. Agar bo‘yi bo‘yicha birinchi qidirish usuli nosimmetrik tarzda bajarilgan bo‘lsa (grafning uchlari darajalar bo‘yicha ko‘rib chiqilgan bo‘lsa), unda bu usul iloji boricha chuqurroq harakat qilishni o‘z ichiga oladi. Keyingi rivojlanishning mumkin emasligi shundan iboratki, keyingi qadam harakatlanishning bir nechta variantiga ega bo‘lgan (ulardan biri to‘liq o‘rganib chiqilgan), ilgari tashrif buyurgan uch oxirgisiga o‘tish bo‘ladi.

Algoritm qanday ishlashini aniq bir misol yordamida ko‘rib chiqamiz. Quyidagi yo‘naltirilmagan bog‘langan grafda jami 5 ta uch mavjud. Avval siz boshlang‘ich uchni tanlashingiz kerak. Qaysi uch tanlangan bo‘lsa ham, har qanday holatda ham graf to‘liq o‘rganib chiqiladi, chunki yuqorida aytib o‘tilganidek, bu bitta yo‘naltirilmagan bog‘langan graf. O‘tish 1 tugundan boshlasin, u holda qarab chiqilgan tugunlar ketma-ketligi tartibi quyidagicha bo‘ladi: 1 2 3 5 4. Agar ijro,

masalan, 3 tugundan boshlangan bo‘lsa, u holda o‘tish tartibi boshqacha bo‘ladi: 3 2 1 5 4.

DFS algoritmi rekursiyaga asoslangan, ya’ni o‘tish funksiyasi o‘zini bajarilayotganda chaqiradi, bu esa kodni umuman ixcham qiladi.

Algoritmning psevdokodi quyidagicha



28-rasm. BFS algoritmi jarayonida graf uchlarini ko‘rish

DFS funksiya sarlavhasi (st)
Chiqish (st)
visited[st] ← tashrif buyurgan;
r = 1 uchun n gacha
Agar (graph[st, r] ≠ 0) va (visited[r] tashrif buyurilmagan) bo‘lsa, u holda DFS (r)

Bu yerda DFS (deep-first search) - bu funksiya nomi. Uning yagona parametri st - dasturning asosiy qismidan argument sifatida uzatiladigan boshlang‘ich uchdir. Mantiqiy qiymatlarni qabul qiladigan massivning har bir elementiga oldindan false (yolg‘on, 0) qiymat beriladi, ya’ni uchlarning har biri dastlab tashrif buyurilmagan deb yoziladi.

Ikki o‘lchovli **graph** massivi grafning qo‘shnilik matritsasi. Natijalar oxirgi satrda to‘planishi kerak. Agar qo‘shnilik matritsasining elementi, qandaydir bosqichda, 1 ga teng bo‘lsa (0 emas) va matritsaning tekshirilgan ustuni bilan bir xil songa ega bo‘lgan uchga tashrif buyurilmagan bo‘lsa, unda funksiya rekursiv ravishda takrorlanadi. Aks holda, funksiya rekursiyaning oldingi bosqichiga qaytadi.

```
#include <iostream>
using namespace std;
const int n=5;
int i, j;
bool *visited=new bool[n];
//qo‘shnilik grafi
int graph[n][n] =
{
   {0, 1, 0, 0, 1},
   {1, 0, 1, 1, 0},
   {0, 1, 0, 0, 1},
   {0, 1, 0, 0, 1},
   {1, 0, 1, 1, 0}
};
//bo‘yi bo‘yicha izlash
void DFS(int st)
{
   int r;
   cout<<st+1<<" ";
   visited[st]=true;
   for (r=0; r<=n; r++)
      if ((graph[st][r]!=0) && (!visited[r]))
         DFS(r);
}
int main()
{
   int start;
   cout<<"Qo‘shnilik matritsasi: "<<endl;
   for (i=0; i<n; i++)
   {
   visited[i]=false;
   for (j=0; j<n; j++)
   cout<<" "<<graph[i][j];
   cout<<endl;
   }
   cout<<"Boshlang‘ich uchni kiriting: >> "; cin>>start;
   // tashrif buyurilgan uchlar massivi
   bool *vis=new bool[n];
   cout<<" O‘tish tartibi: ";
```

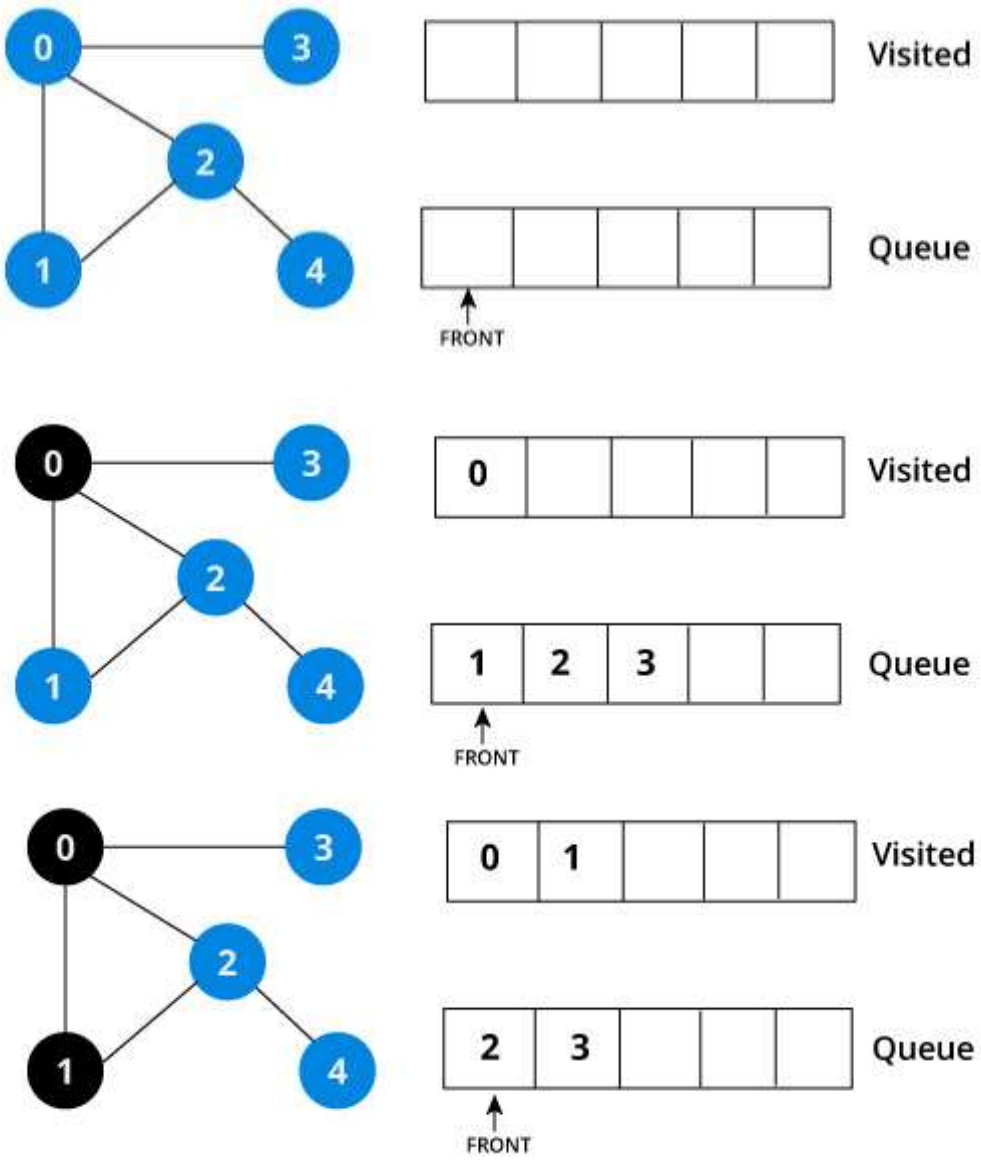
```
    DFS(start-1);
    delete []visited;
    system("pause>>void");
    return 0;
}
```

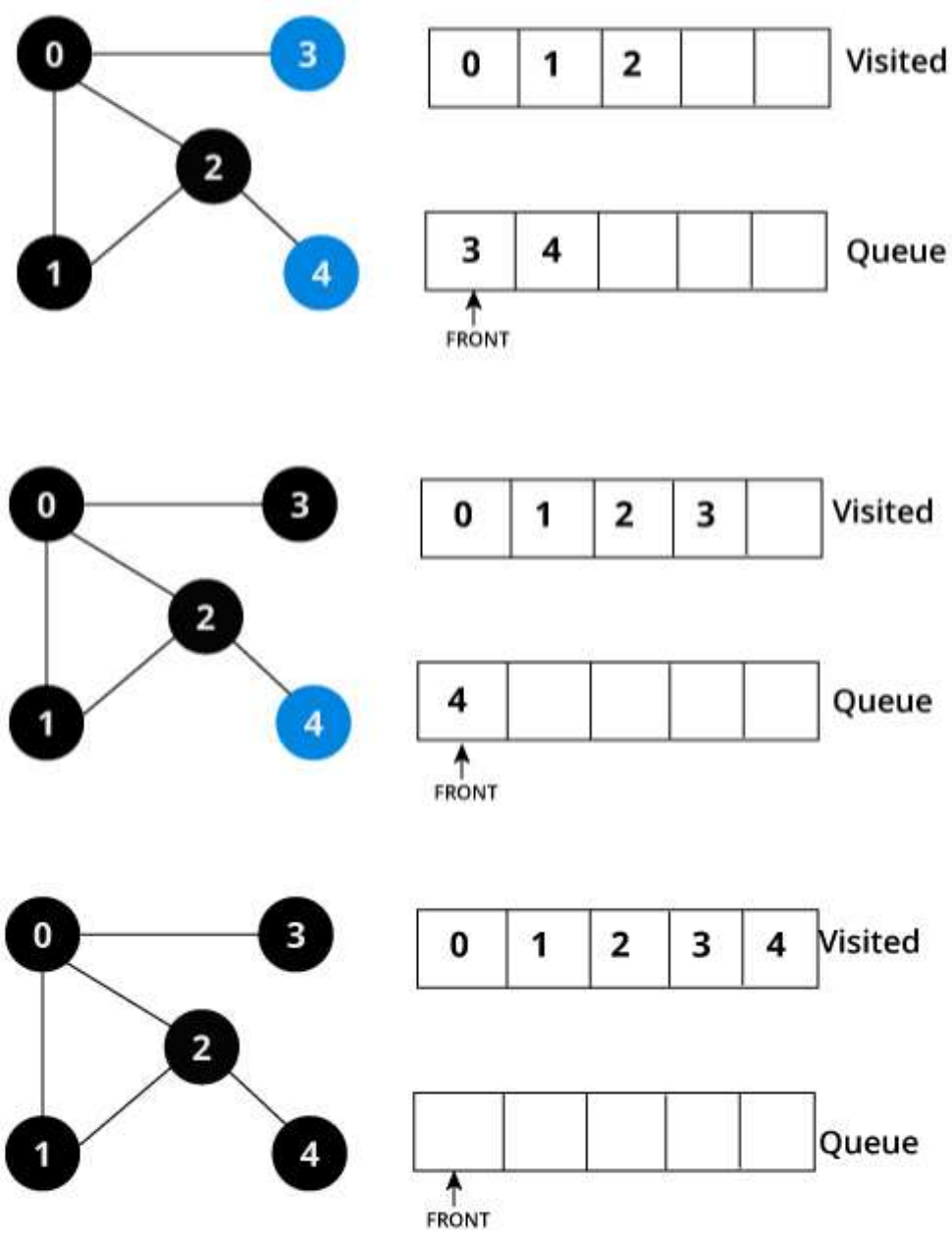
Misollar.

1-misol

BFS algoritmiga misol. Biz ikkita jadvalga ma’lumotlarni joylashtirib boramiz

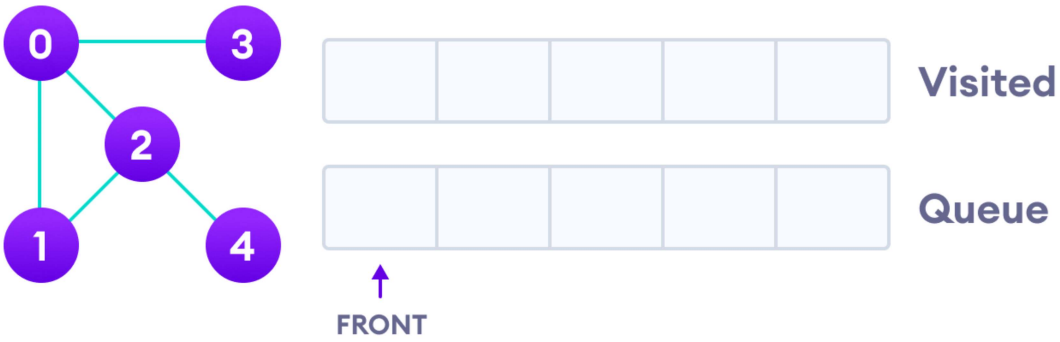
Tashrif buyurilgan uchlar – Visited jadvaliga. Navbatda turgan uchlar esa – Queue jadvaliga



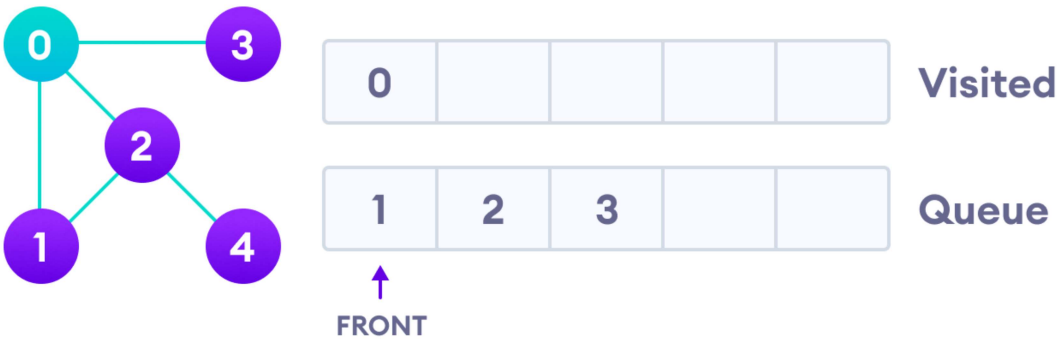


2-misol. Kenglik bo'yicha birinchi qidiruv (BFS)-bu graflar bilan ishlashning ko'plab muhim algoritmlari uchun asos bo'lgan, eng oddiy grafni o'tish algoritmlaridan biri.

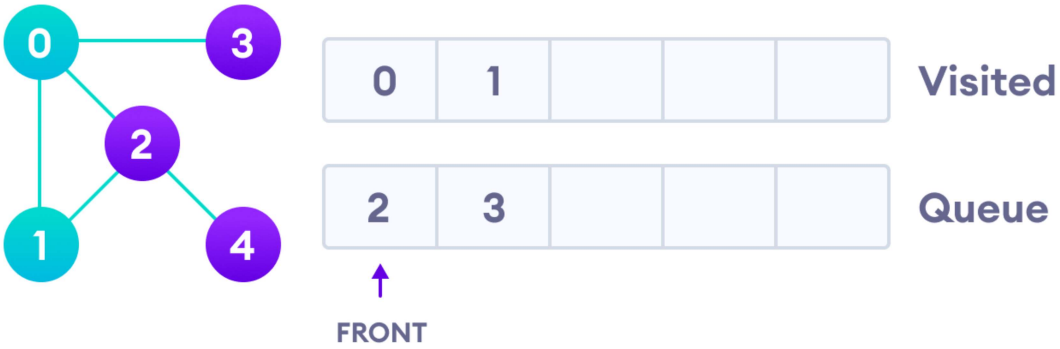
Keling, "Eni bo‘yicha qidirish" algoritmi qanday ishlashini misol bilan ko'rib chiqaylik. Biz 5 ta uchga ega bo'lgan yo'naltirilmagan grafdan foydalanamiz.



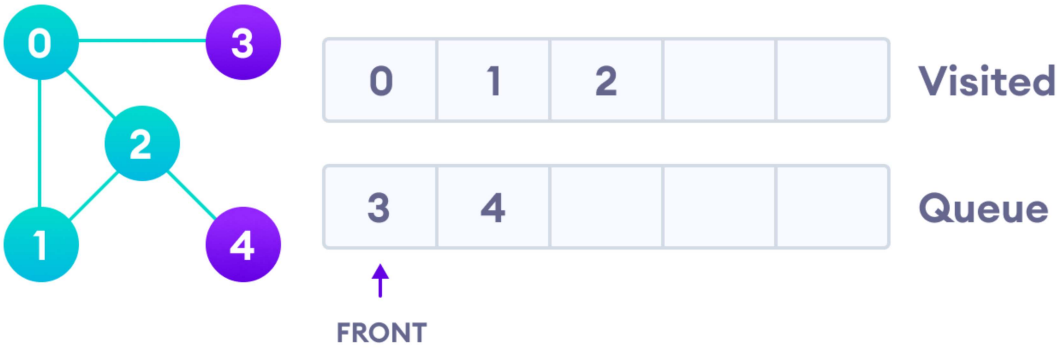
Biz 0 uchdan boshlaymiz, BFS algoritmi uni tashrif buyurilganlar ro'yxatiga qo'yib, uning yonidagi barcha uchlarni stekga qo'yishdan boshlanadi.

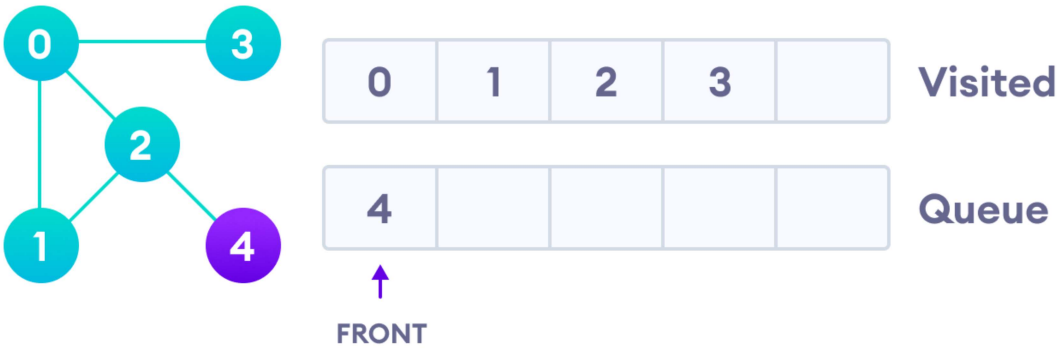


Keyinchalik, biz navbatning old qismidagi elementga tashrif buyuramiz, ya'ni 1 va uning yonidagi uchlarga o'tamiz. 0 tashrif buyurilgani uchun biz uning o'rniga 2 ga tashrif buyuramiz.



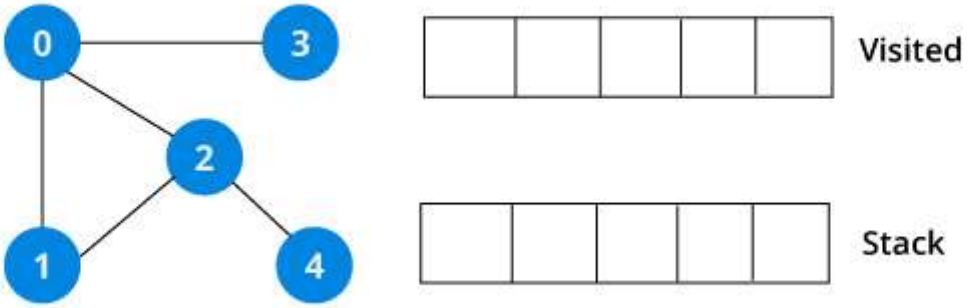
2-uchda ko'rilmagan qo'shni 4-uch bor, shuning uchun biz uni navbatning orqa qismiga qo'shamiz va navbatning oldida joylashgan 3 - ga tashrif buyuramiz.



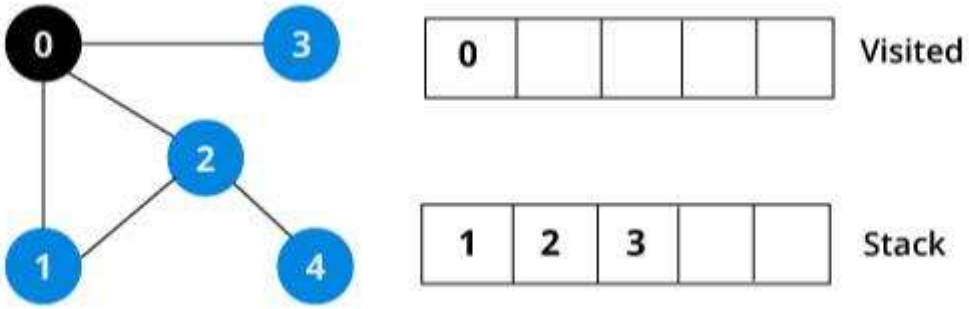


Navbatda faqat 4-uch qoladi, chunki qo'shni yagona 3-uch 3, ya'ni 0 ga tashrif buyurilgan. Biz unga tashrif buyuramiz.

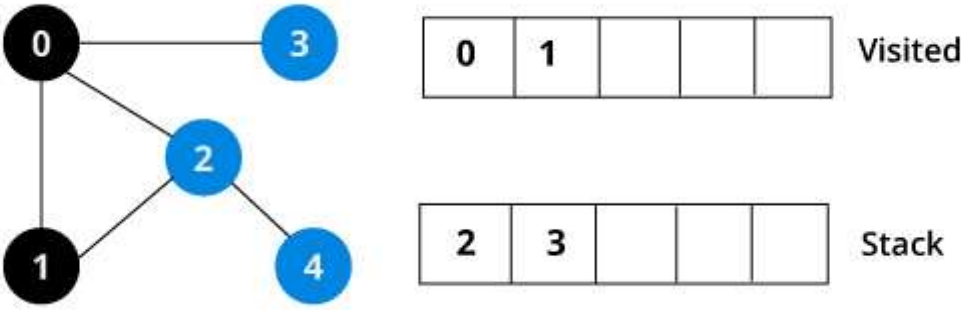
3-misol. DFS algoritmining ishlashi



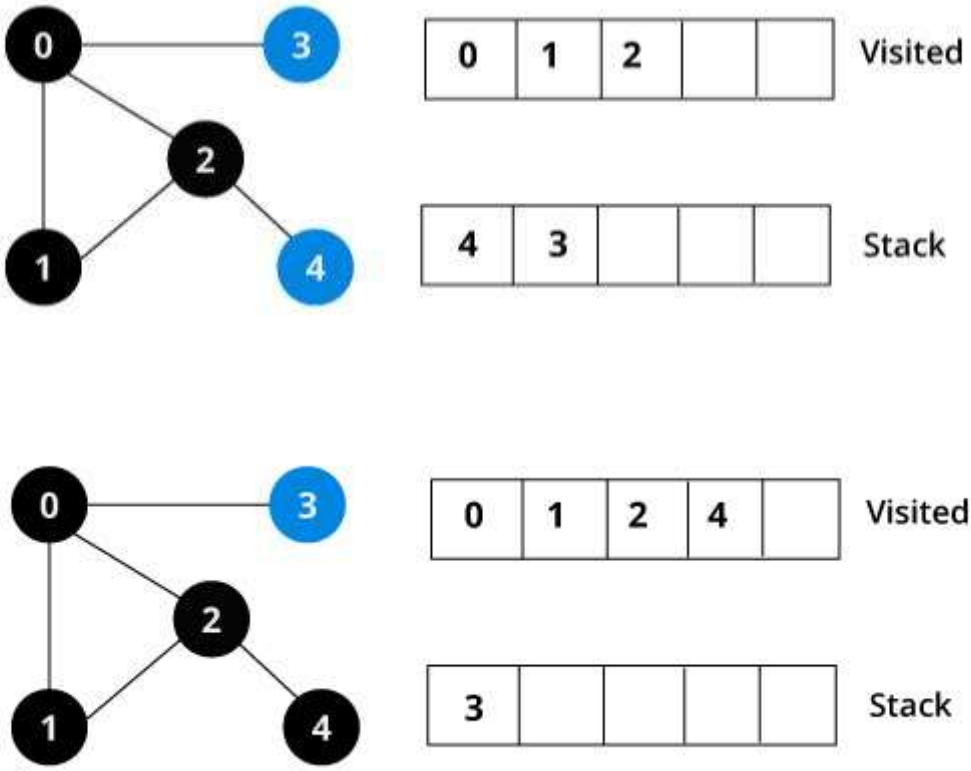
Biz 0 uchdan boshlaymiz, DFS algoritmi uni tashrif buyurilgan ro'yxatga qo'yishdan va barcha qo'shni uchlarni stekka joylashtirishdan boshlanadi.



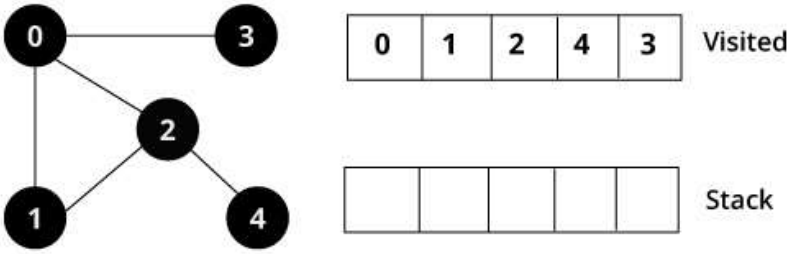
Keyin biz 1-uchning yuqori qismidagi elementga tashrif buyuramiz va qo'shni uchlarga o'tamiz. Biz allaqachon 0 ga tashrif buyurganimiz uchun, uning o'rniga 2 ga tashrif buyuramiz.



2-uchda ko'rilmagan qo'shni 4-uch bor, shuning uchun biz uni to'plamning yuqori qismiga qo'shamiz va tashrif buyuramiz.



Oxirgi 3-bandga tashrif buyurganimizdan so'ng, uning ko'zga ko'rinmas qo'shni uchlar yo'q. Bu grafni birinchi chuqurlik birinchi o'tishini yakunlaydi.



Mavzu yuzasidan savollar:

1. Graflarda o‘tish algoritmlari qanday masala hisoblanadi?
2. BFS algoritmining ishlash prinsipi qanday?
3. DFS algoritmining ishlash prinsipi qanday?

4. Graflarda yana qanday o‘tish algoritmlari mavjud?
5. Yuqorida keltirilgan algoritmlarning murakkabligini baholang