ISSN 2225-9309 (Print)
2025. № 3 (34). С. 115–120

Молодежный
Вестник УГАТУ

http://journal.ugatu.su

УДК 519.6

doi 10.54708/22259309_2025_334115

Алгоритм Форда-Беллмана и его реализация на языке программирования C++

Р. А. Мирсаяпов[1]

[1]roman89007@mail.ru

[1]ФГБОУ ВО «Уфимский университет науки и технологий» (УУНиТ)

Аннотация. В данной работе рассматривается решение проблемы поиска кратчайших путей во взвешенных графах с помощью алгоритма Форда-Беллмана, который способен работать с графами, содержащими рёбра отрицательного веса. Представлены теоретическое обоснование алгоритма, доказательство его корректности, а также эффективная реализация на языке программирования C++. Произведёны анализ временной сложности реализации и сравнение с другими алгоритмами.

Ключевые слова*:* алгоритм Беллмана-Форда; кратчайший путь; графы; отрицательные веса; C++.

ВВЕДЕНИЕ

Одной из фундаментальных проблем в теории графов является задача поиска кратчайших расстояний, которая имеет множество практических применений: маршрутизация в компьютерных сетях, логистика, планирование ресурсов и многие другие области. Алгоритм Форда-Беллмана, разработанный независимо Ричардом Беллманом и Лестером Фордом, позволяет найти в ориентированном взвешенном графе кратчайшие пути от одной вершины до всех остальных.

Основное достоинство данного алгоритма состоит в том, что он применяется к графам, содержащим рёбра с отрицательными весами, что обеспечивает ему более широкую область использования по сравнению с алгоритмом Дейкстры. Тем не менее, если в графе присутствуют отрицательные циклы, достижимые из начальной вершины, задача поиска кратчайшего пути становится неразрешимой, поскольку путь можно неограниченно сокращать, многократно проходя через такой цикл. Стоит отметить, что алгоритм Форда-Беллмана позволяет выявлять подобные циклы.

ОПИСАНИЕ АЛГОРИТМА

Алгоритм реализует принцип последовательной релаксации рёбер, постепенно уточняя оценки расстояний. Чтобы выполнить данную процедуру:

1. Выбираем стартовую вершину s.
2. Инициализируем массив расстояний dist[v] и массив родителей parent[v].
3. Повторяем V - 1 раз. Для каждого ребра обновляем массив расстояний.
4. Проверяем на наличие отрицательных циклов: для каждого ребра (u, v) ∈ E: если dist[u] + w(u, v) < dist[v], то в графе существует отрицательный цикл, достижимый из s.

После всех вышеперечисленных действий получаем массив кратчайших расстояний от вершины s до всех вершин v ∈ V или выводим на экран информацию о том, что был найден отрицательный цикл.

Создадим структуры данных для хранения графа и его ребер для реализации алгоритма на языке программирования C++ (рис. 1).

```
struct Edge {
    int start, finish, weight;
};

class Graph {
public:
    vector<Edge> edges;
    int V, E;

    Graph(int V, int E) : V(V), E(E) {}

    void addEdge(int start, int finish, int weight) {
        edges.push_back({start, finish, weight});
    }
};
```

Рис. 1 Структуры данных

Таким образом, полная реализация алгоритма Беллмана-Форда представлена в приложении.

Доказательство корректности алгоритма Форда-Беллмана основывается на следующих положениях:

1. Индуктивное предположение: после i-й итерации внешнего цикла величина dist[v] соответствует длине кратчайшего пути из начальной вершины s в вершину v, содержащего не более i рёбер.

2. В графе без отрицательных циклов максимальная длина кратчайшего пути ограничена |V| - 1 рёбрами, поскольку в простом пути вершины не повторяются. Следовательно, после выполнения |V| - 1 итераций алгоритм гарантированно находит оптимальные расстояния до всех вершин графа.

Существуют следующие оптимизации для алгоритма Форда-Беллмана:

1. Ранняя остановка: если на очередной итерации не произошло ни одного изменения расстояний, алгоритм можно завершить досрочно.

2. Очередь для обработки: можно отслеживать вершины, расстояния до которых изменились, и на следующей итерации рассматривать только исходящие из них рёбра (алгоритм SPFA).

Для реализации SPFA напишем структуру данных, которая будет хранить граф как список смежности, в отличии от алгоритма Беллмана-Форда, где используется список рёбер (рис. 2).

```cpp
struct Edge {
    int start, finish, weight;
};

class Graph {
public:
    vector<vector<Edge>> list;
    int V, E;

    Graph(int V, int E) : V(V), E(E) {
        list.resize(V + 1);
    }

    void addEdge(int start, int finish, int weight) {
        list[start].push_back({start, finish, weight});
    }
};
```

Рис. 2 Структуры данных для SPFA

Реализация SPFA на языке C++ представлена на рис. 3.

```cpp
bool SPFA(const Graph& graph, int s, vector<int>& dist, vector<int>& parent) {
    int V = graph.V;
    dist[s] = 0;

    queue<int> q;
    vector<bool> inQueue(V + 1, false);
    vector<int> count(V + 1, 0); // Счётчик обновлений вершин

    q.push(s);
    inQueue[s] = true;
    count[s] = 1;

    while (!q.empty()) {
        int u = q.front();
        q.pop();
        inQueue[u] = false;
        for (const auto& edge : graph.list[u]) {
            int v = edge.finish;
            int weight = edge.weight;

            if (dist[u] != INT_MAX && dist[v] > dist[u] + weight) {
                dist[v] = dist[u] + weight;
                parent[v] = u;

                if (!inQueue[v]) {
                    q.push(v);
                    inQueue[v] = true;
                    count[v]++;

                    // Проверка на отрицательный цикл
                    if (count[v] > V) {
                        return false;
                    }
                }
            }
        }
    }
    return true;
}
```

Рис. 3 Реализация SPFA

ВЫЧИСЛИТЕЛЬНЫЙ ЭКСПЕРИМЕНТ

В данном разделе представлены результаты вычислительного эксперимента, направленного на сравнение производительности трёх данных алгоритмов:

1. Классический алгоритм Форда-Беллмана.
2. Алгоритм SPFA (Shortest Path Faster Algorithm).
3. Алгоритм Дейкстры с бинарной кучей.

Основной целью исследования было определить наиболее эффективный алгоритм для различных типов графов с положительными весами рёбер (так как Дейкстра не работает с рёбрами отрицательного веса) и проанализировать факторы, влияющие на их производительность.

Эксперимент проводился на трёх классах графов с неотрицательными весами:

1. Разреженные графы.
2. Графы средней плотности.
3. Плотные графы.

Для каждого класса было сгенерировано по 10 случайных ориентированных графов следующих размеров: V = 100, 500, 1000, 2000.

Веса рёбер распределялись случайным образом в диапазоне [1, 100]. Генерация выполнялась с гарантией связности графа.

Все алгоритмы были реализованы на языке C++ с использованием одинаковых структур данных (где это возможно) для обеспечения корректности сравнения. Тестирование выполнялось на компьютере с процессором Intel Core i5-10600 и 16 ГБ оперативной памяти. Для каждого графа алгоритм запускался 3 раза, после чего вычислялось среднее время выполнения.

В табл. 1 представлены средние значения времени выполнения алгоритмов (в миллисекундах) для разных типов графов.

Таблица 1

Время выполнения алгоритмов

Размер графа (V)	Тип графа	Форд-Беллман	SPFA	Дейкстра (бин. куча)
100,00	Разреженный	1,45	0,28	0,14
100,00	Средней плотности	2,87	0,41	0,22
100,00	Плотный	9,73	2,18	1,08
500,00	Разреженный	37,21	2,54	0,92
500,00	Средней плотности	94,56	6,72	2,37
500,00	Плотный	241,85	53,47	28,62
1000,00	Разреженный	147,82	6,34	2,18
1000,00	Средней плотности	432,68	18,75	6,56
1000,00	Плотный	978,53	215,48	112,73
2000,00	Разреженный	594,36	14,67	4,82
2000,00	Средней плотности	1847,29	42,38	15,93
2000,00	Плотный	3956,74	873,62	472,56

Также был проведён сравнительный анализ асимптотической сложности алгоритмов, представленный в табл. 2.

Таблица 2

Сравнительный анализ асимптотической сложности

Алгоритм	Теоретическая сложность	Практическая сложность
Форд-Беллман	O(V·E)	Подтверждается квадратичным ростом для разреженных графов
SPFA	В среднем O(E), худший случай O(V·E)	Близка к O(E·log V) в большинстве случаев
Дейкстра (бин. куча)	O((V+E)·log V)	Близка к O(E·log V) для разреженных графов

ЗАКЛЮЧЕНИЕ

Результаты вычислительного эксперимента позволяют нам сделать следующие выводы относительно выбора алгоритмов поиска кратчайших путей в графах:

1. Алгоритм «Дейкстра с бинарной кучей» демонстрирует наивысшую производительность на графах с неотрицательными весами рёбер, что делает его предпочтительным вариантом в таких условиях.
2. В качестве универсального алгоритма, способного обрабатывать графы с любыми весами (в том числе с отрицательными) и находить кратчайшее расстояние, целесообразно рассматривать алгоритм SPFA.
3. Классический алгоритм Форда-Беллмана менее эффективен по сравнению с альтернативами, поэтому его применение оправдано лишь в ситуациях, где приоритетом является простота реализации.

Таким образом, для графов с неотрицательными рёбрами оптимальным выбором остаётся алгоритм Дейкстры, тогда как в остальных случаях предпочтение следует отдавать SPFA, превосходящему по скорости алгоритм Форда-Беллмана.

ПРИЛОЖЕНИЕ

```
bool BellmanFord(const Graph& graph, int s, vector<int>& dist, vector<int>& parent) {
   int V = graph.V;
   int E = graph.E;
   // Инициализация расстояний
   dist.assign(V + 1, INT_MAX);
   parent.assign(V + 1, -1);
   dist[s] = 0;
   // Релаксация всех рёбер |V|-1 раз
   for (int i = 1; i <= V - 1; ++i) {
      for (const auto& edge : graph.edges) {
         int u = edge.start;
         int v = edge.finish;
         int weight = edge.weight;
         if (dist[u] != INT_MAX && dist[v] > dist[u] + weight) {
            dist[v] = dist[u] + weight;
            parent[v] = u;
         }
      }
   }
   // Проверка на отрицательные циклы
```

```
    for (const auto& edge : graph.edges) {
        int u = edge.start;
        int v = edge.finish;
        int weight = edge.weight;
        if (dist[u] != INT_MAX && dist[v] > dist[u] + weight) {
            return false;
        }
    }
    return true;
}
```

СПИСОК ЛИТЕРАТУРЫ

1. Беллман Р. Динамическое программирование / Р. Беллман. М.: ИЛ, 1960. 400 с.
2. Форд Л. Р. Потоки в сетях / Л. Р. Форд, Д. Р. Фалкерсон. М.: Мир, 1966. 276 с.
3. Седжвик Р. Фундаментальные алгоритмы на C++ / Р. Седжвик. М.: Вильямс, 2011. 1056 с.
4. Дасгупта С. Алгоритмы / С. Дасгупта, Х. Пападимитриу, У. Вазирани. М.: МЦНМО, 2014. 320 с.
5. Дискретная математика: Учебное пособие / [С. С. Поречный и др.]; Уфимск. гос. авиац. техн. ун-т. Уфа: РИК УГАТУ, 2019. 400 с.

ОБ АВТОРАХ

МИРСАЯПОВ Роман Амирович, студент ПРО ИИМРТ УУНиТ.

METADATA

Title: The Ford-Bellman algorithm and its implementation in the C++ programming language

Authors: R. A. Mirsayapov

Affiliation:

Ufa University of Science and Technology (UUST), Russia.

Email: roman89007@mail.ru

Language: Russian.

Source Molodezhnyj Vestnik UGATU (scientific journal of Ufa University of Science and Technology), no. 3 (34), pp. 115-120, 2025. ISSN 2225-9309 (Print).

Abstract: In this paper, we consider the Ford-Bellman algorithm for finding shortest paths in weighted graphs, which is capable of working with graphs containing edges of negative weight. A theoretical justification of the algorithm, proof of its correctness, as well as an effective implementation in the C++ programming language are presented. The time complexity of the implementation is analyzed.

Key words: Bellman-Ford algorithm, shortest path, graphs, negative weights, C++

About authors:

MIRSAYAPOV, Roman Amirovich, student PRO IIMRT UUST.