KRUSKALA ALGORITMI: SODDALIGI VA SAMARADORLIGI BILAN ENG KAM XARAJATLI YO‘LNI TANLASH

Farmonov Sherzodbek Raxmonjonovich
Farg‘ona davlat universiteti amaliy matematika va informatika kafedrasi katta o‘qituvchisi
e-mail: farmonovsh@gmail.com
Ismoiljonov Hasanboy Dilshodjon o’g’li
Farg’ona davlat universiteti talabasi
e-mail: hasanboyismoiljonov0@gmail.com

Annotatsiya: *Kruskala algoritmi graf nazariyasida minimal ulash daraxtini topish uchun samarali va sodda usullardan biri hisoblanadi. Ushbu algoritm grafdagi barcha qirralarni ularning qiymatiga qarab tartiblaydi va qadam-baqadam eng kam xarajatli ulanishlarni tanlash orqali optimal natijaga erishadi. Kruskala algoritmi tarmoq qurilishi, transport tizimlarini optimallashtirish, energiya ta’minoti va aloqa tarmoqlarini boshqarish kabi sohalarda keng qo‘llaniladi. Uning greedy yondoshuvi orqali har bir bosqichda eng yaxshi lokal yechim tanlanadi va bu global optimal natijaga olib keladi. Samarali ishlash uchun qirralarning tartiblangan bo‘lishi zarur bo‘lib, algoritmning vaqt murakkabligi O(E log E) ko‘rinishida aniqlanadi, bu esa katta graflar bilan ishlash imkonini beradi.*

Kalit so‘zlar: *Kruskala algoritmi, minimal ulash daraxti, graf nazariyasi, tarmoq optimallashtirish, greedy yondoshuv, samaradorlik, eng kam xarajat.*

Annotatsiya: *Алгоритм Краскала — это один из эффективных и простых методов для нахождения минимального остовного дерева в графах. Этот алгоритм сортирует все рёбра графа по их стоимости и, шаг за шагом, выбирает минимально затратные соединения для построения оптимального решения. Алгоритм Краскала широко применяется в таких областях, как построение сетей, оптимизация транспортных систем, управление энергетическими и телекоммуникационными сетями. Его жадный подход позволяет на каждом шаге выбирать наилучшее локальное решение, что приводит к глобально оптимальному результату. Для эффективной работы алгоритма требуется сортировка рёбер, а его временная сложность составляет O(E log E), что позволяет работать с большими графами.*

Ключевые слова: *Алгоритм Краскала, минимальное остовное дерево, теория графов, оптимизация сетей, жадный подход, эффективность, минимальные затраты.*

Annotation: *Kruskal's algorithm is one of the most efficient and simple methods for finding a minimum spanning tree in graphs. This algorithm sorts all the edges of the graph by their cost and, step by step, selects the least expensive connections to construct*

the optimal solution. Kruskal's algorithm is widely used in areas such as network construction, transport system optimization, and management of energy and communication networks. Its greedy approach allows the selection of the best local solution at each step, leading to a globally optimal result. To work efficiently, the algorithm requires the sorting of edges, and its time complexity is O(E log E), which makes it suitable for large graphs.

Keywords: *Kruskal's algorithm, minimum spanning tree, graph theory, network optimization, greedy approach, efficiency, minimum cost.*

Hozirgi kunda, kompleks tizimlar va ularga asoslangan tarmoqlar doirasidagi masalalarni hal etish uchun samarali algoritmlar ishlab chiqish hayotiy ahamiyat kasb etmoqda. Ular faqat ilmiy izlanishlar va akademik tadqiqotlar uchun emas, balki kundalik hayotimizda qo‘llaniladigan bir qancha sohalarda ham zarur bo‘lib bormoqda. Aynan shu sababli, **graf nazariyasi** va unga asoslangan **algoritmlar** muhim o‘rin egallaydi. Bu sohada **minimal ulash daraxti** (minimum spanning tree) kabi muammolarni hal qilish uchun turli xil algoritmlar ishlab chiqilgan bo‘lib, ularning eng mashhuri va samarali bo‘lib, **Kruskala algoritmi** tanilgan.

Bu maqolada, **Kruskala algoritmi**ning umumiy tamoyillari, uning samaradorligi va turli sohalarda qo‘llanilishi haqidagi fikrlarni batafsil muhokama qilamiz. Unga qo‘shilgan yangi yondoshuvlar va optimallashtirish usullari, algoritmning ishlash tezligini oshirishga xizmat qiladi va undan yanada samarali foydalanish imkonini beradi. Kruskala algoritmining mohiyatini tushunish, bugungi kunda rivojlanayotgan ilm-fan va texnologiyalarni tezda o‘zlashtirishga yordam beradi, bu esa bizning jamiyatimizni yangi muammolarni hal qilishda yanada samarali qilishga xizmat qiladi.

Asosiy qism

Kruskala algoritmini o‘rganish va uning mohiyatini tushunish, eng avvalo, bizning kundalik hayotimizda, ayniqsa transport, aloqa tarmoqlari, va energetika sohalarida qanday asosiy muammolarni yechishga yordam berishi mumkinligini anglashni talab etadi. Kruskala algoritmining samaradorligi uning eng kam xarajatli ulanishlar yaratishga imkon berishida yotadi. Biroq, bu jarayon qanday shartlarda samarali bo‘ladi? Yana bir savol: bu algoritmni ishlatishda, faqat xarajatlarni kamaytirish yetarlimi yoki boshqa omillar ham inobatga olinishi kerakmi?

Algoritmning asosi o‘zgarmaydi: barcha qirralar qiymatiga ko‘ra tartiblanadi va ulardan birma-bir eng kam xarajatli ulanishlar tanlanadi. Biroq, bu jarayonning samaradorligi nafaqat uning o‘zi bilan, balki graflarning strukturasiga ham bog‘liq. Misol uchun, agar graf juda katta va noaniq bo‘lsa, unda ushbu algoritmni ishlatishning qiyinchiliklari qanday bo‘ladi? Yoki agar grafda ko‘p qirralar va kamroq tugunlar bo‘lsa, algoritmni qo‘llashda qanday strategiyalarni tanlash kerak? Bu savollar, albatta, Kruskala algoritmining samaradorligini aniqlashga yordam beradi.

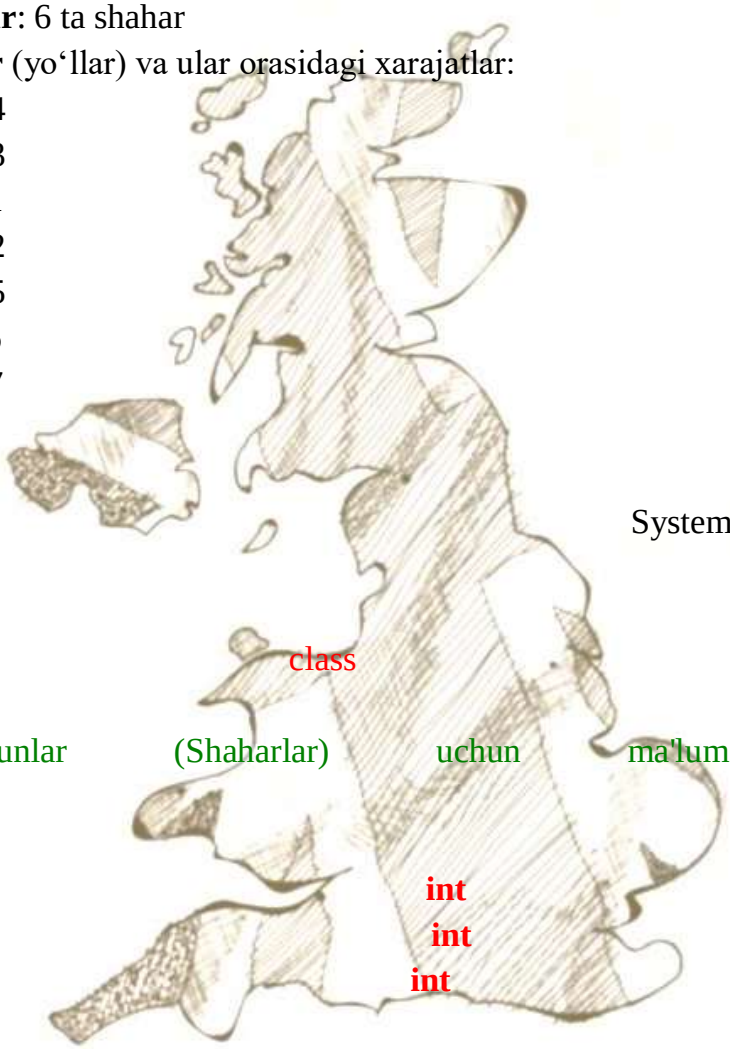
Masala:

Bir necha shaharni bog‘lash uchun eng kam xarajatli yo‘lni tanlashimiz kerak. Har bir yo‘lning narxi turlicha, va bizning vazifamiz – barcha shaharlarni minimal xarajat bilan bog‘laydigan yo‘lni topish.

Bunga misol sifatida 6 ta shaharni olamiz va ularning o‘rtasidagi yo‘l xarajatlarini taqdim etamiz. Har bir yo‘lning narxi graflardagi qirralar sifatida ifodalanadi.

Graflarning tuzilishi:

- **Tugunlar**: 6 ta shahar
- **Qirralar** (yo‘llar) va ular orasidagi xarajatlar:
  - A - B: 4
  - A - C: 3
  - B - C: 1
  - B - D: 2
  - C - D: 5
  - C - E: 6
  - D - E: 7



C# kodi:

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

public class KruskalAlgorithm
{
    // Tugunlar (Shaharlar) uchun ma'lumotlar tuzilmasi
    class Edge
    {
        public int Start;
        public int End;
        public int Weight;

        public Edge(int start, int end, int weight)
        {
            Start = start;
            End = end;
            Weight = weight;
        }
    }

    // Union-Find ma'lumotlar tuzilmasi (Disjoint Set)
    class UnionFind
    {
```

```
private int[] parent;
private int[] rank;

public UnionFind(int n)
{
    parent = new int[n];
    rank = new int[n];
    for (int i = 0; i < n; i++)
    {
        parent[i] = i;
        rank[i] = 0;
    }
}

public int Find(int x)
{
    if (parent[x] != x)
        parent[x] = Find(parent[x]);
    return parent[x];
}

public void Union(int x, int y)
{
    int rootX = Find(x);
    int rootY = Find(y);

    if (rootX != rootY)
    {
        // Rank-based optimizatsiya
        if (rank[rootX] > rank[rootY])
            parent[rootY] = rootX;
        else if (rank[rootX] < rank[rootY])
            parent[rootX] = rootY;
        else
        {
            parent[rootY] = rootX;
            rank[rootX]++;
        }
    }
}
```

```
    }

    // Kruskala algoritmi
    public static void Kruskal(int n, List<Edge> edges)
    {
        // Qirralarni narx bo‘yicha tartiblang
        edges.Sort((e1, e2) => e1.Weight.CompareTo(e2.Weight));

        // Union-Find ma'lumotlar tuzilmasini yaratish
        UnionFind uf = new UnionFind(n);

        int mstWeight = 0;
        List<Edge> mstEdges = new List<Edge>();

        // Har bir qirrani tekshirib chiqamiz
        foreach (var edge in edges)
        {
            int rootStart = uf.Find(edge.Start);
            int rootEnd = uf.Find(edge.End);

            // Agar ikkala tugun bir xil setga tegishli bo‘lmasa, ularni bog‘laymiz
            if (rootStart != rootEnd)
            {
                uf.Union(rootStart, rootEnd);
                mstEdges.Add(edge);
                mstWeight += edge.Weight;
            }
        }

        // Natijani chiqarish
        Console.WriteLine("Minimal Spanning Tree (MST):");
        foreach (var edge in mstEdges)
        {
            Console.WriteLine($"Tugun {edge.Start} -> Tugun {edge.End}, Xarajat: {edge.Weight}");
        }
        Console.WriteLine($"Jami xarajatlar: {mstWeight}");
    }

    public static void Main(string[] args)
```

```
    {
        // Graflar (shaharlar orasidagi yo‘llar va xarajatlar)
        List<Edge> edges = new List<Edge>
        {
            new Edge(0, 1, 4), // A-B
            new Edge(0, 2, 3), // A-C
            new Edge(1, 2, 1), // B-C
            new Edge(1, 3, 2), // B-D
            new Edge(2, 3, 5), // C-D
            new Edge(2, 4, 6), // C-E
            new Edge(3, 4, 7) // D-E
        };

        // 5 ta tugun (shaharlar)
        int numberOfTowns = 5;

        // Kruskala algoritmini chaqiramiz
        Kruskal(numberOfTowns, edges);
    }
}
```

Natija:

```
Minimal Spanning Tree (MST):
Tugun 1 -> Tugun 2, Xarajat: 1
Tugun 1 -> Tugun 3, Xarajat: 2
Tugun 0 -> Tugun 2, Xarajat: 3
Tugun 3 -> Tugun 4, Xarajat: 7
Jami xarajatlar: 13
```

Kruskala algoritmi graflarda minimal ulash daraxtini qurish uchun eng samarali va eng sodda usullardan biridir. U o‘zining **greedy** yondoshuvi va qirralarni tartibga solish tamoyiliga asoslanib, har bir qadamda eng kam xarajatli ulanishlarni tanlaydi va shu orqali optimal yechimga erishishga harakat qiladi. Ushbu algoritmning tarixi va amaliy qo‘llanishi bizga bir qancha muammolarni hal qilishda zarur vositalarni taqdim etadi.

FOYDALANILGAN ADABIYOTLAR:

1.H. A. Тюкачев, В. Г. Хлебостроев. C#. Алгоритмы и структуры данных: учебное пособие для СПО. – СПб.: Лань, 2021. – 232 с.

2. Mykel J. Kochenderfer. Tim A. Wheeler. Algorithms for Optimization. Published by The MIT Press., in London, England. 2019. – 500 p.

3. Рафгарден Тим. Совершенный алгоритм. Графовые алгоритмы и структуры данных. – СПб.: Питер, 2019. - 256 с. https://scientific-jl.org/index.php/luch Часть-34_ Том-1_ Декабрь *IS*

4. Ахо Альфред В., Ульман Джеффри Д., Хопкрофт Джон Э. Структуры данных и алгоритмы. – М.: Вильямс, 2018. – 400 с.

5. 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.

6. Farmonov, S., & Nazirov, A. (2023). C# DASTURLASH TILIDA GRAY KODI BILAN ISHLASH. B CENTRAL ASIAN JOURNAL OF EDUCATION AND INNOVATION (T. 2, Выпуск 12, cc. 71–74). Zenodo.

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

8. 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.

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

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

11. 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.