УДК 004.021
DOI: 10.25559/SITITO.17.202103.585-592

Научная статья

Итерационный алгоритм поиска кратчайшего пути в невзвешенном неориентированном графе

В. В. Сысоев
ПАО «Сбербанк России», г. Москва, Российская Федерация
117997, Российская Федерация, г. Москва, ул. Вавилова, д. 19
Sysoev.V.V@sberbank.ru

Аннотация

Существует проблема поиска наикратчайших путей между двумя вершинами в невзвешенном, неориентированном графе, которая усугубляется тем, что имеющиеся алгоритмы поиска всех путей имеют сложность не меньше $O(n^3)$. Предлагаемый же новый метод поиска кратчайшего пути в невзвешенном неориентированном графе позволяет получить все кратчайшие пути с приемлемой сложностью $O(n^2)$ в худшем случае, а в среднем, $O(n)$. Существующий алгоритм поиск всех путей, алгоритм Флойда-Уоршелла имеет сложность $O(n^3)$, алгоритм Дейкстры, хоть и имеет сложность $O(n^2)$, однако, для нахождения всех кратчайших путей между двумя вершинами, нужно заново пересчитывать пути в графе. Данная статья описывает новый итерационный алгоритм, обосновывает его асимптотическую сложности и сравнивает время и результаты работ с алгоритмом Дейкстры, доказывая тем самым, что обоснование асимптотическая сложность обоснована верно, и алгоритм работает намного быстрее.

Ключевые слова: алгоритм поиска, граф, кратчайший путь, алгоритм Дейкстры, алгоритм Флойда-Уоршелла

Автор заявляет об отсутствии конфликта интересов.

Для цитирования: Сысоев, В. В. Итерационный алгоритм поиска кратчайшего пути в невзвешенном неориентированном графе / В. В. Сысоев. – DOI 10.25559/SITITO.17.202103.585-592 // Современные информационные технологии и ИТ-образование. – 2021. – Т. 17, № 3. – С. 585-592.

Original article

Iterative Algorithm for Finding the Shortest Ways in an Unweighted Undirected Graph

V. V. Sysoev
PJSC "Sberbank of Russia", Moscow, Russian Federation
19 Vavilov St., Moscow 117997, Russian Federation
Sysoev.V.V@sberbank.ru

Abstract

There is a problem of finding the shortest paths between two vertices in an unweighted, undirected graph, which is aggravated by the fact that the available algorithms for finding all paths have a complexity of at least $O(n^3)$. The proposed new method for finding the shortest path in an unweighted undirected graph allows obtaining all shortest paths with acceptable complexity $O(n^2)$, and on average, $O(n)$. The existing algorithm for finding all paths, the Floyd-Warshall algorithm has complexity $O(n^3)$, Deikastra's algorithm, although it has complexity $O(n^2)$, but to find all the paths, you need to recalculate the paths to find all the paths between two vertices in the graph. This article describes a new iterative algorithm, justifies its asymptotic complexity, and compares the time and results of work with Deikasta's algorithm, thereby proving that the justification for asymptotic complexity is justified correctly, and the algorithm works much faster.

Keywords: search algorithm, graph, shortest path, Deikastra's algorithm, Floyd-Warshall algorithm

The author declares no conflict of interest.

For citation: Sysoev V.V. Iterative Algorithm for Finding the Shortest Ways in an Unweighted Undirected Graph. *Sovremennye informacionnye tehnologii i IT-obrazovanie = Modern Information Technologies and IT-Education.* 2021; 17(3):585-592. DOI: https://doi.org/10.25559/SITITO.17.202103.585-592

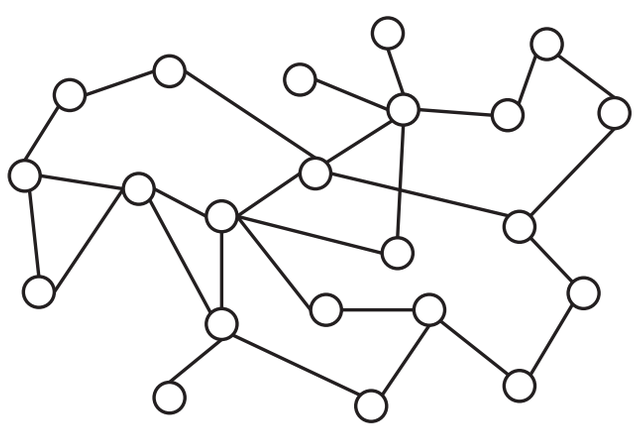
Введение

При поиске кратчайшего пути между двумя вершинами в невзвешенном неориентированном графе чаще всего используется алгоритм Дейкстры (при весах всех ребер = 1)[1] [1, 2, 3, 4]. Алгоритм Флойда-Уоршелла применяется реже[2], поскольку его сложность, в худшем случае, всегда $O(n^3)$ [2, 5, 6, 7], хотя существуют работы, которые показывают, что в малых графах, при $n < 50$ алгоритм Флойда-Уоршелла работает быстрее [8]. Даже по сравнению с алгоритмом A*, который работает на планарных графах, у алгоритма Дейкстры есть преимущества, ввиду его стабильной сложности [9, 11, 12]. Особенность работы алгоритма Дейкстры в режиме поиска всех кратчайших путей в том, что он не способен находить сразу все наикратчайшие пути за один такт своей работы – поэтому его необходимо запускать несколько раз, что приводит к тому, что алгоритм Дейкстры получает сложность $O(n^3)$ [13] в худшем случае[3].

Предлагаемый нами алгоритм находит все кратчайшие пути за один цикл работы и работает за время $O(n^2)$ в худшем случае и время $O(n)$ в среднем, что выводится в данной статье и подтверждается экспериментальным путем.

Описание алгоритма

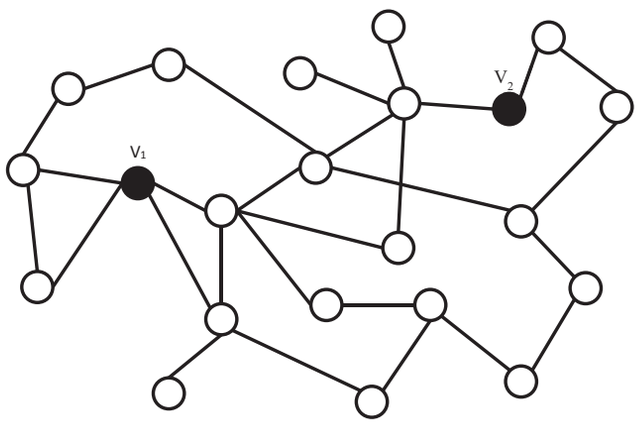
Пусть имеется невзвешенный неориентированный граф G: = (V, E), где V – непустое множество вершин, а E – непустое множество неупорядоченных ребер[4] [14]:



Р и с. 1. Случайный невзвешенный неориентированный граф G
F i g. 1. Random unweighted undirected graph G

Так же каждому узлу графа известно множество его соседей $Nb_v = \{V_1, V_2 \dots V_n\}$, где n – количество соседей узла. Причем данное множество не может быть пустым.

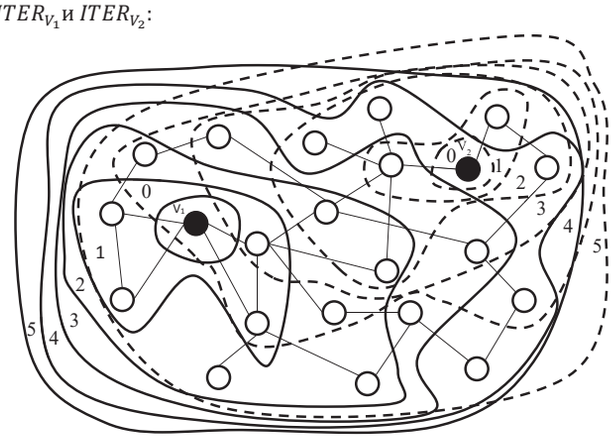
Обозначим вершины, между которыми необходимо найти кратчайшие пути V_1 и V_2:



Р и с. 2. Положение узлов V_1 и V_2 в графе G
F i g. 2. Position of nodes V_1 and V_2 in graph G

Для каждого из вершин V_1 и V_2 определим их итерации. Итерация $ITER_V$ это список вершин, расположенных на удалении N соседей от искомого узла V, $ITER_V = \{\text{ITER}_0, \text{ITER}_1 \dots \text{ITER}_N\}$. Где $ITER_0$ – список узлов на удалении 0 от V (всегда сам узел), $ITER_1$– список узлов на удалении 1 от V (всегда соседи вершины V) и так далее.

Далее, для каждой вершины V_1 и V_2 находим списки итераций $ITER_{V_1}$ и $ITER_{V_2}$:



Р и с. 3. Схематическое отображение итераций для узлов V_1 и V_2
F i g. 3. Schematic display of iterations for nodes V_1 and V_2

На следующем шаге находим индексы вхождений, т.е. определим итерацию, в которую входит противоположная вершина, для каждого из вершин V_1 и V_2. Т.е. во множестве итераций $ITER_{V_1}$ ею будет вершина V_2, и, соответственно, во множестве $ITER_{V_2}$ ею будет являться вершина V_1. В результате получаем

[1] Свидетельство о государственной регистрации программы для ЭВМ № 2018619012 Российская Федерация. Программа построения графов и определения минимальных путей алгоритмом Дейкстры: № 2018615681 : заявл. 31.05.2018 : опубл. 25.07.2018 / Е. Н. Хохлачев, А. Н. Новиков, Е. Е. Новикова; правообладатели Хохлачев Е.Н., Новиков А.Н., Новикова Е.Е. URL: https://www.elibrary.ru/item.asp?id=39299178 (дата обращения: 14.08.2021).

[2] Cormen T.H. Introduction to Algorithms. 3rd Ed. Cambridge, Massachusetts: MIT Press, 2009. 1292 p.

[3] Белоусов А. И., Ткачев С. Б. Математика в техническом университете. 5-е изд. М.: МГТУ им. Н.Э. Баумана, 2007. Вып. 19: Дискретная математика. 2015. 743 с.

[4] Graham R., Knuth D., Patashnik O. Concrete Mathematics: A Foundation for Computer Science. 2nd Ed. Addison-Wesley Professional; 1994. 672 p.

индексы вхождения $ItId_{V_2}$ и $ItId_{V_1}$ – номера итераций, в $ItId_{V_2}$ и $ITER_{V_2}$ в которые входят узлы V_1 и V_2 соответственно.

Далее найдем цепочку путей графа. Для этого выберем два индекса – один из полученных индексов вхождений $ItId_{V_1}$ или $ItId_{V_2}$ и нулевой индекс $ZId = 0$.

Пусть мы выбрали индекс вхождения $ItId_{V_1}$. Исходя из этого, находим общие узлы $ItId_{V_1}[ItId_{V_2}]$ и $ItId_{V_2}[ZId]$, постепенно декрементируя $ItId_{V_2}$ и инкрементируя ZId:

$$W = \left\{\left\{ITER_{V_1}[ItId_{V_2} - \mathrm{i}]\right\} \cap \left\{ITER_{V_2}[ZId + \mathrm{i}]\right\}\right\},$$

где $i = 0 \ldots ItId_{V_2}$. Таким образом, мы получаем список списков вершин

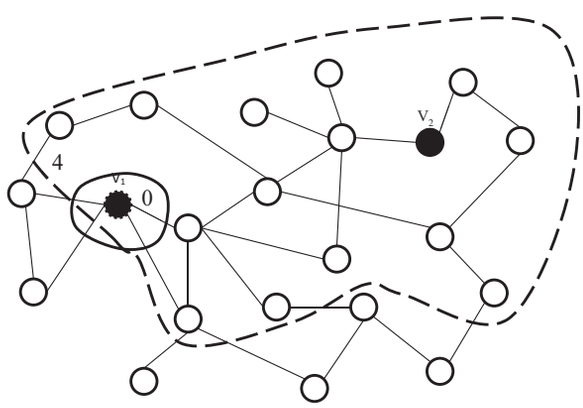
$$W = \{\{V_1\}, \{V_a \ldots V_b\} \ldots \{V_2\}\},$$

где $\{V_a \ldots V_b\}$ – некоторые вершины в произвольной итерации, который при наложении соседей каждой вершины в списке со следующей итерацией даст все наикратчайшие пути:

$$Ways = \{V_1, W_1 \cap Nb_{v_1}, W_2 \cap Nb_{v_2}, V_2\}$$

Для наилучшего понимания работы алгоритма рассмотрим его в схематическом виде:

Рассмотрим ситуацию, когда $ITER_{V_1}[ItId_{V_2}]$ и $ITER_{V_2}[Zid]$:



Р и с. 4. Схематическое изображение результата работы итерации 1
F i g. 4. Schematic representation of the iteration 1 result

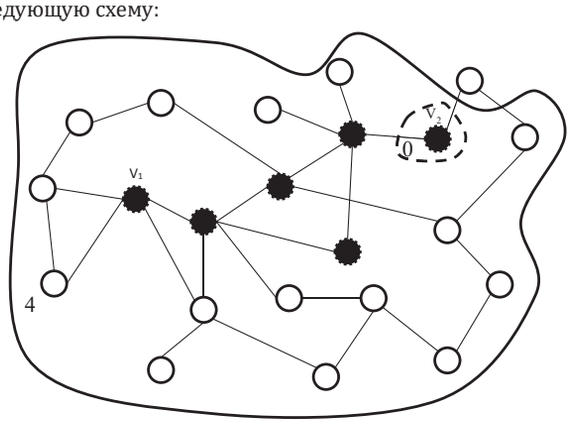
Таким образом, $ItId_{V_2} = 4, Zid = 0$, как договаривались выше. Добавляем вершины пересечения $ITER_{V_1}[ItId_{V_2} - i] \cap ITER_{V_2}[Zid + i]$ в W. На рисунке 4 это будет мы добавляем вершину V_1.

Далее, в результате повторения вышеописанной операции, декрементирования $ItId_{V_2}$ и инкрементирования Zid : $ITER_{V_1}[ItId_{V_2} - i]$ и $ITER_{V_2}[Zid + i]$, получаем следующую схему:

Р и с. 5. Схематическое изображение результата работы итерации 1
F i g. 5. Schematic representation of the iteration 1 result1

Соответственно, добавляем в W вершины пересечения итераций, в нашем примере это вершина V_a .

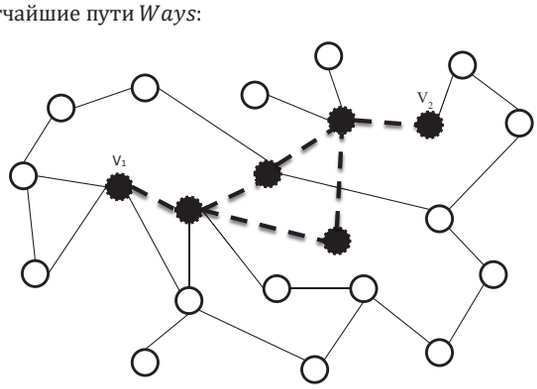
Продолжаем вышеописанный алгоритм $ItId_{V_2}$ раз, и получаем следующую схему:



Р и с. 6. Схематическое изображение результата работы итерационного алгоритма $ItId_{V_2}$ раз
F i g. 6. Schematic representation of the result of the iterative algorithm $ItId_{V_2}$

На схеме вершины, входящие в путь W, обозначены пунктирным окаймлением, они же являются общими для каждой из итераций.

При наложении соседей для каждого узла V_n из W получаем кратчайшие пути $Ways$:



Р и с. 7. Кратчайшие пути между узлами V_1 и V_2
F i g. 7. Shortest paths between nodes V_1 and V_2

Таким образом, в результате работы итерационного алгоритма, получаем все кратчайшие пути между вершинами V_1 и V_2, а, именно, два пути.

Вывод и обоснование сложности алгоритма

Определим асимптотическую сложность по верхнему пределу O [13]. Пусть v – количество вершин в графе G, а e – количество ребер, тогда для худшего случая (граф G полный) асимптотическая сложность по верхнему пределу имеет вид:

$$O = (v-1) + (v-1) + (v-1)\cdot(v-1) + (v-1) \approx v^2, \quad (1)$$

где первое слагаемое представляет собой создание списка итераций для первой вершины, второе слагаемое – соответственно, до второй. Третье слагаемое – это поиск соседей в каждой вершине, обход всех вершин между двумя вершинами и анализ примыкающих к нем ребер. Четвертое слагаемое отображает работу по поиску путей по результирующим массивам.

С точки зрения эффективности наихудшего случая, алгоритм схож с алгоритмом Дейкстры, который имеет ту же сложность $O(v^2)$. Однако, стоит заметить, что алгоритм Дейкстры ищет только один путь, итерационный алгоритм же ищет все пути за раз. Сравнение итерационного алгоритма с алгоритмом Флойда-Уоршелла отпадёт по той причине, что алгоритм Флойда-Уоршелла имеет сложность $O(v^3)$ хотя есть модификация в виде блочного алгоритма, где удавалось добиться сложности $O(nv^2 log b)$, где b – блок размерностью n_b [15]. Сравнение с алгоритмом A* не проводилось, так как алгоритм A* работает только с планарными графами.

Следует так же заметить, что сравнение намеренно указывается именно с классическим алгоритмом, без использования приемов, таких как, куча Фибоначчи [16, 17, 18, 19], параллельной обработки на CPU/GPU [20, 21, 22, 23], использованием подсчета вероятностей [10], [24] или приемов из генетических алгоритмов [25], поскольку наш алгоритм оценивается так же в не модифицированном виде, и применение дополнительных математических/технических приемов, теоретически, может ускорить и его.

Однако, в реальности (в компьютерных сетях, инфраструктуре, компьютерных играх и т.д.) полные графы встречаются крайне редко, чаще всего графы имеют относительно невысокую плотность, при которой $e \ll v^2$. Тогда сложность нашего алгоритма становится $O(v)$, так как второй операнд в операции умножения в формуле (1) представляет собой операции с ребрами, и в условии, когда $e \ll v^2$, им можно пренебречь.

Экспериментальная часть

Для наглядности были проведены эксперименты с нашим алгоритмом и классическим алгоритмом Дейкстры, результаты эксперимента занесены в Таблицу 1.

Таблица 1. **Результаты экспериментов с итерационным алгоритмом и алгоритмом Дейкстры**

Table 1. **Results of experiments with the iterative algorithm and Dijkstra's algorithm**

кол. вершин	№ эксп.	Итерационный		Дейкстры		длина пути	кол. ребер
		t, мс	кол. путей	t, мс	кол. путей		
50	1	12	2	1	1	12	63
	2	0	1	0	1	6	
	3	0	4	0	1	9	
	4	0	3	0	1	5	
	5	0	6	0	1	5	
100	1	0	4	0	1	3	412
	2	0	2	0	1	3	
	3	0	5	0	1	3	
	4	0	5	0	1	3	
	5	0	2	0	1	3	
500	1	23	3	45	1	2	6 499
	2	10	1	44	1	2	
	3	0	1	50	1	1	
	4	24	2	54	1	2	
	5	12	1	60	1	2	
1 000	1	200	34	290	1	3	27 128
	2	118	169	293	1	2	
	3	201	34	297	1	3	
	4	50	4	290	1	2	
	5	97	169	292	1	2	
2 500	1	3 158	3 173	4 784	1	4	154 132
	2	244	9	4 607	1	3	
	3	359 194	21 120	4 747	1	5	
	4	453	4	4 580	1	2	
	5	2 270	90	4 675	1	2	
5 000	1	20 341	3 876	38 530	1	3	646 047
	2	28 076	14 707	36 236	1	3	
	3	41 212	17 220	36 433	1	3	
	4	32 658	9 966	37 519	1	4	
	5	327 699	40	36 721	1	4	
10 000	1	135 075	10 781	175 358	1	3	1 497 971
	2	36 907	324	187 082	1	3	
	3	678	2	175 047	1	2	
	4	124 155	4 764	174 336	1	3	
	5	31 092	108	164 587	1	3	

Эксперименты были проведены на ПК со следующими характеристиками: Intel Core i7-6950X 3.00 Ггц, 64 Гб ОЗУ. Все алгоритмы работали в однопоточном режиме, все графы созданы случайным образом, и такими, что все вершины этих графов достижимы и графы имеют плотность[5] *D=0,05*.

Из таблицы можно просмотреть квадратную зависимость сложности алгоритма Дейкстры $O(v^2)$ от количества вершин и линейную сложность нашего итерационного алгоритма, на которую, по сути, больше влияют количество существующих кратчайших путей между двумя вершинами, чем количество ребер в графах.

[5] Белоусов А. И., Ткачев С. Б. Математика в техническом университете. 5-е изд. М.: МГТУ им. Н.Э. Баумана, 2007. Вып. 19: Дискретная математика. 2015. 743 с.

Заключение

В результате работы алгоритма мы получаем все кратчайшие пути в невзвешенном неориентированном графе. Отличительной чертой работы данного алгоритма является его логическая простота и скорость работы, поскольку он не требует обхода вершин несколько раз для нахождения всех кратчайших путей между двумя вершинами.

Недостатком предложенного алгоритма является повышенный расход памяти для необходимости хранения всем узлами списка своих соседей, а также всех итераций для двух искомых вершин.

Список использованных источников

[1] Galinac Grbac, T. On the Applications of Dijkstra's Shortest Path Algorithm in Software Defined Networks / T. Galinac Grbac, N. Domazet. – DOI 10.1007/978-3-319-66379-1_4 // Intelligent Distributed Computing XI. IDC 2017. Studies in Computational Intelligence; ed. by M. Ivanović, C. Bădică, J. Dix, Z. Jovanović, M. Malgeri, M. Savić. – Springer, Cham, 2018. – Vol. 737. – Pp. 39-45.

[2] Планидин, Р. И. Анализ алгоритмов маршрутизации на примере алгоритмов Дейкстры и Флойда / Р. И. Планидин // Вестник студенческой науки кафедры информационных систем и программирования. – 2017. – № 1(1). – С. 41-47. – URL: https://elibrary.ru/item.asp?id=32677886 (дата обращения: 14.08.2021).

[3] Alam, M. A. Finding Shortest Path for Road Network Using Dijkstra's Algorithm / M. A. Alam, M. O. Faruq. – DOI 10.46281/bjmsr.v1i2.366 // Bangladesh Journal of Multidisciplinary Scientific Research. – 2019. – Vol. 1, issue 2. – Pp. 41-45.

[4] A Heuristic-Based Private Bitcoin Payment Network Formation Using Off-Chain Links / Erdin E. [и др.]. – DOI 10.1109/Blockchain.2019.00046 // 2019 IEEE International Conference on Blockchain (Blockchain). – IEEE Press, Atlanta, GA, USA, 2019. – Pp. 294-301.

[5] Кривошеин, Д. Ю. Алгоритмы пересчёта кратчайших путей в графе при изменении весов ребер / Д. Ю. Кривошеин, А. М. Марченко // Проблемы разработки перспективных микро- и наноэлектронных систем (МЭС). – 2012. – № 1. – С. 263-266. – URL: https://elibrary.ru/item.asp?id=17956216 (дата обращения: 14.08.2021).

[6] Прихожий, А. А. Разнородный блочный алгоритм поиска кратчайших путей между всеми парами вершин графа / А. А. Прихожий, О. Н. Карасик // Системный анализ и прикладная информатика. – 2017. – № 3. – С. 68-75. – URL: https://elibrary.ru/item.asp?id=30731159 (дата обращения: 14.08.2021).

[7] Быкова, В. В. Математические методы анализа рекурсивных алгоритмов / В. В. Быкова // Журнал Сибирского федерального университета. Серия: Математика и физика. – 2008. – Т. 1, № 3. – С. 236-246. – URL: https://www.elibrary.ru/item.asp?id=11482601 (дата обращения: 14.08.2021).

[8] AbuSalim, S. W. G. Comparative Analysis between Dijkstra and Bellman-Ford Algorithms in Shortest Path Optimization / S. W. G. AbuSalim [и др.]. – DOI 10.1088/1757-899X/917/1/012077 // IOP Conference Series: Materials Science and Engineering. – 2020. – Vol. 917. – Article number: 012077.

[9] Yijun, Z. A Fast Bi-Directional A* Algorithm Based on Quad-Tree Decomposition and Hierarchical Map / Z. Yijun, X. Jiadong, L. Chen. – DOI 10.1109/ACCESS.2021.3094854 // IEEE Access. – 2021. – Vol. 9. – Pp. 102877-102885.

[10] Саблин, А. В. Постановка и анализ задачи поиска пути между двумя точками при разработке игр / А. В. Саблин // Информационно-телекоммуникационные технологии и математическое моделирование высокотехнологичных систем. – М.: РУДН, 2019. – С. 295-298. – URL: https://elibrary.ru/item.asp?id=39207936 (дата обращения: 14.08.2021).

[11] Листопад, Н. И. Алгоритмы поиска кратчайшего пути и их модификация / Н. И. Листопад, И. А. Карук, А. А. Хайдер // Информатизация образования. – 2016. – № 1. – С. 48-63. – URL: https://elibrary.ru/item.asp?id=32844256 (дата обращения: 14.08.2021).

[12] Буйнова, Е. Л. Исследование алгоритмов поиска кратчайших расстояний в сетевых моделях / Е. Л. Буйнова, Ф. М. Газизуллина [и др.] // Инновационные научные исследования: теория, методология, тенденции развития. – Уфа: Вестник науки, 2020. – С. 69-78. – URL: https://elibrary.ru/item.asp?id=42641303 (дата обращения: 14.08.2021).

[13] Teaching and assessing discrete mathematics / A. Queiruga-Dios, G. R. Sánchez, Á. M. del Rey, M. Demlova. – DOI 10.1109/EDUCON.2018.8363420 // 2018 IEEE Global Engineering Education Conference (EDUCON). – IEEE Press, Santa Cruz de Tenerife, Spain, 2018. – Pp. 1568-1571.

[14] Knuth, D. The Art of Programming / D. Knuth. – DOI 10.1093/itnow/bwr021 // ITNOW. – 2011. – Vol. 53, issue 4. – Pp. 18-19.

[15] Sao, P. Scalable All-pairs Shortest Paths for Huge Graphs on Multi-GPU Clusters / P. Sao [и др.]. – DOI 10.1145/3431379.3460651 // Proceedings of the 30th International Symposium on High-Performance Parallel and Distributed Computing (HPDC '21). – Association for Computing Machinery, New York, NY, USA, 2021. – Pp. 121-131.

[16] Qu, T. A Fast Isomap Algorithm Based on Fibonacci Heap / T. Qu, Z. Cai. – DOI 10.1007/978-3-319-20469-7_25 // Advances in Swarm and Computational Intelligence. ICSI 2015. Lecture Notes in Computer Science; ed. by Y. Tan, Y. Shi, F. Buarque, A. Gelbukh, S. Das, A. Engelbrecht. – Vol. 9142. – Pp. 225-231. – Springer, Cham, 2015.

[17] Abuaiadh, D. Are Fibonacci heaps optimal / D. Abuaiadh, J. H. Kingston. – DOI 10.1007/3-540-58325-4_210 // Algorithms and Computation. ISAAC 1994. Lecture Notes in Computer Science; ed. by D. Z. Du, X. S. Zhang. – Springer, Berlin, Heidelberg, 1994. – Vol. 834. – Pp. 442-450.

[18] Kozen, D. C. Fibonacci Heaps / D. C. Kozen. – DOI 10.1007/978-1-4612-4400-4_9 // The Design and Analysis of Algorithms. Texts and Monographs in Computer Science. – Springer, New York, NY, 1992. – Pp. 44-47.

[19] Беляев, И. О. Приоритетная очередь на основе бинарной, биномиальной и фибонначиевой куч и ее приме-

нение в многоагентных поисковых системах / И. О. Беляев // RSDN Magazine. – 2012. – № 1. – С. 04-11. – URL: https://elibrary.ru/item.asp?id=17721928 (дата обращения: 14.08.2021).

[20] Zhang, W. Asynchronous Parallel Dijkstra's Algorithm on Intel Xeon Phi Processor / W. Zhang, L. Zhang, Y. Chen. – DOI 10.1007/978-3-030-05051-1_24 // Algorithms and Architectures for Parallel Processing. ICA3PP 2018. Lecture Notes in Computer Science; ed. by J. Vaidya, J. Li. – Springer, Cham, 2018. – Vol. 11334. – Pp. 337-357.

[21] Dong, X. Efficient Stepping Algorithms and Implementations for Parallel Shortest Paths / X. Dong [и др.]. – DOI 10.1145/3409964.3461782 // Proceedings of the 33rd ACM Symposium on Parallelism in Algorithms and Architectures (SPAA '21). – Association for Computing Machinery, New York, NY, USA, 2021. – Pp. 184-197.

[22] Crauser, A. A parallelization of Dijkstra's shortest path algorithm / A. Crauser [и др.]. – DOI 10.1007/BFb0055823 // Mathematical Foundations of Computer Science 1998. MFCS 1998. Lecture Notes in Computer Science; ed. by L. Brim, J. Gruska, J. Zlatuška. – Springer, Berlin, Heidelberg, 1998. – Vol. 1450. – Pp. 722-731.

[23] Khanda, A. A Parallel Algorithm Template for Updating Single-Source Shortest Paths in Large-Scale Dynamic Networks / A. Khanda [и др.]. – DOI 10.1109/TPDS.2021.3084096 // IEEE Transactions on Parallel and Distributed Systems. – 2022. – Vol. 33, no. 4. – Pp. 929-940.

[24] Горбунова, А. В. Решение задачи о выборе оптимального маршрута с использованием вероятностно-статистической модели города / А. В. Горбунова, А. Р. Чичерова, А. К. Демидов // Южно-Уральская молодежная школа по математическому моделированию: сб. тр. III Всерос. НПК / Под ред. Ю. М. Ковалева. – Челябинск: Изд. ЮУрГУ, 2016. – С. 42-45. – URL: https://elibrary.ru/item.asp?id=27559513 (дата обращения: 14.08.2021).

[25] Dudi, T. Shortest Path Evaluation with Enhanced Linear Graph and Dijkstra Algorithm / T. Dudi, R. Singhal, R. Kumar. – DOI 10.23919/SICE48898.2020.9240227 // 2020 59th Annual Conference of the Society of Instrument and Control Engineers of Japan (SICE). – IEEE Press, Chiang Mai, Thailand, 2020. – Pp. 451-456.

Поступила 14.08.2021; одобрена после рецензирования 10.09.2021; принята к публикации 15.09.2021.

Об авторе:

Сысоев Валентин Валерьевич, главный инженер лаборатории кибербезопасности, Блок «Технологии», ПАО «Сбербанк России» (117997, Российская Федерация, г. Москва, ул. Вавилова, д. 19), **ORCID: https://orcid.org/0000-0002-6157-5815,** Sysoev.V.V@sberbank.ru

Автор прочитал и одобрил окончательный вариант рукописи.

References

[1] Galinac Grbac T., Domazet N. On the Applications of Dijkstra's Shortest Path Algorithm in Software Defined Networks. In: Ed. by M. Ivanović, C. Bădică, J. Dix, Z. Jovanović, M. Malgeri, M. Savić. *Intelligent Distributed Computing XI. IDC 2017. Studies in Computational Intelligence*. 2018; 737:39-45. Springer, Cham. (In Eng.) DOI: https://doi.org/10.1007/978-3-319-66379-1_4

[2] Planidin R.I. Programming Alorithm Routing. *Vestnik studencheskoj nauki kafedry informacionnyh sistem i programmirovanija* = The Bulletin of Student Science of the Department of Information Systems and Programming. 2017; (1):41-47. Available at: https://elibrary.ru/item.asp?id=32677886 (accessed 14.08.2021). (In Russ., abstract in Eng.)

[3] Alam M.A., Faruq M.O. Finding Shortest Path for Road Network Using Dijkstra's Algorithm. *Bangladesh Journal of Multidisciplinary Scientific Research*. 2019; 1(2):41-45. (In Eng.) DOI: https://doi.org/10.46281/bjmsr.v1i2.366

[4] Erdin E., Cebe M., Akkaya K., Bulut E., Uluagac A.S. A Heuristic-Based Private Bitcoin Payment Network Formation Using Off-Chain Links. *2019 IEEE International Conference on Blockchain (Blockchain)*. IEEE Press, Atlanta, GA, USA; 2019. p. 294-301. (In Eng.) DOI: https://doi.org/10.1109/Blockchain.2019.00046

[5] Krivoshein D.Yu., Marchenko A.M. Algorithms for dynamic all-pairs shortest path problem. *Problemy razrabotki perspektivnyh mikro- i nanojelektronnyh system (MJeS)* = Problems of Advanced Micro- and Nanoelectronic Systems Development (MES). 2012; (1):263-266. Available at: https://elibrary.ru/item.asp?id=17956216 (accessed 14.08.2021). (In Russ., abstract in Eng.)

[6] Prihozhy A.A., Karasik O.N. Heterogenious blocked all- pairs shortest paths algorithm. *Sistemnyj Analiz i Prikladnaâ Informatika* = System Analysis and Applied Information Science. 2017; (3):68-75. Available at: https://elibrary.ru/item.asp?id=30731159 (accessed 14.08.2021). (In Russ., abstract in Eng.)

[7] Bykova V.V. Mathematical Methods for the Analysis of Recursive Algorithms. *Journal of Siberian Federal University. Mathematics & Physics.* 2008; 1(3):236-246. Available at: https://www.elibrary.ru/item.asp?id=11482601 (accessed 14.08.2021). (In Russ., abstract in Eng.)

[8] AbuSalim S.W.G., et al. Comparative Analysis between Dijkstra and Bellman-Ford Algorithms in Shortest Path Optimization. *IOP Conference Series: Materials Science and Engineering*. 2020; 917:012077. (In Eng.) DOI: https://doi.org/10.1088/1757-899X/917/1/012077

[9] Yijun Z., Jiadong X., Chen L. A Fast Bi-Directional A* Algorithm Based on Quad-Tree Decomposition and Hierarchical Map. *IEEE Access*. 2021; 9:102877-102885. (In Eng.) DOI: https://doi.org/10.1109/ACCESS.2021.3094854

[10] Sablin A.V. Formulation and analysis the problem of finding a path between two points when developing games. *Proceedings of the International Conference on Information and Telecommunication Technologies and Mathematical Modeling of High-Tech Systems.* RUDN University, Moscow;

2019. p. 295-298. Available at: https://elibrary.ru/item.asp?id=39207936 (accessed 14.08.2021). (In Russ., abstract in Eng.)

[11] Listopad N.I., Karuk I.A., Hayder A.A. Algorithms for searching the shortest path and its modification. *Informatizacija obrazovanija* = Informatization of Education. 2016; (1):48-63. Available at: https://elibrary.ru/item.asp?id=32844256 (accessed 14.08.2021). (In Russ., abstract in Eng.)

[12] Buynova E.L., et al. *Issledovanie algoritmov poiska kratchajshih rasstojanij v setevyh modeljah* [Research of algorithms for finding the shortest distances in network models]. Vestnik nauki, Ufa; 2020. p. 69-78. Available at: https://elibrary.ru/item.asp?id=42641303 (accessed 14.08.2021). (In Russ.)

[13] Queiruga-Dios A., Sánchez G.R., del Rey Á.M., Demlova M. Teaching and assessing discrete mathematics. *2018 IEEE Global Engineering Education Conference (EDUCON).* IEEE Press, Santa Cruz de Tenerife, Spain; 2018. p. 1568-1571. (In Eng.) DOI: https://doi.org/10.1109/EDUCON.2018.8363420

[14] Knuth D. The Art of Programming. *ITNOW*. 2011; 53(4):18-19. (In Eng.) DOI: https://doi.org/10.1093/itnow/bwr021

[15] Sao P., Lu H., Kannan R., Thakkar V., Vuduc R., Potok T. Scalable All-pairs Shortest Paths for Huge Graphs on Multi-GPU Clusters. *Proceedings of the 30th International Symposium on High-Performance Parallel and Distributed Computing (HPDC '21).* Association for Computing Machinery, New York, NY, USA; 2021. p. 121-131. (In Eng.) DOI: https://doi.org/10.1145/3431379.3460651

[16] Qu T., Cai Z. A Fast Isomap Algorithm Based on Fibonacci Heap. In: Ed. by Y. Tan, Y. Shi, F. Buarque, A. Gelbukh, S. Das, A. Engelbrecht. *Advances in Swarm and Computational Intelligence. ICSI 2015. Lecture Notes in Computer Science.* 2015; 9142:225-231. Springer, Cham. (In Eng.) DOI: https://doi.org/10.1007/978-3-319-20469-7_25

[17] Abuaiadh D., Kingston J.H. Are Fibonacci heaps optimal? In: Ed. by D. Z. Du, X. S. Zhang. *Algorithms and Computation. ISAAC 1994. Lecture Notes in Computer Science*. 1994; 834:442-450. Springer, Berlin, Heidelberg. (In Eng.) DOI: https://doi.org/10.1007/3-540-58325-4_210

[18] Kozen D.C. Fibonacci Heaps. In: *The Design and Analysis of Algorithms. Texts and Monographs in Computer Science*. Springer, New York, NY; 1992. p. 44-47. (In Eng.) DOI: https://doi.org/10.1007/978-1-4612-4400-4_9

[19] Belyaev I. *Prioritetnaja ochered' na osnove binarnoj, binomial'noj i fibonnachievoj kuch i ee primenenie v mnogoagentnyh poiskovyh sistemah* [Priority queue based on binary, binomial and Fibonacian heaps and its application in multiagent search systems]. *RSDN Magazine.* 2012; (1):04-11. Available at: https://elibrary.ru/item.asp?id=17721928 (accessed 14.08.2021). (In Russ.)

[20] Zhang W., Zhang L., Chen Y. Asynchronous Parallel Dijkstra's Algorithm on Intel Xeon Phi Processor. In: Ed. by J. Vaidya, J. Li. *Algorithms and Architectures for Parallel Processing. ICA3PP 2018. Lecture Notes in Computer Science*. 2018; 11334:337-357. Springer, Cham. (In Eng.) DOI: https://doi.org/10.1007/978-3-030-05051-1_24

[21] Dong X., Gu Y., Sun Y., Zhang Y. Efficient Stepping Algorithms and Implementations for Parallel Shortest Paths. *Proceedings of the 33rd ACM Symposium on Parallelism in Algorithms and Architectures (SPAA '21).* Association for Computing Machinery, New York, NY, USA; 2021. p. 184-197. (In Eng.) DOI: https://doi.org/10.1145/3409964.3461782

[22] Crauser A., Mehlhorn K., Meyer U., Sanders P. A parallelization of Dijkstra's shortest path algorithm. In: Ed. by L. Brim, J. Gruska, J. Zlatuška. *Mathematical Foundations of Computer Science 1998. MFCS 1998. Lecture Notes in Computer Science*. 1998; 1450:722-731. Springer, Berlin, Heidelberg. (In Eng.) DOI: https://doi.org/10.1007/BFb0055823

[23] Khanda A., Srinivasan S., Bhowmick S., Norris B., Das S.K. A Parallel Algorithm Template for Updating Single-Source Shortest Paths in Large-Scale Dynamic Networks. *IEEE Transactions on Parallel and Distributed Systems*. 2022; 33(4):929-940. (In Eng.) DOI: https://doi.org/10.1109/TPDS.2021.3084096

[24] Gorbunova A.V., Chicherova A.R., Demidov A.K. *Reshenie zadachi o vybore optimal'nogo marshruta s ispol'zovaniem verojatnostno-statisticheskoj modeli goroda* [Solving the problem of choosing the optimal route using a probabilistic-statistical model of the city]. In: Ed. by Yu. M. Kovalev. *Proceedings of the South Ural Youth School on Mathematical Modeling*. Chelyabinsk, SUSU Publ.; 2016. p. 42-45. Available at: https://elibrary.ru/item.asp?id=27559513 (accessed 14.08.2021). (In Russ.)

[25] Dudi T., Singhal R., Kumar R. Shortest Path Evaluation with Enhanced Linear Graph and Dijkstra Algorithm. *2020 59th Annual Conference of the Society of Instrument and Control Engineers of Japan (SICE).* IEEE Press, Chiang Mai, Thailand; 2020. p. 451-456. (In Eng.) DOI: https://doi.org/10.23919/SICE48898.2020.9240227

Submitted 14.08.2021; approved after reviewing 10.09.2021; accepted for publication 25.09.2021.

About the author:

Valentin V. Sysoev, Senior engineer of the Cybersecurity Laboratory, PJSC "Sberbank of Russia" (19 Vavilov St., Moscow 117997, Russian Federation), **ORCID: https://orcid.org/0000-0002-6157-5815,** Sysoev.V.V@sberbank.ru

The author has read and approved the final manuscript.