УДК 519.17

СРАВНИТЕЛЬНЫЙ АНАЛИЗ СЛОЖНОСТИ АЛГОРИТМОВ ФЛОЙДА, ДЕЙКСТРЫ И БЕЛЛМАНА-ФОРДА ДЛЯ ГРАФОВ С РАЗЛИЧНЫМ КОЛИЧЕСТВОМ ВЕРШИН

Алексеев П.А.

студент,

Российский университет транспорта (МИИТ),

Москва, Россия

Дубгорный Д.Д.

студент,

Российский университет транспорта (МИИТ),

Москва, Россия

Русских Д.С.

студент,

Российский университет транспорта (МИИТ),

Москва, Россия

Иванова А.П.

к.ф.-м.н., доцент

Российский технологический университет – МИРЭА,

Российский университет транспорта (МИИТ),

Москва, Россия

Аннотация:

В статье проведено исследование трёх алгоритмов построения матрицы кратчайших расстояний: алгоритма Флойда, алгоритма Дейкстры и алгоритма Беллмана-Форда. Рассмотрены графы разной размерности и разной плотности (под плотностью графа понимается число, равное отношению числа рёбер графа к максимально возможному числу рёбер в графе). Была написана программа на языке Python. Приведены графики зависимости времени работы всех алгоритмов

от количества вершин для графов разной плотности. В результате исследования на случайных полных графах получено, что наиболее эффективным алгоритмом является алгоритм Дейкстры.
Ключевые слова: матрица кратчайших расстояний, алгоритм Флойда, алгоритм Дейкстры, алгоритма Беллмана-Форда.

COMPARATIVE ANALYSIS OF THE COMPLEXITY OF FLOYD, DIJKSTRA, AND BELLMAN-FORD ALGORITHMS FOR GRAPHS WITH DIFFERENT NUMBERS OF VERTICES

Alekseev P.A.
student,
Russian University of Transport (MIIT),
Moscow, Russia
Dubgorny D.D.
student,
Russian University of Transport (MIIT),
Moscow, Russia
Russkikh D.S.
student,
Russian University of Transport (MIIT),
Moscow, Russia
Ivanova A.P.
Ph.D., Associate Professor
MIREA – Russian Technological University,
Russian University of Transport (MIIT),
Moscow, Russia

Abstract: The article examines three algorithms for constructing the shortest distance matrix: Floyd's algorithm, Dijkstra's algorithm, and the Bellman-Ford algorithm.

Graphs of different dimensions and densities are considered (graph density is understood as a number equal to the ratio of the number of edges of the graph to the maximum possible number of edges in the graph). The program was written in Python. Graphs of the dependence of the running time of all algorithms on the number of vertices for graphs of different densities are given. As a result of the study on random complete graphs, it was found that the most effective algorithm is Dijkstra's algorithm.
Keywords: shortest distance matrix, Floyd algorithm, Dijkstra algorithm, Bellman-Ford algorithm.

Для нахождения матрицы кратчайших расстояний между всеми парами вершин графа используют алгоритм Флойда, однако, можно использовать и другие алгоритмы, например, алгоритм Дейкстры и алгоритм Беллмана-Форда, которые разрабатывались для отыскания кратчайших расстояний от одной фиксированной вершины до всех остальных. При этом алгоритмы Дейкстры и Беллмана-Форда нужно будет применять для каждой вершины графа в качестве начальной. Цель данной работы сравнить время работы данных алгоритмов.

Пусть задан граф $G = (V, E)$ где V – множество вершин, E – множество рёбер. Граф G – взвешенный, C = $\left(c_{ij}\right)_{n\times n}$ – матрица весов, $n = |V|$.

Так как алгоритмы Дейкстры и Беллмана-Форда могут использоваться только для графов с неотрицательными матрицами весов, будем предполагать, что:

1) все веса рёбер неотрицательные: $\forall c_{ij} \geq 0$,
2) в графе нет циклов с суммарным отрицательным весом (если это не так, то обойдя по такому циклу достаточно много раз, получим путь со сколь угодно малым весом, стремящемся к $-\infty$),
3) матрица весов, вообще говоря, не удовлетворяет неравенству треугольника ($c_{ij} \leq c_{ik} + c_{kj}$),
4) если в графе нет дуги (v_i, v_j), то её вес полагаем равным ∞ ($c_{ij} = \infty$).

Далее дадим описание всех трёх алгоритмов.

1. Алгоритм Флойда [2,4,10] базируется на использовании последовательностей из n преобразований (итераций) начальной матрицы весов. При этом на k-ой итерации матрица представляет длины кратчайших путей между каждой парой вершин с тем ограничением, что путь между вершинами v_i и v_j (для любых v_i и v_j) содержит в качестве промежуточных только вершины из множества $\{v_1, v_2, \dots, v_k\}$.

Предположим, что в начальной матрице весов c_{ii}= 0 для всех i = 1,2,…,n и $c_{ij} = \infty$ если в графе отсутствует дуга (v_i, v_j).

Присвоение начальных значений

Шаг 1. Положить k = 0.

Итерация

Шаг 2. $k = k + 1$.

Шаг 3. Для всех $i \neq k$, таких, что $c_{ik} \neq \infty$, и для всех $j \neq k$, таких, что $c_{kj} \neq \infty$, введём операцию

$$c_{ij} = min\left[c_{ij}, \left(c_{ik} + c_{kj}\right)\right].$$

Проверка на окончание

Шаг 4. (а) Если $c_{ii} < 0$, то в графе G существует цикл отрицательного веса, содержащий вершину v_i, и решения нет.

(б) Если все $c_{ii} \geq 0$ и $k = n$, то получено решение. Матрица $C = \left(c_{ij}\right)_{n\times n}$ даёт длины всех кратчайших путей. Остановка.

(в) Если все $c_{ii} \geq 0$, но $k < n$, то вернуться к шагу 2.

Проведём измерения скорости выполнения алгоритма Флойда от количества вершин n и плотностей графа. Плотностью графа [9] будем называть число, равное отношению числа рёбер графа $|E|$ к максимально возможному числу рёбер в графе $\frac{|V|(|V|-1)}{2}$.

Будем генерировать графы различной плотности (0.25, 0.5, 0.75 и 1.00) случайным образом. Для этого создадим функцию, которая по количеству

вершин *n* и указанной плотности *p* будем псевдослучайным образом генерировать матрицу весов. В цикле проводится изменение содержимого матрицы следующим образом: функция random() модуля [7] random генерирует случайно число из отрезка [0,1], если оно меньше переданной в качестве аргумента функции параметра коэффициента плотности *p*, то данному элементу матрицы присваивается псевдослучайное значение из полуинтервала [1,20), см. рис. 1.

```
def rnd(n, p):
    ar = [[0 for _ in range(n)] for _ in range(n)]
    for i in range(n):
        for j in range(n):
            if i != j:
                if random.random() < p:
                    ar[i][j] = random.randint(1,20)
                else:
                    ar[i][j] = math.inf
    return ar
```

Рис. 1 – Функция генерации матрицы весов графа. Авторская разработка

Для подсчёта времени работы генерируются 10 графов, замеряется время для каждого графа, затем время усредняется. Результаты для количества вершин 100, 200, …, 1000 и разных плотностей 0.25, 0.5, 0.75 и 1.00 графов приведены в табл. 1 и на рис. 2.

Таблица 1. – Зависимость времени работы алгоритма Флойда от количества вершин графа при разной плотности. Авторская разработка

Количество вершин	0,25	0,50	0,75	1,00
100	0,11	0,11	0,11	0,11
200	0,84	0,84	0,84	0,84
300	2,81	2,81	2,80	2,81
400	6,65	6,64	6,62	6,64

500	12,94	12,96	12,94	12,96
600	22,37	22,32	22,37	22,29
700	35,53	35,46	35,47	35,44
800	53,29	53,02	53,05	53,00
900	75,81	75,69	75,66	75,58
1000	104,05	103,91	104,04	103,74

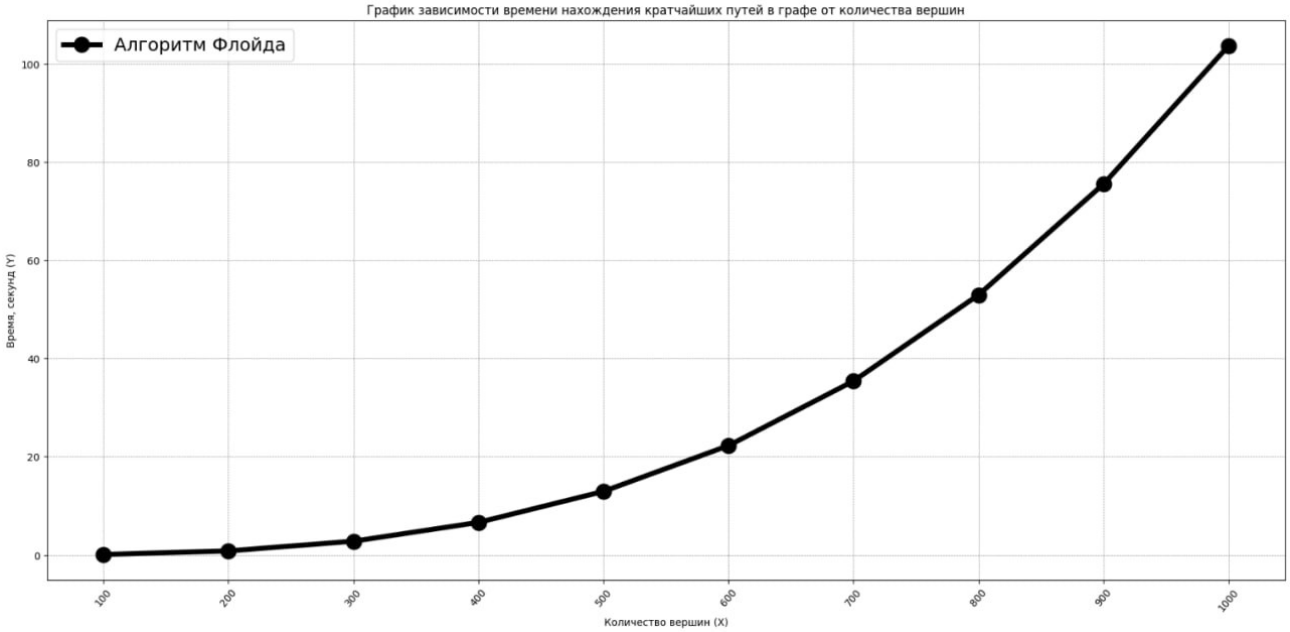

Рис. 2 – Зависимость времени работы алгоритма Флойда от количества вершин полного графа. Авторская разработка

Из табл. 1 и рис. 2 можно сделать вывод, что плотность графа не влияет на время работы алгоритма Флойда, поэтому далее будем применять алгоритм только к полным графам.

2. Алгоритм Дейкстры [2,4,10] позволяет искать кратчайший путь между исходной заданной вершиной и всеми остальными вершинами графа.

Каждой вершине присваивается вес-метка, отражающая минимальное расстояние от исходной вершины до данной. Эта метка может быть временной или постоянной. Временная метка может быть обновлена, если будет найден более короткий путь. Если же обновление не происходит, метка становится

постоянной, и её значение указывает на длину кратчайшего пути от начальной вершины до рассматриваемой.

На каждом шаге алгоритма ровно одна из временных меток становится постоянной.

Будем искать кратчайшие пути из вершины s во все остальные.

Обозначим $l(v_i)$ метку вершины v_i.

Присвоение начальных значений

Шаг 1. Присвоить $l(s)=0$ и считать эту метку постоянной. Присвоить $l(v_i)=\infty \ \ \forall v_i \neq s$ – временные метки. Присвоить $p=s$ – текущая вершина.

Обновление меток

Шаг 2. $\forall v_i \in \Gamma(p)$, метки которых временные, изменить метки:

$$l(v_i)=\min\{l(v_i), l(p)+c(p,v_i)\},$$

где $c(p,v_i)$ – вес ребра, соединяющего текущую вершину p с вершиной v_i.

Превращение меток в постоянные

Шаг 3. Среди всех вершин с временными метками найти вершину v_i^*:

$$l(v_i^*)=\min\left\{l(v_i)\right\}.$$

Шаг 4. Считать метку вершины v_i^* постоянной, обновить текущую вершину: $p=v_i^*$.

Шаг 5. Если все вершины имеют постоянные метки, то эти метки равны длинам кратчайших путей, иначе переход на шаг 2.

Расчёты представлены в табл. 2 и на рис. 3.

Таблица 2. – Зависимость времени работы алгоритма Дейкстры от количества вершин графа при разной плотности. Авторская разработка

Количество вершин	Плотность графа			
	0.25	0.5	0.75	1
100	0.03	0.04	0.06	0.07
200	0.18	0.30	0.42	0.54

300	0.58	1.00	1.42	1.89
400	1.37	2.51	3.57	4.67
500	2.71	5.03	7.13	9.11
600	4.71	8.61	12.06	15.24
700	7.52	13.25	18.49	24.19
800	11.10	19.04	27.38	35.98
900	15.21	26.50	38.74	50.98
1000	20.24	36.26	53.08	69.24

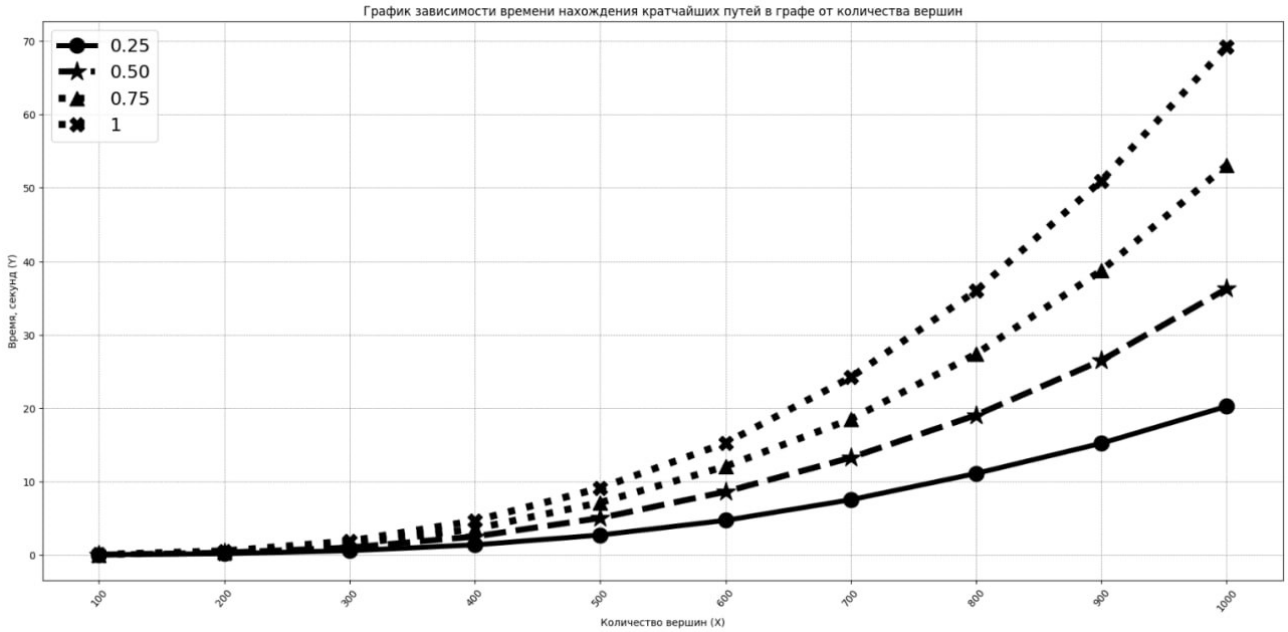

Рис. 3. Зависимость времени работы алгоритма Дейкстры от количества вершин графа для графов разной плотности (0.25, 0.5, 0.75 и 1.00). Авторская разработка

Проанализируем данные: на время работы алгоритма Дейкстры влияет плотность графа, т.к. у менее плотного графа производится меньше итераций (в силу меньшего числа дуг), соответственно алгоритм выполняет свою работу быстрее.

3. Алгоритм Беллмана-Форда [2,4,10] в классической реализации позволяет найти кратчайшие пути от одной вершины до всех остальных, поэтому

для решения задачи необходимо запускать алгоритм ровно столько раз, сколько вершин в графе. Приведём описание алгоритма.

Описываемый метод является итерационным и основан на пометках вершин, причём в конце k-ой итерации пометки равны длинам тех кратчайших путей (от вершины s ко всем остальным вершинам), которые содержат не более k +1 дуг. В отличие от алгоритма Дейкстры никакая из пометок во время этого процесса не рассматривается как окончательная.

Пусть $l^k(v_i)$– пометка вершины v_i в конце (k +1)-ой итерации.

Присвоение начальных значений

Шаг 1. Положим $S = \Gamma(s),\ k = 1,\ l^1(s) = 0,\ l^1(v_i) = c(s, v_i)$ для всех $v_i \in \Gamma(s)$ и $l^1(v_i) = \infty$ для всех остальных.

Обновление пометок

Шаг 2. Для каждой вершины $v_i \in \Gamma(s),\ v_i \neq s$ изменить её пометку следующим образом:

$$l^{k+1}(v_i) = \min[l^k(v_i), \min_{v_j \in T_i}\{l^k(v_j) + \mathrm{c}(v_j, v_i)\}],$$

где $T_i = \Gamma^{-1}(v_i) \cap S$. (Множество S содержит теперь все вершины, для которых кратчайшие пути из s состоят из k дуг).

Множество T_i содержит те вершины, для которых текущие кратчайшие пути из s состоят из k дуг (т. е. те, вершины которых лежат в S) и для которых существуют дуги к вершине v_i. Отметим, что если $v_i \notin \Gamma(s)$, то кратчайший путь от s к v_i не может состоять из k +1 дуг и поэтому изменять пометку вершины v_i не нужно. Для вершин $v_i \notin \Gamma(s)$, положим $l^{k+1}(v_i) = l^k(v_i)$.

Проверка на окончание

Шаг 3. (а) Если $k \leq n-1$ и $l^{k+1}(v_i) = l^k(v_i)$ для всех v_i, то получен оптимальный ответ и пометки равны длинам кратчайших путей. Останов.

(б) Если $k < n-1$ и $l^{k+1}(v_i) \neq l^k(v_i)$ для некоторой вершины v_i , то перейти к шагу 4.

(в) Если $k = n - 1$ и $l^{k+1}(v_i) \neq l^k(v_i)$ для некоторой вершины v_i , то в графе существует цикл отрицательного веса и задача не имеет решения. Останов.

Подготовка к следующей итерации

Шаг 4. Обновить множество S следующим образом:

$$S = \{v_i | l^{k+1}(v_i) \neq l^k(v_i)\}.$$

(Новое множество S содержит теперь все вершины, кратчайшие пути до которых из s имеют длину k +1).

Шаг 5. Положить $k = k + 1$ и перейти к шагу 2.

Как только получены длины кратчайших путей от s к каждой другой вершине, то сами пути находятся просто. Так как вершина v_i' непосредственно предшествует вершине v_i в кратчайшем пути от s к v_i, то для любой вершины v_i соответствующую вершину v_i' можно найти как одну из оставшихся вершин, для которой выполняется условие:

$$l(v_i') + c(v_i', v_i) = l(v_i).$$

Расчёты представлены в табл. 3 и на рис. 4.

Таблица 3. – Зависимость времени работы алгоритма Беллмана-Форда от количества вершин графа при разной плотности. Авторская разработка

Количество вершин	Плотность графа			
	0.25	0.5	0.75	1
100	0,14	0,24	0,3	0,39
200	1,13	2,04	3,04	3,77
300	4,1	7,04	10,06	11,03
400	10,43	17,97	20,2	22,78
500	20,71	30,07	35,43	47,3
600	31,75	47,9	63,32	80,21
700	44,88	76,83	98,23	136,92
800	63,4	114,71	156,41	225,03
900	90,65	151,19	238	306,37
1000	128,45	212,63	311,1	450,87

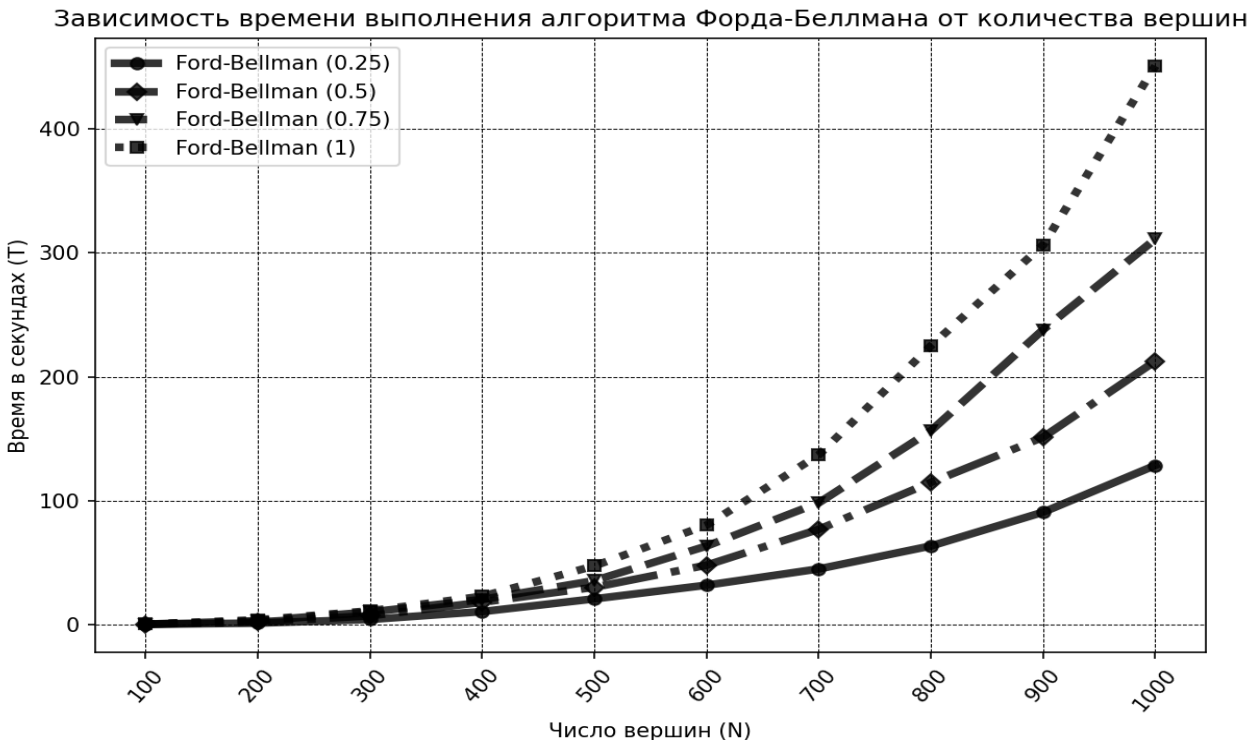

Рис. 4. Зависимость времени работы алгоритма Беллмана-Форда от количества вершин графа для графов разной плотности. Авторская разработка

По графикам рис. 4 видно, что с возрастанием количества вершин время тоже возрастает. То же самое происходит и когда увеличивается плотность графа – время выполнения алгоритма увеличивается. Самое наименьшее время выполнения имеет наименее плотный граф, наибольшее – полный граф.

Теперь сравним все три алгоритма. Будем генерировать случайные графы по 10 штук для каждой размерности от 100 до 1000 вершин (с шагом 100). Измерения будем проводить только для полных графов. В табл. 4 приведено усреднённое время работы алгоритмов.

Таблица 4. – Зависимость времени работы алгоритмов от количества вершин для полных графов. Авторская разработка

Количество вершин	Алгоритм Флойда	Алгоритм Дейкстры	Алгоритм Беллмана-Форда
100	0,11	0,06	0,22
200	0,84	0,49	2,26

300	2,83	1,72	6,94
400	7,19	4,64	16,83
500	14,45	9,38	35,87
600	25,77	15,98	62,06
700	41,34	25,46	105,93
800	59,93	37,25	167,22
900	87,07	54,15	230,53
1000	117,33	73,18	329,25

На рис. 5 приведены графики зависимости времени работы алгоритмов от количества вершин графов.

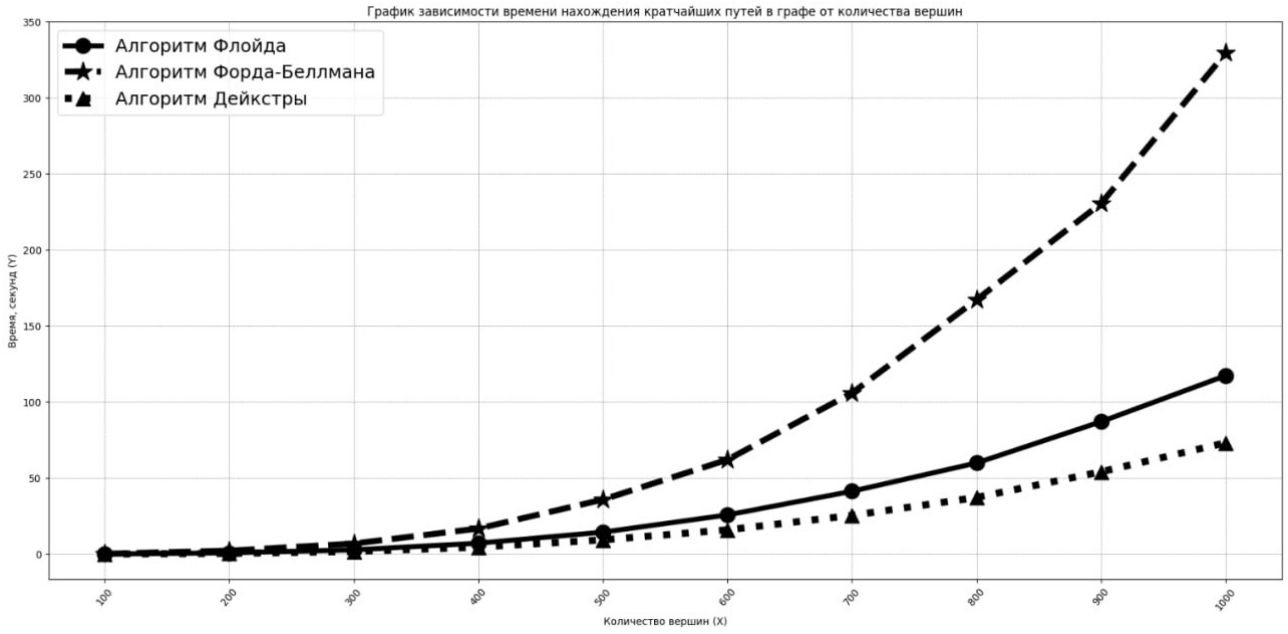

Рис. 5. Зависимость времени работы алгоритмов Флойда, Беллмана-Форда и Дейкстры от количества вершин графа для полных графов. Авторская разработка

Заключение. На основании проведённых исследований (см. табл. 4 и рис. 5) можно сделать вывод, что несмотря на то, что для получения матрицы кратчайших расстояний алгоритм Дейкстры нужно запускать для каждой

вершины графа, он работает быстрее всех. На втором месте по времени работы – алгоритм Флойда, самый медленный – алгоритм Беллмана-Форда.

Библиографический список:

1. Алексеев В.Е, Захарова Д.В. Теория графов: Учебное пособие. – Нижний Новгород: Нижегородский госуниверситет, 2017. – 119 с.

2. Дистель Р. Теория графов: Пер. с англ. – Новосибирск: Изд-во Ин-та математики, 2002. – 336 с.

3. Кормен, Томас Х., Лейзерсон, Чарльз И., Ривест, Рональд Л., Штайн, Клиффорд. Алгоритмы: построение и анализ, 2-е издание.: Пер. с англ. – М.: Издательский дом «Вильямс», 2022. – 1296 с.

4. Кристофидес Н. Теория графов. Алгоритмический подход. – М.: Мир, 1978. – 432 с.

5. Лекции по теории графов / Емеличев В.А., Мельников О.И., Сарванов В.И., Тышкевич Р.И. – М.: Наука, 1990. – 384 с.

6. Новиков Ф.А. Дискретная математика для программистов. Учебник для вузов. 2-е изд. – СПб.: Питер, 2007. – 364 с.

7. Озерова, Г.П. Основы программирования на языке Python в примерах и задачах: учебное пособие для вузов / Политехнический институт ДВФУ. – Владивосток: Изд-во Дальневост. федерал. ун-та, 2022. – 128 с.

8. Оре О. Теория графов. – 2-е изд. - М.: Наука, 1980. – 336 с.

9. Сапожников Н.А., Иванова А.П. Сравнительный анализ алгоритмов Краскала и Прима для графов с различным количеством вершин // Дневник науки. – 2024. – № 12 [Электронный ресурс]. (Дата обращения 28.12.2024).

10. Теория графов. Часть 1: учебное пособие по дисциплине «Теория графов»: Иванова А.П. – Москва: Янус-К, 2024. – 96 с.

Оригинальность 82%