KOMPYUTER GRAFIKASI VA O'YIN DASTURLASHDA JOHNSON ALGORITMINING AHAMIYATI

Farmonov Sherzodbek Raxmonjonovich
Farg'ona davlat universiteti amaliy matematika va informatika kafedrasi katta o'qituvchisi
farmonovsh@gmail.com
Odiljonova Dilnora Azizjon qizi
Farg'ona davlat universiteti talabasi
dinaraodiljonova1@gmail.com

Annotatsiya. Johnson algoritmi — bu yo'nalishli va og'irliklari musbat yoki manfiy bo'lishi mumkin bo'lgan grafalarda barcha juftliklar o'rtasidagi eng qisqa yo'llarni topish uchun ishlatiladigan algoritm. U Bellman-Ford va Dijkstra algoritmlarini birlashtiradi. Birinchi bosqichda, Bellman-Ford algoritmi yordamida yangi tepadan barcha tepalarga bo'lgan eng qisqa yo'llar topiladi, so'ngra grafaning og'irliklari qayta hisoblanadi (re-weighting). Keyin, har bir tepadan Dijkstra algoritmi yordamida eng qisqa yo'llar hisoblanadi.

Kalit so'zlar. Kompyuter grafikasi va o'yin dasturlashida Johnson algoritmiga oid kalit so'zlar quyidagilar bo'lishi mumkin: eng qisqa yo'l, grafik nazariyasi, manfiy og'irlikli aylanishlar, bellman-Ford algoritmi,dijkstra algoritmi,grafni optimallashtirish,yo'nalish algoritmlari, yo'l toppish, yo'l topishda samaradorlik,grafni ifodalash, qo'shni matritsa, chegara og'irliklari, eng qisqa yo'l muammosi, barcha juftliklar o'rtasidagi eng qisqa yo'l, o'yin sun'iy intellekti (AI).

Annotation. The Johnson algorithm is used to find the shortest paths between all pairs of vertices in a directed graph, where edge weights can be either positive or negative. It combines the Bellman-Ford and Dijkstra algorithms. In the first step, the Bellman-Ford algorithm is used to calculate the shortest paths from a newly added source vertex to all other vertices, and then the edge weights are recalculated (re-weighting). Afterward, the Dijkstra algorithm is applied from each vertex to compute the shortest paths.

Keywords. Johnson Algorith,Shortest Path,Graph Theory,Negative Weight Cycles, Bellman-Ford Algorithm,Dijkstra's Algorithm,Graph Optimization,Routing Algorithms,Pathfinding, Efficiency in Pathfinding,Graph Representation, Adjacency Matrix,Edge Weights,Shortest Path Problem,All-Pairs Shortest Path,Game AI.

Аннотация. Алгоритм Джонсона используется для нахождения кратчайших путей между всеми парами вершин в направленных графах, у которых веса рёбер могут быть как положительными, так и отрицательными. Он объединяет алгоритмы Беллмана-Форда и Дейкстры. На первом этапе с

помощью алгоритма Беллмана-Форда вычисляются кратчайшие пути от вновь добавленной вершины ко всем остальным вершинам, после чего пересчитываются веса рёбер (перевзвешивание). Затем для каждой вершины с помощью алгоритма Дейкстры вычисляются кратчайшие пути.

Ключевыми словами. Кратчайший путь, теория графов,отрицательные циклы с весами, алгоритм Беллмана-Форда, алгоритм Дейкстры, оптимизация графов, алгоритмы маршрутизации, Поиск пути, эффективность поиска пути,представление графа, матрица смежности, веса рёбер, задача кратчайшего пути, кратчайший путь между всеми парами вершин.

Johnson algoritmi — bu og'irliklari musbat yoki manfiy bo'lishi mumkin bo'lgan yo'naltirilgan grafda barcha juftliklar o'rtasidagi eng qisqa yo'llarni topish uchun ishlatiladigan algoritmdir. Bu algoritm, ayniqsa, manfiy og'irlikli grafiklarda ishlashda foydalidir, chunki u manfiy og'irlikli sikllarni ham boshqaradi. Johnson algoritmi Bellman-Ford algoritmi va Dijkstra algoritmning kombinatsiyasidan foydalanadi.

Johnson algoritmi ishlash tartibi:

1. Yangi tepa qo'shish: Grafiga yangi bir tepa qo'shiladi va bu yangi tepa barcha boshqa tepalarga nol og'irlikdagi yo'llar bilan bog'lanadi.

2. Bellman-Ford algoritmini ishga tushirish: Yangi tepadan (ya'ni, dan) Bellman-Ford algoritmi ishlatiladi. Bu algoritm barcha tepalarga eng qisqa yo'llarni hisoblaydi va har bir tepa uchun nomli qiymatlarni topadi. Bu qiymatlar yangi tepaga nisbatan har bir tepaning eng qisqa masofasini ifodalaydi. Agar Bellman-Ford algoritmi grafda manfiy og'irlikli siklni aniqlasa, algoritm to'xtaydi va "eng qisqa yo'l mavjud emas" deb xabar beradi, chunki manfiy sikllar eng qisqa yo'lni belgilab bo'lmaydi.

3. Dijkstra algoritmini ishlatish: Har bir tepadan Dijkstra algoritmini ishlatib, qayta og'irlashtirilgan grafikda barcha tepalarga bo'lgan eng qisqa masofalarni hisoblash kerak.

Johnson algoritmini C# tilida soddaroq misol orqali tushuntirish uchun, biz avvalo grafaning barcha juftlar o'rtasidagi eng qisqa yo'llarni topish masalasini hal qilishga e'tibor qaratamiz. Bu misolda, bizning grafigimizdagi og'irliklar musbat bo'lishi kerak, va algoritmning asosi bo'lgan Bellman-Ford va Dijkstra algoritmlarini qo'llaymiz.

C# tilidagi Johnson algoritmining dasturlash kodi:

```
using System;
using System.Collections.Generic;
class Graph
{
   public int V; // Vertex count
```

```
    public List<Tuple<int, int, int>> Edges; // Edges (start, end, weight)
    public Graph(int V)
    {
        this.V = V;
        this.Edges = new List<Tuple<int, int, int>>();
    }
    // Add edge to graph
    public void AddEdge(int u, int v, int w)
    {
        Edges.Add(Tuple.Create(u, v, w));
    }
    // Bellman-Ford algorithm to find the shortest paths from a source vertex
    public int[] BellmanFord(int source)
    {
        int[] dist = new int[V];
        for (int i = 0; i < V; i++) dist[i] = int.MaxValue;
        dist[source] = 0;
        for (int i = 1; i < V; i++)
        {
            foreach (var edge in Edges)
            {
                int u = edge.Item1, v = edge.Item2, w = edge.Item3;
                if (dist[u] != int.MaxValue && dist[u] + w < dist[v])
                {
                    dist[v] = dist[u] + w;
                }
            }
        }
        return dist;
    }
    // Dijkstra's algorithm to find shortest paths from a source vertex (for re-weighted
graph)
    public int[] Dijkstra(int source)
    {
        int[] dist = new int[V];
        bool[] visited = new bool[V];
        for (int i = 0; i < V; i++) dist[i] = int.MaxValue;
        dist[source] = 0;
        for (int i = 0; i < V; i++)
```

```
    {
        int u = -1;
        for (int j = 0; j < V; j++)
        {
            if (!visited[j] && (u == -1 || dist[j] < dist[u]))
                u = j;
        }
        visited[u] = true;
        foreach (var edge in Edges)
        {
            int v = edge.Item2, w = edge.Item3;
            if (dist[u] != int.MaxValue && dist[u] + w < dist[v])
            {
                dist[v] = dist[u] + w;
            }
        }
    }
    return dist;
}
// Johnson's algorithm to find all-pairs shortest paths
public void Johnson()
{
    // Step 1: Add a new vertex 's' and connect it to all vertices with 0-weight edges
    Graph gNew = new Graph(V + 1);
    foreach (var edge in Edges)
    {
        gNew.AddEdge(edge.Item1, edge.Item2, edge.Item3);
    }
    for (int i = 0; i < V; i++)
    {
        gNew.AddEdge(V, i, 0);
    }
    // Step 2: Run Bellman-Ford from the new vertex 's' to find the potential values
    int[] h = gNew.BellmanFord(V); // h[v] stores the shortest path from 's' to 'v'
   // Step 3: Re-weight the edges using the potential values
    List<Tuple<int, int, int>> reWeightedEdges = new List<Tuple<int, int, int>>();
    foreach (var edge in Edges)
    {
        int u = edge.Item1, v = edge.Item2, w = edge.Item3;
```

```
            reWeightedEdges.Add(Tuple.Create(u, v, w + h[u] - h[v]));
        }
        // Step 4: Run Dijkstra for each vertex to find the shortest paths
        for (int i = 0; i < V; i++)
        {
            Console.WriteLine($"Shortest paths from vertex {i}:");
            int[] dist = Dijkstra(i);
           for (int j = 0; j < V; j++)
            {
                // Adjust the distances back to the original values
                if (dist[j] != int.MaxValue)
                {
                    dist[j] = dist[j] + h[j] - h[i];
                    Console.WriteLine($"To vertex {j}: {dist[j]}");
                }
                else
                {
                  Console.WriteLine($"To vertex {j}: No path");
                }
            }
        }
    }
}
class Program
{
    static void Main()
    {
        // Create a graph with 4 vertices
        Graph g = new Graph(4);
      // Add edges (u, v, weight)
        g.AddEdge(0, 1, -1);
        g.AddEdge(0, 2, 4);
        g.AddEdge(1, 2, 3);
        g.AddEdge(1, 3, 2);
        g.AddEdge(3, 1, 1);
        g.AddEdge(3, 2, 5);
   // Run Johnson's algorithm to find all-pairs shortest paths
        g.Johnson();
    }
```

}

Kodning ishlash prinsipi:

1. Bellman-Ford algoritmi - Bu algoritm birinchi qadamda barcha juftliklar o'rtasidagi eng qisqa yo'llarni hisoblash uchun ishlatiladi.
2. Dijkstra algoritmi - Ikkinchi qadamda, har bir tepadan eng qisqa yo'lni topish uchun Dijkstra algoritmi ishlatiladi, lekin grafa qayta og'irlashtirilgan bo'lib, Dijkstra faqat musbat og'irlikli graflarda ishlaydi.
3. Re-weighting - Bellman-Ford algoritmi yordamida yangi tepadan har bir tepa uchun eng qisqa masofalar topilganidan so'ng, grafa og'irliklari qayta hisoblanadi (re-weighting), va Dijkstra algoritmi yordamida har bir juftlik orasidagi eng qisqa yo'llar topiladi.

Chiqish:

Bu kodda, g.Johnson() chaqirig'i orqali grafa bo'yicha barcha juftliklar o'rtasidagi eng qisqa yo'llar hisoblanadi va chiqariladi.

Misol uchun, quyidagi grafada:

og'irligi -1
og'irligi 4
og'irligi 3
og'irligi 2
og'irligi 1
og'irligi 5

Natija quyidagi shaklda bo'lishi mumkin:

Shortest paths from vertex 0:
To vertex 0: 0
To vertex 1: -1
To vertex 2: 2
To vertex 3: 1

Johnson algoritmi, grafa nazariyasida keng qo'llaniladigan va samarali usul bo'lib, barcha juftliklar o'rtasidagi eng qisqa yo'llarni topish uchun ishlatiladi. U Bellman-Ford va Dijkstra algoritmlarining kombinatsiyasi bo'lib, musbat va manfiy og'irliklarga ega bo'lgan graflarda ham ishlaydi. Biroq, uning samaradorligi va resurslarni talab etishi grafa kattaligi va tuzilishiga bog'liq bo'lishi mumkin.

Foydalanilgan adabiyotlar:

1. "Introduction to Algorithms" ,Thomas H. Cormen, Charles E. Leiserson, Ronald L. Rivest, Clifford Stein,2009 (3-nashr), MIT Press (https://mitpress.mit.edu/books/introduction-algorithms)
2. "Algorithms", Robert Sedgewick, Kevin Wayne,2011 (4-nashr), (https://www.pearson.com/store/p/algorithms/P100000198143)

3. "Graph Theory", Reinhard Diestel, 2017 (5-nashr),Springer (https://link.springer.com/book/10.1007/978-3-662-53422-8)
4. "Data Structures and Algorithm Analysis in C++", Mark Allen Weiss, 2013 (4-nashr), Pearson (https://www.pearson.com/store/p/data-structures-and-algorithm-analysis-in-c/P100000198144)
5. **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. 6. Farmonov, S., & Toirov, S. (2023). NETDA DASTURLASHNING ZAMONAVIY TEXNOLOGIYALARINI O'RGANISH.** *Theoretical aspects in the formation of pedagogical sciences*, *2***(22), 90-96.**