GRAF STRUKTURALARIDA DFS: ALGORITMNING XUSUSIYATLARI VA AMALIYOTDAGI QO‘LLANILISHI

Farmonov Sherzodbek Rahmonjonovich
Farg’ona davlat universiteti amaliy matematika va informatika kafedrasi katta o’qituvchisi
farmonovsh@gmail.com
Qirg’izboyev Diyorbek Akmaljon o’g’li
Farg’ona davlat Universiteti
Kompyuter ilmlari va dasturlash texnologiyalari yo’nalishi
2-kurs talabasi
Diyorbekqirgizboyev91@gmail.com

Anotatsiya: ***DFS (Depth-First Search)*** *algoritmi — grafni chuqurdan chuqurga qidirish algoritmi bo'lib, grafdagi barcha tugunlarni (yoki uchrashgan tugunlarni) qidirish uchun ishlatiladi. DFS algoritmi orqali biror tugundan boshlab, barcha qo'shni tugunlarni kuzatib chiqamiz, har bir qo'shni tugunning ham qo'shni tugunlarini yana chuqurroq qidiramiz. Bu jarayon tugunlar bo'ylab yuqoridan pastga qarab ketadi.*

Kalit so’zlar: *DFS (Depth-First Search) , graf, daraxt, rekursiya, stack, indeks, Graph Traversal, Adjacency List, tugun, Chuqur qidirish, Visitation, Backtracking, Iterativ DFS.*

Annotation: *The DFS (Depth-First Search) algorithm is a graph traversal algorithm used to visit all vertices of a graph (or the encountered vertices). Using the DFS algorithm, starting from one vertex, we explore all its neighbors, and then for each neighbor, we explore their neighbors, going deeper each time. This process moves from vertex to vertex, like "top-down."*

Keywords: *DFS (Depth-First Search), graph, tree, recursion, stack, index, graph traversal, adjacency list, vertex, deep search, visitation, backtracking, Iterative DFS.*

Аннотация: *DFS (Depth-First Search) алгоритм — это алгоритм поиска в графе, который используется для обхода всех вершин графа (или встреченных вершин). С помощью алгоритма DFS начиная с одной вершины, мы исследуем всех её соседей, затем для каждого соседа ищем его соседей, углубляясь всё дальше. Этот процесс идет от вершины к вершине, как бы "сверху вниз".*

Ключевые слова: *DFS (Поиск в глубину), граф, дерево, рекурсия, стек, индекс, обход графа, список смежности, вершина, глубокий поиск, посещение, бэктрекинг, Итеративный DFS.*

DFS (Depth-First Search) algoritmi — bu graf yoki daraxtlar kabi ma'lumotlar tuzilmalarini qidirishda ishlatiladigan asosiy algoritmlardan biridir. Algoritmning asosiysi, berilgan grafda birinchi navbatda chuqur qidiruvni amalga oshirishdir. Bu algoritm bir nuqtadan boshlab graflarni yoki daraxtlarni o‘rganadi va har bir yangi tugun yoki cho‘qqini chuqurlik bo‘yicha izlaydi, to‘xtamadan oxirigacha boradi. DFS algoritmida graflar ko‘pincha tugunlar va ularni bog‘laydigan qirralar (yoki oraliqlar) orqali ifodalanadi. DFS algoritmi grafning barcha tugunlarini to‘liq o‘rganishi uchun har bir tugun faqat bir marta o‘rganiladi. Bu algoritmning boshqalaridan ajralib turadigan xususiyati, u chuqurroq o‘rganishga intiluvchi yondashuvni qo‘llaydi va bu jarayonni ko‘pincha rekursiya yordamida amalga oshiradi. Rekursiv yondashuvda har bir yangi tugun uchun DFS yana yangi qidiruvni boshlaydi, bu esa chuqurlikda to‘xtamaydigan izlanishga olib keladi. DFS algoritmi odatda graflarni yoki daraxtlarni chuqur qidirishda ishlatiladi, bunda har bir node yoki tugun o‘zining barcha bolalari orqali alohida o‘rganiladi. Bu algoritmning ishlash prinsipida o‘ziga xos xususiyatlar mavjud: algoritm faqat kerakli tugunni topish uchun faqat bitta yo‘ldan harakat qiladi va barcha noaniq qirralar o‘rganilib bo‘lgach, natija qaytariladi. DFS algoritmida har bir tugunning holati ko‘pincha "qolgan", "o‘rganilgan" va "tugallanmagan" kabi holatlar bilan belgilanadi, bu esa algoritmga tugunni qayta ishlashda yordam beradi. DFS algoritmi aslida oddiy ko‘rinishda bo‘lsa-da, uning amaliy qo‘llanilishi juda kengdir, chunki u har xil turdagi masalalarni hal qilishda juda samarali bo‘lishi mumkin, ayniqsa graf strukturasi bilan ishlashda. DFS algoritmi grafik algoritmlar orasida eng oddiylaridan biri hisoblanadi, ammo u juda samarali va muhim vositadir. DFSning amaliy qo‘llanilishi juda keng, chunki uning yordamida ko‘plab turli xil muammolarni hal qilish mumkin. Misol uchun, bu algoritm loyihalarni boshqarishda, tarmoqni tahlil qilishda yoki hatto o‘yinlarda ishlatiladi. Masalan, labirintni o‘rganishda DFS yordamida har bir yo‘lni chuqurroq tahlil qilib, topilgan yo‘lni qayta ishlash mumkin. DFSning boshqa bir muhim qo‘llanilishi esa topologik saralashda bo‘ladi. Topologik saralash, bir-biriga bog‘langan tugunlar (masalan, dasturiy ta’minotdagi modullar yoki jarayonlar) bo‘yicha o‘zaro bog‘lanishni tartibga solishda ishlatiladi. DFS algoritmi shu tarzda barcha tugunlarni tartibga solishda ishlatiladi. Bu turdagi muammolarni hal qilishda DFSning chuqurlik bo‘yicha yondashuvi juda samarali hisoblanadi.

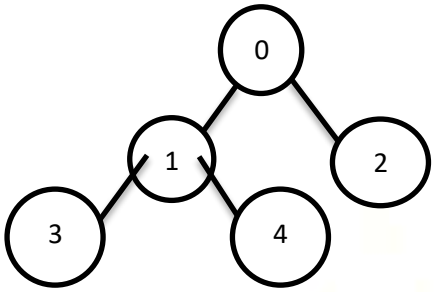
DFS algoritmining ishlash prinsipi:

1. **Boshlanish nuqtasi**: Algoritm graflarni yoki daraxtlarni qidirishda, boshlang'ich tugundan boshlanadi.
2. **Chuqurdan-qidirish**: Algoritm har bir tugunni tekshiradi, agar tugun yangi bo'lsa (ya'ni, unga hali tashrif buyurilmagan bo'lsa), uning qo'shni tugunlarini ham tekshiradi. Bunday tarzda, u tugunning barcha qo'shni tugunlariga chuqurroq kirib boradi.

DFS algoritmi odatda **rekursiv** tarzda amalga oshiriladi, lekin uni **stack** (stak) yordamida ham iterativ ravishda amalga oshirish mumkin.

DFSning C# kodidagi misoli (rekursiv):

Biz quyidagi graf bo'yicha ish yuritib ko'ramiz.



Quyida DFS algoritmining rekursiv usulda ishlashini ko'rsatadigan misol keltirilgan. Bu misolda, grafni ro'yxatlar (adjacency list) yordamida ifodalash va DFSni amalga oshirish ko'rsatiladi.

C# dasturlash tilidagi kodi:

```
using System;
using System.Collections.Generic;

class Program
{
    static Dictionary<int, List<int>> graph = new Dictionary<int, List<int>>();
    static void DFS(int node, HashSet<int> visited)
    {
        if (visited.Contains(node))
            return;
        visited.Add(node);

        Console.WriteLine(node);
        foreach (var neighbor in graph[node])
        {
            DFS(neighbor, visited);
        }
    }

    static void Main()
    {
        graph[0] = new List<int> { 1, 2 };
        graph[1] = new List<int> { 0, 3, 4 };
        graph[2] = new List<int> { 0 };
        graph[3] = new List<int> { 1 };
        graph[4] = new List<int> { 1 };
        HashSet<int> visited = new HashSet<int>();
        Console.WriteLine("DFS natijasi:");
```

```
        DFS(0, visited);
    }
}
```

DFS algoritmining ishlash jarayoni:

1.**Graf** quyidagicha tuzilgan:

1) 0 → 1, 2
2) 1 → 0, 3, 4
3) 2 → 0
4) 3 → 1
5) 4 → 1

2.DFS(0, visited) chaqirilganda, boshlang'ich tugun 0 bo'ladi. DFS algoritmi 0 ni tashrif buyurib, uning qo'shni tugunlariga (1 va 2) o'tadi va ularni ham tekshiradi.

3.Algoritm, 0 ning qo'shnisi 1 ni tekshiradi, so'ngra 1 ning qo'shnilari 3 va 4 ni tekshiradi. Keyin 0 ni tugatib, boshqa tugunlarga o'tadi.

DFS algoritmining boshqa implementatsiyasi (iterativ usul):

Agar rekursiyadan foydalanmaslik kerak bo'lsa, DFS algoritmini **stack** (stak) yordamida iterativ tarzda ham amalga oshirish mumkin.

C# dasturlash tilidagi kodi:

```
using System;
using System.Collections.Generic;

class Program
{
    static Dictionary<int, List<int>> graph = new Dictionary<int, List<int>>();

    static void DFS(int start)
    {
        Stack<int> stack = new Stack<int>();
        HashSet<int> visited = new HashSet<int>();
        stack.Push(start);

        while (stack.Count > 0)
        {
            int node = stack.Pop();
            if (!visited.Contains(node))
            {
                visited.Add(node);
                Console.WriteLine(node);
                foreach (var neighbor in graph[node])
                {
```

```
                if (!visited.Contains(neighbor))
                {
                    stack.Push(neighbor);
                }
            }
        }
    }

    static void Main()
    {
        graph[0] = new List<int> { 1, 2 };
        graph[1] = new List<int> { 0, 3, 4 };
        graph[2] = new List<int> { 0 };
        graph[3] = new List<int> { 1 };
        graph[4] = new List<int> { 1 };

        Console.WriteLine("Iterativ DFS natijasi:");
        DFS(0);
    }
}
```

DFS algoritmining afzalliklari:

Chuqurdan qidirish: DFS graflar va daraxtlar bilan ishlashda qidiruvning chuqur tomonini, ya'ni biror yo'nalishda qidirish imkonini beradi.

Xotira tejash: Agar graf keng bo'lsa, DFS rekursiv ravishda ko'plab tugunlarni ko'rib chiqishi mumkin va xotira ishlatishda samarali bo'lishi mumkin.

DFSning kamchiliklari:

Chuqur grafda stack to'lishi mumkin: Agar graf juda chuqur bo'lsa, rekursiv chaqiruvlar va stack'lar juda katta bo'lishi mumkin va bu "stack overflow" xatoligiga olib kelishi mumkin.

Optimal yo'lni topmaslik: DFS ba'zan eng qisqa yo'lni topmasligi mumkin. Agar eng qisqa yo'l kerak bo'lsa, BFS (Breadth-First Search) yaxshiroq variant bo'lishi mumkin.

DFS algoritmi grafni tahlil qilishda, masalan, grafning bog'liqligini tekshirish, tsiklni aniqlash, topologik saralash kabi muammolarni hal qilishda keng qo'llaniladi. Rekursiv usulda amalga oshirilganda, DFS algoritmi oddiy va tushunarli bo'lsa-da, chuqur graf tuzilmalarida "stack overflow" xatoliklariga olib kelishi mumkin. Iterativ usulda esa stack yordamida DFS ni amalga oshirish mumkin, bu esa xotira va rekursiv chaqiruvlar bilan bog'liq muammolarni oldini oladi. DFS algoritmining asosiy afzalliklari orasida chuqur qidirish va keng graf tuzilmalarida samaradorlikni ta'minlash, shuningdek, xotira tejash bor.

FOYDALANILGAN ADABIYOTLAR:

1. “INTRODUCTION TO ALGORITHMS” BY THOMAS H. CORMEN, CHARLES E. LEISERSON, RONALD L. RIVEST, AND CLIFFORD STEIN
2. “ALGORITHMS” BY ROBERT SEDGEWICK AND KEVIN WAYNE
3. “THE ART OF COMPUTER PROGRAMMING” BY DONALD E. KNUTH
4. “DATA STRUCTURES AND ALGORITHMS IN C++” BY ADAM DROZDEK
5. “GRAPH THEORY AND COMPLEX NETWORKS” BY MAARTEN VAN STEEN
6. “ALGORITHM DESIGN MANUAL” BY STEVEN S. SKIENA
7. “DATA STRUCTURES AND ALGORITHMS” BY MICHAEL T. GOODRICH, ROBERTO TAMASSIA, AND MICHAEL H. GOLDWASSER
8. “COMPETITIVE PROGRAMMING” BY STEVEN HALIM AND FELIX HALIM
9. Farmonov, S., & Nazirov, A. (2023). C# DASTURLASH TILIDA GRAY KODI BILAN ISHLASH. B CENTRAL ASIAN JOURNAL OF EDUCATION AND INNOVATION (T. 2, Выпуск 12, сс. 71–74). Zenodo.
10. Farmonov, S., & Toirov, S. (2023). NETDA DASTURLASHNING ZAMONAVIY TEXNOLOGIYALARINI O'RGANISH. *Theoretical aspects in the formation of pedagogical sciences*, *2*(22), 90-96
11. Raxmonjonovich, F. S. (2023). Array ma’lumotlar tizimini talabalarga o’qitishda Blockchain metodidan foydalanish. *Yangi O'zbekiston taraqqiyotida tadqiqotlarni o'rni va rivojlanish omillari*, *2*(2), 541-547.
12. Raxmonjonovich, F. S. (2023). Dasturlashda interfeyslardan foydalanishning ahamiyati. *Yangi O'zbekiston taraqqiyotida tadqiqotlarni o'rni va rivojlanish omillari*, *2*(2), 425-429.
13. Raxmonjonovich, F. S. (2023). Dasturlashda obyektga yo’naltirilgan dasturlashning ahamiyati. *Yangi O'zbekiston taraqqiyotida tadqiqotlarni o'rni va rivojlanish omillari*, *2*(2), 434-438.