KRUSKAL ALGORITMI: MINIMAL QOPLOVCHI DARAXT (MST) - TO'LIQ QO'LLANMA

Onarkulov Maksadjon Karimberdiyevich
f.m.f.f.d (PhD) -Amaliy matematika va informatika kafedrasi dotsenti
maxmaqsad@gmail.com
Ne`matova Shohsanam Nodirbek qizi
Farg'ona davlat universiteti talabasi
nematovashohsanam5@gmail.com

Anotatsiya: *Kruskal algoritmi — og'irlikli graf uchun minimal qoplovchi daraxtni (MST) topishga mo'ljallangan mashhur algoritmdir. Ushbu algoritm grafning barcha qirralarini og'irlik bo'yicha saralaydi va har gal eng kichik og'irlikdagi qirrani tanlab, uni sikl hosil qilmasa daraxtga qo'shadi. Bu jarayon barcha cho'qqilar yagona bog'langan graf bo'lib qolguncha davom etadi. Algoritm ko'pincha disjoint-set (union-find) ma'lumot tuzilmasidan foydalanadi. U ko'p hollarda tarmoqlarni optimallashtirish, tarmoq loyihalash va boshqa kombinatorika muammolarini hal qilishda qo'llanadi.*

Kalit so'zlar: *Kruskal algoritmi, Minimal qoplovchi daraxt, Og'irlikli graf, Qirra, Cho'qqi, Cikl, Disjoint-set (union-find), Saralash, Tarmoq optimallashtirish, Kesh invalidatsiyasi, Samaradorlikni oshirish, Tarmoq loyihalash, Algoritmlar, Xotira boshqaruvi.*

Аннотация: *Алгоритм Краскала — это известный алгоритм для нахождения минимального остовного дерева (MST) в взвешенном графе. Алгоритм сортирует все рёбра графа по весу и поочередно выбирает ребро с наименьшим весом, добавляя его в дерево только в том случае, если оно не образует цикла. Этот процесс продолжается до тех пор, пока все вершины не будут объединены в одну связанную компоненту. Алгоритм часто использует структуру данных «система непересекающихся*

множеств» (union-find). Широко применяется для оптимизации сетей, проектирования сетей и решения других комбинаторных задач.

Ключевые слова: *Алгоритм Краскала, Минимальное остовное дерево (MST), Взвешенный граф, Ребро, Вершина, Цикл, Система непересекающихся множеств (union-find), Сортировка, Оптимизация сетей, Инвалидация кэша, Оптимизация производительности, Проектирование сети, Алгоритмы, Управление памятью.*

Annotation: *Kruskal's algorithm is a popular algorithm designed to find the Minimum Spanning Tree (MST) of a weighted graph. The algorithm sorts all the edges in the graph by weight and then iteratively selects the edge with the smallest weight, adding it to the tree only if it does not form a cycle. This process continues until all the vertices are connected in a single connected component. The algorithm often uses a disjoint-set (union-find) data structure. It is widely used in network optimization, network design, and other combinatorial problems.*

Keywords: *Kruskal's algorithm, Minimum Spanning Tree (MST), Weighted graph, Edge, Vertex, Cycle, Disjoint-set (union-find), Sorting, Network optimization, Cache invalidation, Performance optimization, Network design, Algorithms, Memory management.*

Kirish: Graf Algoritmlarining Ahamiyati. Zamonaviy informatika va amaliy dasturlashda graf nazariyasi asosiy o‘rinni egallaydi. Graf algoritmlari - bu nafaqat nazariy matematik tushunchalar, balki haqiqiy dunyo muammolarini hal qilishning kuchli vositasi hisoblanadi. Bugungi kunda graf algoritmlari quyidagi sohalarda keng qo‘llaniladi:

- **Tarmoq dizayni va optimizatsiyasi** (Internet, aloqa tarmoqlari)
- **Transport tizimlari** (yo‘nalishlarni topish, logistika)
- **Ijtimoiy tarmoqlar tahlili** (do‘stlik tarmoqlari, ma'lumotlar oqimi)
- **Kompyuter grafikasi va o‘yin dasturlash** (obyektlararo bog‘lanishlar)

- **Sun'iy intellekt va mashina o‘rganishi** (ma'lumotlar strukturasi)

Graf algoritmlari orasida eng ko‘p qo‘llaniladiganlaridan biri - bu **minimal qoplovchi daraxt (MST)** topishga bag‘ishlangan algoritmlardir. Ushbu algoritmlar orasida Kruskal algoritmi o‘zining soddaligi, tushunarli logikasi va samaradorligi bilan ajralib turadi.

MST ning Matematik Ta'rifi:

Formal ravishda, berilgan G = (V, E) bog‘langan graf uchun minimal qoplovchi daraxt - bu:

1. Grafning barcha V tugunlarini qamrab oluvchi
2. |V|-1 ta qirradan iborat
3. Hech qanday siklga ega bo‘lmagan
4. Barcha mumkin bo‘lgan qoplovchi daraxtlar ichida eng kichik umumiy vaznga ega bo‘lgan daraxt hisoblanadi.

MST ni topish quyidagi amaliy muammolarni hal qilishda asos bo‘la oladi:

1. Shahararo yo‘l tarmog‘ini qurish - barcha shaharlarni eng kam umumiy yo‘l uzunligi bilan bog‘lash
2. Elektr tarmog‘ini loyihalash - barcha uylarni eng kam sim ishlatib ulanma
3. Kompyuter tarmoqlarini optimallashtirish - serverlar orasidagi eng samarali aloqa yo‘llarini tanlash
4. Irrigatsiya tizimini qurish - barcha dalalarni eng kam quvur ishlatib sug‘orish

Kruskal Algoritmi: Tarixi va Rivojlanishi

Kruskal algoritmi 1956-yilda amerikalik matematik **Joseph Kruskal** tomonidan ishlab chiqilgan. Ushbu algoritm:

- **Greedy (oqilona tanlov) algoritmlar** sinfiga kiradi
- **Graph Theory** jurnalida birinchi marta e'lon qilingan
- Dastlab faqat nazariy maqolada taqdim etilgan bo‘lsa, keyinchalik keng amaliy qo‘llanma topgan

- 1970-yillarda kompyuter injineringi rivojlanishi bilan algoritmning samaradorligi oshgan

Algoritmning asosiy qadamlari:

1. Barcha qirralarni vazn bo‘yicha saralash
2. Eng kichik vaznli qirradan boshlab tanlash
3. Tsikl hosil qilmasligini tekshirish
4. Jarayonni MST tugaguncha davom ettirish

Kruskal Algoritmining Afzalliklari

Boshqa MST algoritmlariga (masalan, Prim algoritmi) nisbatan Kruskal algoritmining asosiy afzalliklari:

1. **Kodning soddaligi** - algoritmni tushunish va amalga oshirish oson
2. **Parallelizatsiya qobiliyati** - qirralarni saralash va tekshirish jarayonlarini parallel ravishda bajarish mumkin
3. **Sparse graflarda samaradorlik** - kam qirrali graflarda ayniqsa samarali
4. **Diskda ishlash imkoniyati** - katta graflarni diskda saqlab, qismma-qism ishlash mumkin

Kruskal algoritmini dasturiy tarzda amalga oshirishda quyidagi jihatlar muhim ahamiyat kasb etadi:

1. **Ma'lumotlar strukturasi tanlovi**:
   - Qirralarni saqlash uchun ro‘yxat yoki massiv
   - Union-Find tuzilmasi uchun optimallashtirilgan algoritmlar
2. **Tezlik optimizatsiyasi**:
   - Path compression
   - Union by rank
3. **Xotira boshqaruvi**:
   - Katta graflar bilan ishlashda samarali xotira foydalanish
4. **Kodning o‘qilishi**:
   - Toza va izohlangan kod yozish
   - Modulli yondashuv

Kruskal Algoritmini C# da Implementatsiya Qilish

DSU (Disjoint Set Union) Klassi

```
public class DSU
{
    private int[] parent;
    private int[] rank;

    public DSU(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]); // Path compression
        return parent[x];
    }

    public bool Union(int x, int y)
    {
        int xRoot = Find(x);
        int yRoot = Find(y);
```

```
        if (xRoot == yRoot)
            return false; // Tsikl hosil qiladi

        // Union by rank
        if (rank[xRoot] < rank[yRoot])
            parent[xRoot] = yRoot;
        else if (rank[xRoot] > rank[yRoot])
            parent[yRoot] = xRoot;
        else
        {
            parent[yRoot] = xRoot;
            rank[xRoot]++;
        }
        return true;
    }
}
```

Edge Klassi va Kruskal Algoritmi

```
public class Edge : IComparable<Edge>
{
    public int U { get; set; }
    public int V { get; set; }
    public int Weight { get; set; }
    public Edge(int u, int v, int weight)
    {
        U = u;
        V = v;
        Weight = weight;
    }
    public int CompareTo(Edge other)
```

```
    {
        return Weight.CompareTo(other.Weight);
    }
}
public class KruskalMST
{
    public List<Edge> FindMST(List<Edge> edges, int numNodes)
    {
        edges.Sort(); // Vazn bo'yicha saralash
        DSU dsu = new DSU(numNodes);
        List<Edge> mst = new List<Edge>();

        foreach (var edge in edges)
        {
            if (dsu.Union(edge.U, edge.V))
            {
                mst.Add(edge);
                if (mst.Count == numNodes - 1)
                    break;
            }
        }
        return mst;
    }
}
```

Asosiy Dastur (Main)

```
public class Program
{
    public static void Main()
    {
```

```
    // Grafni yaratish (A=0, B=1, C=2, D=3)
    List<Edge> edges = new List<Edge>
    {
        new Edge(0, 2, 1), // A-C:1
        new Edge(1, 3, 3), // B-D:3
        new Edge(2, 3, 4), // C-D:4
        new Edge(0, 1, 5), // A-B:5
        new Edge(0, 3, 6)  // A-D:6
    };
    KruskalMST kruskal = new KruskalMST();
    var mst = kruskal.FindMST(edges, 4); // 4 ta tugun (A,B,C,D)
    Console.WriteLine("Minimal Qoplovchi Daraxt (MST) Qirralari:");
    foreach (var edge in mst)
        Console.WriteLine($"{edge.U} - {edge.V} : {edge.Weight}");
    int totalWeight = mst.Sum(edge => edge.Weight);
    Console.WriteLine($"Umumiy Vazn: {totalWeight}");
  }
}
```

Natija:Minimal Qoplovchi Daraxt (MST) Qirralari:

```
0 - 2 : 1
1 - 3 : 3
2 - 3 : 4
Umumiy Vazn: 8
```

5. Kruskal Algoritmining Tezlik Tahlili

- Saralash: O(E log E) (qirralarni vazn bo'yicha saralash).
- Union-Find operatsiyalari: Har bir Find va Union O(α(N)) (α - invers Ackermann funksiyasi, deyarli doimiy).
- Umumiy tezlik: O(E log E) yoki O(E log V) (chunki $E \leq V^2$).

Kruskal vs Prim Algoritmlari

Kruskal	Prim
Qirralarni saralaydi.	Tugunlarni qo'shib boradi.
O(E log E) tezlik.	O(V²) yoki O(E + V log V) tezlik.
Sparse graflar uchun afzal.	Dense graflar uchun afzal.
Union-Find tuzilmasini talab qiladi.	Priority Queue dan foydalanadi.

Qaysi Algoritmni Tanlash Kerak?

Kruskal Tanlang Agar:

- Graf katta va tarqoq bo'lsa
- Qirralar soni kam bo'lsa (E ≈ V)
- Parallel hisoblash imkoniyati kerak bo'lsa
- Butun graf bir vaqtning o'zida mavjud bo'lmasa

Prim Tanlang Agar:

- Graf zich bo'lsa (E ≈ V²)
- Faqat bog'langan komponent kerak bo'lsa
- Graf dinamik o'zgarib turuvchi bo'lsa
- Boshlang'ich tugun ma'lum bo'lsa

C# da Implementatsiya Farqlari

Kruskal Implementatsiyasi

// Asosiy komponentlar:

- Edge klass (qirralar)
- DSU (Union-Find) struktura
- Qirralarni saralash

Prim Implementatsiyasi

// Asosiy komponentlar:

- PriorityQueue (Min-Heap)
- Visited tugunlar massivi
- Faqat bog'langan tugunlarni ko'rib chiqish

Xulosa va Tavsiyalar

1. Kruskal - universal yechim, ayniqsa katta va tarqoq graflar uchun

2. Prim - zich graflar va real vaqt rejimidagi amaliyotlar uchun
3. Ikkala algoritm ham O(E log V) ga yaqin tezlikda ishlashi mumkin (to'g'ri implementatsiya bilan)
4. Amaliy loyihalarda - grafnig xususiyatlariga qarab tanlov qiling

Yakuniy qoida: Agar qanday algoritmdan foydalanishni bilmasangiz, Kruskal - ishonchli universal tanlov, Prim esa zich graflarda yaxshiroq samara beradi.

Foydalanilgan adabiyotlar:

1. S. Jain. "Tabiatdan ilhomlangan optimallashtirish algoritmlari" (2022)
2. S. Saha, Sh. Shukla. "Qo'shma ma'lumotlar tuzilmalari" (2020)
3. M. Oshurov, Sh. Sattarova, Sh. Usmonqulov. "Algoritmlar" (2018)
4. N. Karumanchi. "Osongina ma'lumotlar tuzilmalari va algoritmlar" (2017)
5. A. Bhargava. "Algoritmlarni tushunish" (2017)
6. P. Evdokimov. "C# misollarda" (2016)
7. Robinson, Weber, Eifrem. "Graf ma'lumotlar bazalari" (2016)
8. S. Dasgupta, C. Papadimitriou, U. Vazirani. "Algoritmlar" (2014)
9. Niklaus Wirth. "Algoritmlar va ma'lumotlar tuzilmalari" (2010)
10. Marcin Jamro. "C# Data Structures and Algorithms" (2-nashr, 2024)
11. J. Erikson. "Algoritmlar" (2023)
12. Hemant Jain. "Data Structures & Algorithms using Kotlin" (2-nashr, 2022)
13. N.A. Tyukachev, V.G. Khlebostrov. "C#. Algoritmlar va ma'lumotlar tuzilmalari" (2021)
14. Mykel J. Kochenderfer, Tim A. Wheeler. "Algorithms for Optimization" (2019)

15. Tim Roughgarden. "Mukammal algoritm. Graf algoritmlari va ma'lumotlar tuzilmalari" (2019)

16. Aho, Ullman, Hopcroft. "Ma'lumotlar tuzilmalari va algoritmlar" (2018)

17. G. Heineman, G. Pollice, S. Selkow. "Algoritmlar. C, C++, Java va Python misollari" (2017)

Ilm fan taraqqiyotida raqamli iqtisodiyot va zamonaviy ta'limning o'rni hamda rivojlanish omillari