ELEKTR TARMOQLARINI ULASHDA KRUSKAL ALGORITMINING AHAMIYATI

Farmonov Sherzodbek Rahmonjonovich
Farg‘ona davlat universiteti amaliy matematika va informatika kafedrasi katta o‘qituvchisi
farmonovsh@gmail.com
Alijonova Ruxshona Baxtiyorjon qizi
Farg‘ona davlat universiteti talabasi
r66045003@gmail.com

Anotatsiya. *Maqolada Kruskal algoritmining minimal bog‘lovchi daraxtni qurishdagi samaradorligi va elektr tarmoqlarini ulashdagi amaliy qo‘llanilishi va cheklovlari tahlil qilinadi. Graf nazariyasi asosida algoritmning nazariy va dasturiy asoslari qisqacha ko‘rib chiqiladi.*

Kalit so‘zlar. *Kruskal algoritmi, minimal bog‘lovchi daraxt, elektr tarmoqlari, graf nazariyasi, samaradorlikni oshirish va algoritm cheklovlari.*

Annotation. *The article analyzes the efficiency of Kruskal's algorithm in constructing a minimum spanning tree and its practical application in connecting electrical networks, along with its limitations. The theoretical and programming foundations of the algorithm are briefly discussed based on graph theory.*

Keywords. *Kruskal's algorithm, minimum spanning tree, electrical networks, graph theory, improving efficiency and algorithm limitations.*

Аннотация. *В статье анализируется эффективность алгоритма Крускала при построении минимального остовного дерева и его практическое применение для подключения электрических сетей, а также его ограничения. Кратко рассматриваются теоретические и программные основы алгоритма на основе теории графов.*

ISSN 3060-4567

Ключевые слова. *Алгоритм Крускала, минимальное остовное дерево, теория графов электрические сети, повышение эффективности и ограничения алгоритма.*

Kruskal algoritmi - minimum bo'luvchi daraxt (MST) qurish uchun ishlatiladigan samarali algoritm bo'lib, uning asosiy maqsadi grafdagi barcha cho'qqilarni bog'laydigan daraxtni topishdir. Algoritm og'irlikli graf bo'yicha eng kichik umumiy og'irlikka ega bo'lgan qirralarni tanlashni ta'minlaydi, bu esa uni ko'plab amaliy sohalarda qo'llashga imkon yaratadi.

Kruskal algoritmining samarali ishlashi uchun berilganlar strukturasi va algoritmik yondashuvlar muhim rol o'ynaydi. Xususan, Union-Find strukturasidan foydalanish, komponentlar o'rtasidagi bog'lanishlarni tezda aniqlash va yangilash imkoniyatini beradi, bu esa algoritmni samarali va tezkor bajarishga yordam beradi. Dasturlash tili sifatida C# dasturlash tili orqali Kruskal algoritmini amaliyotga tadbiq qilish, nafaqat algoritmning ishlash prinsiplari, balki muayyan berilganlar strukturasi bilan qanday ishlashni ham o'rganish imkonini beradi. Ushbu maqola Kruskal algoritmining algoritmlar va berilganlar strukturasi fanlari nuqtai nazaridan tahlilini taqdim etadi. Shuningdek, algoritmning afzalliklari, kamchiliklari va uni real hayotdagi masalalarda qanday qo'llash mumkinligi haqida batafsil ma'lumot beriladi.

MST (Minimum Spanning Tree) grafdagi barcha cho'qqilarni bog'laydigan, lekin og'irligi minimal bo'lgan daraxtdir. Og'irlikli grafda har bir qirra o'zining og'irligi bilan taqdim etiladi. Kruskal algoritmi ushbu og'irliklarga qarab eng arzon yo'llarni tanlash orqali MSTni quradi. Agar grafda siklsiz daraxt mavjud bo'lsa, bu daraxtning barcha cho'qqilari bir-biriga bog'langan va hech qanday siklni o'z ichiga olmaydi.

Kruskal algoritmining murakkabligi asosan qirralarni saralash jarayoniga bog'liq. Qirralarni saralashning vaqt murakkabligi O(E log E), bu yerda E - grafdagi qirralar soni. Union-Find jarayonining vaqt murakkabligi O(V + E)

bo‘lib, bunda V – cho’qqilar soni. Shuning uchun umumiy murakkablik O(E log E) deb hisoblanadi.

Graf nazariyasida siklsiz daraxt yoki spanning tree grafning barcha cho’qqilarini bog’lashni, lekin sikl hosil qilmaslikni ta’minlaydi. Kruskal algoritmi aynan shunday daraxtni topish uchun ishlatiladi. Siklsiz daraxt har doim bog’langan bo’ladi, ammo undan hech qanday doimiy qaytadigan yo’l bo’lmaydi, bu esa tarmoqda ma’lumotlar uzatishni samarali qiladi.

Qirralarni saralash jarayoni Kruskal algoritmining asosiy qismidir. Bu qadamda, og’irlikli grafdagi barcha qirralar og’irliklar bo’yicha saralanadi. Dastlabki saralashdan so’ng, algoritm eng kichik og’irlikdagi qirralarni tanlab, daraxtni quradi. Bu jarayon samarali ishlashi uchun dasturiy yechimlar ko’p hollarda yordamchi ma’lumotlar tuzilmalarini masalan, heaps yoki priority queues talab qiladi.

Muhandislikda qo’llanilishi Kruskal algoritmining keng tarqalgan sohalaridan biridir. Bu algoritm tarmoq qurilishi, logistika, kompyuter tarmoqlari, telekommunikatsiya va boshqa ko‘plab muhandislik sohalarida qo’llaniladi.

- Tarmoq qurilishi: Telefon tarmoqlari yoki internet tarmoqlarini qurishda, eng optimal yo’llarni tanlash uchun Kruskal algoritmi ishlatiladi. Bu yerda MSTni topish, tarmoqdagi barcha nuqtalarni eng kam narxda bog’lashni ta’minlaydi.
- Elektr energiyasi taqsimoti: Elektr energiyasi tizimlarini qurishda ham MSTni topish uchun Kruskal algoritmi ishlatiladi. Bu energiya taqsimotini optimallashtirish va samarali ta’minot uchun zarurdir.
- Transport tizimlari: Yangi yo’llar yoki transport tarmoqlari qurishda eng arzon va samarali yo’llarni aniqlash uchun Kruskal algoritmi qo’llaniladi.

Kruskal algoritmi minimal bog‘lovchi daraxtni (MST) qurishda samarali va oddiy usul hisoblansada , uning ham ba'zi cheklovlari mavjud. Quyida Kruskal algoritmining asosiy cheklovlari keltirilgan:

- Yo‘nalishli grafiklar bilan ishlamaydi: Kruskal algoritmi yo‘nalishsiz grafiklar bilan ishlashga mo‘ljallangan. Agar grafda qirralar yo‘nalgan bo‘lsa

(ya'ni, har bir qirra boshlang‘ich va tugash tuguniga ega bo‘lsa), Kruskal algoritmi ishlamaydi.

- Manfiy og‘irlikli qirralar bilan ishlamaydi: Kruskal algoritmi faqat ijobiy og‘irlikli qirralar bilan ishlaydi. Agar grafikda manfiy og‘irliklar mavjud bo‘lsa, algoritm noto‘g‘ri natijalar berishi mumkin. Bu xususiyat, masalan, krizisli xarajatlar yoki sarmoyalarni ko‘rib chiqish uchun kerak bo‘lganda muammo bo‘lishi mumkin.
- Katta va zich grafiklarda samaradorlik pasayishi: Kruskal algoritmi qirralarni saralashga asoslangan bo‘lib, bu jarayon katta va zich grafiklarda samaradorlikni pasaytirishi mumkin. Grafikdagi qirralar soni juda ko‘p bo‘lsa, qirralarni saralash jarayoni vaqt jihatdan katta yuk yaratadi. Bu jarayonning samaradorligi bo‘lsa ham, juda katta grafiklarda bu vaqt jiddiy pasayishiga olib kelishi mumkin.
- Barcha qirralarni ko‘rib chiqish kerak: Kruskal algoritmi barcha qirralarni ko‘rib chiqadi, shuning uchun qirralar soni juda ko‘p bo‘lsa, algoritm juda sekin ishlashi mumkin. Bu holatda algoritmni ishlatish samarali bo‘lmasligi mumkin, chunki u ko‘p va noaniq qirralarni tahlil qilishga vaqt sarflaydi.

Algoritmning ishlash jarayoni. Kruskal algoritmi graf nazariyasida minimal bog‘lovchi daraxt (MST)ni topish uchun ishlatiladi. Bu algoritmning maqsadi — grafdagi barcha tugunlarni bog‘laydigan minimal xarajatga ega daraxtni qurishdir. Kruskal algoritmi, asosan, yo‘nalishsiz grafiklar uchun mo‘ljallangan bo‘lib, har bir qirra (yo‘l)ning og‘irliklari (xarajatlari) belgilangan bo‘ladi. Algoritmning ishlash jarayoni qadam-baqadam quyidagicha amalga oshiriladi:

1. **Qirralarni saralash:** Dastlab, barcha qirralar og‘irliklariga qarab o‘sish tartibida saralanadi. Bu bosqichda qirralarning eng kichik og‘irlikli qirra birinchi tanlanadi.

2. **Union-Find tuzilmasini tashkil etish:** Union-Find ma'lumot tuzilmasi yordamida har bir tugun o‘zining alohida to‘plamiga joylashtiriladi. Bu tuzilma yordamida tugunlarni birlashtirish va tsiklni oldini olish uchun kerakli

operatsiyalarni bajarish mumkin. Union-Find ma'lumot tuzilmasi yordamida har bir tugun o‘zining alohida to‘plamiga joylashtiriladi. Bu tuzilma yordamida tugunlarni birlashtirish va tsiklni oldini olish uchun kerakli operatsiyalarni bajarish mumkin.

3. **Eng kichik qirra tanlash:** Saralangan qirralardan eng kichik og‘irlikka ega bo‘lgan qirra tanlanadi. Agar tanlangan qirra sikl hosil qilmasa (ya'ni, qirra ikki tugunni birlashtirishda ularni birlashtirish hali amalga oshmagan bo‘lsa), bu qirra minimal bog‘lovchi daraxtga qo‘shiladi.

4. **Union operatsiyasini bajarish:** Tanlangan qirra qo‘shilgandan so‘ng, Union operatsiyasi orqali tegishli tugunlar birlashtiriladi. Bu jarayon orqali grafdagi tugunlar birlashtiriladi va siklning oldini olish uchun yangi qirralar qo‘shilishi mumkin.

5. **Takrorlash jarayoni**: Bu jarayon har bir qirra uchun takrorlanadi. Qirralar saralangan holda eng kichik og‘irlikli qirra tanlanib, minimal bog‘lovchi daraxt qurilishini davom ettiradi.

6. **Barcha tugunlar bog‘langach, to‘xtash:** Barcha tugunlar birlashtirilgan va minimal bog‘lovchi daraxt qurilganidan so‘ng, algoritm to‘xtaydi. Agar barcha tugunlar bog‘langan bo‘lsa, minimal bog‘lovchi daraxt olingan bo‘ladi.

Matematik Tavsif. Kruskal algoritmi graf nazariyasiga asoslanadi. Algoritmdagi asosiy tushunchalar quyidagilardir:

- **Graf(G):** Tugunlar va qirralardan tashkil topgan struktura bo’lib, G = (V, E) orqali belgilanadi. Bu yerda V- tugunlar to‘plami, ya'ni grafdagi barcha nuqtalar , E-qirralar to‘plami, ya'ni tugunlar orasidagi bog‘lanishlar yoki yo‘llar.
- **Tugun (Node):** Grafdagi har bir nuqta yoki cho‘qqi tugun deb ataladi. Har bir tugun o‘z o‘rniga ega va boshqa tugunlar bilan bog‘langan.
- **Qirra (Edge):** bu ikki tugunni bog‘lovchi yo‘l. Har bir qirra o‘ziga xos og‘irlikga ega bo‘lishi mumkin, bu esa o‘sha yo‘lning masofasini yoki xarajatini ifodalaydi.

- **Og'irlik (Weight):** bu har bir qirraning qiymati bo‘lib, masofani yoki xarajatni ifodalaydi. Masalan, A va B tugunlari orasidagi qirra og‘irlik (xarajat) sifatida 4 birlikni olishi mumkin, bu esa o‘sha qirra bo‘ylab o‘tish uchun talab qilinadigan xarajatni anglatadi.

Masala:

1. **Tumanlar:** Bagdad, Oltiariq, Rishton, Beshariq, Quva, Furqat, Marhamat, Kosonsoy, Kitob.

2. **Tumanlar orasidagi masofalar(km):**

- Bagdad → Oltiariq: 310 km
- Bagdad → Furqat: 320 km
- Bagdad → Rishton: 370 km
- Oltiariq → Beshariq: 270 km
- Oltiariq → Furqat: 500 km
- Oltiariq → Marhamat: 800 km
- Rishton → Quva: 450 km
- Rishton → Kitob: 600 km
- Beshariq → Quva: 200 km
- Quva → Marhamat: 150 km
- Furqat → Kosonsoy: 70 km
- Marhamat → Kosonsoy: 60 km
- Kitob → Kosonsoy: 1200 km

Maqsad: Bu tumanlar orasidagi elektr tarmoqlar uchun minimal xarajat bilan ulanishni ta'minlaydigan minimal bog‘lovchi daraxtni qurish.

```
using System;
using System.Collections.Generic;
namespace ConsoleApp17
{  class KruskalAlgorithm
{ public class UnionFind // Union-Find ma'lumot tuzilmasi
{ private int[] parent, rank;
```

```
public UnionFind(int n)
{
parent = new int[n];
rank = new int[n];
for (int i = 0; i < n; i++)
parent[i] = i; }
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)
{if (rank[rootX] > rank[rootY])
parent[rootY] = rootX;
else if (rank[rootX] < rank[rootY])
parent[rootX] = rootY;
else{parent[rootY] = rootX; rank[rootX]++;}
}public static void Kruskal(int n, List<Edge> edges)
{// Qirralarni og'irliklarga qarab saralash edges.Sort((a, b) =>
a.Weight.CompareTo(b.Weight));
UnionFind uf = new UnionFind(n);
List<Edge> mst = new List<Edge>();
foreach (var edge in edges)
{int rootX = uf.Find(edge.Start);
int rootY = uf.Find(edge.End);
if (rootX != rootY) {mst.Add(edge); uf.Union(rootX, rootY);}
   }
```

```
        // Natijani chiqarish
        Console.WriteLine("Minimal bog'lovchi daraxt:");
        int totalWeight = 0;
        foreach (var edge in mst){Console.WriteLine($"{edge.Start} → {edge.End}: {edge.Weight} km");totalWeight += edge.Weight;}
Console.WriteLine($"Jami xarajatlar: {totalWeight} km"); }
public class Edge //  Qirra tuzilmasi
        {public int Start { get; }
        public int End { get; }
        public int Weight { get; }
        public Edge(int start, int end, int weight)
        {Start = start;End = end;Weight = weight;}
          }internal class Program{static void Main(string[] args)
           {   // Tumanlar soni (Bagdad, Oltiariq, Rishton, Beshariq, Quva, Furqat, Marhamat, Kosonsoy, Kitob -> 0, 1, 2, 3, 4, 5, 6, 7, 8)int n = 9; // Qirralar (yo‘llar) va ularning og‘irliklariList<Edge> edges = new List<Edge>
            {new Edge(0, 1, 310), // Bagdad → Oltiariq
            new Edge(0, 5, 320), // Bagdad → Furqat
            new Edge(0, 2, 370), // Bagdad → Rishton
            new Edge(1, 3, 270), // Oltiariq → Beshariq
            new Edge(1, 5, 500), // Oltiariq → Furqat
            new Edge(1, 6, 800), // Oltiariq → Marhamat
            new Edge(2, 4, 450), // Rishton → Quva
            new Edge(2, 8, 600), // Rishton → Kitob
            new Edge(3, 4, 200), // Beshariq → Quva
            new Edge(4, 6, 150), // Quva → Marhamat
            new Edge(5, 7, 70),  // Furqat → Kosonsoy
            new Edge(7, 6, 60),  // Marhamat → Kosonsoy
            new Edge(8, 6, 1200), // Kitob → Kosonsoy};
            // Kruskal algoritmini chaqirish
```

```
Kruskal(n, edges);      }}
```

Natija:

7 → 6: 60 km

5 → 7: 70 km

4 → 6: 150 km

3 → 4: 200 km

1 → 3: 270 km

0 → 1: 310 km

0 → 2: 370 km

2 → 8: 600 km

Jami xarajatlar: 2030 km

Kruskal algoritmi minimal bog‘lovchi daraxtni qurish uchun ishlatiladigan eng samarali usullardan biridir. U grafdagi barcha tugunlarni bog‘laydigan eng kam xarajatli tarmoqni yaratadi, bu esa resurslarni optimallashtirish va xarajatlarni kamaytirish uchun juda muhimdir. Algoritmning ishlash prinsipi oddiy va aniq bo‘lib, uni katta graflar bilan ishlashda ham qo‘llash mumkin, ammo graf qirralari soni ko‘p bo‘lsa, samaradorligi pasayishi mumkin. Kruskal algoritmi, shuningdek, Union-Find ma'lumot tuzilmasi yordamida sikl hosil bo‘lishining oldini oladi va bu tarmoqda barcha tugunlarni xavfsiz va samarali ravishda bog‘lash imkonini beradi. Bu algoritm transport tarmoqlari, internet provayderlari va boshqa infratuzilmalarda keng qo‘llaniladi. Misol uchun, elektr tarmoqlarini qurishda yoki telekommunikatsiya tizimlarini loyihalashda, Kruskal algoritmi orqali eng kam xarajatli ulanishlar tanlanadi va tarmoq samarali tarzda tashkil etiladi. Shu bilan birga, Kruskal algoritmining ishlash tezligi va samaradorligi grafning strukturasiga, ya'ni tugunlar va qirralarning soniga bog‘liq bo‘lib, keng miqyosli tizimlar uchun turli optimizatsiya usullari kerak bo‘lishi mumkin. Biroq, uning oddiyligi va ishlash samaradorligi uni ko‘plab sohalarda qo‘llaniladigan samarali vositaga aylantiradi.

FOYDALANILGAN ADABIYOTLAR RO'YHATI

1. Cormen, T. H., Leiserson, C. E., Rivest, R. L., & Stein, C. (2009). Introduction to Algorithms (3rd ed.). The MIT Press.

2. Li, H., Xia, Q., & Wang, Y. (2017). Research and improvement of kruskal algorithm. Journal of Computer and Communications, 5(12), 63.

3. Guttoski, P. B., Sunye, M. S., & Silva, F. (2007, September). Kruskal's algorithm for query tree optimization. In 11th international database engineering and applications symposium (IDEAS 2007) (pp. 296-302). IEEE..

4. Melnikov, B. F., & Terentyeva, Y. Y. (2021, February). Building communication networks: on the application of the Kruskal's algorithm in the problems of large dimensions. In IOP Conference Series: Materials Science and Engineering (Vol. 1047, No. 1, p. 012089). IOP Publishing..

5. Broutin, N., Devroye, L., & McLeish, E. (2010). Note on the Structure of Kruskal's Algorithm. Algorithmica, 56, 141-159.

6. 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.

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

8. GeeksforGeeks. (n.d.). Kruskal's Algorithm for Minimum Spanning Tree (MST). Retrieved from https://www.geeksforgeeks.org/kruskals-algorithm-for-minimum-spanning-tree/

9. Stack Overflow. (n.d.). Kruskal's Algorithm Implementation in C#. Retrieved from https://stackoverflow.com/questions/21910216/kruskals-algorithm-implementation-in-c-sharp