TASHQI RESURSLARNING OPTIMAL TAQSIMOTINI REJALASTISHDA PRIM ALGORITMINING QO’LLANILISHI

Farmonov Sherzodbek Raxmonjonovich
Farg’ona davlat unversiteti amaliy matematika va informatika kafedrasi katta o‘qituvchisi
Farmonovsh@gmail.com
Akramjonova Gulsanam Olimjon qizi
Farg’ona davlat universiteti 2-kurs talabasi
malaknur1m.o5@gmail.com

Annotatsiya: Ushbu maqolada Prim algoritmi, graf nazariyasida minimum spanning tree (MST) topish uchun ishlatiladigan samarali algoritm sifatida ko'rib chiqiladi. Prim algoritmi, berilgan grafdagi eng kichik og'irlikka ega bo'lgan bog'lanishlarni tanlab, birinchi nuqtadan boshlanib, bosqichma-bosqich yangi nuqtalarni qo'shish orqali MST ni quradi. Maqolada algoritmning ishlash prinsipi, uning murakkabligi, shuningdek, amaliyotda qo'llanilishi va boshqa MST algoritmlari bilan taqqoslanishi batafsil bayon etiladi. Shuningdek, Prim algoritmining turli xil variantlari va ularning afzalliklari ham ko'rib chiqiladi.

Kalit so'zlar: Prim algoritmi, Minimum spanning tree(MST), Graf nazariyasi, algoritm murakkabligi, amaliy qo’llanish, MST algoritmlari,bog’lanishlar,og’irlik,turli xil variantlar

Annotation: This article examines Prim's algorithm as an efficient method for finding the minimum spanning tree (MST) in graph theory. Prim's algorithm selects the smallest weight connections in a given graph, starting from an initial point and incrementally adding new points to construct the MST. The article details the working principle of the algorithm, its complexity, as well as its practical applications and comparisons with other MST algorithms. Additionally, various variants of Prim's algorithm and their advantages are discussed.

Keywords: Prim's algorithm, Minimum spanning tree (MST), Graph theory, Algorithm complexity, Practical applications, MST algorithms, Connections, Weight, Various variants

Аннотация:В данной статье рассматривается алгоритм Прима как эффективный алгоритм для поиска минимального остовного дерева (MST) в теории графов. Алгоритм Прима строит MST, выбирая соединения с наименьшим весом, начиная с одной вершины и постепенно добавляя новые вершины. В статье подробно описываются принцип работы алгоритма, его сложность, а также применение на практике и сравнение с другими алгоритмами

MST. Также рассматриваются различные варианты алгоритма Прима и их преимущества.

Ключевые слова: Алгоритм Прима, Минимальное остовное дерево (MST), Теория графов, Сложность алгоритма, Практическое применение, Алгоритмы MST, Соединения, Вес, Различные варианты

Prim algoritmi – bu graf nazariyasida minimal ostov daraxtini (MST) topish uchun ishlatiladigan samarali algoritm. Ushbu algoritm, berilgan bog'lanishlar (graf) ichidan eng kichik og'irlikka ega bo'lgan ostov daraxtini qurishga qaratilgan.

Qo'llanilishi: Prim algoritmi ko'plab amaliy masalalarda, masalan, tarmoq dizayni, yo'l qurilishi va boshqa sohalarda qo'llaniladi. U asosan bog'lanishlarning og'irliklari bilan ishlaydigan masalalarda samarali hisoblanadi. Shuningdek, Prim algoritmi, Kruskal algoritmi bilan birga, minimal ostov daraxtini topishning eng mashhur usullaridan biri hisoblanadi.

Prim algoritmining bosqichlari

Prim algoritmi minimal ostov daraxtini (MST) topish uchun quyidagi bosqichlarda amalga oshiriladi:

1.Boshlanish:

Tugmalarni tanlash: Algoritmni boshlash uchun biror bir tugmani (vertex) tanlaymiz.Bu tugma MSTning boshlanish nuqtasi bo'ladi.

Ostov daraxtini yaratish: Tanlangan tugmani MSTga qo'shiladi.

2.Bog’lanishlarni kuzatish:

Qayta ko'rib chiqish: Hozirgi MSTga qo'shilgan tugmalar atrofidagi bog'lanishlarni (edges) kuzating. Bu bog'lanishlar MSTga qo'shilishi mumkin bo'lgan tugmalar bilan bog'langan bo'lishi kerak.

3.Eng kichik bog'lanishni tanlash:

Tanlov: Hozirgi MSTga qo'shilgan tugmalar atrofidagi bog'lanishlar orasidan eng kichik og'irlikka ega bo'lgan bog'lanishni tanlang. Bu bog'lanish orqali yangi tugma MSTga qo'shiladi.

4. Yangi tugmani qo'shish:

Qo'shish: Tanlangan bog'lanish orqali yangi tugmani MSTga qo'shing va uni MSTning bir qismi sifatida belgilab qo'ying.

5. Takrorlash:

Qadamlarni takrorlash: 2-4 bosqichlarni, barcha tugmalar MSTga qo'shilguncha takrorlang. Har safar yangi tugma qo'shilganda, MST kengayadi.

6.Tugash:

Natijani olish: Barcha tugmalar MSTga qo'shilgach, algoritm tugaydi va natijada minimal ostov daraxti hosil bo'ladi.

Qo'shimcha izohlar:

• Prioritet navbatlari: Algoritm samaradorligini oshirish uchun, eng kichik bog'lanishni tezda topish uchun prioritet navbatlari (masalan, min-heap) ishlatilishi mumkin.

• Og'irliklar: Agar grafda og'irliklar berilmagan bo'lsa, u holda bog'lanishlar teng deb hisoblanadi va har qanday bog'lanish tanlanishi mumkin.

Bu bosqichlar orqali Prim algoritmi minimal ostov daraxtini samarali tarzda topadi.

Prim algoritmining qo’llanilish sohalari

Prim algoritmi, minimal ostov daraxtini (MST) topish uchun ishlatiladigan samarali algoritm bo'lib, ko'plab sohalarda qo'llaniladi. Quyida uning asosiy qo'llanilish sohalari keltirilgan:

1.Tarmoq dizayni:

• Telekommunikatsiya tarmoqlari: Telefon va internet tarmoqlarini qurishda minimal xarajatli bog'lanishlarni aniqlash uchun.

• Transport tarmoqlari: Yo'llar, temir yo'llar yoki boshqa transport infratuzilmasini loyihalashda optimal yo'nalishlarni belgilashda.

2.Ma'lumotlar tuzilishi va algoritmlar

• Grafikalar: Katta grafikalar ustida ishlashda, masalan, ijtimoiy tarmoqlar yoki ma'lumotlar bazalaridagi aloqalarni aniqlashda.

• O'qituvchi tizimlar: Ma'lumotlarni optimallashtirish va samarali strukturalarni yaratishda.

2.Kiberxavfsizlik

• Tarmoq xavfsizligi: Tarmoqdagi zaif joylarni aniqlash va xavfsizlikni ta'minlashda minimal bog'lanishlarni aniqlashda.

4. O'yin Dizayni

• O'yin xaritalari: O'yinlarda xaritalarni tuzishda yoki resurslarni optimallashtirishda minimal xarajatli yo'llarni aniqlash uchun.

5. Sun'iy intellekt va mashinani o'qitish

• Rekomendatsiya tizimlari: Foydalanuvchilar o'rtasidagi aloqalarni tahlil qilish va eng samarali tavsiyalarni berish uchun.

6. Logistika va ta'minot zanjiri

• Resurslarni taqsimlash: Resurslarni optimal taqsimlash va yetkazib berish yo'llarini belgilashda.

7. Biologiya va ekologiya

• Ekologik tadqiqotlar: O'simliklar va hayvonlar o'rtasidagi bog'lanishlarni tahlil qilishda, masalan, ekosistemalarda resurslarning taqsimlanishini o'rganishda.

Prim algoritmi ko'plab sohalarda qo'llanilib, turli xil muammolarni hal qilishda yordam beradi. U samaradorligi va oddiyligi bilan ajralib turadi, bu esa uni keng qo'llaniladigan vosita qiladi.

Prim algoritmi tashqi resurslarning optimal taqsimotini rejalashtirishda quyidagi jarayonlarda qo'llanilishi mumkin:

1. Resurs ta'minoti: Mamlakatlar yoki shaharlar o'rtasida resurslarni (masalan, suv, elektr energiyasi yoki gaz) taqsimlashda, minimal xarajatlar bilan ta'minot tarmog'ini tashkil qilish uchun Prim algoritmidan foydalanish mumkin. Bu holatda, resurs manbalaridan iste'molchilar ga yetkazish yo'llarini tanlash zarur.

2. Transport tarmoqlari: Mahsulotlarni bir joydan boshqasiga olib o'tish uchun optimal маршрутlarini aniqlashda. Misol uchun, fabrikadan do'konlarga yoki bir shahar ichida turli nuqtalarga mahsulotlarni tarqatishda.

3. Tarmoqlarni qurish: Masalan, aloqa tarmoqlari yoki internet infratuzilmasini qurish jarayonida, turli simlar yoki bog'lanishlar o'rtasida eng tejamkor masofalarni tanlashda ham Prim algoritmi qo'llanilishi mumkin. Maxsus holda, to'garakda joylashgan stansiyalar o'rtasida aloqa simlarini o'rnatishda minimal narxni ta'minlash uchun.

4. Yerdan foydalanish: Qishloq xo'jaligi yoki boshqa ishlab chiqarish sohalaridagi resurslarni (masalan, ekin maydonlari yoki o'rmonlar) optimal taqsimlash va ulardan samarali foydalanishni rejalashtirishda**.**

Prim algoritmining afzalliklari va kamchiliklari

Prim algoritmi graf nazariyasi doirasida ishlatiladigan muhim algoritmlardan biridir. Bu algoritm minimal bog'lanish daraxtini (MST - Minimum Spanning Tree) topish uchun mo'ljallangan. Quyida Prim algoritmining afzalliklari va kamchiliklari haqida ma'lumot beraman:

Afzalliklari:

1. Soddaligi: Prim algoritmi ancha sodda va tushunarli bo'lib, uni amalga oshirish oson.

2. Ishlash tezligi: Kichik grafiklar uchun (masalan, n < 100) Prim algoritmi juda tez ishlaydi. Uning vaqti O(E log V) bo'lishi mumkin, bu yerda E - chekkalar soni, V - tugunlar soni.

3. To‘g‘ridan-to‘g‘ri qo‘llash: Agar grafikni matritsa ko‘rinishida ifodalangan bo‘lsa, bu algoritm juda samarali ishlaydi.

4. Minimal bog'lanish: Har qanday bog'lanish grafikasi uchun minimal bog'lanish daraxtini topishga qodir.

5. Qo'shimcha ma'lumotlarni hisoblash: Algoritmni qo'llash davomida boshqa ko'plab ma'lumotlarni (masalan, eng qisqa yo'l) hisoblash imkoniyatiga ega bo'lishingiz mumkin.

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

```
class Program
{
    static void Main(string[] args)
    {
        // Grafning og'irliklarini belgilash
        int[,] graph = new int[,]
        {
            { 0, 2, 0, 6, 0 },
            { 2, 0, 3, 8, 5 },
            { 0, 3, 0, 0, 7 },
            { 6, 8, 0, 0, 9 },
            { 0, 5, 7, 9, 0 }
        };

        PrimMST(graph);
    }

    static void PrimMST(int[,] graph)
    {
        int verticesCount = graph.GetLength(0);
        bool[] inMST = new bool[verticesCount];
        int[] key = new int[verticesCount];
        int[] parent = new int[verticesCount];

        // Barcha kalitlarni cheksizga o'rnatamiz
        for (int i = 0; i < verticesCount; i++)
        {
            key[i] = int.MaxValue;
            inMST[i] = false;
        }

        // Birinchi tugunni tanlaymiz
        key[0] = 0; // Birinchi tugun uchun kalit
        parent[0] = -1; // Birinchi tugun uchun ota tugun yo'q

        for (int count = 0; count < verticesCount - 1; count++)
        {
            // Minimal kalitni tanlash
            int u = MinKey(key, inMST);
            inMST[u] = true;

            // Qo'shni tugunlar orqali o'tish
            for (int v = 0; v < verticesCount; v++)
            {
```

```
                if (graph[u, v] != 0 && !inMST[v] && graph[u, v] < key[v])
                {
                    parent[v] = u;
                    key[v] = graph[u, v];
                }
            }
        }

        PrintMST(parent, graph);
    }

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

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

    static void PrintMST(int[] parent, int[,] graph)
    {
        Console.WriteLine("Minimal Ostov Daraxti:");
        for (int i = 1; i < parent.Length; i++)
        {
            Console.WriteLine($"Tugun {parent[i]} - Tugun {i}: Og'irlik {graph[i,
parent[i]]}");
        }
    }
}
```

Kodni tushuntirish:

1. Grafni belgilash: graph massivi orqali grafning og'irliklarini belgilaymiz. 0 qiymati bog'lanmagan tugunlar orasidagi aloqani ifodalaydi.

2. PrimMST metodi: Bu metod minimal ostov daraxtini hisoblash uchun asosiy algoritmni amalga oshiradi.

3. MinKey metodi: Bu metod minimal kalit qiymatini va unga tegishli tugunni topadi.

Foydalanilgan adabiyotlar:

1.Karp, R. M. (1991). An introduction to randomized algorithms. Discrete 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, 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.

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