ENG QISQA MARSHRUTLARNI OPTIMALLASHTIRISH

Farmonov Sherzodbek Raxmonjonovich
Farg‘ona davlat universiteti amaliy matematika
va informatika kafedrasi katta o’qituvchisi
farmonovsh@gmail.com
Imomova Marjona Jahongir qizi
Farg‘ona davlat universiteti talabasi
imomovamarjona2002@gmail.com
https://doi.org/10.5281/zenodo.14498667

Annotatsiya: “Eng qisqa marshrutlarni optimallashtirish: Jonson algoritmi” Ushbu maqolada Jonson algoritmi va uning graflarda eng qisqa yo‘llarni topishdagi o‘rni ko‘rib chiqiladi. Ayniqsa, algoritmning siyrak graflar bilan samarali ishlashi va transport yo‘llarini optimallashtirish, tarmoqlarni tahlil qilish hamda resurslarni rejalashtirish kabi real hayotiy masalalardagi qo‘llanilishi tahlil qilinadi. Maqolada Jonson algoritmining ishlash bosqichlari, uning murakkablik darajasi va boshqa usullarga nisbatan afzalliklari batafsil yoritilgan. Ushbu algoritmning zamonaviy texnologiyalarda tutgan o‘rni va imkoniyatlari keng tahlil qilinadi.

Kalit so‘zlar: Jonson algoritmi, eng qisqa yo‘l, siyrak graflar, graflar tahlili, transport tizimlari, optimallashtirish, tarmoqlar tahlili, marshrutlarni rejalashtirish.

Аннотация: “Оптимизация кратчайших маршрутов: Алгоритм Джонсона”

В данной статье рассматривается алгоритм Джонсона, который является одним из ключевых инструментов для поиска кратчайших путей в графах. Особое внимание уделяется его способности эффективно работать с разреженными графами и его применению в реальных задачах, таких как оптимизация транспортных маршрутов, анализ сетей и планирование ресурсов. Также обсуждаются основные этапы работы алгоритма, его сложность и преимущества по сравнению с другими методами. Статья предлагает подробный анализ возможностей алгоритма Джонсона и его роли в современных технологиях.

Ключевые слова: алгоритм Джонсона, кратчайший путь, разреженные графы, графы, транспортные системы, оптимизация, анализ сетей, планирование маршрутов.

Annotatsiya: ”Optimizing Shortest Paths: Johnson’s Algorithm” This article explores Johnson's algorithm, a vital tool for finding shortest paths in graphs. Special emphasis is placed on its efficiency in handling sparse graphs and its applications in real-world problems such as optimizing transportation routes, network analysis, and resource planning. The article discusses the algorithm's key steps, complexity, and advantages over other methods. It provides a detailed analysis of Johnson's algorithm's capabilities and its importance in modern technologies.

Keywords: Johnson's algorithm, shortest path, sparse graphs, graph analysis, transportation systems, optimization, network analysis, route planning.

Zamonaviy dunyo murakkab tizimlar va ulkan ma'lumotlar bilan ishlashga asoslangan. Transport tarmoqlaridan tortib, sun’iy intellektga asoslangan algoritmlargacha bo‘lgan turli sohalarda samarali va tezkor yo‘l topish masalasi dolzarb bo‘lib qolmoqda. Ayniqsa, graflar orqali modellashtirilgan tizimlarda qisqa marshrutlarni aniqlash, resurslardan samarali foydalanish va vaqtni tejash muhim vazifalar sirasiga kiradi. Shu nuqtai nazardan, **Jonson algoritmi** graflar nazariyasining muhim yutuqlaridan biri sifatida ajralib turadi.

Mashhur informatika olimi Donald Knutning so‘zlariga ko‘ra, “algoritmlar nafaqat muammolarni hal qilish vositasi, balki murakkab jarayonlarni boshqarish va optimallashtirishning intellektual asbobidir.” Jonson algoritmi ushbu tamoyilga mos ravishda yaratilgan bo‘lib, u graflardagi qisqa yo‘lni topishda samarali va universal yondashuvni taklif etadi. Uning asosiy afzalligi — graflardagi musbat va manfiy og‘irliklar bilan ishlash qobiliyati hamda murakkab tizimlarni tahlil qilishga moslashuvchanligi.

Bugungi kunda Jonson algoritmi nafaqat nazariy ahamiyatga ega, balki real hayotiy sohalarda ham keng qo‘llanilmoqda. Transport tarmoqlarini optimallashtirish, logistika tizimlarini rejalashtirish, kompyuter tarmoqlarida samarali marshrutlash va hatto biologik tahlillarni amalga oshirishda ushbu algoritmning imkoniyatlari beqiyosdir. Ushbu algoritmning murakkab graflar bilan ishlashda aniqlik va tezlikni ta'minlay olishi uni boshqa algoritmlardan ajratib turadi.

Ushbu maqolada Jonson algoritmining nazariy asoslari, ishlash tamoyillari va uning turli sohalardagi qo‘llanilish imkoniyatlari haqida batafsil so‘z yuritiladi. Ayniqsa, transport tarmoqlari va tarmoq tizimlaridagi ahamiyati chuqur tahlil qilinadi. Jonson algoritmi nafaqat qisqa marshrutlarni topish vositasi, balki texnologiyalarning rivojlanishida muhim ahamiyatga ega bo‘lgan algoritmik tamoyillarning yorqin namunasi hisoblanadi. Shunday ekan, ushbu

algoritmni chuqurroq o‘rganish va amaliy qo‘llanilishi haqida bilimga ega bo‘lish kelajak texnologiyalariga yo‘l ochadi.

Asosiy qism: Jonson algoritmi graflarda eng qisqa marshrutlarni topish uchun ishlatiladigan eng samarali usullardan biridir. Ushbu algoritm Bellman-Ford va Dijkstra algoritmlarining kombinatsiyasiga asoslanib, musbat va manfiy og‘irlikli qirralarni birgalikda tahlil qila oladi. Lekin o‘quvchi savol berishi mumkin: agar boshqa algoritmlar mavjud bo‘lsa, nima uchun Jonson algoritmi tanlanadi? Javob shuki, bu algoritm bir vaqtning o‘zida ko‘plab marshrutlarni aniqlashga qodir va bu jarayonda murakkablikni minimallashtiradi.

Jonson algoritmining ishlash jarayoni bir nechta bosqichlarni o‘z ichiga oladi. Avval graflarning barcha qirralarini musbat og‘irliklarga moslashtirish uchun Bellman-Ford algoritmi qo‘llaniladi. So‘ngra Dijkstra algoritmi yordamida har bir tugundan boshqa barcha tugunlarga eng qisqa yo‘l topiladi. Bu ikki yondashuvning uyg‘unligi Jonson algoritmini noyob qiladi. Ammo savol tug‘iladi: bu jarayon qachon samarali va qachon qiyinchiliklarga duch kelishi mumkin?

Samaradorlik asosan graflarning tuzilishiga bog‘liq. Jonson algoritmi siyrak graflar bilan ishlaganda juda samarali, chunki uning vaqt murakkabligi O(V2log⁡V+VE)O(V^2 \log V + VE)O(V2logV+VE) ga teng. Ammo graf zich bo‘lsa, ushbu algoritmning samaradorligi pasayishi mumkin. Shunday ekan, Jonson algoritmini qo‘llashdan oldin grafning xususiyatlarini tahlil qilish zarur.

Internet va kompyuter tarmoqlarining tezkor ishlashi ma’lumotlarni samarali uzatish va tahlil qilishga bog‘liq. Tarmoqlarda har bir tugun serverni, qirralar esa ular o‘rtasidagi aloqa yo‘llarini ifodalaydi. Ma'lumotlar oqimini optimallashtirishda Jonson algoritmi eng qisqa marshrutlarni aniqlash uchun ishlatiladi. Masalan, katta ma’lumotlar markazlarida tarmoq yukini teng taqsimlash uchun ushbu algoritm qo‘llanilishi mumkin.

Shu nuqtada savol tug‘iladi: katta hajmdagi dinamik ma’lumotlar bilan ishlaydigan tarmoqlarda Jonson algoritmi qanchalik samarali? Tarmoqlardagi real vaqt rejimidagi ma'lumotlar oqimida ushbu algoritm statik qirralar va tugunlarga asoslanishi sababli, u doim ham ideal bo‘lmasligi mumkin. Lekin tarmoqning umumiy konfiguratsiyasi ma’lum bo‘lsa, Jonson algoritmi tez va samarali ishlash imkonini beradi.

Graflar nazariyasi faqat texnologik tizimlarda emas, balki biologik tahlillarda ham qo‘llaniladi. Masalan, genetik tadqiqotlarda yoki nerv tizimlarining tahlilida graflar yordamida muloqot tarmoqlari modellashtiriladi. Jonson algoritmi yordamida ushbu tizimlarda qisqa va samarali marshrutlarni aniqlashga erishiladi. Masalan, genlar o‘rtasidagi o‘zaro bog‘liqlikni tahlil

qilishda yoki biologik jarayonlarni optimallashtirishda ushbu algoritm amaliy ahamiyatga ega.

Statistik tahlil sohasida esa Jonson algoritmi katta hajmdagi ma'lumotlar to‘plamidan kerakli xulosalarni chiqarishga yordam beradi. Masalan, iqtisodiy tahlillarda yoki bozor dinamikasini o‘rganishda graflar orqali bog‘liqlikni aniqlash va optimallashtirish imkonini beradi.

Masala: Sizga tugunlar va qirralar to‘plamidan tashkil topgan yo‘nalgan og‘irlikli graf G(V,E)G(V, E)G(V,E) berilgan. Har bir qirra og‘irligi musbat yoki manfiy bo‘lishi mumkin (lekin manfiy og‘irlikli sikllar yo‘q). Graflarda har bir tugun uuu-dan boshqa barcha tugun vvv-larga bo‘lgan eng qisqa yo‘llarni topish talab etiladi.

C# da kod

Quyida masala uchun Jonson algoritmi to‘liq C# kod ko‘rinishida berilgan:

```
using System;
using System.Collections.Generic;
using System.Linq;
class Graph
{
 public int Vertices { get; }
  public List<(int, int, int)> Edges { get; }
  public Graph(int vertices)
{
Vertices = vertices;
Edges = new List<(int, int, int)>();
}
public void AddEdge(int u, int v, int w)
{
 Edges.Add((u, v, w));
 }
private int[] BellmanFord(int src)
 {
 int[] distance = Enumerable.Repeat(int.MaxValue, Vertices + 1).ToArray();
 distance[src] = 0;
for (int i = 0; i < Vertices; i++)
{
foreach (var (u, v, w) in Edges)
```

```
        {
 if (distance[u] != int.MaxValue && distance[u] + w < distance[v])
        {
distance[v] = distance[u] + w;

            }
          }
        }

        foreach (var (u, v, w) in Edges)

        {
          if (distance[u] != int.MaxValue && distance[u] + w < distance[v])

          {
            throw new InvalidOperationException("Graph contains a negative
weight
        cycle.");
          }
        }
        return distance;
        }
    private int[] Dijkstra(int src, List<(int, int)>[] modifiedGraph)
        {
        var distance = Enumerable.Repeat(int.MaxValue, Vertices).ToArray();
         var priorityQueue = new SortedSet<(int distance, int vertex)>();
         distance[src] = 0;
         priorityQueue.Add((0, src));
         while (priorityQueue.Any())
         {
           var (dist, u) = priorityQueue.Min;
          priorityQueue.Remove(priorityQueue.Min);

           foreach (var (v, w) in modifiedGraph[u])

           {
             if (distance[u] + w < distance[v])
```

```
            {
                priorityQueue.Remove((distance[v], v));

                distance[v] = distance[u] + w;

                priorityQueue.Add((distance[v], v));

            }
        }
    }

    return distance;

}

public Dictionary<int, int[]> Johnson()

{
    // Add a new helper vertex connected to all others with weight 0

    for (int i = 0; i < Vertices; i++)

    {
        AddEdge(Vertices, i, 0);

    }

    // Step 1: Run Bellman-Ford to compute h values

    int[] h = BellmanFord(Vertices);


    // Step 2: Reweight edges

    var modifiedGraph = new List<(int, int)>[Vertices];
```

```
for (int i = 0; i < Vertices; i++)

{
   modifiedGraph[i] = new List<(int, int)>();

}

foreach (var (u, v, w) in Edges)

{
   if (u != Vertices)

   {
      int newWeight = w + h[u] - h[v];

      modifiedGraph[u].Add((v, newWeight));

   }
}

// Remove the extra helper vertex

Edges.RemoveAll(e => e.Item1 == Vertices);


// Step 3: Run Dijkstra from each node

var allPairsShortestPaths = new Dictionary<int, int[]>();

for (int u = 0; u < Vertices; u++)

{
   allPairsShortestPaths[u] = Dijkstra(u, modifiedGraph);

}
return allPairsShortestPaths;
```

```
    }
}

class Program

{
    static void Main()

    {
        Graph g = new Graph(5);

        g.AddEdge(0, 1, -1);
            g.AddEdge(0, 2, 4);

        g.AddEdge(1, 2, 3);

        g.AddEdge(1, 3, 2);

        g.AddEdge(1, 4, 2);

        g.AddEdge(3, 2, 5);

        g.AddEdge(3, 1, 1);

        g.AddEdge(4, 3, -3);

        try
        {
            var result = g.Johnson();
            foreach (var src in result.Keys)

            {
                Console.WriteLine($"Shortest paths from node {src}:");

                for (int i = 0; i < result[src].Length; i++)
```

```
            {
                Console.WriteLine($"  To {i}: {result[src][i]}");

            }
        }
    }
    catch (Exception ex)
    {
        Console.WriteLine(ex.Message);
    }
  }
}
```

Har bir tugundan boshqa barcha tugunlarga bo‘lgan eng qisqa yo‘llar:
Shortest paths from node 0:
To 0: 0
To 1: -1
To 2: 2
To 3: -2
To 4: 1
Shortest paths from node 1:
To 0: inf
To 1: 0
To 2: 3
To 3: -1
To 4: 2

Jonson algoritmi graflardagi eng qisqa yo‘llarni topishda innovatsion yondashuvni taklif qiluvchi universal vosita sifatida o‘z o‘rnini mustahkam egallagan. U murakkab tizimlarni boshqarishda yuqori samaradorlikni ta’minlaydi va nazariy hamda amaliy masalalarga moslashuvchan yechimlar taklif etadi. Ushbu algoritmning o‘ziga xosligi nafaqat musbat, balki manfiy og‘irlikli graflarda ham ishlay olish qobiliyatida namoyon bo‘ladi. Bu esa uni boshqa qidiruv algoritmlaridan ajratib turadi.

Transport va logistika sohalarida Jonson algoritmi nafaqat eng qisqa marshrutlarni topadi, balki operatsion xarajatlarni optimallashtirishga xizmat qiladi. Internet tarmoqlaridagi ma’lumotlarni uzatishda yoki katta ma’lumotlar bazalarida tezkor qidiruvni amalga oshirishda uning imkoniyatlari keng va

xilma-xildir. Biologik tadqiqotlar va statistik tahlillarda Jonson algoritmi murakkab bog‘lanishlarni tahlil qilish va tizimlar samaradorligini oshirishga yordam beradi.

Biroq, ushbu algoritm ham har doim ham ideal emas. Uning samaradorligi graflarning zichligi, og‘irliklarning taqsimoti va tizimning dinamik xususiyatlariga bog‘liq. Shuning uchun ushbu algoritmdan foydalanishdan avval tizim talablari va cheklovlari chuqur o‘rganilishi zarur

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, S., & Nazirov, A. (2023). C# DASTURLASH TILIDA GRAY KODI BILAN ISHLASH. B CENTRAL ASIAN JOURNAL OF EDUCATION AND INNOVATION (Т. 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 from http://web-journal.ru/index.php/ilmiy/article/view/2480.
21. 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.