DFS ALGORITMINING NAZARIY ASOSLARI VA UNING AMALIY QO‘LLANILISHI

Farmonov Sherzodbek Raxmonjonovich
Farg‘ona davlat universiteti amaliy matematika va informatika kafedrasi katta o’qituvchisi
farmonovsh@gmail.com
Topvoldiyev Islomjon Ikromjon o‘g‘li
Farg‘ona davlat universiteti talabasi
topvoldiyevislombek564@gmail.com

Anotatsiya: DFS (Depth-First Search) algoritmi graflarda chuqur qidiruvni amalga oshirishda ishlatiladigan samarali usuldir. Ushbu maqolada DFS algoritmining nazariy asoslari va uning amaliy qo‘llanilishi batafsil tahlil qilingan. DFS algoritmining ishlash tamoyili, uning rekursiv va iterativ versiyalari, shuningdek, graflarda bog‘liqlik komponentlarini aniqlash, aylanishlarni tekshirish.

Kalit so'zlar: DFS, Depth-First Search, graf, algoritm, rekursiya, iteratsiya, bog‘liqlik komponentlari, topologik saralash, tarmoq tahlili, sun'iy intellekt, o‘yin dizayni, ma'lumotlar bazasi, C#, dasturlash, algoritmik yondashuvlar

Annotation: DFS (Depth-First Search) algorithm is an efficient method used for performing deep search in graphs. This paper provides a detailed analysis of the theoretical foundations of the DFS algorithm and its practical applications. It explores the working principle of the DFS algorithm, its recursive and iterative versions, as well as its use in identifying connected components and detecting cycles in graphs.

Keywords: DFS, Depth-First Search, graph, algorithm, recursion, iteration, connected components, topological sorting, network analysis, artificial intelligence, game design, databases, C#, programming, algorithmic approaches

Аннотация: Алгоритм DFS (Depth-First Search) — это эффективный метод для выполнения глубинного поиска в графах. В данной статье подробно рассматриваются теоретические основы алгоритма DFS и его практическое применение. Изучены принципы работы алгоритма DFS, его рекурсивные и итеративные версии, а также использование алгоритма для определения связанных компонентов и поиска циклов в графах.

Ключевые слова: DFS, Depth-First Search, граф, алгоритм, рекурсия, итерация, связанные компоненты, топологическая сортировка, анализ сети, искусственный интеллект, разработка игр, базы данных, C#, программирование, алгоритмические подходы

Depth-First Search (DFS) algoritmi graflarni va daraxtlarni chuqur tekshirishga asoslangan muhim qidiruv algoritmidir. U dastlabki bosqichda bir tugundan boshlanib, graflarning barcha qirralarini imkon qadar chuqurroq

o‘rganishga harakat qiladi. DFS algoritmining asosiy tamoyili "birinchi o‘rganilgan yo‘lni oxirigacha sinash" tamoyiliga asoslanadi. Ya’ni, algoritm dastlab bir yo‘nalishni tanlaydi va uni oxiriga yetguncha davom ettiradi. Agar bu yo‘nalishdan hech qanday maqsadga erishilmasa yoki grafikning tugallangan qismiga chiqib qolinsa, algoritm ortga qaytadi va keyingi yo‘nalishni ko‘rib chiqishni davom ettiradi. DFS algoritmi graflar va daraxtlar kabi ierarxik tuzilmalarda keng qo‘llaniladi, chunki u ma’lumotlarni tizimli ravishda tekshirish va ularni osonlikcha izlash imkonini beradi. Ushbu algoritm stack (to‘plam) ma’lumotlar tuzilmasiga tayanadi, chunki u chuqurdan yuqoriga qaytishni talab qiladi. Rekursiv yondashuv esa DFSning mashhur implementatsiyalaridan biridir, chunki u tabiiy ravishda stack tuzilmasiga asoslangan. DFS algoritmini nafaqat dasturchilar va matematiklar, balki sun’iy intellekt, biologiya, va boshqa ko‘plab ilmiy sohalarda tadqiqot olib boruvchi mutaxassislar ham o‘z ishlarida qo‘llaydi.

DFS algoritmi murakkab masalalarni yechish uchun o‘zining oddiy, ammo samarali ishlash tamoyillari tufayli muhimdir. Uning ishlash prinsipi o‘yinlardagi qidiruv daraxtlaridan tortib, tarmoqdagi ulanish muammolarini aniqlashgacha bo‘lgan ko‘plab sohalarda qo‘llanilishi mumkin. DFSning vaqt murakkabligi O(V+E) (bu yerda V - tugunlar soni, E - qirralar soni) ekanligi sababli u juda katta graflar uchun ham samarali ishlashi mumkin. Shuningdek, ushbu algoritm xotira jihatidan ham tejamkor bo‘lib, qo‘shimcha joy talab qilish uchun faqat stack tuzilmasidan foydalanadi. Ammo, grafikning chuqurlik darajasi juda yuqori bo‘lsa, recursion stack hajmining oshib ketishi sababli muammo yuzaga kelishi mumkin.

DFS (Depth-First Search) algoritmi graflar va daraxtlarni chuqur tekshirishga asoslangan qidiruv algoritmi bo‘lib, uning ishlash tamoyili ma’lumot tuzilmasining chuqur qatlamlariga imkon qadar tezroq kirib borishni o‘z ichiga oladi. Bu algoritm rekurent (rekursiv) yondashuvga asoslanib yoki stack (to‘plam) tuzilmasi yordamida amalga oshiriladi. DFSning asosiy maqsadi — graflarni yoki daraxtlarni to‘liq tekshirish, barcha tugunlar bilan aloqa qilish va ular orasidagi bog‘lanishlarni tahlil qilishdir. Algoritm odatda ma’lum bir boshlang‘ich tugundan ishga tushadi va har bir tugunni bir marta tashrif buyurilgan deb belgilaydi, bu esa takroriy tekshiruvning oldini olish imkonini beradi. Vaqt murakkabligi O(V+E) ko‘rinishida hisoblanadi, bu yerda V- graflar yoki daraxtlarning tugunlari soni, E esa qirralar sonidir. Ushbu murakkablik shuni ko‘rsatadiki, algoritm graflarning har bir tuguni va har bir qirrasini faqat bir marta tekshiradi. Xotira murakkabligi esa graflarning chuqurlik darajasiga bog‘liq bo‘lib, u stackda saqlanadigan tugunlar soniga qarab O(D) (bu yerda D - graflarning maksimal chuqurligi) bo‘lishi mumkin. DFSning bunday samaradorligi katta hajmdagi ma’lumotlar tuzilmalarini tahlil qilish va qidiruv jarayonlarini amalga oshirish uchun uni muhim vositaga aylantiradi.

DFS algoritmining nazariy asoslari graflar nazariyasiga chuqur bog‘langan. Bu algoritm yordamida graflardagi bog‘liqlik komponentlarini aniqlash, aylanishlarni topish, topologik tartiblash kabi ko‘plab muammolarni hal qilish mumkin. DFS nafaqat yo‘nalmagan graflar, balki yo‘nalgan graflar bilan ham samarali ishlaydi. Bundan tashqari, DFS yordamida ma’lum bir tugundan boshqa tugunga yetish mumkinligini aniqlash, ma’lum bir yo‘lning mavjudligini tekshirish yoki grafda eng qisqa yo‘lni topish kabi vazifalarni bajarish mumkin. DFS shuningdek, graflarning ba’zi elementlarini klassifikatsiya qilish, masalan, ularni "kashf etilgan", "tugallangan" yoki "hali ko‘rilmagan" tugunlarga ajratishda qo‘llaniladi.

DFS algoritmi, shuningdek, boshqa qidiruv algoritmlari, masalan, BFS (Breadth-First Search) bilan taqqoslanadi. DFS chuqurlikka asoslangan bo‘lsa, BFS kenglikka asoslangan yondashuvni qo‘llaydi. DFS asosan grafikning ma’lum bir yo‘nalishidagi ma’lumotlarni tahlil qilish yoki chuqur muammolarni topish uchun qulaydir. Ammo, DFSning asosiy kamchiligi shundaki, u eng qisqa yo‘lni topishda samarali emas. Shu sababli, algoritmni tanlashda muammoning xususiyatlari hisobga olinishi lozim. DFS algoritmi nazariyasi graflar bilan ishlashning ko‘plab dolzarb masalalarida asos bo‘lib xizmat qiladi. Uning rekurent tabiati va chuqur tahlil qilish imkoniyatlari uni nafaqat matematik va algoritmik muammolar, balki amaliy ilovalarda ham samarali vositaga aylantiradi. Shu sababli, DFS algoritmini tushunish va uni qo‘llash ko‘plab texnik sohalarda muhim qadam hisoblanadi. DFSning bir qancha amaliy qo‘llanilishlariga masalan, o‘yinlar va sun'iy intellektda ham duch kelish mumkin. O‘yin dizaynida DFS algoritmi qarorlar daraxtini tekshirishda ishlatiladi. Misol uchun, chess yoki boshqa strategik o‘yinlarda DFS orqali o‘yinchi harakatlarini tahlil qilish va eng yaxshi harakatni tanlash mumkin. DFS bu kabi o‘yinlarda barcha imkoniyatlarni o‘rganish va ulardan eng yaxshisini tanlashda yordam beradi. Sun'iy intellekt tizimlarida esa DFSni qarorlar qabul qilish, qidiruv muammolarini hal qilish va rejalashtirishda qo‘llash mumkin. Bu tizimlar DFS orqali barcha mumkin bo‘lgan holatlarni chuqur tekshiradi va eng optimal yechimni topishga harakat qiladi.

Misol

Bu misolda biz grafni DFS yordamida chuqur qidiramiz va bog‘liqlik komponentlarini aniqlanadi. Grafni adjacency list yordamida ifodalab, ya'ni har bir tugunga qo‘shni tugunlar ro‘yxatini saqlab. DFSning rekursiv va iterativ usullarida korasataman.

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

class Graph
```

```
{
    private int vertices;
    private List<int>[] adjList;

    public Graph(int v)
    {
        vertices = v;
        adjList = new List<int>[v];
        for (int i = 0; i < v; i++)
        {
            adjList[i] = new List<int>();
        }
    }
    public void AddEdge(int u, int v)
    {
        adjList[u].Add(v);
        adjList[v].Add(u);
    }
    private void DFSUtil(int v, bool[] visited)
    {
        visited[v] = true;
        Console.Write(v + " ");
        foreach (int neighbor in adjList[v])
        {
            if (!visited[neighbor])
            {
                DFSUtil(neighbor, visited);
            }
        }
    }
    public void ConnectedComponents()
    {
        bool[] visited = new bool[vertices];
        int componentCount = 0;
        for (int v = 0; v < vertices; v++)
        {
            if (!visited[v])
            {
                Console.WriteLine("Komponent " + (++componentCount) + ":");
```

```
                DFSUtil(v, visited);
                Console.WriteLine();
            }
        }
    }
}

class Program
{
    static void Main(string[] args)
    {
        Graph g = new Graph(7);
        g.AddEdge(0, 1);
        g.AddEdge(0, 2);
        g.AddEdge(1, 3);
        g.AddEdge(4, 5);
        g.AddEdge(4, 6);
        Console.WriteLine("Bog'liqlik komponentlari:");
        g.ConnectedComponents();
    }
}
```

Natija

Bog’liqlik komponentlari:
Komponent 1:
0 1 3 2
Komponent 2:
4 5 6

1. **Graph sinfi**:
    - vertices — grafdagi tugunlar soni.
    - adjList — adjacency list, bu erda har bir tugun uchun qo‘shni tugunlar ro‘yxati saqlanadi.
    - AddEdge(int u, int v) — grafga yangi qirra (edge) qo‘shadi. Bu metod har ikki yo‘nalishda (undirected graph) qirra qo‘shadi.
    - DFSUtil(int v, bool[] visited) — DFS algoritmining asosiy rekursiv funksiyasi. Bu funksiya boshlang‘ich tugundan boshlanib, uning qo‘shni tugunlarini tekshiradi va barcha bog‘liq tugunlarni ko‘rsatadi.
    - ConnectedComponents() — graflardagi barcha bog‘liqlik komponentlarini aniqlash uchun ishlatiladi. Har bir yangi

komponentni topish uchun DFS chaqiriladi va yangi komponentni tekshirish uchun tugunlar tashrif buyuriladi.

2. **Program sinfi**:
    - Main metodida graf yaratilib, unga tugunlar va qirralar qo‘shiladi.
    - So‘ngra ConnectedComponents() metodi chaqirilib, grafdagi bog‘liqlik komponentlari aniqlanadi.

DFS algoritmi - bu graflarni chuqur qidiruv orqali tahlil qilishning samarali usulidir. U asosan rekursiv yoki stack yordamida ishlaydi va graflarning barcha tugunlari va qirralarini faqat bir marta tekshirishni ta’minlaydi. DFSning asosiy afzalligi uning chuqur tekshiruvni amalga oshirishi va murakkab strukturalarda ham samarali ishlashidir. Algoritmning ishlash tamoyili oddiy va tushunarli bo‘lib, graflarda bog‘liqlik komponentlarini aniqlash, aylanishlarni tekshirish, topologik saralash kabi ko‘plab amaliy masalalarni hal qilishda qo‘llaniladi. DFSning graflarda o‘ziga xos o‘rni bor. Uning murakkablik komponentlarini ajratish, masalalarni kombinatorik tarzda hal qilish va optimallashtirishda ishlatilishi juda muhim. Boshqa algoritmlar bilan taqqoslaganda, DFS chuqur tahlilni amalga oshiradi va ba'zi muammolarda eng yaxshi yechimlarni taqdim etadi. Bu algoritmni dasturlashda qo‘llash oson va ko‘plab sohalarda, masalan, tarmoq tahlili, sun'iy intellekt, o‘yin dizayni, va ma'lumotlar bazalarida muvaffaqiyatli ishlatiladi.

Foydalanilgan adabiyotlar:

1. Cormen, T. H., Leiserson, C. E., Rivest, R. L., & Stein, C. (2009). *Introduction to Algorithms* (3rd ed.). MIT Press.
2. Kleinberg, J., & Tardos, É. (2006). *Algorithm Design*. Pearson Education.
3. Sedgewick, R., & Wayne, K. (2011). *Algorithms* (4th ed.). Addison-Wesley.
4. Tarjan, R. E. (1972). Depth-First Search and Linear Graph Algorithms. *SIAM Journal on Computing*, 1(2), 146-160.
5. Knuth, D. E. (1973). *The Art of Computer Programming, Volume 1: Fundamental Algorithms*. Addison-Wesley.
6. Robert Sedgewick and Kevin Wayne. (2016). *Algorithms, Part I*. Coursera, Princeton University.
7. Cormen, T. H., Leiserson, C. E., Rivest, R. L., & Stein, C. (2011). *Introduction to Algorithms* (4th ed.). MIT Press.
8. Alon, N. (2000). *Graph Theory and Combinatorics: An Introduction*. Springer.
9. Turing, A. M. (1936). On Computable Numbers, with an Application to the Entscheidungsproblem. *Proceedings of the London Mathematical Society*, 2(42), 230-265.

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.
14. Raxmonjonovich, F. S. (2023). Dasturlash tillarida fayllar bilan ishlash mavzusini Blended Learning metodi yordamida o'qitish. Yangi O'zbekiston taraqqiyotida tadqiqotlarni o'rni va rivojlanish omillari, 2(2), 464-469.
15. Raxmonjonovich, F. S. (2023). DASTURLASHDA ISTISNOLARNING AHAMIYATI. Yangi O'zbekiston taraqqiyotida tadqiqotlarni o'rni va rivojlanish omillari, 2(2), 475-481.
16. Raxmonjonovich, F. S. (2023). Dasturlashda abstraksiyaning o’rni. Yangi O'zbekiston taraqqiyotida tadqiqotlarni o'rni va rivojlanish omillari, 2(2), 482-486.
17. Raxmonjonovich, F. S., & Ravshanbek o’g’li, A. A. (2023). Zamonaviy dasturlash tillarining qiyosiy tahlili. Yangi O'zbekiston taraqqiyotida tadqiqotlarni o'rni va rivojlanish omillari, 2(2), 430-433.
18. Raxmonjonovich, F. S. (2023). C# dasturlash tilida fayl operatsiyalari qo’llashning qulayliklari haqida. Yangi O'zbekiston taraqqiyotida tadqiqotlarni o'rni va rivojlanish omillari, 2(2), 439-446.
19. Raxmonjonovich, F. S. (2023). C# tilida ArrayList bilan ishlashning afzalliklari. Yangi O'zbekiston taraqqiyotida tadqiqotlarni o'rni va rivojlanish omillari, 2(2), 470-474.
20. Farmonov Sherzodbek Raxmonjonovich, & Rustamova Humoraxon Sultonbek qizi. (2024). C# DASTURLASH TILIDA TO’PLAMLAR BILAN ISHLASH. Ta’lim Innovatsiyasi Va Integratsiyasi, 11(10), 210–214. Retrieved from http://web-journal.ru/index.php/ilmiy/article/view/2480.
21. Raxmonjonovich, F. S., & Ravshanbek o’g’li, A. A. (2023). Zamonaviy dasturlash tillarining qiyosiy tahlili. Yangi O'zbekiston taraqqiyotida tadqiqotlarni o'rni va rivojlanish omillari, 2(2), 430-433.
22. Farmonov, S., & Rasuljonova, Z. (2024). OB'EKTGA YO'NALTIRILGAN DASTURLASH ZAMONAVIY DASTURLASHNING ASOSI

SIFATIDA. Центральноазиатский журнал образования и инноваций, 3(1), 83-86.
23. Farmonov, S., & Ro‘zimatov, J. (2024). DASTURLASH TILLARINI O‘RGANISHDA ONLINE TA’LIM PLATFORMALARIDAN FOYDALANISH. Theoretical aspects in the formation of pedagogical sciences, 3(1), 5-10.
24. Farmonov, S. R., & qizi Xomidova, M. A. (2024). C# VA JAVA DASTURLASH TILLARIDA FAYLLAR BILAN ISHLASHNING TURLI USULLARINING SAMARADORLIGI HAQIDA. Zamonaviy fan va ta'lim yangiliklari xalqaro ilmiy jurnal, 1(9), 45-51.
25. Raxmonjonovich, F. S. (2024). C# VA MASHINA TILI. Ta'lim innovatsiyasi va integratsiyasi, 12(1), 59-62.
26. Farmonov, S. (2023). C# DASTURLASH TILIDA GRAY KODI BILAN ISHLASH. Центральноазиатский журнал образования и инноваций, 2(12 Part 2), 71-74.
27. Farmonov, S., & Jo’rayeva, M. (2023, December). DASTURLASHDA POLIMORFIZMNING AHAMIYATI. In Международная конференция академических наук (Vol. 2, No. 13, pp. 5-8).
28. Farmonov, S., & Usmonaliyev, U. (2024). O'ZBEKISTON RESPUBLIKASI IT SOHASINING RIVOJLANISH ISTIQBOLLARI. Бюллетень педагогов нового Узбекистана, 2(1), 59-62.
29. Raxmonjonovich, F. S., & Xasan o'g'li, X. O. (2023). DASTURLASHDA SANA VA VAQTLAR BILAN ISHLASH. Ta'lim innovatsiyasi va integratsiyasi, 11(11), 3-6.