УДК 519.17

СРАВНИТЕЛЬНЫЙ АНАЛИЗ СЛОЖНОСТИ АЛГОРИТМОВ ФЛОЙДА И БЕЛЛМАНА-ФОРДА ДЛЯ ГРАФОВ С РАЗЛИЧНЫМ КОЛИЧЕСТВОМ РЁБЕР

Абрамова О.А.

студент,

Российский университет транспорта (МИИТ),

Москва, Россия

Ковалева В.Е.

студент,

Российский университет транспорта (МИИТ),

Москва, Россия

Ларина В.Е.

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 работе проведено исследование сложности алгоритмов Флойда и Беллмана-Форда. Рассмотрены случайные графы разной плотности. Была написана программа на языке Python. Приведены результаты эмпирической оценки методом наименьших квадратов сложности алгоритмов путём подбора функций-многочленов, описывающих зависимость времени работы алгоритмов от количества вершин для графов разной плотности. В результате исследования на

случайных графах получено, что степень многочлена равна трём и не зависит от плотности графа для обоих алгоритмов.

Ключевые слова: матрица кратчайших расстояний, алгоритм Флойда, алгоритма Беллмана-Форда, сложность алгоритма, метод наименьших квадратов.

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

Abramova O.A.
student,
Russian University of Transport (MIIT),
Moscow, Russia
Kovaleva V.E.
student,
Russian University of Transport (MIIT),
Moscow, Russia
Larina V.E.
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 paper investigates the complexity of Floyd and Bellman-Ford algorithms. Random graphs of different densities are considered. The program was

written in Python. The results of an empirical least squares estimation of the complexity of algorithms by selecting polynomial functions describing the dependence of the running time of algorithms on the number of vertices for graphs of different densities are presented. As a result of the study on random graphs, it was found that the degree of the polynomial is equal to three and does not depend on the graph density for both algorithms.

Keywords: shortest distance matrix, Floyd algorithm, Bellman-Ford algorithm, algorithm complexity, least squares method.

Алгоритмы поиска кратчайших путей в графах находят широкое применение в различных областях, включая компьютерные сети, транспортные системы, социальные сети, био-информатику, игровую индустрию, робототехнику и оптимизацию бизнес-процессов. Целью данной работы является проведение сравнительного анализа алгоритмов отыскания кратчайших путей Флойда и Беллмана-Форда. Эти алгоритмы основаны на различных подходах и имеют свои особенности, влияющие на производительность в зависимости от структуры графа.

Для нахождения матрицы кратчайших расстояний между всеми парами вершин графа используют алгоритм Флойда, однако, можно использовать и другие алгоритмы, например, алгоритм Беллмана-Форда. Алгоритм Беллмана-Форда в этом случае необходимо будет применять для каждой вершины графа, выбирая её в качестве начальной.

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, $n = |V|,\ m = |E|$.

Так как алгоритм Беллмана-Форда можно применять 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 ($c_{ij} \le c_{ik} + c_{kj}$),
4) если в графе нет дуги (v_i, v_j), то её вес полагаем равным ∞ ($c_{ij} = \infty$).

Кратко приведём описание алгоритмов.

Алгоритм Флойда [4,5,9,10]. Предположим, что в начальной матрице весов $c_{ii} = 0$ для всех $i = 1,2,\ldots,n$ 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, 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. Останов.

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

Алгоритм Беллмана-Форда [4,5,10] основан на пометках вершин. Обозначим за $l^k(v_i)$ пометку 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:

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

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:

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

Для выполнения исследования сложности алгоритмов в данной работе использовались случайно сгенерированные графы. Такой подход позволяет объективно оценить эффективность алгоритмов на графах с различной структурой, не ограничиваясь заранее определёнными параметрами.

Функция generate_random_graph генерирует случайный неориентированный граф в виде матрицы смежности. Функция принимает три параметра: size (количество вершин), density (плотность графа, определяющая отношение фактического количества рёбер к максимальному возможному) и max_weight (максимальный вес рёбер). Функция создаёт матрицу, заполненную нулями, и добавляет рёбра между вершинами, присваивая им случайные веса в заданном диапазоне. В итоге возвращается матрица смежности, где ненулевые элементы представляют веса рёбер между вершинами (см. рис. 1).

```python
def generate_random_graph(size, density=0.6, max_weight=30):
    matrix = np.zeros((size, size), dtype=int)
    nodes = list(range(size))
    random.shuffle(nodes)

    for i in range(size - 1):
        u, v = nodes[i], nodes[i + 1]
        matrix[u][v] = matrix[v][u] = random.randint(1, max_weight)

    required_edges = int(size * (size - 1) * density // 2)
    current_edges = size - 1
    while current_edges < required_edges:
        u, v = random.sample(range(size), 2)
        if matrix[u][v] == 0:
            matrix[u][v] = matrix[v][u] = random.randint(1, max_weight)
            current_edges += 1

    return matrix
```

Рисунок 1. – Функция генерации матрицы весов случайного неориентированного графа с заданным количеством вершин и заданной плотности. Авторская разработка

На рис. 2-5 приведены примеры графов, сгенерированных функцией generate_random_graph.

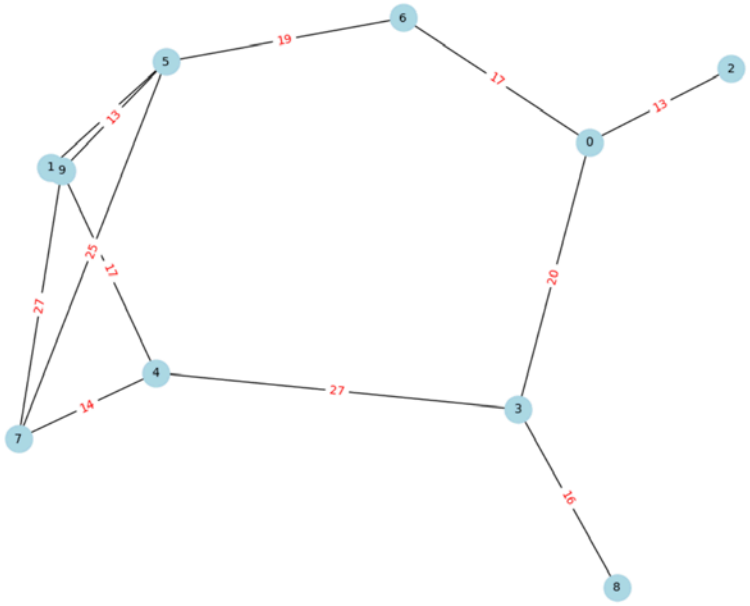

Рисунок 2. – Пример сгенерированного графа с 10 вершинами. Плотность 0.1. Авторская разработка

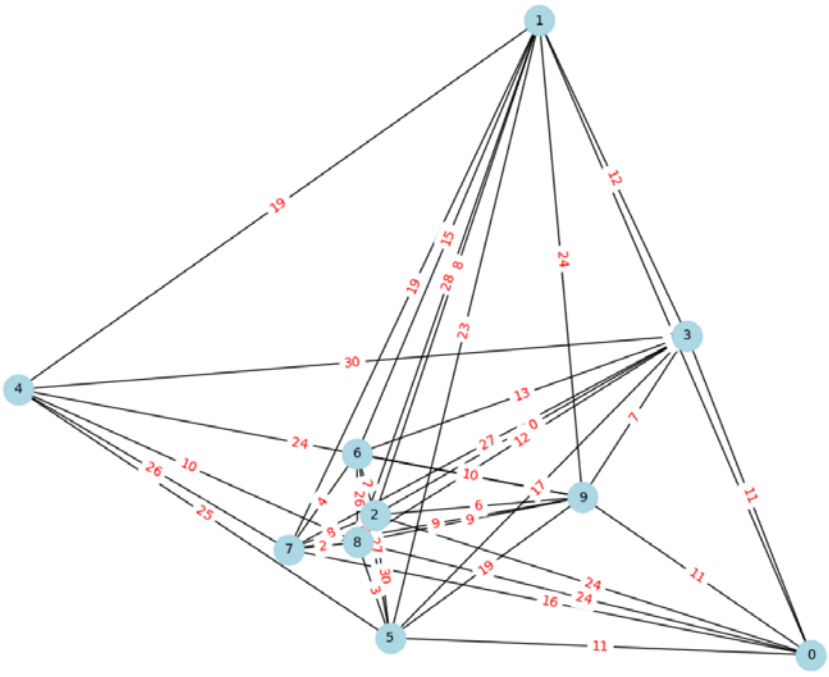

Рисунок 3. – Пример сгенерированного графа с 10 вершинами. Плотность 0.4. Авторская разработка

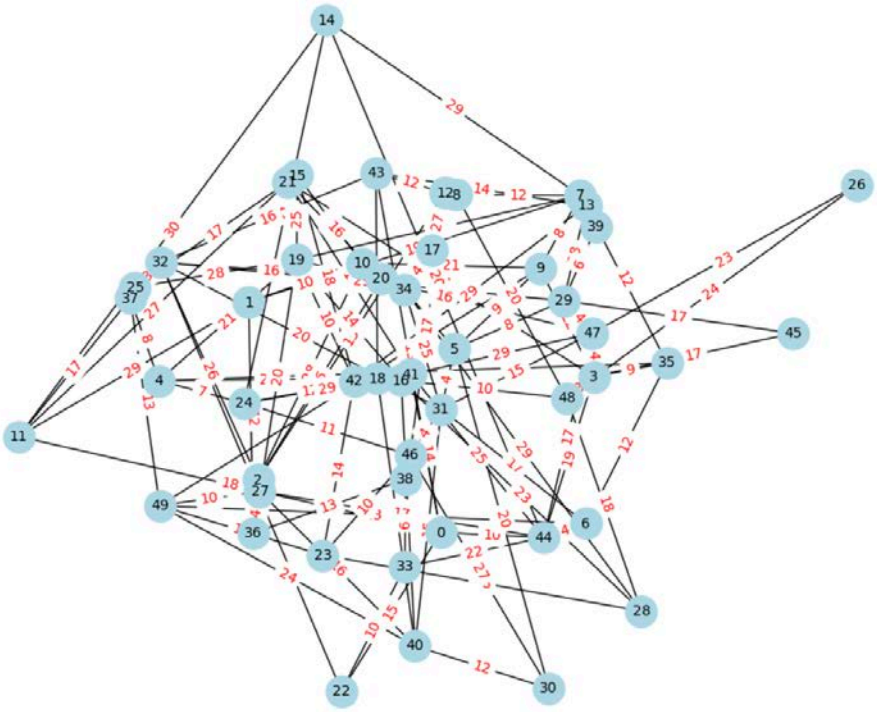

Рисунок 4. – Пример сгенерированного графа с 10 вершинами. Плотность 0.7.

Авторская разработка

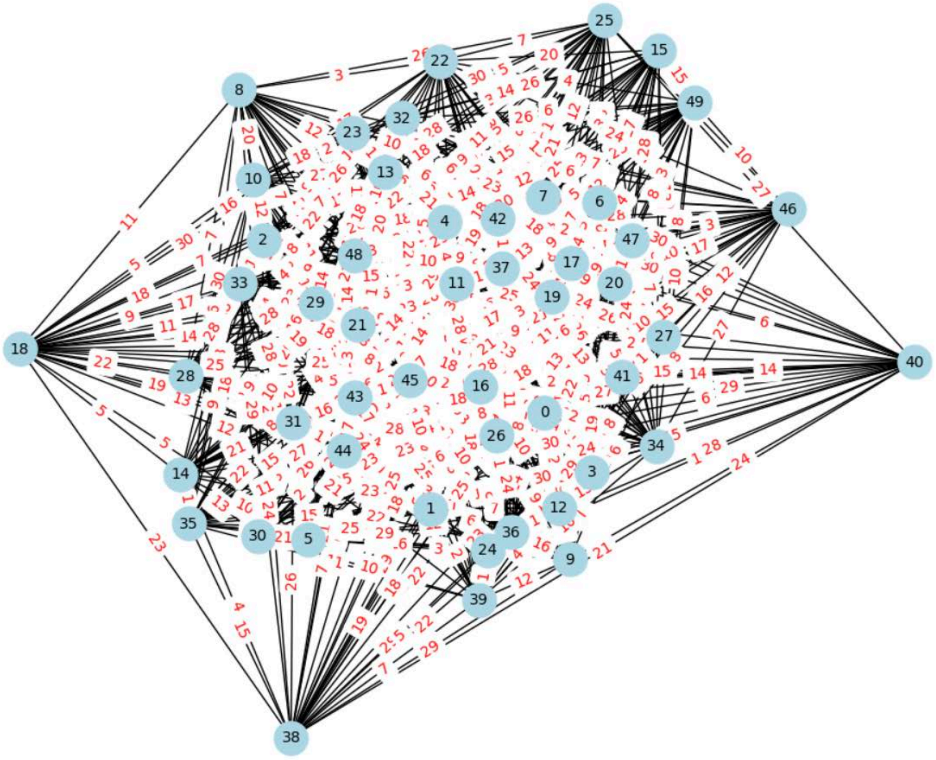

Рисунок 5. – Пример сгенерированного графа с 10 вершинами. Плотность 1.0.

Авторская разработка

Будем подбирать теоретическую функцию зависимости времени работы алгоритмов от количества вершин графа, используя метод наименьших квадратов (МНК). Рассмотрим три варианта.

1) Квадратичная зависимость. Теоретическую функцию $y(x)$ будем искать в следующем виде:

$$y(x) = a_0 + a_1 x + a_2 x^2. \quad (1)$$

Согласно МНК неизвестные параметры a_0, a_1, a_2 выбираются таким образом, чтобы сумма квадратов отклонения эмпирических значений от значений, найденных по формуле (1), была минимальной:

$$\sum_{i=1}^{N} (y_i - Y_i)^2 = \sum_{i=1}^{n} (a_0 + a_1 x_i + a_2 {x_i}^2 - Y_i)^2 \to min.$$

Применяя необходимое условие экстремума для функции трёх переменных $F(a_0, a_1, a_2) = \sum_{i=1}^{N} (a_0 + a_1 x_i + a_2 {x_i}^2 - Y_i)^2$, приравняем к нулю её частные производные:

$$\begin{cases} \dfrac{\partial F}{\partial a_0} = \sum\limits_{i=0}^{N} 2(a_0 + a_1 x_i + a_2 {x_i}^2 - Y_i) = 0, \\ \dfrac{\partial F}{\partial a_1} = \sum\limits_{i=0}^{N} 2(a_0 + a_1 x_i + a_2 {x_i}^2 - Y_i) x_i = 0, \\ \dfrac{\partial F}{\partial a_2} = \sum\limits_{i=0}^{N} 2(a_0 + a_1 x_i + a_2 {x_i}^2 - Y_i) {x_i}^2 = 0. \end{cases}$$

После преобразования получим следующую систему нормальных уравнений:

$$\begin{cases} a_0 N + a_1 \sum_{i=1}^{N} x_i + a_2 \sum_{i=1}^{N} x_i^2 = \sum_{i=1}^{N} Y_i, \\ a_0 \sum_{i=1}^{N} x_i + a_1 \sum_{i=1}^{N} x_i^2 + a_2 \sum_{i=1}^{N} x_i^3 = \sum_{i=1}^{N} Y_i x_i, \\ a_0 \sum_{i=1}^{N} x_i^2 + a_1 \sum_{i=1}^{N} x_i^3 + a_2 \sum_{i=1}^{N} x_i^4 = \sum_{i=1}^{N} Y_i x_i^2. \end{cases}$$

Решив эту систему, получим значения коэффициентов квадратичной функции (1).

2) Кубическая зависимость. Функцию будем искать в следующем виде:

$$y(x) = a_0 + a_1 x + a_2 x^2 + a_3 x^3. \quad (2)$$

Система нормальных уравнений в этом случае имеет вид:

$$\begin{cases} a_0 N + a_1 \sum_{i=1}^{N} x_i + a_2 \sum_{i=1}^{N} x_i^2 + a_3 \sum_{i=1}^{N} x_i^3 = \sum_{i=1}^{N} Y_i, \\ a_0 \sum_{i=1}^{N} x_i + a_1 \sum_{i=1}^{N} x_i^2 + a_2 \sum_{i=1}^{N} x_i^3 + a_3 \sum_{i=1}^{N} x_i^4 = \sum_{i=1}^{N} Y_i x_i, \\ a_0 \sum_{i=1}^{N} x_i^2 + a_1 \sum_{i=1}^{N} x_i^3 + a_2 \sum_{i=1}^{N} x_i^4 + a_3 \sum_{i=1}^{N} x_i^5 = \sum_{i=1}^{N} Y_i x_i^2, \\ a_0 \sum_{i=1}^{N} x_i^3 + a_1 \sum_{i=1}^{N} x_i^4 + a_2 \sum_{i=1}^{N} x_i^5 + a_3 \sum_{i=1}^{N} x_i^6 = \sum_{i=1}^{N} Y_i x_i^3. \end{cases}$$

3) Многочлен 4 степени. Функцию будем искать в следующем виде:

$$y(x) = a_0 + a_1 x + a_2 x^2 + a_3 x^3 + a_4 x^4. \quad (3)$$

Система нормальных уравнений в этом случае имеет вид:

$$\begin{cases} a_0 N + a_1 \sum_{i=1}^{N} x_i + a_2 \sum_{i=1}^{N} x_i^2 + a_3 \sum_{i=1}^{N} x_i^3 + a_4 \sum_{i=1}^{N} x_i^4 = \sum_{i=1}^{N} Y_i, \\ a_0 \sum_{i=1}^{N} x_i + a_1 \sum_{i=1}^{N} x_i^2 + a_2 \sum_{i=1}^{N} x_i^3 + a_3 \sum_{i=1}^{N} x_i^4 + a_4 \sum_{i=1}^{N} x_i^5 = \sum_{i=1}^{N} Y_i x_i, \\ a_0 \sum_{i=1}^{N} x_i^2 + a_1 \sum_{i=1}^{N} x_i^3 + a_2 \sum_{i=1}^{N} x_i^4 + a_3 \sum_{i=1}^{N} x_i^5 + a_4 \sum_{i=1}^{N} x_i^6 = \sum_{i=1}^{N} Y_i x_i^2, \\ a_0 \sum_{i=1}^{N} x_i^3 + a_1 \sum_{i=1}^{N} x_i^4 + a_2 \sum_{i=1}^{N} x_i^5 + a_3 \sum_{i=1}^{N} x_i^6 + a_4 \sum_{i=1}^{N} x_i^7 = \sum_{i=1}^{N} Y_i x_i^3, \\ a_0 \sum_{i=1}^{N} x_i^4 + a_1 \sum_{i=1}^{N} x_i^5 + a_2 \sum_{i=1}^{N} x_i^6 + a_3 \sum_{i=1}^{N} x_i^7 + a_4 \sum_{i=1}^{N} x_i^8 = \sum_{i=1}^{N} Y_i x_i^4. \end{cases}$$

Решим полученные системы нормальных уравнений и получим неизвестные коэффициенты функций (1), (2) и (3). Также для оценки качества подбора теоретической зависимости вычислим сумму квадратов отклонений теоретических и эмпирических значений:

$$Sum = \sum_{i=1}^{n} (y(x_i) - Y_i)^2, \quad (4)$$

здесь Y_i – эмпирические значения, $y(x_i)$ – теоретические значения.

В таблицах 1 и 2 приведены значения коэффициентов a_0, a_1, a_2, a_3, a_4 и величины *Sum* для графов различной плотности 0.1, 0.5 и 0.9, вычисленные для алгоритмов Флойда и Беллмана-Форда соответственно.

Таблица 1. – Коэффициенты функций (1), (2) и (3) и значение *Sum* для графов различной плотности 0.1, 0.5 и 0.9, вычисленные для алгоритма Флойда.

Авторская разработка

	Плотность графа		
	0.1	0.5	0.9
Квадратичная	a_0 = 0.0745	a_0 = 0.0743	a_0 = 0.0742

$y(x) = a_0 + a_1x + a_2x^2$	$a_1 = -0.0066$ $a_2 = 0.0001$	$a_1 = -0.0066$ $a_2 = 0.0001$	$a_1 = -0.0066$ $a_2 = 0.0001$
	Sum = 0.0024	*Sum* = 0.0023	*Sum* = 0.0023
Кубическая $y(x) = a_0 + a_1x + a_2x^2 + a_3x^3$	$a_0 = -4.7\cdot10^{-4}$ $a_1 = 6.5\cdot10^{-5}$ $a_2 = -1.5\cdot10^{-6}$ $a_3 = 8.7\cdot10^{-7}$	$a_0 = -8.0\cdot10^{-5}$ $a_1 = 1.8\cdot10^{-5}$ $a_2 = 1.4\cdot10^{-7}$ $a_3 = 8.7\cdot10^{-7}$	$a_0 = -2.2\cdot10^{-4}$ $a_1 = 2.2\cdot10^{-5}$ $a_2 = 2.1\cdot10^{-6}$ $a_3 = 8.7\cdot10^{-7}$
	Sum = $2.4\cdot10^{-6}$	*Sum* = $2.0\cdot10^{-7}$	*Sum* = $6.0\cdot10^{-7}$
Многочлен 4 степени $y(x) = a_0 + a_1x + a_2x^2 + a_3x^3 + a_4x^4$	$a_0 = 2.7\cdot10^{-4}$ $a_1 = -2.9\cdot10^{-5}$ $a_2 = 1.9\cdot10^{-6}$ $a_3 = 8.3\cdot10^{-7}$ $a_4 = 2.1\cdot10^{-10}$	$a_0 = -2.3\cdot10^{-4}$ $a_1 = 3.7\cdot10^{-5}$ $a_2 = -5.7\cdot10^{-7}$ $a_3 = 8.8\cdot10^{-7}$ $a_4 = -4.4\cdot10^{-10}$	$a_0 = -1.1\cdot10^{-2}$ $a_1 = 1.2\cdot10^{-4}$ $a_2 = -1.8\cdot10^{-6}$ $a_3 = 9.2\cdot10^{-7}$ $a_4 = -2.4\cdot10^{-10}$
	Sum = $3.1\cdot10^{-7}$	*Sum* = $1.4\cdot10^{-7}$	*Sum* = $6.8\cdot10^{-8}$

Таблица 2. – Коэффициенты функций (1), (2) и (3) и значение *Sum* для графов различной плотности 0.1, 0.5 и 0.9, вычисленные для алгоритма Беллмана-Форда. Авторская разработка

	Плотность графа		
	0.1	0.5	0.9
Квадратичная $y(x) = a_0 + a_1x + a_2x^2$	$a_0 = 3.4\cdot10^{-2}$ $a_1 = -3.0\cdot10^{-3}$	$a_0 = 6.0\cdot10^{-2}$ $a_1 = -5.3\cdot10^{-3}$	$a_0 = 8.6\cdot10^{-2}$ $a_1 = -7.6\cdot10^{-3}$

	$a_2 = 6.4 \cdot 10^{-5}$	$a_2 = 1.1 \cdot 10^{-4}$	$a_2 = 1.6 \cdot 10^{-4}$
	$Sum = 4.9 \cdot 10^{-4}$	$Sum = 1.5 \cdot 10^{-3}$	$Sum = 3.2 \cdot 10^{-3}$
Кубическая $y(x) = a_0 + a_1 x + a_2 x^2 + a_3 x^3$	$a_0 = 4.7 \cdot 10^{-5}$ $a_1 = 6.0 \cdot 10^{-6}$ $a_2 = -6.5 \cdot 10^{-7}$ $a_3 = 4.0 \cdot 10^{-7}$	$a_0 = -1.6 \cdot 10^{-4}$ $a_1 = 2.2 \cdot 10^{-5}$ $a_2 = -164 \cdot 10^{-6}$ $a_3 = 7.0 \cdot 10^{-7}$	$a_0 = -1.1 \cdot 10^{-3}$ $a_1 = 1.4 \cdot 10^{-4}$ $a_2 = -5.4 \cdot 10^{-6}$ $a_3 = 1.0 \cdot 10^{-6}$
	$Sum = 3.1 \cdot 10^{-8}$	$Sum = 3.5 \cdot 10^{-7}$	$Sum = 1.4 \cdot 10^{-5}$
Многочлен 4 степени $y(x) = a_0 + a_1 x + a_2 x^2 + a_3 x^3 + a_4 x^4$	$a_0 = 2.8 \cdot 10^{-4}$ $a_1 = -2.3 \cdot 10^{-5}$ $a_2 = 4.3 \cdot 10^{-7}$ $a_3 = 3.8 \cdot 10^{-7}$ $a_4 = 6.7 \cdot 10^{-11}$	$a_0 = -2.2 \cdot 10^{-4}$ $a_1 = 3.0 \cdot 10^{-5}$ $a_2 = -1.9 \cdot 10^{-6}$ $a_3 = 7.0 \cdot 10^{-7}$ $a_4 = -1.6 \cdot 10^{-11}$	$a_0 = 1.4 \cdot 10^{-4}$ $a_1 = -2.8 \cdot 10^{-45}$ $a_2 = 6.5 \cdot 10^{-7}$ $a_3 = 9.3 \cdot 10^{-7}$ $a_4 = 3.7 \cdot 10^{-10}$
	$Sum = 7.1 \cdot 10^{-9}$	$Sum = 1.2 \cdot 10^{-7}$	$Sum = 2.2 \cdot 10^{-6}$

Заключение. На основании проведённых исследований (см. табл. 1 и 2) можно сделать следующие выводы:

1. Плотность графа не оказывает существенного влияния на значение величины *Sum*, т.е. на точность подбора теоретической функции $y(x)$ по методу наименьших квадратов.

2. Предпочтение следует отдать многочлену третьей степени, что согласуется с теоретическими данными. Для алгоритма Флойда сложность оценивается $O(n^3)$ [4,9]. Для алгоритма Беллмана-Форда сложность оценивается $O(n \cdot m)$ [4] для задачи нахождения кратчайших путей от одной фиксированной

вершины. Т.к. мы запускали алгоритм для каждой из n вершин, сложность будет оцениваться $O(n^2 \cdot m)$.

Библиографический список:

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

2. Алексеев П.А., Дубгорный Д.Д., Русских Д.С., Иванова А.П. Сравнительный анализ сложности алгоритмов Флойда, Дейкстры и Беллмана-Форда для графов с различным количеством вершин // Дневник науки. – 2025. – №1 [Электронный ресурс]. (Дата обращения 04.02.2025).

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

4. 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.

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

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

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

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

9. Сэджвик Р. Фундаментальные алгоритмы на C++. Алгоритмы на графах: Пер. с англ. – СПб.: ООО «ДиаСофтЮП», 2002. – 496 с.

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

Оригинальность 80%