УДК 004.272.2(075.8)

СРАВНИТЕЛЬНЫЙ АНАЛИЗ АЛГОРИТМОВ ФЛОЙДА-УОРШЕЛЛА И ДЕЙКСТРЫ ДЛЯ НАХОЖДЕНИЯ КРАТЧАЙШИХ ПУТЕЙ ВО ВЗВЕШЕННОМ ГРАФЕ

Стельмак К.Д., студент гр.381570, Анацко Д.Д., студент гр. 381570

Белорусский государственный университет информатики и радиоэлектроники,
Институт информационных технологий,
г. Минск, Республика Беларусь

Скудняков Ю.А. – канд. техн. наук, доцент, доцент каф. ИСиТ

Аннотация. Решение задачи нахождения кратчайшего пути в графах является весьма актуальным и практически чрезвычайно важным, поскольку при этом достигается улучшение показателей качества различных объектов и процессов деятельности многих предприятий и организаций разного профиля. В этом случае достигается минимизация издержек в различных сферах человеческой деятельности, приводящих к экономии финансовых, материальных, временных, эксплуатационных и других ресурсов, повышению надежности, долговечности, компактности изделий различного назначения.

Ключевые слова. Кратчайший путь, граф, алгоритм Флойда–Уоршелла, алгоритм Дейкстры, сложность алгоритма.

Введение.

Задачи поиска кратчайшего пути заключаются в нахождении минимального по весу маршрута между вершинами взвешенного графа. Взвешенный граф состоит из вершин (узлов) и рёбер (связей), где каждому ребру может быть присвоен вес (стоимость, расстояние, время и т. д.). Задачи поиска кратчайшего пути широко применяются в картографических сервисах, логистике, компьютерных сетях, в решении задач топологического синтеза радио- и электронных вычислительных средств, экономике, социальных сетях и др. Задача нахождения кратчайшего пути во взвешенном графе [1–9] решается в различных вариантах с использованием ориентированного или неориентированного, разреженного или плотного графа, со взвешенными ребрами и/или взвешенными вершинами, положительными или отрицательными весами, между парой вершин или всеми парами вершин, при обязательном проходе всех вершин или необязательном проходе и т. д. Различные алгоритмы отличаются вычислительной сложностью: поиск кратчайшего пути между парой вершин решается за квадратичное время алгоритмом Дейкстры, поиск кратчайшего пути между всеми парами вершин решается за кубическое время алгоритмом Флойда-Уоршелла, поиск кратчайшего пути при обязательном проходе всех вершин полного графа является *NP*-трудной задачей [9].

Для решения задач поиска кратчайшего пути во взвешенных графах широко применяются алгоритмы Флойда–Уоршелла и Дейкстры. Оба алгоритма решают схожие задачи, но их эффективность и область применения различаются. В данной статье приводится сравнительный анализ эффективности этих алгоритмов.

Алгоритм Флойда-Уоршелла решает задачу нахождения кратчайших путей между всеми парами вершин во взвешенном графе с положительным или отрицательным весом ребер (но без отрицательных циклов). Этот алгоритм постепенно улучшает известные пути, проверяя, можно ли добраться из вершины *i* в вершину *j* через какую-то промежуточную вершину *k*.

Описание алгоритма:

1 Создание матрицы расстояний $dist[V][V]$ (где V – количество вершин в графе), где элемент $dist[i][j]$ обозначает расстояние между вершинами i и j и равен $w(i,j)$ (весу ребра, если такое существует) и ∞ (если ребра нет). Элемент $dist[i][i]$ равен 0 (расстояние от вершины до самой себя).

2 Последовательный перебор всех вершин k и проверка, можно ли улучшить путь из вершины i в вершину j через промежуточную вершину k : $dist[i][j] = min(dist[i][j], dist[i][k]+dist[k][j])$.

3 После выполнения алгоритма $dist[i][j]$ содержит длину кратчайшего пути между любыми двумя вершинами.

Псевдокод алгоритма Флойда–Уоршелла представлен в листинге 1.

Листинг 1 – Псевдокод алгоритма Флойда–Уоршелла

```
1   for k from 1 to V:
2   for i from 1 to V:
3   for j from 1 to V:
4   if dist[i][j] > dist[i][k] + dist[k][j]:
5   dist[i][j] = dist[i][k] + dist[k][j]
```

Алгоритм прост в реализации, однако высокие временная $O(V^3)$ и пространственная $O(V^2)$ сложности делают его неэффективным для больших графов.

Алгоритм Дейкстры – алгоритм поиска кратчайшего пути в графе из заданной вершины во все остальные (single-source shortest path problem).

Алгоритм Дейкстры предназначен для поиска кратчайшего пути от одной вершины ко всем остальным. Работает только для графов без рёбер отрицательного веса.

Описание алгоритма:

1 Задаётся граф с V вершинами и матрицей смежности, где $graph[i][j]$ хранит вес ребра между вершинами i и j (если ребро отсутствует, используется ∞).

2 Создаётся массив $dist[V]$, в котором $dist[i]$ обозначает кратчайшее расстояние от исходной вершины до i. Изначально $dist[i] = \infty$, кроме начальной вершины ($dist[start] = 0$).

3 Используется множество или приоритетная очередь для хранения необработанных вершин.

4 Из множества необработанных вершин выбирается вершина u с минимальным значением $dist[u]$.

5 Для каждой смежной вершины v проверяется, можно ли улучшить её текущее расстояние $dist[v]$ через вершину u. После обработки вершина u помечается как посещённая.

6 Алгоритм продолжается, пока не будут обработаны все вершины или не найдены кратчайшие пути до нужных вершин. По завершении $dist[i]$ содержит кратчайшее расстояние от исходной вершины до каждой другой.

Временная сложность алгоритма:

– $O(V^2)$ при использовании обычного массива для хранения расстояний;

– $O((V + E) \log V)$ при использовании приоритетной очереди (например, двоичной кучи).

Пространственная сложность:

$O(V^2)$ при использовании матрицы смежности;

– $O(V + E)$ при использовании списка смежности.

Псевдокод алгоритма Дейкстры представлен в листинге 2.

Листинг 2 – Псевдокод алгоритма Дейкстры

```
1   func dijkstra(s):
2   for v ∈V v ∈V
3   d[v] == ∞ ∞
4   used[v] = false
5   d[s] = 0
6   for i ∈V i ∈V
7   v = null
8   for j ∈V j ∈V
9   if !used[j] and (v == null or d[j] < d[v])
10  v = j
11  if d[v] == ∞ ∞
12  break
13  used[v] = true
14  for e: исходящие из v рёбра
15  if d[v] + e.len < d[e.to]
16  d[e.to] = d[v] + e.len
```

Плюсы алгоритма Дейкстры: гарантированно находит кратчайшие пути в графе с неотрицательными рёбрами, эффективен при использовании приоритетной очереди, хорошо подходит для маршрутизации. Минусы: работает медленно на плотных графах с большим числом вершин при реализации через матрицу смежности, не применим для графов с отрицательными рёбрами.

Пример применения алгоритмов.

В качестве примера рассмотрим работу алгоритмов Флойда-Уоршелла и Дейкстры для графа, изображенного на рисунке 1.

Представлен взвешенный ориентированный граф без отрицательных ребер с пятью вершинами.

Матрица смежности для данного графа представлена на рисунке 2.

Результат работы программы, реализующей алгоритма Флойда-Уоршелла представлен на рисунке 3.

Результат работы программы, реализующей алгоритм Дейкстры представлен на рисунке 4.

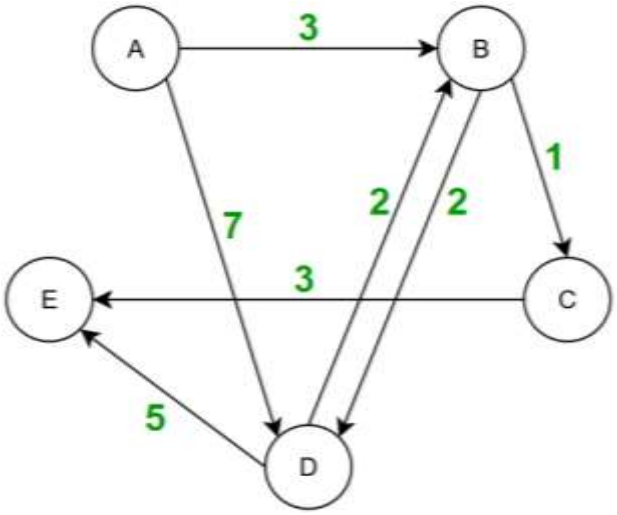

Рисунок 1 – Взвешенный ориентированный граф

	A	B	C	D	E
A	0	3	∞	7	∞
B	∞	0	1	2	∞
C	∞	∞	0	∞	3
D	∞	2	∞	0	5
E	∞	∞	∞	∞	0

Рисунок 2 – Матрица смежности

```
Матрица кратчайших расстояний:
0       3       4       5       7
INF     0       1       2       4
INF     INF     0       INF     3
INF     2       3       0       5
INF     INF     INF     INF     0
```

Рисунок 3 – Результат работы программы, реализующей алгоритм Флойда-Уоршелла

```
Кратчайшие пути от вершины A:
Вершина A: 0, путь: A
Вершина B: 3, путь: A -> B
Вершина C: 4, путь: A -> B -> C
Вершина D: 5, путь: A -> B -> D
Вершина E: 7, путь: A -> B -> C -> E

----------------------------------------

Кратчайшие пути от вершины C:
Вершина A: ∞, путь: нет пути
Вершина B: ∞, путь: нет пути
Вершина C: 0, путь: C
Вершина D: ∞, путь: нет пути
Вершина E: 3, путь: C -> E
```

Рисунок 4 – Результат работы программы, реализующей алгоритм Дейкстры при стартовых вершинах A и C

Сравнение алгоритмов.

Алгоритм Дейкстры оказывается более эффективным для разреженных графов, поскольку его сложность при использовании приоритетной очереди составляет O(ElogV), что делает его предпочтительным при наличии относительно небольшого количества рёбер. В случае плотных графов, где количество рёбер близко к V2V^2, алгоритм Флойда-Уоршелла с временной сложностью

O(V3)O(V^3) становится конкурентоспособным и широко применяется благодаря простоте реализации.

Ключевое различие между алгоритмами заключается в их применимости к графам с отрицательными весами: алгоритм Флойда-Уоршелла корректно работает даже в таких условиях (если отсутствуют отрицательные циклы), тогда как алгоритм Дейкстры не гарантирует нахождения кратчайших путей при наличии рёбер с отрицательным весом.

Заключение.

Оба алгоритма имеют свои сильные и слабые стороны. Алгоритм Дейкстры выигрывает на больших разреженных графах, а алгоритм Флойда–Уоршелла подходит для плотных графов с небольшим количеством вершин. Выбор подходящего алгоритма зависит от конкретной задачи.

Алгоритм Дейкстры может быть использован только в задаче поиска кратчайшего пути с одним источником, а алгоритм Флойда-Уоршелла доступен для поиска кратчайшего пути между любыми двумя точками. Он подходит для поиска кратчайшего пути среди всех вершин или в небольшом объеме данных.

Актуальным и перспективным направлением решения рассмотренных в статье задач является разработка и практическое применение квантовых алгоритмов, обладающих по сравнению с существующими значительно большей производительностью и обеспечивающих существенное сокращение времени обработки информации для получения необходимых, весьма точных и крайне важных, результатов решения сложных проблем в различных сферах человеческой деятельности [11].

Список использованных источников:

1.Дольников, В. Л. Основные алгоритмы на графах : Д 65 текст лекций / В. Л. Дольников, О. П. Якимова; Яросл. гос. ун-т им. П. Г. Демидова. – Ярославль : ЯрГУ, 2011. – 80 с.

2. Floyd, R. W. Algorithm 97: Shortest path / R. W. Floyd // Communications of the ACM, 1962, 5(6), p. 345.

3. Hofner, P. Dijkstra, Floyd and Warshall Meet Kleene/ P. Hofner and B. Moller // Formal Aspect of Computing, Vol.24, No.4, 2012, № 2, pp. 459–476.

4. Venkataraman, G. A Blocked All-Pairs Shortest Paths Algorithm / G. Venkataraman, S. Sahni, S. Mukhopadhyaya // Journal of Experimental Algorithmics (JEA), Vol 8, 2003, pp. 857–874.

5. Park, J. S. Optimizing graph algorithms for improved cache performance / J. S. Park, M. Penner, and V. K. Prasanna // IEEE Trans. on Parallel and Distributed Systems, 2004, 15(9), pp. 769–782.

6. Singh, A. Performance Analysis of Floyd Warshall Algorithm vs Rectangular Algorithm / A. Singh, P. K. Mishra // International Journal of Computer Applications, Vol.107, No.16, 2014, pp. 23–27.

7. Madduri, K. An Experimental Study of a Parallel Shortest Path Algorithm for Solving Large-Scale Graph Instances / K Madduri,. D. Bader, J. W. Berry, J. R. Crobak // Proceedings of the Ninth Workshop on Algorithm Engineering and Experi- ments (ALENEX), 2007, pp. 23–35.

8. Prihozhy, A. Synthesis and Optimization of Pipelines for HW Implementations of Dataflow Programs / A. Prihozhy, E. Bezati, H. Rahman, M. Mattavelli // IEEE Trans. on CAD of Integrated Circuits and Systems, Vol. 34, No. 10, 2015, pp. 1613–1626.

9.Прихожий, А.А. Разнородный блочный алгоритм поиска кратчайших путей между всеми парами вершин графа / А.А. Прихожий,О.И. Карасик, Системный анализ и прикладная информатика. – Минск: БНТУ,2017, – С.68-75.

10.Троелсен, Э. C# и платформа .NET 3.0, специальное издание / Э. Троелсен. – СПб.: Питер, 2008. – 1456 с.

11.Хадиев, К.Р. Квантовый алгоритм для нахождения кратчайшего пути в ациклическом ориентированном графе / К.Р. Хадиев, Л.И. Сафина // Вести МГУ: Вычислительная математика и кибернетика, сер.15, №1. – М.: МГУ, 2017.– С.48-52.

UDC 004.272.2(075.8)

COMPARATIVE ANALYSIS OF FLOYD-WARSHALL AND DIJKSTRA ALGORITHMS FOR FINDING SHORTEST PATHS IN A WEIGHTED GRAPH

Stelmak K.D., student, Anatsko D.D., student

Institute of Information Technologies of the Belarusian State University of Informatics and Radioelectronics, Minsk, Republic of Belarus

Skudnyakov Yu.A.– Candidate of Technical Sciences, Associate Professor

Annotation. Solving the problem of finding the shortest path in graphs is very relevant and practically extremely important, since it improves the quality of various facilities and processes of many enterprises and organizations of various profiles. In this case, minimization of costs in various spheres of human activity is achieved, leading to savings in financial, material, time, operational and other resources, increasing reliability, durability, compactness of products for various purposes.

Keywords. Shortest path, graph, Floyd–Warshall algorithm, Dijkstra algorithm, algorithm complexity.