## 7 - AMALIY MASHG‘ULOT. *GRAFDA O‘TISH ALGORITMLARI.*

***Ishdan maqsad:***Talabalarni grafiklarni aylanib o‘tish algoritmlari bilan tanishtirish. Breadth First Search (BFS) va Depth First Search (DFS) protseduralari o’rtasidagi farqni bilib oling. C++ tilida algoritmlarga misollar yarating.

***Nazariy qism:*** Graflar - ob’ektlar orasidagi munosabatlarni aks ettirish uchun ishlatiladigan ma’lumotlar tuzilmalari. Graflarni tarmoqlash algoritmlari graflarni tahlil qilish va qayta ishlashda muhim rol o’ynaydi. Ular sizga eng qisqa yo’llarni topish, graflarni kesib o’tish, ulangan komponentlarni qidirish va boshqa ko’p narsalarni qilish imkonini beradi.

Graf - matematik struktura bo‘lib, bu cho‘qqilar to‘plami va bu uchlarni bog‘lovchi qirralar to‘plamidir.

Grafdagi ***yo‘l*** har bir qo‘shni cho‘qqi jufti chekka bilan bog‘langan cho‘qqilar ketma-ketligidir.

***Tsikl*** - birinchi va oxirgi uchlari mos keladigan yo’l.

***Yo’naltirilgan graf*** - qirralari yo’nalishga ega bo’lgan graf.

***Og’irlikdagi graf*** - har bir chekkasi raqam (vazn) bilan bog’langan grafdir.

***O’tish algoritmi*** - bu grafdagi yo’l yoki tsiklni topish jarayoni.

***Chuqurlik-Birinchi o’tish Qidiruv, DFS***

- ✓ Bu boshlang’ich cho’qqini tanlash va unga o’tish bilan boshlanadi.
- ✓ Keyin hali tashrif buyurmagan yaqin cho’qqiga o’tadi.
- ✓ Kirilmagan qo’shnilari bo’lmagan cho’qqiga yetguncha shu tarzda davom etadi.
- ✓ Shundan so’ng, u avvalgi cho’qqiga qaytadi va jarayonni takrorlaydi.

DFS grafdagi tsikllarni topish va ulangan komponentlarni aniqlash uchun ishlatiladi.

***Aylanib o’tish V kenglik (Breadth-First Search, BFS)***

- ✓ Boshlang’ich cho’qqini tanlash va uni navbatga qo’yish bilan boshlanadi.
- ✓ Keyin u navbatdan vertexni olib tashlaydi va qo’shnilariga o’tadi.
- ✓ Shundan so’ng barcha qo’shnilar navbatga qo’shiladi.
- ✓ Navbat bo’sh qolguncha jarayon takrorlanadi.

BFS o’lchovsiz grafdagi eng qisqa yo’lni topish uchun ishlatiladi.

***Dijkstra algoritmi Algoritm***

- ✓ Og’irlangan grafdagi eng qisqa yo’lni topish uchun ishlatiladi.
- ✓ Boshlanish cho’qqisini tanlash va unga 0 masofani berish bilan boshlanadi.
- ✓ Keyin barcha qo’shni cho’qqilargacha bo’lgan masofalar yangilanadi.

- ✓ Barcha cho'qqilarga tashrif buyurilgunga qadar takrorlanadi.

Dijkstra algoritmi manfiy qirralari bo'lmagan grafdagi eng qisqa yo'llarni topishda samarali.

***Floyd - Warshall algoritmi Algoritm***

- ✓ Og'irlangan grafdagi barcha cho'qqi juftlari orasidagi eng qisqa yo'llarni topish uchun foydalaniladi.
- ✓ Barcha cho'qqi juftlari orasidagi masofalar matritsasi tuziladi.
- ✓ Ushbu matritsa optimal qiymatlarga erishilgunga qadar takroriy yangilanadi.

Graf o'tish algoritmlari algoritmik nazariyaning muhim qismidir. Ular kompyuter tarmoqlarini marshrutlash, ijtimoiy tarmoqlarni tahlil qilish, graf ma'lumotlar bazalari va boshqalar kabi turli sohalarda qo'llaniladi. Ushbu algoritmlarni tushunish va ulardan foydalanish turli xil graf masalalarni samarali hal qilish imkonini beradi.

***Amaliy qism:*** Quyidagi C++ dastur kodi daraxtning BFS o'tish algoritmiga misol ko'rsatadi. Ushbu dastur daraxtning yuqori saqlanish (root)idan boshlab barcha elementlarni "Bo'shlikni Tashqaridan Qidirish" tartibida chiqaradi.

```
#include <iostream>
#include <queue>
// Daraxt uchun uzilishi
struct TreeNode {
    int val;
    TreeNode* left;
    TreeNode* right;
    TreeNode(int x) : val(x), left(NULL), right(NULL) {}
};
// BFS algoritmi
void BFS(TreeNode* root) {
    if (root == NULL)
        return;
    // Queue (tayyor)
    std::queue<TreeNode*> q;
    q.push(root);
    while (!q.empty()) {
        // Queue dan elementni olib chiqarish
        TreeNode* node = q.front();
        q.pop();
        // Olib chiqarilgan elementni chiqarish
        std::cout << node->val << " ";
        // Balandlik tartibida bo'yoqlarni qo'shish
        if (node->left != NULL)
```

```
            q.push(node->left);
        if (node->right != NULL)
            q.push(node->right);
    }
}
int main() {
    // Daraxtni yaratish
    TreeNode* root = new TreeNode(1);
    root->left = new TreeNode(2);
    root->right = new TreeNode(3);
    root->left->left = new TreeNode(4);
    root->left->right = new TreeNode(5);
    root->right->left = new TreeNode(6);
    root->right->right = new TreeNode(7);
    // BFS algoritmini ishga tushirish
    std::cout << "Daraxt bo'yoqlari (BFS qidiruvi tartibida): ";
    BFS(root);
    return 0;
}
```

Ushbu dastur konsolga "Daraxt bo'yoqlari (BFS qidiruvi tartibida): 1 2 3 4 5 6 7" chiqaradi. Bu yerda, daraxt yuqori saqlanish 1 dan boshlanadi va barcha elementlar bo'yoqlar to'plamlari "Bo'shlikni Tashqaridan Qidirish" tartibida chiqariladi.

Quyidagi C++ dastur kodi daraxtning DFS (Depth-First Search) o'tish algoritmiga misol ko'rsatadi. Ushbu dastur daraxtning yuqori saqlanishidan boshlab barcha elementlarni "Oqimli O'tish" (Pre-order traversal), "Ichkaridan O'tish" (In-order traversal), va "So'nggi O'tish" (Post-order traversal) tartibida chiqaradi.

```
#include <iostream>
#include <stack>
// Daraxt uchun uzilishi
struct TreeNode {
    int val;
    TreeNode* left;
    TreeNode* right;
    TreeNode(int x) : val(x), left(NULL), right(NULL) {}
};
// Pre-order DFS algoritmi
void preOrderDFS(TreeNode* root) {
    if (root == NULL)
        return;
    // Stack (tayyor)
```

```
    std::stack<TreeNode*> s;
    s.push(root);
    while (!s.empty()) {
        // Stack dan elementni olib chiqarish
        TreeNode* node = s.top();
        s.pop();
        // Olib chiqarilgan elementni chiqarish
        std::cout << node->val << " ";
        // Elementning o'ng va chap bo'yoqlarini qo'shish
        if (node->right != NULL)
            s.push(node->right);
        if (node->left != NULL)
            s.push(node->left);
    }
}
int main() {
    // Daraxtni yaratish
    TreeNode* root = new TreeNode(1);
    root->left = new TreeNode(2);
    root->right = new TreeNode(3);
    root->left->left = new TreeNode(4);
    root->left->right = new TreeNode(5);
    root->right->left = new TreeNode(6);
    root->right->right = new TreeNode(7);
    // Pre-order DFS algoritmini ishga tushirish
    std::cout << "Daraxt bo'yoqlari (Pre-order DFS tartibida): ";
    preOrderDFS(root);
    return 0;
}
```

Ushbu dastur konsolga "Daraxt bo'yoqlari (Pre-order DFS tartibida): 1 2 4 5 3 6 7" chiqaradi. Bu yerda, daraxt yuqori saqlanish 1 dan boshlanadi va barcha elementlar "Oqimli O'tish" tartibida chiqariladi.

***1-topshiriq:*** Og‘irlanmagan grafda ikkita cho‘qqi orasidagi eng qisqa yo‘lni toping.?

***Misol:*** A, B, C, D, E cho‘qqilari va AB, AC, BC, BD, DE qirralari bo‘lgan graf mavjud bo‘lsin. Biz A cho’qqisidan E cho’qqigacha bo’lgan eng qisqa yo’lni topishimiz kerak.

***Yechim:*** A cho‘qqisidan boshlab kenglik bo‘yicha birinchi o‘tish (BFS) algoritmidan foydalaning. Keyin E cho‘qqisiga yetguncha grafning chetlarini kuzatib boring.

***2-topshiriq:*** Barcha bog‘langan komponentlarni yo‘naltirilmagan grafda qidiring.?

***Misol:*** 1, 2, 3, 4 uchlari va 12, 23, 34 qirralari bo'lgan grafni ko'rib chiqaylik. Barcha bog'langan komponentlarni topish kerak.

***Yechim:*** Hali tashrif buyurmagan har bir cho‘qqidan boshlab chuqurlikdan birinchi o‘tish (DFS) algoritmidan foydalaning. Har bir o'tish uchun biz ulangan komponentni olamiz.

***3-topshiriq:*** Grafdagi ikkita cho‘qqi orasidagi barcha yo‘llarni toping.?

***Misol:*** A, B, C, D uchlari va AB, BC, CD qirralari bo'lgan graf. A cho'qqidan D cho'qqigacha bo'lgan barcha yo'llarni toping.

***Yechim:*** Siz topilgan barcha yo'llarni saqlagan holda chuqurlikdan birinchi o'tish (DFS) algoritmi bilan rekursiv yondashuvdan foydalanishingiz mumkin.

***4-vazifa:*** Grafdagi barcha sikllarni toping.?

***Yechim:*** Har bir tepadan boshlab barcha sikllarni topish uchun chuqurlikdan birinchi o‘tish (DFS) algoritmidan foydalaning.

***5-vazifa:*** Grafdagi barcha ulangan komponentlarni toping.?

***Yechim:*** Har bir cho‘qqidan boshlab va tashrif buyurilgan cho‘qqilarni belgilab, chuqurlikdan birinchi o‘tish (DFS) yoki kenglikdan birinchi o‘tish (BFS) algoritmidan foydalaning.

**Mustaqil bajarish uchun topshiriqlar.**

1. Xaritadan ikki shahar orasidagi eng qisqa yo‘lni toping.

2. Ijtimoiy tarmoqdagi ikki foydalanuvchi o'rtasida yo'l mavjudligini aniqlang.

3. Tovarlarni ombordan belgilangan joyga yetkazishning barcha mumkin bo'lgan yo'llarini toping.

4. Graf daraxt ekanligini aniqlang.

5. Bir nechta shaharlar bo'ylab sayyohlik sayohati uchun barcha mumkin bo'lgan marshrut kombinatsiyalarini toping.

6. Grafda sikl mavjudligini aniqlang.

7. Shaxmat taxtasidagi buyumni bir kvadratdan ikkinchisiga o‘tkazish uchun eng qisqa yo‘lni toping.

8. Grafning ikki tomonlama ekanligini aniqlang.

9. Elektr tarmog'ida manbadan iste'molchiga elektr energiyasini uzatish yo'lini toping.

10. Grafning cheklari berilgan sonli uchlari bor yoki yo‘qligini aniqlang.

11. Robotning labirintda harakatlanishi uchun eng samarali marshrutni toping.

12. Navigatsiyani ta’minlash uchun veb-sayt sahifalari o’rtasida yo’l mavjudligini aniqlang.

13. Vazifalarni ijrochilar o’rtasida eng kam xarajat bilan taqsimlashning barcha mumkin bo’lgan variantlarini toping.

14. Grafning ulanganligini aniqlang.

15. Posilkani jo‘natish joyidan shahardagi adresatga yetkazishning eng qisqa yo‘lini toping.

16. Grafning qo‘shnilari berilgan sonli uchlari bor yoki yo‘qligini aniqlang.

17. Logistika tarmog’idagi korxonalar o’rtasida resurslarni ko’chirishning optimal yo’lini toping.

18. Graf Eylerian ekanligini aniqlang.

19. Kompyuter tarmog’idagi ma’lumotlarni manbadan qabul qiluvchiga ko’chirishning eng samarali yo’lini toping.

20. Grafda qo‘shnilari bo‘lmagan cho‘qqilar bor yoki yo‘qligini aniqlang.

21. Grafning ikki cho‘qqisi orasidagi bog‘lanishni uzish uchun olib tashlash kerak bo‘lgan eng kichik qirralarni toping.

22. Grafning tekis ekanligini aniqlang.

23. Aeroportlar orasidagi reysni o’tkazish uchun eng qisqa marshrutni toping.

24. Grafda kiruvchi qirralarning berilgan soniga ega cho‘qqi bor yoki yo‘qligini aniqlang.

25. Bir necha fanlar bo’yicha o’quv kursini yakunlashning barcha mumkin bo’lgan usullarini toping.