IQRO JURNALI , IQRO JOURNAL, ИКРО ЖУРНАЛИ
ISSN:2181-4341| wordlyknowledge.uz/index.php/iqro
Volume 24. Issue 01. Oktabr, 2026

FLOYD ALGORITMI YORDAMIDA PUNKTLARARO QATNOVNING OPTIMAL VARIANTLARINI ANIQLASH

Is'xakova Dilafruz Elbek qizi
Toshkent davlat texnika universiteti
huzuridagi akademik litseyi
oliy toifali o'qituvchisi
https://doi.org/10.5281/zenodo.23158279

Annotatsiya:
Ushbu maqolada bir nechta punktlar orasidagi qatnovda eng kam vaqt va yo'llarning optimal variantlarini aniqlash masalasi tahlil qilinadi. Zamonaviy transport va logistika tizimlarida vaqtni tejash va resurslarni to'g'ri taqsimlash muhim ahamiyat kasb etadi. Tadqiqotda ushbu muammoni yechish uchun Floyd algoritmidan foydalanish ko'rsatilgan. Algoritm barcha punktlar orasidagi eng qisqa yo'llarni topish imkonini beradi. Maqolada Floyd algoritmining matematik asoslari, ishlash bosqichlari va amaliy ahamiyati yoritib berilgan. Shuningdek, bir nechta punktlar orasidagi qatnovda eng kam vaqt va yo'llarning optimal variantlarini aniqlashda ushbu algoritmning samaradorligi asoslab berilgan.
Kalit so'zlar: Floyd algoritmi, optimal masofa, qisqa vaqt, eng qisqa yo'l, dinamik dasturlash.

Аннотация:
В данной статье анализируется проблема определения кратчайшего времени и оптимальных вариантов маршрутов между несколькими точками. В современных транспортных и логистических системах экономия времени и правильное распределение ресурсов имеют большое значение. В исследовании показано применение алгоритма Флойда для решения этой проблемы. Алгоритм позволяет находить кратчайшие маршруты между всеми точками. В статье рассматриваются математические основы, этапы работы и практическое значение алгоритма Флойда. Также обосновывается эффективность этого алгоритма в определении кратчайшего времени и оптимальных вариантов маршрутов между несколькими точками.
Ключевые слова: алгоритм Флойда, оптимальное расстояние, кратчайшее время, кратчайший маршрут, динамическое программирование.

Abstract:
This article analyzes the problem of determining the shortest travel time and optimal routes between multiple points. In modern transport and logistics systems, saving time and the efficient allocation of resources are of great importance. The study demonstrates the application of the Floyd algorithm to solve this problem. The algorithm enables the identification of the shortest routes between all points. The article examines the mathematical foundations, operational stages, and practical significance of the Floyd algorithm. It also substantiates the algorithm's effectiveness in determining the shortest travel time and optimal routes between multiple points.

Keywords: Floyd algorithm, optimal distance, shortest time, shortest route, dynamic programming.

KIRISH

Bugungi kun va zamon talabidan kelib chiqgan holda samarali algoritmga bo'lgan ehtiyoj tobora oshib bormoqda. Algoritmning qo'llanish ko'lami juda keng bo'lib, foydalanuvchilar uchun ko'plab qulayliklar taqdim etadi. Mazkur maqolada ham aholi yashash punktlarini jamoat joylari, yirik bino va tashkilotlarga borishda optimal masofaga erishish masalasi o'rganildi.

TADQIQOT METODOLOGIYASI

Biror n ta joyga eng kam vaqt va yo'ldan foydalangan holda qanday qilib yetib borish mumkin? Bunda bir manzildan ikkinchi manzilga boradigan yollar uchun kerak bo'ladigan yo'lning uzunligi berilgan.

Ushbu masalani yechishda Floyd-Warshall algoritmidan samarali foydalaniladi. Floyd-Warshall algoritmi - vaznli yo'naltirilgan grafikning barcha uchlari orasidagi eng qisqa masofani topish uchun dinamik algoritm. 1962 yilda Robert Floyd va Stiven Vorshel tomonidan ishlab chiqilgan. Shu bilan birga, algoritm birinchi marta 1959 - yilda Bernard Roy tomonidan ishlab chiqilgan va nashr etilgan. Floyd-Warshall algoritmiga ko'ra bitta bajarilishi barcha vertikal juftliklar orasidagi eng qisqa yo'llarning uzunliklarini (to'plangan og'irliklar) topadi. Ushbu algoritm ham yo'naltirilgan ham yo'naltirilmagan og'irlikdagi grafiklar uchun ishlaydi. Biz grafikning uchlari 1 dan n gacha ketma-ket raqamlangan deb taxmin qilamiz. Floyd algoritmida eng qisqa yo'llarning uzunligini hisoblash uchun n x n A matritsadan foydalaniladi. Bunda A[i, j] matritsa elementlari i dan j gacha bo'lgan masofaning uzuznligiga teng. Avvaliga barcha $i \neq j$ uchun A[i, j] = C[i, j] ni qo'yamiz. Agar i dan j gacha bo'lgan masofa bo'lmasa, biz C[i, j] = да deb hisoblaymiz.

Keyin A matritsasi bo'yicha n ta iteratsiya qilamiz. K - iteratsiyasidan keyin ,A[i, j] qiymat uchun raqami k dan yuqori bo'lgan uchdan o'tmaydigan i uchdan j uchigacha bo'lgan yo'lning eng kichik uzunligi bo'ladi. Boshqacha qilib aytganda, i va j yo'lning oxirgi uchlari orasida raqamlari k dan kichik yoki unga teng bo'lgan uchlar bo'lishi mumkin. Ichki uchta sikl doimiy ravishda bajariladigan operatsiyani o'z ichiga oladi ya'ni algoritm kubik murakkablikka ega va oddiy kengaytma yordamida siz eng qisqa yo'llar haqida ma'lumot olishingiz mumkin.

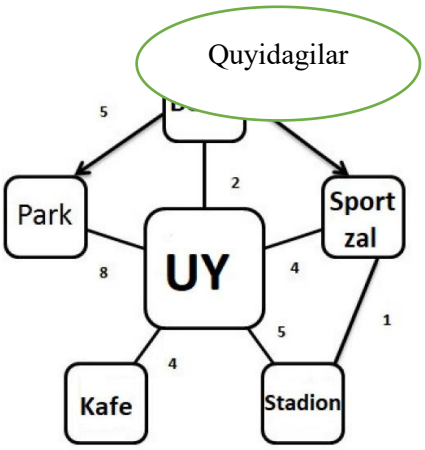
NATIJALAR VA TAHLIL

O'rganilgan misol asosida tadqiqotda 6 ta shaharning joylashuvi va ularni tutashtiruvchi qirralarga mos km qiymatlari asosida eng qisqa yo'l aniqlangan.

Masala. 6 ta shaharning turli manzillari berilgan. Qanday qilib ushbu joylarga eng qisqa yo'ldan foydalangan holda boorish mumkin? Shaharning joylashuvi va ularni tutashtiruvchi qirralarga mos qiymatlari (qiymatlar km da berilgan) ko'rsatilgan.

Siz uydan va qaror qildingiz. sizning yo’llaringiz:

Siz uydan va qaror qildingiz. sizning yo’llaringiz



Parkga borishga

1. Uydan Parkga. (8 km)

2. Uydan Bozorga keyin Parkga (5 + 2 = 7 km). (Bozordan keyin burilmasdan bordingiz deb tassavvur qilaylik.)

1-yo‘nalishni tanlarmidiniz? [8<7 bo’lmasa, yo’q albatta!]

Xuddi shu usul bilan stadionga bormoqchi bo’lsangiz quyidagi yo’ldan borishingiz mumkin:

Uy – > Bozor –> Sport zal – > Stadion

Boshqa yo’llarga nisbatan stadiongacha eng maqbul yo’l shu (2 + 1 + 1 = 4 km) bo’ladi.

To'g'ridan-to'g'ri yo’lning o'rniga siz tranzit o'tishni qo'llashga qaror qildingiz.

A –> B mavjud va B – > C, keyin A –> C mavjud. (1)

Siz bu yo’lni faqatgina A → B → C → A → C (to’g’ridan to’g’ri yo’l) dan qisqaroq ekanligi uchungina ishlatdingiz.(2)

Bu yerda (1) transitive yopilish uchun Warshall Algoritmi va (2) Floyd Algoritmi. U Floyd-Warshall algoritmi deb ataladi, chunki Warshallning tranzitivlik nazariyasi masofalarni yangilash uchun kerak.

Sezgan bo’lsangiz, Parkdan Bozorga yo’l yo’q, Bozordan parkga bor bo’lishiga qaramasdan. Floyd algoritmi asosan yo’nalishli grafiklar uchun. Modomiki u yo’nalishli grafiklar bilan ishlar ekan, yo’nalishsiz grafiklar bilan ham ishlay oladi.

Bunday holatda parkdan uyga qaytishni xohlasangiz, brinchi yo’ldan(8 km) borishingizga to’g’ri keladi. Chunki, faqat bir tomonlama yo’ bo’lganligi sababli siz 2- yo’ldan foydalana olmaysiz.

U manfiy qirrali grafiklar bilan ham ishlaydi. Har bir bog’lovchiga berilgan o’lchovlar masofalar hisoblanadi. Lekin bu uni musbat bo’lishi kerakligini anglatmaydi.

Masofa matritsasini yangilash uchun kod parchasi quyidagicha bo'ladi:

```
for (int k = 0; k < n; k++)
{
for (int i = 0; i < n; i++)
{
for (int j = 0; j < n; j++)
{
if (A[i, k] + A[k, j] < A[i, j])
```

```
A[i, j] = A[i, k] + A[k, j];
}
}
}
```

Bu $O(n^3)$ algoritmidir, chunki sizga n * n * n takrorlash kerak.

Avval sizning uy → sport zali qo'shni matritsada 4 o'rniga 3 ga yangilanadi, marshrut uy – > bozor – > sport zali orqali.

U takrorlash orqali eng qisqa yo'lni quradi. UY – > Bozor – > Sport zal – >Stadion yo'nalishi bir marta yangilanishlik emas.

Keyin sport zali – > stadion qiymati uy – > stadion qiymatini yangilash uchun ishlatiladi. [yani, Agar i dan k gacha minimum masofa va k dan j gacha minimum masofa yig'indisi, i dan j gacha masofadan qisqaroq bo'lsa, yangilang]

Algoritmning C# dasturlash tilidagi kodi:

```
using System;
using System.Collections.Generic; using System.Linq; using System.Text; using System.Threading.Tasks; namespace Algoritm_Floyd {
class Program
{
static void chiqarish(double [,] masofalar, int[,]next)
{
Console.WriteLine("Uchlar masofa yo'nalish"); for (int i = 0; i < next.GetLength(0); i++)
{
for (int j = 0; j < next.GetLength(1); j++)
{
if (i != j)
{
int u = i + 1;
int v = j + 1;
string path = string.Format("{0} → {1} {2,2:G} {3}", u, v, masofalar[i, j], u); do {
u = next[u - 1, v - 1]; path += " – > " + u;
}
while (u != v);
Console.WriteLine(path);
}
}
}
}
static void hisoblash(double[,] a,int n){int[,] next = new int[n, n]; for (int i = 0; i < n; i++)
{
for (int j = 0; j < n; j++)
{
if (i != j)
{
```

```
next[i, j] = j + 1;
}
}
}
for (int i = 0; i < n; i++)
{
for (int j = 0; j < n; j++)
{
for (int k = 0; k < n; k++)
{
if (a[j, k] > a[j, i] + a[i, k])
{
a[j, k] = a[j, i] + a[i, k]; next[j,k]=next[j,i];
}
}
}
}
chiqarish(a,next);
}
static void Main(string[] args)
{
Console.Write("Nuqtalar sonini kiriting:");
int n = int.Parse(Console.ReadLine());
double[,]masofalar=new double[n,n];
Console.WriteLine("Agar ikki nuqta orasidagi masofa cheksiz bo'lsa x ni ezing!");
for (int i = 1; i <=n; i++)
{
for (int j = 1; j <= n; j++)
{
if (i == j)
{
masofalar[i - 1, j - 1] = 0; continue;
}
Console.Write(i + "-→" + j + "="); var x=Console.ReadLine(); if(x=="x")
{
masofalar[i-1, j-1] = double.PositiveInfinity;
}
else
{
masofalar[i-1, j-1] = double.Parse(x);
}
}
}
```

```
Console.WriteLine("Natija");hisoblash(masofalar,n) Console.ReadKey();}
}
}
}
```

1[1] file:///C:/Users/niluf/Desktop/AIgoritm Floyd/Algoritm Floyd/bin/Debug/Algoi

2. Multimodal va vizual kommunikatsiya vositalari

Nuqtalar sonini kiriting:6
Agar ikki njqta orasidagi masofa cheksiz bo'ha x ni ezing!
1->2=2
1->3=8
1- – >4=4
1- – >5=4
1- – >6=5
2- >1=2
2- >3=5
2- >4=6
2- >5=x
2– >6=x
3– >1=8
3- >2=x
3- >4=x
3- >5=x
3- >6=x
4- >1=4
4- >2=x
4- >3=x
4- >5=x
4- >6=1
5- >1=4
5->2=x
5- >3=x
5- >4=x
5- >6=x
6-- >1=5
6- >2=x
6- >3=x

Natija: [6”)4=1]

Uchlar	Masofa	Yo'nalish	Ustun 4	Ustun 5
1 -> 2	1	2	-	-
1 -> 3	1	7	2	3
1 ->4	4	1	4	5

Uchlar	Masofa	Yo'nalish	Ustun 4	Ustun 5
1 -> 5	1	4	6	1
1 -> 6	5	1	3	4
2 -> 1	2	2	1	5
2 -> 3	2	5	1	6
2 -> 4	6	2	1	-
2 -> 5	2	6	1	2
2 -> 6	7	2	1	4
3 -> 1	3	5	1	5
3 -> 2	**3**	3	1	6
3 -> 4	3	5	1	-
4 -> 5	5	12	3	-
4 -> 6	6	12	3	-
4 -> 1	4	13	1	2
4 -> 2	4	4	1	2
4 -> 3	4	4	1	5
4 -> 5	6	4	-	-
4 -> 6	6	11	4	6
5 -> 1	5	8	4	1
5 -> 2	5	1	4	1
5 -> 3	5	4	5	1
5 -> 4	5	6	5	1
5 -> 6	6	1	1	1
6 -> 2	6	8	5	1
6 -> 3	6	9	5	1
6 -> 4	6	5	6	1
6 -> 5	5	7	6	4
	7	12	6	1
	1		6	
	9		6	

XULOSA

Mazkur maqolada Floyd-Warshall algoritmi yordamida bir nechta punktlar orasidagi eng qisqa yo'l va minimal vaqtni aniqlash masalasi tadqiq etildi. 6 ta shahar misolida algoritmning amaliy samaradorligi ko'rsatildi hamda C# dasturlash tilida yozilgan dasturiy ta'minot orqali natijalar tasdiqlandi. Algoritm yo'nalishli va manfiy qirrali grafiklarda ham ishlash qobiliyatiga ega ekanligi asoslab berildi.

FOYDALANILGAN ADABIYOTLAR:

1. Atallah, M. J. (Ed.). "Basic Graph Algorithms." Ch. 6 in Algorithms and Theory of Computation Handbook. Boca Raton, FL: CRC Press, 1998.
2. Floyd, R. W. "Algorithm 97." Comm. ACM 5-6, 345, 1962.
3. Larson, R. and Odoni, A. "Shortest Paths between All Pairs of Nodes." §6.2.2 in Urban Operations Research. 1981.
4. Loerch, U. "Floyd's Algorithm." Auckland, New Zealand: Dept. Computer Science, University of Auckland, 2000.