YO'L VA MASOFANI OPTIMALLASHTIRISHDA JONSON ALGORITIMINI QO'LLANILISHI

Farmonov Sherzodbek Raxmonjonovich
Farg'ona davlat unversiteti amaliy matematika va informatika kafedrasi katta o'qituvchisi
farmonovsh@gmail.com
Kamolova Rayhona Foziljon qizi
Farg'ona davlat unversiteti 2-kurs talabasi
Kamolovarayhona14@gmail.com

Annotatsiya. Ushbu maqolada transport, logistika va marshrutni belgilash kabi sohalarda yo'llarni optimallashtirish(qisqartirish) muhim hisoblanadi. Ayniqsa, yo'llar orasidagi eng qisqa masofani yoki optimal(qisqa va tez) marshrutni topish kerak bo'lganda, grafik algoritmlaridan foydalaniladi. Shunday algoritmlardan biri - Jonson algoritmi bo'lib, u katta grafiklarda barcha juftliklar orasidagi eng qisqa masofani topish uchun mo'ljallangan.

Kalit so'zlar. Jonson algoritmi, Bellman-Ford algoritmi, Dijkstra algoritimi og'irlikli Graf, skil.

Annotation: This article emphasizes the importance of optimizing (shortening) routes in fields such as transportation, logistics, and route planning. Especially when it is necessary to determine the shortest distance or the optimal (short and fast) route between paths, graph algorithms are used. One such algorithm is Johnson's algorithm, which is designed to find the shortest paths between all pairs in large graphs.

Аннотация.В данной статье подчеркивается важность оптимизации (сокращения) маршрутов в таких областях, как транспорт, логистика и планирование маршрутов. Особенно это актуально, когда требуется найти кратчайшее расстояние или оптимальный (краткий и быстрый) маршрут между пунктами. Для решения таких задач используются графовые алгоритмы. Одним из таких алгоритмов является алгоритм Джонсона, который предназначен для поиска кратчайших расстояний между всеми парами вершин в больших графах.

Ключевые слова. Алгоритм Джонсона, алгоритм Беллмана-Форда, алгоритм Дейкстры, взвешенный граф, цикл.

Jonson algoritmi. Barcha juftlik orasidagi eng qisqa yo'llarni topish uchun foydalaniladi. Bu algoritm asosan yo'nalishi musbat yoki manfiy bo'lgan yo'nalgan grafiklar uchun mo'ljallangan. Ammo Bellman-Ford algoritmiga asoslangan bo'lganligi sababli, Jonson algoritmi manfiy siklli grafiklarda ishlamaydi, ya'ni unda hech qanday sikl bo'lmasligi kerak.Ushbu maqolada og'irlikli yo'l izlash

algoritimlaridan foydalanish mumkin bo;lgan soxalar , shuningdek shuningdek ushbu soxalarda foydalanish mumkin bo'lgan ilovalar keltirilgan. Jonson algoritimi yordamida yo'l va masofani optimallashtirishga oid masalalarni tahlil qilish va yechish ko'rib chiqiladi. Yo'l va masofani optimallashtirish masalasi bu eng qisqa yoki eng samarali yo'lini topish jarayonidir. Bu masala turli o'lchovlarga asoslanadi:

- **Masofa(eng qisqa yo'l)-** maqsad manzilga(tugunga) eng qisqa masofadagi yo'lni topish .
- **Vaqt-** yetib boorish uchun qisqa vaqt sarflanadigan yo'l.
- **Narx-** yoqilg'I sarfi, transport xarajatlari yoki tariflarni minimallashtirish

Masalan, transport kompaniyalari yuk tashish xarajatlarini kamaytirish uchun yuk mashinalarini eng qisqa va tez yo'nalishlar bo'yicha harakatlantiradi.SHuningdek, onlayn xaritalash tizimlari(masalan, Google Maps yoki Yandex Maps) foydalanuvchiga ma'lum manzilga yetib borish uchun eng qisqa va tez yetib boriladigan yo'lni tavsiya qiladi. Yo'l va masofani optimallashtirishda turli algoritm va usullardan foydalaniladi.

- Dijkstra algoritmi og'irligi musbat bo'lgan grafiklarda manba tugundan barcha boishqa tugunlargacha bo'lgan eng qisqa yo'llarni topish uchun mo'ljallangan u transport va yo'nalishlarni optimallashtirishda keng qo'llaniladi.
- Bellman-Ford algoritmi: ma'nfiy og'irlikka ega grafiklarda ham ishlaydi.U eng qisqa masofani topish uchun mo'ljallangan. Bu algoritm asosiy kamchilgi unda Dijkstra algoritmidan ko'proq vaqt sarflaydi.

Johnson Algoritmi Bo'yicha Bosqichlar

Grafni yaratish: Shaharlar orasidagi masofalarni qirralar orqali ifodalaymiz.

Yangi manba tugun qo'shish: Barcha shaharlar uchun bir xil masofalarni hisoblash maqsadida grafikga yangi manba tugun qo'shiladi va bu tugundan barcha boshqa tugunlarga 0 og'irlikli qirralar orqali ulanish o'rnatiladi.

Asl grafdagi masofalarni qayta hisoblash: Qayta o'lchangan grafikdagi natijalarni asl grafikda qo'llab, kerakli natijalarni topamiz.

Masala yordamida Jonson algoritimi ishlash jarayonini korib chiqaylik

Masala: Bir logistika kompaniyasi 5 ta shaharga xizmat ko'rsatadi. Kompaniya har bir shahar orasidagi masofalarni optimallashtirib, har bir shahardan boshqa barcha shaharlar orasidagi eng qisqa yo'l masofalarini topishi kerak.

Shaharlar orasidagi yo'llar va ularning masofalari quyidagicha:

Shaharlar: A, B, C, D, E

Yo'llar va masofalar:

A -> B : 4 km, A -> C : 2 km, B -> C : -1 km (masofa qisqarishi mumkin bo'lgan yo'l), B -> D : 5 km, C -> D : 8 km, C -> E : 10 km, D -> E : 2 km, E -> B : -3 km (masofa qisqarishi mumkin bo'lgan yo'l)

C# da bu masalani kodini yozamiz:

C# tilidagi kodi quyidagicha:

```
using System;
using System.Collections.Generic;
public class Tugun
{ public int Manba { get; set; }
 public int Manzil { get; set; }
   public int Ogirlik { get; set; }
  public Tugun(int manba, int manzil, int ogirlik)
   { Manba = manba;
     Manzil = manzil;
      Ogirlik = ogirlik;
 }
}public class JohnsonAlgoritmi
{ public static Dictionary<int, int>[] EngQisqaYo'lniTopish(int tugunlarSoni, List<Tugun> qirralar)
   { // Yangi manba tugun qo'shish
      int yangiManba = tugunlarSoni;  for (int i = 0; i < tugunlarSoni; i++)
      { qirralar.Add(new Tugun(yangiManba, i, 0));  }
  // Bellman-Ford algoritmi yordamida og'irliklarni qayta hisoblash
      int[] h = new int[tugunlarSoni + 1];
      Array.Fill(h, int.MaxValue);
      h[yangiManba] = 0;   for (int i = 0; i < tugunlarSoni; i++)
      { foreach (var qirra in qirralar)
         {  if (h[qirra.Manba] != int.MaxValue && h[qirra.Manba] + qirra.Ogirlik < h[qirra.Manzil])
            {
 h[qirra.Manzil] = h[qirra.Manba] + qirra.Ogirlik;
   }   } }   // Qayta og'irliklash
      foreach (var qirra in qirralar)
      { qirra.Ogirlik = qirra.Ogirlik + h[qirra.Manba] - h[qirra.Manzil]; }
  // Har bir tugundan Dijkstra algoritmini qo'llash
      Dictionary<int, int>[] engQisqaYo'llar = new Dictionary<int, int>[tugunlarSoni];
      for (int i = 0; i < tugunlarSoni; i++)
   {
     engQisqaYo'llar[i] = Dijkstra(tugunlarSoni, qirralar, i);
      }
     // Asl grafikda eng qisqa yo'llarni qayta hisoblash
      for (int i = 0; i < tugunlarSoni; i++)
```

```
        {
    foreach (var kalit in engQisqaYo‘llar[i].Keys)
            {
                engQisqaYo‘llar[i][kalit] = engQisqaYo‘llar[i][kalit] + h[kalit] - h[i];
            }
        }   return engQisqaYo‘llar;
      }
       // Dijkstra algoritmi
      private static Dictionary<int, int> Dijkstra(int tugunlarSoni, List<Tugun>
qirralar, int manba)
      {
        Dictionary<int, int> masofalar = new Dictionary<int, int>();
        for (int i = 0; i < tugunlarSoni; i++)
        {
          masofalar[i] = int.MaxValue;
        }
        masofalar[manba] = 0;
         var navbat = new SortedSet<(int masofa, int tugun)>();
        navbat.Add((0, manba));
          while (navbat.Count > 0)
        {
          var (masofa, tugun) = navbat.Min;
          navbat.Remove(navbat.Min);
           foreach (var qirra in qirralar)
          {
            if (qirra.Manba == tugun)
        {
           int yangiMasofa = masofa + qirra.Ogirlik;
            if (yangiMasofa < masofalar[qirra.Manzil])
         {          navbat.Remove((masofalar[qirra.Manzil], qirra.Manzil));
                    masofalar[qirra.Manzil] = yangiMasofa;
                    navbat.Add((yangiMasofa, qirra.Manzil));
                }
              }   } }
       return masofalar;
      }
    }
    public class Program
    {
```

```
        public static void Main()
        {
            // Tugunlar va qirralarni aniqlash
        List<Tugun> qirralar = new List<Tugun>
            {
                new Tugun(0, 1, 4),   // A -> B : 4 km
                new Tugun(0, 2, 2),   // A -> C : 2 km
                new Tugun(1, 2, -1),  // B -> C : -1 km
                new Tugun(1, 3, 5),   // B -> D : 5 km
                new Tugun(2, 3, 8),   // C -> D : 8 km
                new Tugun(2, 4, 10),  // C -> E : 10 km
                new Tugun(3, 4, 2),   // D -> E : 2 km
                new Tugun(4, 1, -3)   // E -> B : -3 km
            };   int tugunlarSoni = 5;
       // Johnson algoritmini qo‘llash
            var engQisqaYo‘llar = JohnsonAlgoritmi.EngQisqaYo’lniTopish(tugunlarSoni, qirralar); // Natijalarni chiqarish
            for (int i = 0; i < tugunlarSoni; i++)
            {
                for (int j = 0; j < tugunlarSoni; j++)
                {
                    Console.WriteLine($"Shahar {i} dan shahar {j} gacha eng qisqa yo'l masofasi: {engQisqaYo‘llar[i][j]} km");
                }
            }
        }
    }
```

Izoh: Graf tuzish: Tugun klassi orqali shaharlar orasidagi masofalarni grafik ko‘rinishida belgilab olamiz. Og‘irliklarni qayta hisoblash: Bellman-Ford algoritmi orqali har bir tugun uchun minimal masofalarni qayta hisoblaymiz.Ushbu kodda Jonson algoritimi orqali barcha tugunlar orasidagi eng qisqa masofalar hisoblanadi.Jonson algoritmi katta grafiklar samarali, real vaqt tizimlarida yo’nalish va transport optimallashtirishda ishlatiladi.

Yo‘l va masofani optimallashtirish muammolari logistika, transport va ishlab chiqarish sohalarida muhim ahamiyat kasb etadi. Ushbu jarayonlarda Jonson algoritmining qo‘llanilishi muhim rol o‘ynaydi, chunki u minimal vaqt va resurslar bilan jarayonlarni samarali tashkil etishga yordam beradi.Jonson algoritmi ikki yoki

undan ortiq resurslar o‘rtasida navbatni optimallashtirish uchun foydalaniladi. Bu algoritm yordamida mashinalar va operatsiyalar uchun navbatni shunday tuzish mumkinki, umumiy vaqtni qisqartirish va jarayonni samaradorligini oshirish mumkin.

Foydalanilgan 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 (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.