GRAF VA DARAXT TUZILMALARIDA BREADTH-FIRST SEARCH ALGORITMINING QO'LLANILISHI

Farmonov Sherzodbek Raxmonjonovich
Farg'ona Davlat Universiteti Amaliy matematika va informatika kafedrasi katta o'qituvchisi
farmonovsh@gmail.com
Madatova Ruxshona Bunyodbek qizi
Fizika -Matematika fakulteti "Kompyuter ilmlari va dasturlash texnologiyalari " yo'nalishi 23.12-guruh 2-bosqich talabasi
ruxshonamadatova4@gmail.com

Annotatsiya

Ushbu maqolada Breadth-First Search (BFS) algoritmi haqida batafsil ma'lumot berilgan. BFS algoritmi graf va daraxt tuzilmalarida qidiruv va traversiya uchun ishlatiladigan asosiy usul bo'lib, boshlang'ich tugundan boshlab, uning qo'shni tugunlarini birma-bir ko'rib chiqadi. Algoritmning asosiy qadamlariga navbatni boshqarish, tugunlarni ko'rib chiqish va takrorlash kiritilgan. Maqolada BFS algoritmining afzalliklari, jumladan soddaligi va eng qisqa yo'llarni topish imkoniyati, shuningdek, uning kamchiliklari, xususan xotira sarfi va katta grafiklar bilan bog'liq muammolar ko'rib chiqilgan.

Kalit so'zlar: Kenglik bo'yicha qidirish, graf, daraxt, qidirish algoritmi, navbat, boshlang'ich nuqta, yuqoridan pastga, tugunlar, bog'lanish, tizim.

Аннотация: В данной статье представлена подробная информация об алгоритме поиска в ширину (BFS). Алгоритм BFS является основным методом для поиска и обхода в графах и деревьях, начиная с начальной вершины и последовательно рассматривая её соседние вершины. В основные шаги алгоритма входят управление очередью, обход вершин и повторение. В статье рассмотрены преимущества алгоритма BFS, включая простоту и возможность нахождения кратчайших путей, а также его недостатки, в частности, потребление памяти и проблемы, связанные с большими графами.

Ключевые слова: Поиск в ширину (BFS) , Граф, Дерево, Алгоритм поиска, Очередь (queue), Начальная точка, Сверху вниз, Узлы (твердые), Связь, Система.

Annotation

This article provides detailed information about the Breadth-First Search (BFS) algorithm. The BFS algorithm is a fundamental method used for searching and traversing in graph and tree structures, starting from a source node and examining its neighboring nodes one by one. Key steps of the algorithm include managing the queue,

exploring nodes, and iterating. The article discusses the advantages of the BFS algorithm, including its simplicity and ability to find the shortest paths, as well as its disadvantages, particularly memory consumption and issues related to large graphs.

Key words: Breadth-First Search (BFS), Graph, Tree, Search algorithm, Queue, Starting point, Node, Connection, System, Search process.

Kirish

BFS (Kenglik bo'yicha qidirish) algoritmi graf va daraxt tuzilmalarida keng qo'llaniladi. Quyida BFS algoritmining graf va daraxtlarda qanday qo'llanilishi haqida batafsil ma'lumot beraman:

Graflarda BFS

1. Eng Qisqa Yo'lni Topish:

• BFS, agar grafda barcha qirralar teng vaznga ega bo'lsa, eng qisqa yo'lni topish uchun ishlatiladi. Misol uchun, shaharlar orasidagi transport yo'llarini ifodalovchi grafda, bir shahridan boshqasiga borish uchun eng qisqa yo'lni aniqlashda foydalaniladi.

2. Komponentlarni Topish:

• BFS yordamida grafning alohida komponentlarini aniqlash mumkin. Agar grafda alohida bog'lanmagan qismlar mavjud bo'lsa, BFS har bir komponentni qidirib topishda foydali bo'ladi.

3. Ijtimoiy Tarmoqlar:

• Ijtimoiy tarmoqlarda foydalanuvchilar o'rtasidagi aloqalarni tahlil qilishda, masalan, "do'stlar do'sti" kabi munosabatlarni aniqlashda BFS algoritmi qo'llaniladi.

4. Tarmoq Tarqatish:

• Tarmoq dasturlashida ma'lumot paketlarini uzatishda va marshrutlashda, eng samarali yo'llarni aniqlashda BFS yordamida tarmoqning strukturasini o'rganish mumkin.

5. Grafning O'zgarishi:

• Grafda tugunlar yoki qirralar qo'shilganda yoki o'chirilganda, BFS yordamida yangi holatlarni tahlil qilish va yangilangan grafni ko'rib chiqish mumkin.

Daraxtlarda BFS

1. Darajalar Bo'yicha O'qish:

• Daraxt tuzilmasida BFS algoritmi yordamida tugunlarni darajalari bo'yicha o'qish amalga oshiriladi. Bu, masalan, daraxtdagi har bir darajadagi tugunlarni birinchi darajadan oxirgi darajagacha ketma-ket o'qishga imkon beradi.

2. Daraxtning Balansini Tekshirish:

• BFS yordamida daraxtning balansi va tuzilishini tekshirish mumkin. Agar daraxtning har bir darajasidagi tugunlar soni teng bo'lsa, bu daraxt muvozanatli

hisoblanadi.

3. Foydalanuvchi Interfeyslari:

• Kompyuter o'yinlari yoki grafik interfeyslarida foydalanuvchi uchun ma'lumotlarni ko'rsatishda, masalan, menyu strukturasini ko'rsatishda BFS algoritmi yordamida har bir darajadagi elementlarni ko'rsatish mumkin.

4. Qidiruv va Filtratsiya:

Kenglik bo'yicha qidirish (BFS) — bu graf yoki daraxtdagi tugunlarni (nod) kenglik bo'yicha o‘rganish uchun ishlatiladigan algoritm. BFS algoritmi tugunlarni darajalari bo‘yicha o‘rganadi, ya'ni avval birinchi darajadagi tugunlarni, so‘ng ikkinchi darajadagi tugunlarni, va hokazo qidiradi. Bu usul, asosan, grafning eng qisqa yo‘llarini topish va turli xil muammolarni hal qilishda qo‘llaniladi.

2. Ishlash Printsipi

BFS algoritmining ishlash printsipi quyidagi bosqichlardan iborat:

1. Boshlang'ich Tugunni Tanlash: Algoritm boshlanishi uchun birinchi tugunni tanlaydi va uni "tashrif buyurilgan" deb belgilaydi.

2. Qatorni Ishlatish: Tashrif buyurilgan tugunlar qatorini (queue) yaratadi. Bu qator orqali tugunlar kenglik bo'yicha o'rganiladi.

3. Tugunlarni O'rganish:

• Boshlang'ich tugunni qatorga qo'shadi.

• Qator bo'sh bo'lmaguncha, tugunlarni qatordan olib, uning qo'shni tugunlarini ko'rib chiqadi.

• Qo'shni tugunlar hali tashrif buyurilmagan bo'lsa, ularni qatorga qo'shadi va "tashrif buyurilgan" deb belgilaydi.

4. Natijani Qaytarish: Algoritm tugunlar bilan ishlash jarayonida kerakli natijalarni qaytaradi (masalan, eng qisqa yo'l yoki tugunlar ro'yxati).

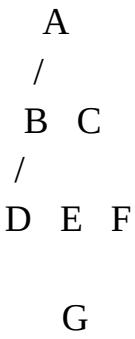
3. Algoritmning Pseudokodi

BFS algoritmining oddiy pseudokodi:

```
BFS(Grafo G, Tugun s):
    Q = bo'sh qator
    Q.enqueue(s)
    belgilangan[s] = to'g'ri

    while Q not bo'sh:
        v = Q.dequeue()
        for each qo'shni tugun u of v:
            if not belgilangan[u]:
                belgilangan[u] = to'g'ri
                Q.enqueue(u)
```

Misol:



grafda tugunlar va qirralar quyidagicha ifodalangan:
• Tugunlar: A, B, C, D, E, F, G
• Qirralar:

• A - B

• A - C

• B - D

• B - E

• C - F

• E - G

BFS Algoritmini Qo'llash

Biz BFS algoritmini A tugunidan boshlaymiz. Algoritmning har bir bosqichida qaysi tugunlar o'rganilayotganini va qaysi tugunlar qator (queue)da saqlanayotganini ko'rsatamiz.

1. Boshlang'ich Tugunni Tanlash:

• A tugunini tanlaymiz.
• Qator: [A]
• Tashrif buyurilgan tugunlar: {A}

2. BFS Jarayoni:

1-qadam: A tugunini chiqaramiz va uning qo'shni tugunlarini qo'shamiz (B va C).

• Qator: [B, C]
• Tashrif buyurilgan tugunlar: {A, B, C}

2-qadam: B tugunini chiqaramiz va uning qo'shni tugunlarini qo'shamiz (D va E).

• Qator: [C, D, E]
• Tashrif buyurilgan tugunlar: {A, B, C, D, E}

3-qadam: C tugunini chiqaramiz va uning qo'shni tugunini qo'shamiz (F).

• Qator: [D, E, F]
• Tashrif buyurilgan tugunlar: {A, B, C, D, E, F}

4-qadam: D tugunini chiqaramiz. D tugunining qo'shni tugunlari yo'q.

• Qator: [E, F]
• Tashrif buyurilgan tugunlar: {A, B, C, D, E, F}

5-qadam: E tugunini chiqaramiz va uning qo'shni tugunini qo'shamiz (G).

• Qator: [F, G]
• Tashrif buyurilgan tugunlar: {A, B, C, D, E, F, G}

6-qadam: F tugunini chiqaramiz. F tugunining qo'shni tugunlari yo'q.

• Qator: [G]
• Tashrif buyurilgan tugunlar: {A, B, C, D, E, F, G}

7-qadam: G tugunini chiqaramiz. G tugunining qo'shni tugunlari yo'q.

• Qator: [] (bo'sh)
• Tashrif buyurilgan tugunlar: {A, B, C, D, E, F, G}

Natija

BFS algoritmi A tugunidan boshlanib barcha tugunlarni kenglik bo'yicha o'rgandi. Natijada olingan tartib:

• O'rganilgan tugunlar tartibi: A → B → C → D → E → F → G

C# dasturlash tili:

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

class Graph
{
    private int vertices;
    private List<int>[] adjacencyList;

    public Graph(int v)
    {
```

```
        vertices = v;
        adjacencyList = new List<int>[v];

        for (int i = 0; i < v; i++)
        {
            adjacencyList[i] = new List<int>();
        }
    }
    public void AddEdge(int v, int w)
    {
        adjacencyList[v].Add(w);
        adjacencyList[w].Add(v);
    }

    public void BFS(int startVertex)
    {
        bool[] visited = new bool[vertices];
        Queue<int> queue = new Queue<int>();

        visited[startVertex] = true;
        queue.Enqueue(startVertex);

        while (queue.Count > 0)
        {
            int currentVertex = queue.Dequeue();
            Console.Write(currentVertex + " ");
            foreach (int neighbor in adjacencyList[currentVertex])
            {
                if (!visited[neighbor])
                {
                    visited[neighbor] = true;
                    queue.Enqueue(neighbor);
                }
            }
        }
    }
}

class Program
```

```
{
    static void Main(string[] args)
    {
        Graph graph = new Graph(7);
        graph.AddEdge(0, 1); // A - B
        graph.AddEdge(0, 2); // A - C
        graph.AddEdge(1, 3); // B - D
        graph.AddEdge(1, 4); // B - E
        graph.AddEdge(2, 5); // C - F
        graph.AddEdge(4, 6); // E - G

        Console.WriteLine("BFS natijasi (A dan boshlanadi):");
        graph.BFS(0); // 0 - A tuguni
    }
}
```

Xulosa

Breadth-First Search (BFS) algoritmi graf va daraxt tuzilmalarida qidiruv va traversiya uchun muhim vosita hisoblanadi. Ushbu algoritmning asosiy afzalliklari uning soddaligi, qisqa yo'llarni topish qobiliyati va tugunlarni darajalarga ajratish imkoniyatidir. Biroq, BFS algoritmining xotira sarfi katta bo'lishi mumkinligi va kengaytirilgan grafiklarda samarali ishlamasligi kabi kamchiliklari ham mavjud.

BFS algoritmi turli sohalarda, masalan, yo'l qidirish, ijtimoiy tarmoqlar va o'yin dasturlarida keng qo'llaniladi. Ushbu maqolada BFS algoritmining asosiy qadamlarini, afzalliklarini va kamchiliklarini ko'rib chiqdik, shuningdek, uning amaliyotdagi qo'llanilishi haqida ma'lumot berdik. Umuman olganda, BFS algoritmi zamonaviy dasturlash va ma'lumotlar tuzilmalari nazariyasida muhim rol o'ynaydi va uning qo'llanilishi ko'plab sohalarda foydali bo'lishi mumkin.

Foydalanilgan adabiyotlar:

1. "Network Flows: Theory, Algorithms, and Applications" - Ravindra K. Ahuja, Thomas L. Magnanti, James B. Orlin
2. "Optimization Methods in Operations Research and Systems Analysis" - A. Ravindran, D. T. Phillips, J. J. Solberg
3. "Introduction to Operations Research" - Frederick S. Hillier, Gerald J. Lieberman
4. Operations Research: Applications and Algorithms" - Wayne L. Winston
5. Mykel J. Kochenderfer. Tim A. Wheeler. Algorithms for Optimization. Published by The MIT Press., in London, England. 2019. – 500 p.
6. Рафгарден Тим. Совершенный алгоритм. Графовые алгоритмы и структуры данных. – СПб.: Питер, 2019. - 256 с.

7. Ахо Альфред В., Ульман Джеффри Д., Хопкрофт Джон Э. Структуры данных и алгоритмы. – М.: Вильямс, 2018. – 400 с.
8. Дж.Хайнеман, Г.Поллис, С.Стэнли. Алгоритмы. Справочник с примерами на C, C++, Java и Python, 2-е изд.: Пер. с англ. — СпБ.: ООО "Альфа-книга", 2017. — 432 с.
9. Farmonov, S., & Nazirov, A. (2023). C# DASTURLASH TILIDA GRAY KODI BILAN ISHLASH. B CENTRAL ASIAN JOURNAL OF EDUCATION AND INNOVATION (Т. 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.
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.