Министерство образования и науки Российской Федерации

Федеральное государственное автономное образовательное учреждение высшего образования

«Уральский федеральный университет имени первого Президента России Б.Н. Ельцина» (УрФУ)

Институт радиоэлектроники и информационных технологий – РтФ

Школа профессионального и академического образования

Оценка \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

Руководитель: Мирвода С.В.

Дата защиты: \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

**АЛГОРИТМ БЕЛЛМАНА-ФОРДА**

Проект по модулю "Алгоритмы обработки данных во внешней памяти"

Студенты: Велимович М.В.

Петров В. А.

(ФИО) (Подпись)

Группа: РИМ-171226

Екатеринбург

2018

**ОГЛАВЛЕНИЕ**

[КРАТКОЕ ОПИСАНИЕ АЛГОРИТМА ДЕЙКСТРЫ 3](#_Toc516608085)

[ОПИСАНИЕ АЛГОРИТМА БЕЛЛМАНА-ФОРДА 6](#_Toc516608086)

[Оценка сложности 8](#_Toc516608087)

[ОТЛИЧИЯ ОТ АЛГОРИТМА ДЕЙКСТРЫ 9](#_Toc516608088)

[СПИСОК ЛИТЕРАТУРЫ 10](#_Toc516608089)

[ПРИЛОЖЕНИЕ А 11](#_Toc516608090)

КРАТКОЕ ОПИСАНИЕ АЛГОРИТМА ДЕЙКСТРЫ

Алгоритм Дейкстры — [алгоритм](https://ru.wikipedia.org/wiki/%D0%90%D0%BB%D0%B3%D0%BE%D1%80%D0%B8%D1%82%D0%BC) на [графах](https://ru.wikipedia.org/wiki/%D0%93%D1%80%D0%B0%D1%84_(%D0%BC%D0%B0%D1%82%D0%B5%D0%BC%D0%B0%D1%82%D0%B8%D0%BA%D0%B0)), изобретённый нидерландским учёным [Эдсгером Дейкстрой](https://ru.wikipedia.org/wiki/%D0%94%D0%B5%D0%B9%D0%BA%D1%81%D1%82%D1%80%D0%B0,_%D0%AD%D0%B4%D1%81%D0%B3%D0%B5%D1%80_%D0%92%D0%B8%D0%B1%D0%B5" \o "Дейкстра, Эдсгер Вибе) в [1959 году](https://ru.wikipedia.org/wiki/1959_%D0%B3%D0%BE%D0%B4). Находит кратчайшие пути от одной из вершин графа до всех остальных. Алгоритм работает только для графов без [рёбер](https://ru.wikipedia.org/wiki/%D0%A0%D0%B5%D0%B1%D1%80%D0%BE_(%D1%82%D0%B5%D0%BE%D1%80%D0%B8%D1%8F_%D0%B3%D1%80%D0%B0%D1%84%D0%BE%D0%B2)) отрицательного [веса](https://ru.wikipedia.org/wiki/%D0%A1%D0%BB%D0%BE%D0%B2%D0%B0%D1%80%D1%8C_%D1%82%D0%B5%D1%80%D0%BC%D0%B8%D0%BD%D0%BE%D0%B2_%D1%82%D0%B5%D0%BE%D1%80%D0%B8%D0%B8_%D0%B3%D1%80%D0%B0%D1%84%D0%BE%D0%B2#%D0%92).

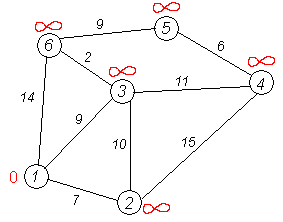
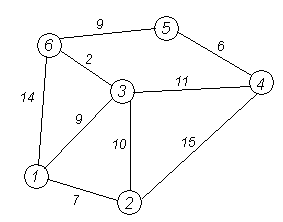
Каждой вершине из *V* сопоставим метку — минимальное известное расстояние от этой вершины до *a*. Алгоритм работает пошагово — на каждом шаге он «посещает» одну вершину и пытается уменьшать метки. Работа алгоритма завершается, когда все вершины посещены.

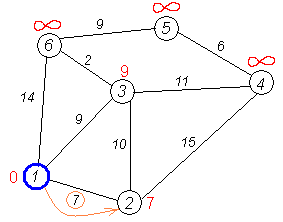
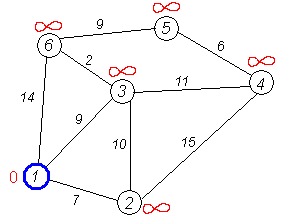
1. Инициализация.

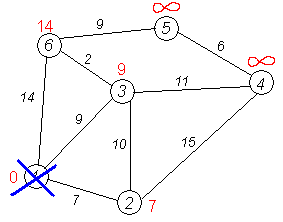
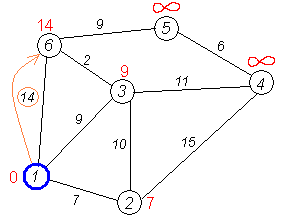
Метка самой вершины *a* полагается равной 0, метки остальных вершин — бесконечности. Это отражает то, что расстояния от *a* до других вершин пока неизвестны. Все вершины графа помечаются как непосещённые.

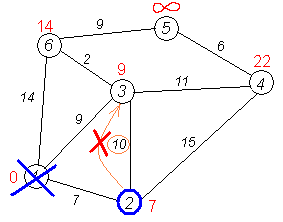
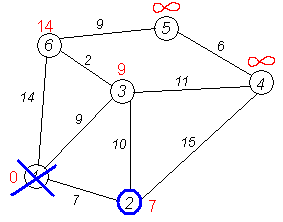
1. Шаг алгоритма.

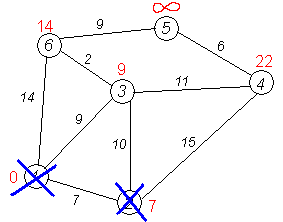
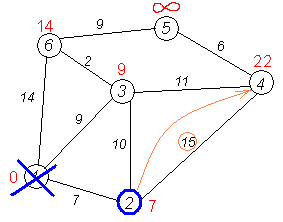
Если все вершины посещены, алгоритм завершается. В противном случае, из ещё не посещённых вершин выбирается вершина *u*, имеющая минимальную метку. Мы рассматриваем всевозможные маршруты, в которых *u* является предпоследним пунктом. Вершины, в которые ведут рёбра из *u*, назовём *соседями* этой вершины. Для каждого соседа вершины *u*, кроме отмеченных как посещённые, рассмотрим новую длину пути, равную сумме значений текущей метки *u* и длины ребра, соединяющего *u* с этим соседом. Если полученное значение длины меньше значения метки соседа, заменим значение метки полученным значением длины. Рассмотрев всех соседей, пометим вершину *u* как посещённую и повторим [шаг алгоритма](https://ru.wikipedia.org/wiki/%D0%90%D0%BB%D0%B3%D0%BE%D1%80%D0%B8%D1%82%D0%BC_%D0%94%D0%B5%D0%B9%D0%BA%D1%81%D1%82%D1%80%D1%8B#%D0%A8%D0%B0%D0%B3).

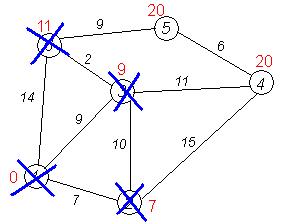
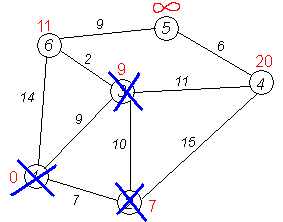












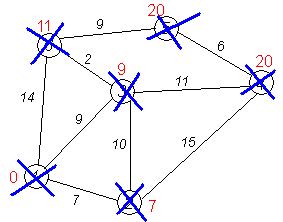
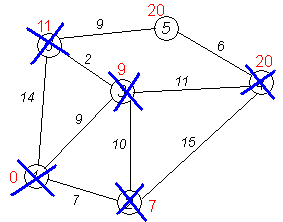


Рис. 1 – Демонстрация работы алгоритма Дейкстры

# ОПИСАНИЕ АЛГОРИТМА БЕЛЛМАНА-ФОРДА

Алгоритм Беллмана-Форда — алгоритм поиска кратчайшего [пути](https://ru.wikipedia.org/wiki/%D0%9F%D1%83%D1%82%D1%8C_(%D1%82%D0%B5%D0%BE%D1%80%D0%B8%D1%8F_%D0%B3%D1%80%D0%B0%D1%84%D0%BE%D0%B2)) во [взвешенном графе](https://ru.wikipedia.org/wiki/%D0%92%D0%B7%D0%B2%D0%B5%D1%88%D0%B5%D0%BD%D0%BD%D1%8B%D0%B9_%D0%B3%D1%80%D0%B0%D1%84). За время *O(|V| × |E|)* алгоритм находит кратчайшие пути от одной [вершины](https://ru.wikipedia.org/wiki/%D0%92%D0%B5%D1%80%D1%88%D0%B8%D0%BD%D0%B0_(%D0%B3%D1%80%D0%B0%D1%84)) графа до всех остальных. В отличие от [алгоритма Дейкстры](https://ru.wikipedia.org/wiki/%D0%90%D0%BB%D0%B3%D0%BE%D1%80%D0%B8%D1%82%D0%BC_%D0%94%D0%B5%D0%B9%D0%BA%D1%81%D1%82%D1%80%D1%8B), алгоритм Беллмана — Форда допускает [рёбра](https://ru.wikipedia.org/wiki/%D0%A0%D0%B5%D0%B1%D1%80%D0%BE_(%D1%82%D0%B5%D0%BE%D1%80%D0%B8%D1%8F_%D0%B3%D1%80%D0%B0%D1%84%D0%BE%D0%B2)) с отрицательным [весом](https://ru.wikipedia.org/wiki/%D0%92%D0%B5%D1%81_(%D1%82%D0%B5%D0%BE%D1%80%D0%B8%D1%8F_%D0%B3%D1%80%D0%B0%D1%84%D0%BE%D0%B2)). Предложен независимо [Ричардом Беллманом](https://ru.wikipedia.org/wiki/%D0%91%D0%B5%D0%BB%D0%BB%D0%BC%D0%B0%D0%BD,_%D0%A0%D0%B8%D1%87%D0%B0%D1%80%D0%B4) и [Лестером Фордом](https://ru.wikipedia.org/wiki/%D0%A4%D0%BE%D1%80%D0%B4,_%D0%9B%D0%B5%D1%81%D1%82%D0%B5%D1%80).

* V – множество вершин графа
* E – множество рёбер графа
* w[ij] – вес (длина) ребра ij
* a – вершина, расстояния от которой ищутся
* U – множество посещенных вершин
* d[u] – по окончании работы алгоритма равно длине кратчайшего пути из a до вершины u
* p[u] – по окончании работы алгоритма содержит кратчайший путь из а в u
* v – вершина графа

1. Формулировка задачи.

Дан ориентированный или неориентированный [граф](https://ru.wikipedia.org/wiki/%D0%93%D1%80%D0%B0%D1%84_(%D0%BC%D0%B0%D1%82%D0%B5%D0%BC%D0%B0%D1%82%D0%B8%D0%BA%D0%B0)) *G* со взвешенными рёбрами. Длиной пути назовём сумму весов рёбер, входящих в этот путь. Требуется найти кратчайшие пути от выделенной вершины *s* до всех вершин графа.

Заметим, что кратчайших путей может не существовать. Так, в графе, содержащем цикл с отрицательным суммарным весом, существует сколь угодно короткий путь от одной вершины этого цикла до другой (каждый обход цикла уменьшает длину пути). Цикл, сумма весов рёбер которого отрицательна, называется *отрицательным циклом*.

**Алгоритм Беллмана-Форда**позволяет решить задачу о кратчайшем пути из одной вершины в общем случае, когда вес каждого из ребер может быть отрицательным. Для заданного взвешенного ориентированного графа G = (V, Е) с истоком s и весовой функцией w : Е —» R алгоритм Беллмана-Форда возвращает логическое значение, указывающее на то, содержится ли в графе цикл с отрицательным весом, достижимый из истока. Если такой цикл существует, в алгоритме указывается, что решения не существует. Если же таких циклов нет, алгоритм выдает кратчайшие пути и их вес.

В этом алгоритме используется ослабление, в результате которого величина d[v], представляющая собой оценку веса кратчайшего пути из истока s к каждой из вершин v є V, уменьшается до тех пор, пока она не станет равна фактическому весу кратчайшего пути из s в v. Значение TRUE возвращается алгоритмом тогда и только тогда, когда граф не содержит циклов с отрицательным весом, достижимых из истока.

1. Формальное описание (псевдокод).

**Bellman\_Ford(G, w, s)**

1 Initialize\_Single\_Source(G, s)

2 **for** i «- l to |V[G]|-1

3 **do for** (для) каждого ребра (u, v) є E[G]

4 **do** RELAX(u,v,w)

5 **for** (для) каждого ребра (u, v) є E[G]

6 **do if** d[v] > d[u] + w(u, v)

7 **then return** FALSE

8 **return** TRUE

После инициализации в строке 1 всех значений d и prev, алгоритм осуществляет |V| — 1 проходов по ребрам графа. Каждый проход соответствует одной итерации цикла for в строках 2-4 и состоит из однократного ослабления каждого ребра графа. После |V| — 1 проходов в строках 5-8 проверяется наличие цикла с отрицательным весом и возвращается соответствующее булево значение.

**Оценка сложности**

Алгоритм Беллмана-Форда завершает свою работу в течение времени **O(V\*E)**, поскольку инициализация в строке 1 занимает время O(V), на каждый из |V| — 1 проходов по ребрам в строках 2-4 требуется время в O(E), а на выполнение цикла for в строках 5-7 — время O(Е).

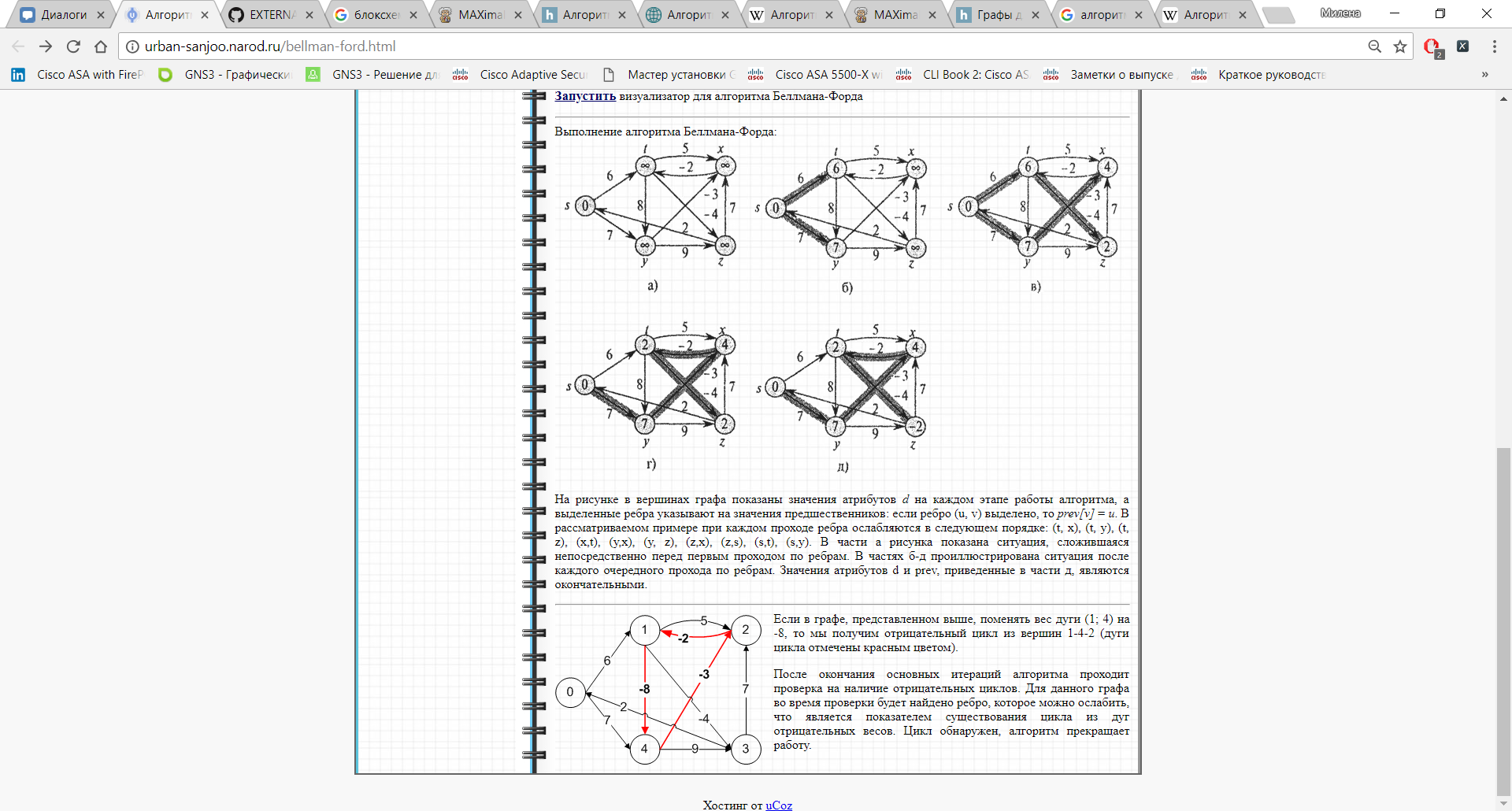


Рис. 2 – Демонстрация работы алгоритма Беллмана-Форда

На рисунке в вершинах графа показаны значения атрибутов d на каждом этапе работы алгоритма, а выделенные ребра указывают на значения предшественников: если ребро (u, v) выделено, то prev[v]=u. В рассматриваемом примере при каждом проходеребра ослабляются в следующем порядке: (t, x), (t, y), (t, z), (x, t), (y, x), (y, z), (z, x), (z, s), (s, t), (s, y). В части а рисунка показана ситуация, сложившаяся непосредственно перед первым проходом по ребрам. В частях б-д происллюстрирована ситуация после каждого очередного прохода по ребрам. Значения атрибутов d и prev, приведенные в части д, являются окончательными.

# ОТЛИЧИЯ ОТ АЛГОРИТМА ДЕЙКСТРЫ

Граф с отрицательными циклами

Алгоритм Беллмана-Форда позволяет очень просто определить, существует ли в графе *G* отрицательный цикл, достижимый из вершины *s*. Достаточно произвести внешнюю итерацию цикла не {\displaystyle |V|-1}|V| - 1, a ровно {\displaystyle |V|}|V| раз. Если при исполнении последней итерации длина кратчайшего пути до какой-либо вершины строго уменьшилась, то в графе есть отрицательный цикл, достижимый из *s*. На основе этого можно предложить следующую оптимизацию: отслеживать изменения в графе и, как только они закончатся, сделать выход из цикла (дальнейшие итерации будут бессмысленны).

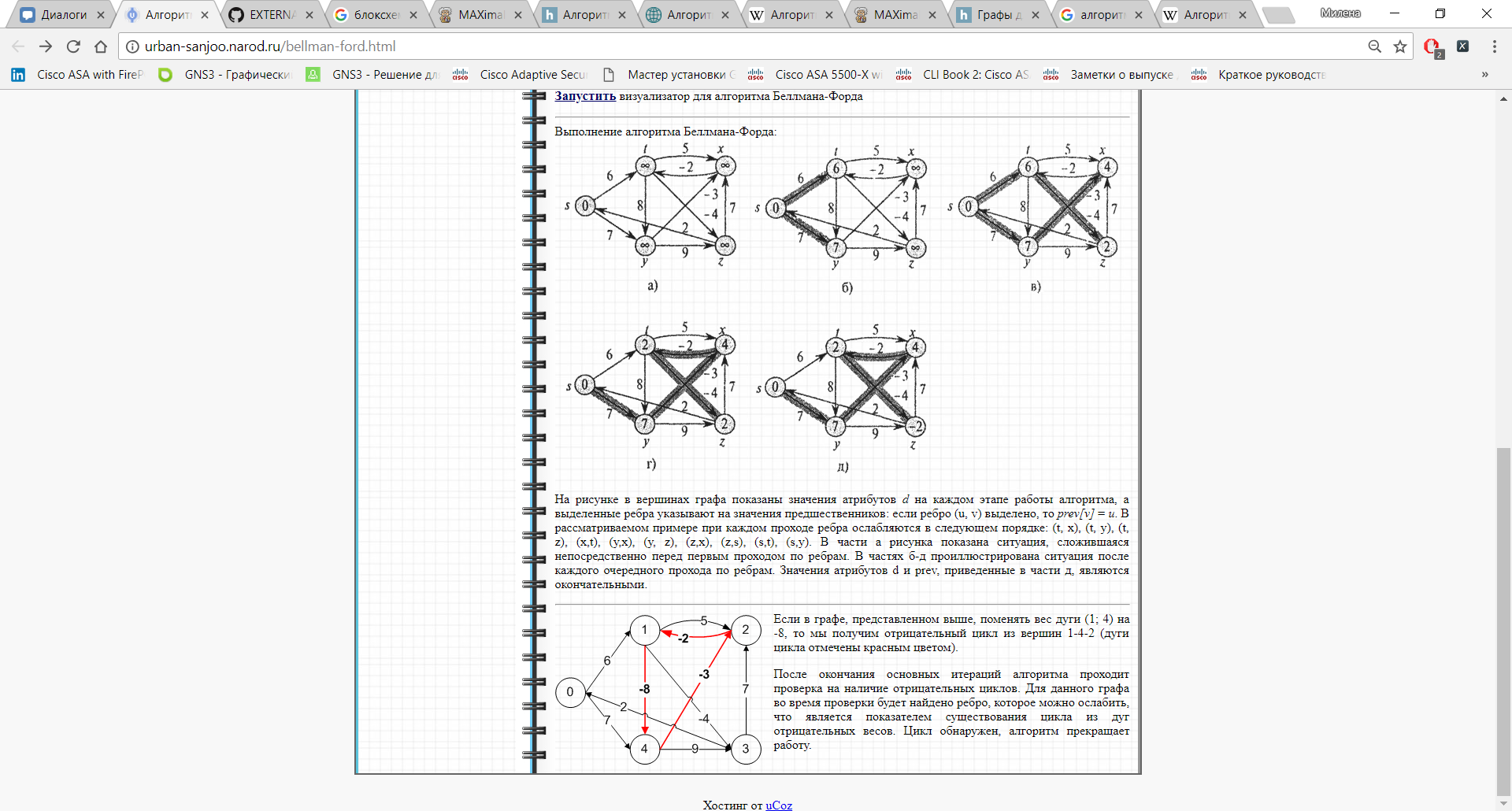


Рис. 3 – Демонстрация работы алгоритма Беллмана-Форда для графа с отрицательными весами

Если в графе, представленном выше на рис.2 (рис. 3), поменять вес дуги (1; 4) на -8, то мы получим отрицательный цикл из вершин 1 – 4 – 2 (дуги цикла отмечены красным цветом).

После окончания основных итераций алгоритма проходит проверка на наличие отрицательных циклов. Для данного графа во время проверки будет найдено ребро, которое можно ослабить, что является показателем существования цикла из дуг отрицательных весов. Цикл обнаружен, алгоритм прекращает работу.

# СПИСОК ЛИТЕРАТУРЫ

1. Портал искусственного интеллекта // Электронный ресурс // https://intellect.ml/algoritm-bellmana-forda-4397
2. Визуализированные структуры данных и алгоритмы // Электронный ресурс // <https://visualgo.net/ru/sssp>
3. Библиотека алгоритмов на графах // Электронный ресурс // <http://urban-sanjoo.narod.ru/bellman-ford.html#example>
4. MAXimal // Нахождение кратчайших путей от заданной вершины до всех остальных вершин алгоритмом Дейкстры // Электронный ресурс // <http://e-maxx.ru/algo/dijkstra>
5. Хабр // Алгоритм Дейкстры. Поиск оптимальных маршрутов на графе // Электронный ресурс // <https://habr.com/post/111361/>
6. Хабр // Графы для самых маленьких: Ford & Bellman или как понять, что ты попал в бесконечно далекое прошлое // Электронный ресурс // <https://habr.com/post/201588/>
7. Kvodo // Электронный ресурс // <http://kvodo.ru/bellman-ford-algorithm.html>

# ПРИЛОЖЕНИЕ А

using System;

using System.Collections.Generic;

using System.Linq;

namespace BellmanFord

{

class Path

{

public string From;

public string To;

public int Cost;

public Path(string from, string to, int cost)

{

From = from;

To = to;

Cost = cost;

}

}

class Program

{

static string[] vertices = { "S", "A", "B", "C", "D", "E" };

static List<Path> graph = new List<Path>()

{

new Path("S", "A", 4),

new Path("S", "E", -5),

new Path("A", "C", 6),

new Path("B", "A", 3),

new Path("C", "B", -2),

new Path("D", "C", 3),

new Path("D", "A", 10),

new Path("E", "D", 8)

};

static Dictionary<string, int> memo = new Dictionary<string, int>()

{

{ "S", 0 },

{ "A", int.MaxValue },

{ "B", int.MaxValue },

{ "C", int.MaxValue },

{ "D", int.MaxValue },

{ "E", int.MaxValue }

};

static bool Iterate()

{

bool doItAgain = false;

foreach (var fromVertex in vertices)

{

Path[] edges = graph.Where(x => x.From == fromVertex).ToArray();

foreach (var edge in edges)

{

int potentialCost = memo[edge.From] == int.MaxValue ? int.MaxValue : memo[edge.From] + edge.Cost;

if (potentialCost < memo[edge.To])

{

memo[edge.To] = potentialCost;

doItAgain = true;

}

}

}

return doItAgain;

}

static void Main()

{

for (int i = 0; i < vertices.Length; i++)

{

if (!Iterate())

break;

}

foreach (var keyValue in memo)

{

Console.WriteLine($"{keyValue.Key}: {keyValue.Value}");

}

Console.ReadLine();

}

}

}