GRAFIK MUAMMOLARNI HAL QILISHDA PRIM ALGORITMIDAN FOYDALANISH

Farmonov Sherzodbek Raxmonovich
Farg'ona davlat universiteti amaliy matematika va informatika kafedrasi katta o'qituvchisi
farmonovsh@gmail.com
Karimova Muxsina Anvarbek qizi
Farg'ona davlat universiteti 2-kurs talabasi
muxsinakarimova883@gmail.com

Annotatsiya: Ushbu maqolada Prim algoritmining asosiy tamoyillari, uning ishlash jarayoni va grafik nazariyasidagi ahamiyati ko'rib chiqiladi. Prim algoritmi, og'irlikni grafda minimal tarqalish daraxtini (MST) topish uchun ishlatiladi va ko'plab amaliy muammolarni hal qilishda qo'llaniladi.

Kalit so'zlar: Iteratsiya, min-heap, prioritet, graf nazariyasi, MST, tezislar.

Annotation: The article discusses the basic principles of Prim's algorithm, its working process, and its significance in graph theory. Prim's algorithm is used to find the Minimum Spanning Tree (MST) in a weighted graph and is applied in solving many practical problems.

Keywords: Iteration, min-heap, graph theory, minimum spanning tree, theses.

Аннотация: В этой статье рассматриваются основные принципы алгоритма Прима, его процесс работы и значение в теории графов. Алгоритм Прима используется для нахождения минимального остовного дерева (MST) в графе с весами и применяется для решения множества практических задач.

Ключевые слова: Итерация, Мин-куча, Приоритет, Графовая теория, Минимальное остовное дерево (MST), Алгоритм Прима, Применение

Grafik nazariyasi, turli xil muammolarni modellashtirish va hal etish uchun keng qo'llaniladigan matematik soha hisoblanadi. Ushbu maqolada, og'irlikli grafda minimal tarqalish daraxtini topish uchun mo'ljallangan Prim algoritmi batafsil tahlil qilinadi. Ushbu algoritm, ko'plab real hayotdagi muammolarni hal etishda muhim rol o'ynaydi.

Prim algoritmi — graf nazariyasida minimal bog'lanish daraxtini (MST) topish uchun ishlatiladigan algoritm. Boshlanishi algoritm biror bir boshlang'ich tugunni tanlaydi va uni MST ga qo'shadi. Bu tugun "tanlangan" tugunlar to'plamiga kiritiladi. Chegaralarni yangilash tanlangan tugunlar to'plamiga kiritilgan tugunlardan chiqadigan barcha qirralarni ko'rib chiqamiz. Har bir qirra uchun, agar u tanlanmagan tugun bilan bog'langan bo'lsa, u holda bu qirra "chegaraviy" qirra hisoblanadi. Ushbu qirralarning vaznlarini saqlab qolamiz. Eng kichik qirra

tanlashi- chegaraviy qirralar orasidan eng kichik vaznga ega bo'lganini tanlaymiz va uni MST ga qo'shamiz. Tanlangan qirra orqali bog'langan tugun "tanlangan" tugunlar to'plamiga qo'shiladi. Takrorlanishi 2-3 bosqichlarni, ya'ni chegaralarni yangilash va eng kichik qirrani tanlash jarayonini, barcha tugunlar tanlangan bo'lmaguncha takrorlaymiz. Natijada barcha tugunlar tanlangan bo'lganda, MST hosil bo'ladi.

Prim Algoritmining Roli

Prim algoritmi — bu minimal bog'lanish daraxtini (MST) topish uchun ishlatiladigan samarali algoritm. Uning roli berilgan og'irlikli grafda minimal bog'lanish daraxtini topishda ishlatiladi. Bu daraxt grafdagi barcha tugunlarni bog'laydi va qirralar vaznlarining yig'indisi minimal bo'ladi. MST topish, masalan, tarmoq dizayni va transport yo'llarini optimallashtirishda muhimdir. Prim algoritmi tushunish va amalga oshirish oson bo'lgan oddiy algoritmlardan biridir. Bu uning o'qitish va o'rganish jarayonida qulay bo'lishini ta'minlaydi. Algoritm bir tugundan boshlanadi va har safar eng kichik vaznli qirra orqali yangi tugunni qo'shib boradi. Bu jarayon tugunlar to'plami to'liq bog'langanicha davom etadi. Prim algoritmini Kruskal algoritmi bilan solishtirganda, Prim algoritmi ko'proq o'zaro bog'langan grafalar uchun samarali bo'lishi mumkin, chunki u har safar eng yaqin tugunni tanlaydi, bu esa tez-tez bog'langan strukturalarda afzallik beradi.

Umuman olganda, Prim algoritmi graf nazariyasida va amaliy muammolarni hal qilishda muhim rol o'ynaydi, chunki u samarali va ishonchli yechimlarni taqdim etadi.

Prim Algoritmining Afzalliklari

Prim algoritmining samaradorligi uning murakkabligi va ishlash tezligiga bog'liq. Prim algoritmi grafdagi minimal bog'lanish daraxtini (MST) topish uchun ishlatiladi va uning samaradorligi quyidagi jihatlarga asoslanadi:

Prim algoritmining murakkabligi grafning tuzilishiga va foydalanilayotgan ma'lumotlar tuzilmasiga qarab farq qiladi:

• **Oddiy ro'yxat bilan:** Agar grafning tugunlari oddiy ro'yxat yoki massivda saqlansa, Prim algoritmining murakkabligi $O(V^2)$ bo'ladi, bu yerda V - grafning tugunlari soni.

• **Prioritet navbati (min-heap) bilan:** Agar prioritet navbati (masalan, min-heap) ishlatilsa, murakkablik $O(E \log V)$ ga tushadi, bu yerda E - grafning qirralari soni. Bu variant katta grafiklar uchun samaraliroq hisoblanadi.

• **Ishlash tezligi**: Prim algoritmi o'zining oddiyligini va ko'p hollarda samaradorligini ta'minlaydi. U har bir tugunni bir marta ko'rib chiqadi va har safar eng kichik qirrani tanlaydi. Bu jarayon grafning barcha tugunlari uchun amalga oshiriladi.

• **Grafning Tuzilishi:** Grafning tuzilishi ham samaradorlikka ta'sir qiladi. Masalan, agar graf juda zich (ya'ni, qirralar soni tugunlar sonidan ancha ko'p bo'lsa) bo'lsa, Prim algoritmi yanada samarali bo'lishi mumkin.

Amaliy qo'llanilishi

1. Boshlang'ich nuqtani tanlash: Algoritm biror bir boshlang'ich tugmani tanlash bilan boshlanadi. Bu tugma MSTning birinchi tugmasi bo'ladi.

2. Qirralarni tanlash: Tanlangan tugma bilan bog'liq bo'lgan barcha qirralar (ya'ni, u tugmadan chiqarilgan qirralar) ko'rib chiqiladi. Har bir qirraning og'irligi (yoki narxi) hisobga olinadi.

3. Eng kichik qirralarni qo'shish: Har bir iteratsiyada eng kichik og'irlikka ega bo'lgan qirra tanlanadi va MSTga qo'shiladi. Usgbu qirra bilan bog'liq yangi tugmalar ham MSTga qo'shiladi.

4. Takrorlash: Ushbu jarayon, barcha tugmalar MSTga qo'shilguncha davom etadi. Har safar eng kichik qirra tanlanganda, MST kengayadi va yangi tugmalar qo'shiladi.

5. Tugallanish: Barcha tugmalar MSTga qo'shilgach, algoritm tugaydi va natijada minimal bog'lanish daraxti hosil bo'ladi.

Prim algoritmi, o'zining oddiyligi va samaradorligi sababli ko'plab amaliy muammolarni hal qilishda qo'llaniladi.

Prim algoritmiga C# dasturida masala:

Masala: Berilgan og'irlikli grafikning minimal bog'lanish daraxtini Prim algoritmi yordamida toping va bu daraxtga kiruvchi qirralar bilan ularning umumiy og'irligini hisoblang.

C# dasturi: Prim algoritmi

```
namespace PrimAlgoritmi
{
   internal class Program
   {

         static void Main(string[] args)
         {
            int verticesCount = 5;
            List<Tuple<int, int, int>> edges = new List<Tuple<int, int, int>>()
            {
                Tuple.Create(0, 1, 2),
                Tuple.Create(0, 3, 6),
                Tuple.Create(1, 2, 3),
                Tuple.Create(1, 3, 8),
                Tuple.Create(1, 4, 5),
                Tuple.Create(2, 4, 7),
```

```
            Tuple.Create(3, 4, 9)
        };

        Prim(verticesCount, edges);
    }

    static void Prim(int verticesCount, List<Tuple<int, int, int>> edges)
    {
        List<Tuple<int, int>>[] graph = new List<Tuple<int, int>>[verticesCount];
        for (int i = 0; i < verticesCount; i++)
        {
            graph[i] = new List<Tuple<int, int>>();
        }
        foreach (var edge in edges)
        {
            graph[edge.Item1].Add(Tuple.Create(edge.Item2, edge.Item3));
            graph[edge.Item2].Add(Tuple.Create(edge.Item1, edge.Item3));
        }

        bool[] inMST = new bool[verticesCount];
        int[] minEdge = new int[verticesCount];
        int[] parent = new int[verticesCount];
        for (int i = 0; i < verticesCount; i++)
        {
            minEdge[i] = int.MaxValue;
            parent[i] = -1;
        }

        minEdge[0] = 0;

        for (int count = 0; count < verticesCount - 1; count++)
        {
            int u = GetMinKey(minEdge, inMST);
            inMST[u] = true;

            foreach (var neighbor in graph[u])
            {
                int v = neighbor.Item1;
                int weight = neighbor.Item2;
```

```
                if (!inMST[v] && weight < minEdge[v])
                {
                    minEdge[v] = weight;
                    parent[v] = u;
                }
            }
        }

        PrintMST(parent, graph);
    }

    static int GetMinKey(int[] minEdge, bool[] inMST)
    {
        int min = int.MaxValue;
        int minIndex = -1;

        for (int v = 0; v < minEdge.Length; v++)
        {
            if (!inMST[v] && minEdge[v] < min)
            {
                min = minEdge[v];
                minIndex = v;
            }
        }

        return minIndex;
    }

    static void PrintMST(int[] parent, List<Tuple<int, int>>[] graph)
    {
        Console.WriteLine("Minimal bog'lanish daraxti:");
        for (int i = 1; i < parent.Length; i++)
        {
            foreach (var neighbor in graph[i])
            {
                if (neighbor.Item1 == parent[i])
                {
                    Console.WriteLine($"Tugun {parent[i]} - Tugun {i}
og'irlik: {neighbor.Item2}");
```

```
                }
            }
        }
        Console.ReadKey();
    }
  }
}
```

Masalaning berilishi:

- Tugunlar soni n=5.
- Qirralar ro‘yxati:{(0,1,2),(0,3,6),(1,2,3),(1,3,8),(1,4,5),(2,4,7),(3,4,9)}.

Natija:

- Minimal bog‘lanish daraxti qirralari: {(0,1),(1,2),(1,4),(0,3)}
- Daraxtning umumiy og‘irligi: 2+3+5+6=16

Dastur qanday ishlaydi:

1. Grafni belgilash: Grafni qo'shni ro'yxat shaklida matritsa (graph) orqali ifodalaymiz.

2. Prim algoritmi: PrimMST metodi orqali MST hisoblanadi.

- inMST massivida MSTga kiritilgan tugunlar saqlanadi.
- minEdge va parent massivlari yordamida eng kichik qirra tanlanadi.

3. Natijani chiqarish: PrintMST metodi orqali MSTning qirralari va og'irliklari konsolga chiqariladi.

Foydalanilgan adabiyotlar :

1. Marcin Jamro. C# Data Structures and Algorithms. Second Edition. Published by Packt Publishing Ltd., in Birmingham, UK. 2024. – 349 p.

2. Дж.Эриксон. Алгоритмы.: – М.: " ДМК Пресс ", 2023. – 528 с.

3. Hemant Jain. Data Structures & Algorithms using Kotlin. Second Edition. in India. 2022. – 572 p.

4. Н. А. Тюкачев, В. Г. Хлебостроев. C#. Алгоритмы и структуры данных: учебное пособие для СПО. – СПб.: Лань, 2021. – 232 с.

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, Sh., & 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, Sh., & Toirov, S. (2023). NETDA DASTURLASHNING ZAMONAVIY TEXNOLOGIYALARINI O'RGANISH. Theoretical aspects in the formation of pedagogical sciences, 2(22), 90-96

11. Raxmonjonovich, F. Sh. (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. Sh. (2023). Dasturlashda interfeyslardan foydalanishning ahamiyati. Yangi O'zbekiston taraqqiyotida tadqiqotlarni o'rni va rivojlanish omillari, 2(2), 425-429.

13. Raxmonjonovich, F. Sh. (2023). Dasturlashda obyektga yo'naltirilgan dasturlashning ahamiyati. Yangi O'zbekiston taraqqiyotida tadqiqotlarni o'rni va rivojlanish omillari, 2(2), 434-438.

14. Raxmonjonovich, F. Sh. (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. Sh. (2023). DASTURLASHDA ISTISNOLARNING AHAMIYATI. Yangi O'zbekiston taraqqiyotida tadqiqotlarni o'rni va rivojlanish omillari, 2(2), 475-481.

16. Raxmonjonovich, F. Sh. (2023). Dasturlashda abstraksiyaning o'rni. Yangi O'zbekiston taraqqiyotida tadqiqotlarni o'rni va rivojlanish omillari, 2(2), 482-486.

17. Raxmonjonovich, F. Sh., & 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. Sh. (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. Sh. (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).