

Министерство образования Республики Беларусь  
Учреждение образования  
БЕЛОРУССКИЙ ГОСУДАРСТВЕННЫЙ УНИВЕРСИТЕТ  
ИНФОРМАТИКИ И РАДИОЭЛЕКТРОНИКИ

Факультет компьютерных систем и сетей  
Кафедра электронных вычислительных машин

Отчет  
по учебной практике (ознакомительной)

Студент гр.150505  
Власов Р.Е.

Руководитель:  
кандидат технических наук, доцент  
Селезнёв И.Л.

МИНСК 2022

## СОДЕРЖАНИЕ

ВВЕДЕНИЕ.....	3
1 ОБЗОР ИСТОЧНИКОВ ПО ТЕМЕ ЗАДАНИЯ.....	5
2 МЕСТО ПРОВЕДЕНИЯ ПРАКТИКИ.....	6
2.1 СФЕРА ДЕЯТЕЛЬНОСТИ И ПРОИЗВОДСТВО.....	6
2.2 ОРГАНИЗАЦИОННАЯ СТРУКТУРА.....	7
3 СУТЬ АЛГОРИТМА.....	11
4 БЛОК-СХЕМА И РЕАЛИЗАЦИЯ ПРОГРАММЫ.....	12
ЗАКЛЮЧЕНИЕ.....	19

## ВВЕДЕНИЕ

Тема работы: реализация алгоритма по нахождению кратчайшего пути с помощью языка программирования СИ.

Алгоритм Дейкстры является алгоритмом кратчайшего пути из одной вершины в другую и решает проблему кратчайшего пути в ориентированных графах.

Алгоритм широко применяется в программировании, например, его используют протоколы маршрутизации OSPF и IS-IS.

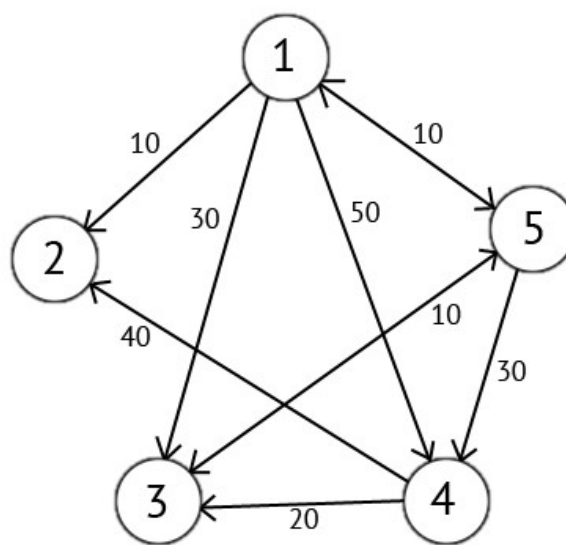


Рисунок 1.1 – Пример ориентированного графа

Этот граф можно представить в виде матрицы:

	1	2	3	4	5
1		10	30	50	10
2					
3					10
4		40	20		
5	10		10	30	

Рисунок 1.2 – Граф, представленный матрицей

Берется в качестве источника вершина 1. Это значит что будет идти поиск кратчайших маршрутов из вершины 1 в вершины 2, 3, 4 и 5.

Данный алгоритм пошагово перебирает все вершины графа и назначает им метки, которые являются известным минимальным расстоянием от вершины источника до конкретной вершины.

## **1 ОБЗОР ИСТОЧНИКОВ ПО ТЕМЕ ЗАДАНИЯ**

В ходе разработки были использованы такие источники как:

- различные веб-ресурсы для IT-специалистов, такие как Habr, Stackoverflow, GitHub и т.д.;
- книга Архитектура компьютера, Э.Тоненбаум
- Электронный ресурс: <https://habr.com/ru/post/111361/> Алгоритм Дейкстры. Поиск оптимальных маршрутов на графе:

## **2 МЕСТО ПРОВЕДЕНИЯ ПРАКТИКИ**

### **2.1 СФЕРА ДЕЯТЕЛЬНОСТИ И ПРОИЗВОДСТВО**

Отсчет своей истории Белорусский государственный институт стандартизации и сертификации (БелГИСС) ведет с 1972 года, когда свою деятельность начал Минский филиал Центрального конструкторского бюро «Эталон». Пройдя структурные преобразования, в 1992 году предприятие получило статус института.

Через пять лет решением Правительства, институт передается в ведение Государственного комитета по стандартизации Республики Беларусь (Госстандарта), что стало важнейшей вехой в истории БелГИСС.

Сегодня институт является центральным государственным научно-практическим предприятием Госстандарта в области технического нормирования, стандартизации, оценки соответствия, информационно-технического обеспечения и системного менеджмента.

В соответствии с постановлением Госстандарта от 10 июля 2017 г. №58 БелГИСС определен в качестве национального института по стандартизации.

Успешно реализуемые государственные направления и проекты позволяют, с одной стороны, поддерживать уникальность института в республике и за ее пределами, а с другой – находиться в конкурентной среде, постоянно улучшаться, искать и предлагать новые решения, динамично продвигаться к новым горизонтам.

Обеспечение высокого научно-технического и организационно-методического уровня выполняемых работ и оказываемых услуг – главная задача института, в выполнение которой вкладывают энергию, знания, опыт и профессионализм более 360 специалистов.

Изучив устав, можно сказать, что миссией предприятия является генерация и распространение новых знаний в целях оказания помощи бизнесу, обществу и государству быть безопаснее, эффективнее и лучше путем создания условий для:

- снижения технических барьеров;
- повышения качества и конкурентоспособности отечественной продукции;
- содействия защите интересов потребителей;
- внедрения инноваций, используя инструменты технического нормирования, стандартизации, оценки соответствия, информационного обеспечения и системного менеджмента.

## **2.2 ОРГАНИЗАЦИОННАЯ СТРУКТУРА**

Белорусский государственный институт стандартизации и сертификации (БелГИСС) находится в подчинении Государственного комитета по стандартизации Республики Беларусь и является центральным государственным научно-практическим предприятием Госстандарта в области технического нормирования и стандартизации, оценки соответствия и систем менеджмента. В настоящее время директором института является Александр Скуратов. Организационная структура БелГИСС представлена на рисунке 1.1:

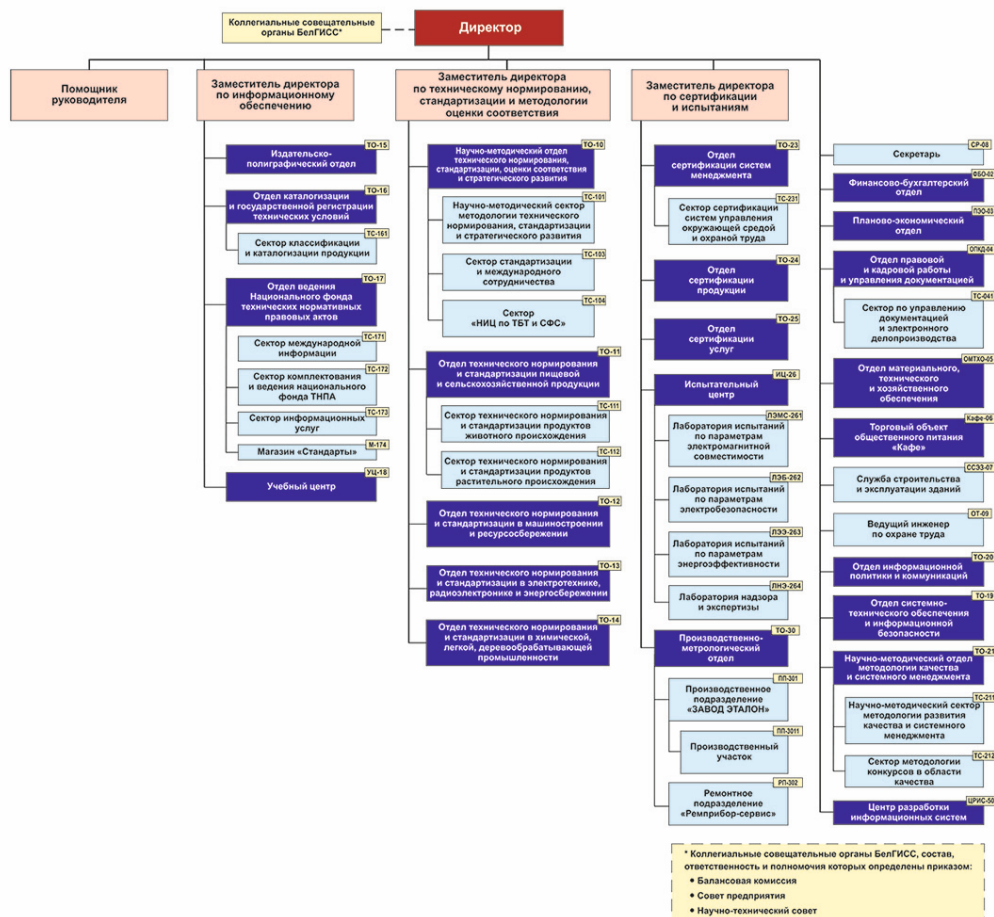


Рис. 2.1– Организационная структура БелГИСС

Рассмотрим структуру производственно-метрологический отдела (ТО-30) и сферу его деятельности.

Руководство деятельностью ТО-30 осуществляет начальник ТО-30, который подчиняется непосредственно заместителю директора по сертификации и испытаниям.

На должность начальника ТО-30 назначается лицо, имеющее высшее профессиональное образование и стаж работы по одной и более специальностям, соответствующим основным видам деятельности ТО-30 на руководящих должностях не менее 4 лет.

Организационную структуру и штатную численность ТО-30 утверждает директор института в установленном порядке.

ТО-30 имеет внутреннюю организационную структуру представленную в таблице 2.1:



Таблица 2.1 – Организационная структура отдела ТО-30

<b>Начальник ТО-30</b>	
<b>Заместитель начальника ТО-30</b>	
Производственно-метрологический отдел (ТО-30)	
<b>Начальник ПП-301</b>	<b>Начальник РП-302</b>
Производственное подразделение «ЗАВОД ЭТАЛОН»	Ремонтное подразделение «Ремприбор-сервис»
<b>Начальник ПП-3011</b>	
Производственный участок	

в составе:

- производственное подразделение «ЗАВОД ЭТАЛОН» (ПП-301);
- ремонтное подразделение «Ремприбор-сервис» (РП-302).

Руководство деятельностью ПП-301 и РП-302 осуществляют начальники соответствующего подразделения, которые непосредственно подчиняются начальнику ТО-30. Начальник подразделения назначается и освобождается от занимаемой должности приказом директора института по представлению начальника ТО-30, согласованному с заместителем директора по сертификации и испытаниям.

ТО-30 включает должности: начальника ТО-30, заместителя начальника ТО-30, начальника подразделения ПП-301, начальник РП-302, ведущего инженера, инженеров, техника-метролога и рабочих.

«Завод Эталон» — производственное подразделение БелГИСС по выпуску продуктов безопасности: арочных и ручных металлодетекторов, а также весоизмерительной техники промышленного бытового назначения. На предприятии освоен широкий ассортимент электронных весов, предназначенных для оснащения торговых, промышленных, агропромышленных предприятий. «Завод Эталон» — производственное подразделение БелГИСС по выпуску продуктов безопасности: арочных и ручных металлодетекторов, а также весоизмерительной техники

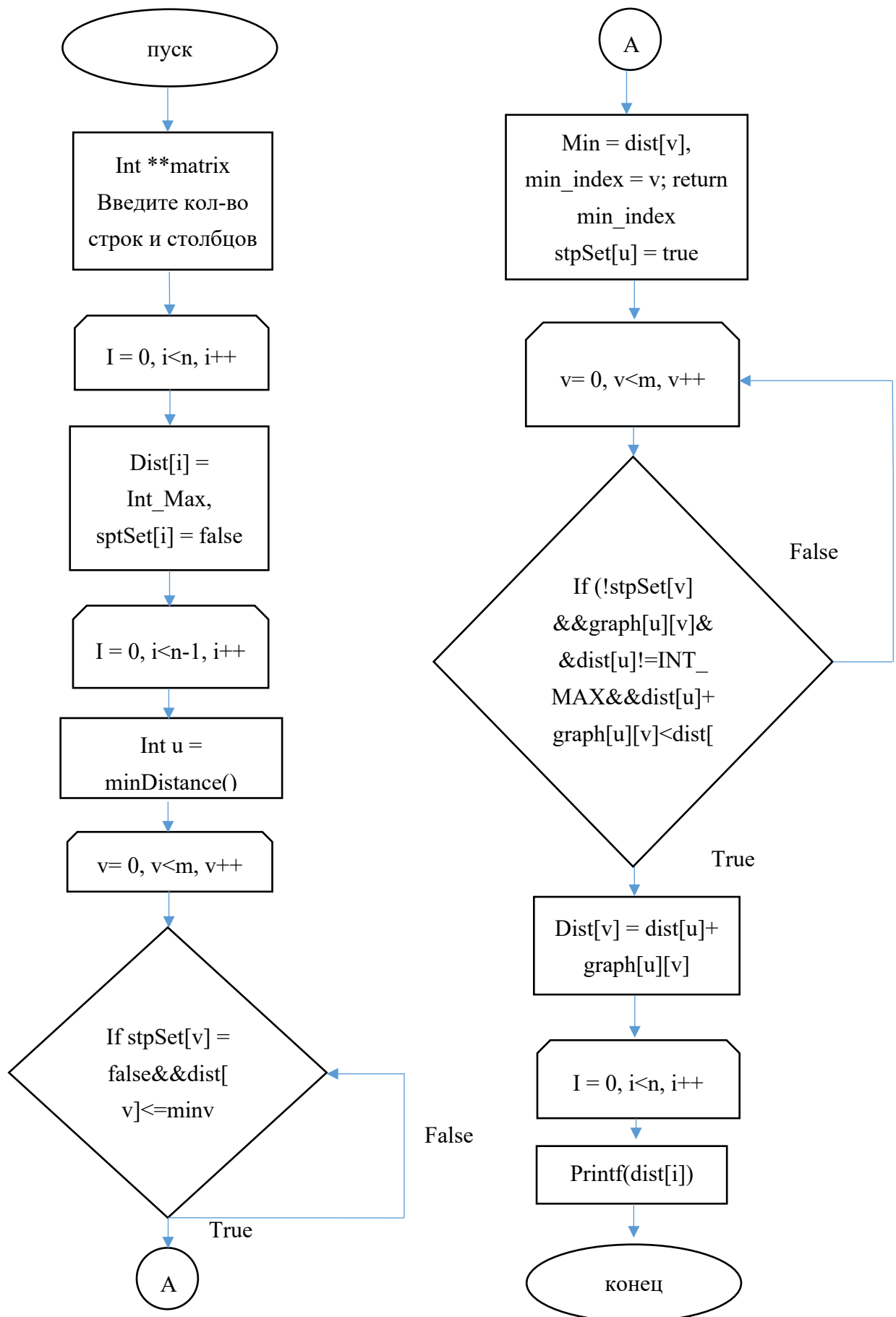
промышленного бытового назначения. На предприятии освоено широкий ассортимент электронных весов, предназначенных для оснащения торговых, промышленных, агропромышленных предприятий. Институт занимается услугами ручного монтажа электронных компонентов, сборкой радиоэлектронных изделий и электротехнической продукции из давальческих комплектующих, ремонтом и поверкой средств измерений. Так же БелГИСС предоставляет множество услуг по следующим направлениям деятельности:

- техническое нормирование и стандартизация;
- информационное обеспечение;
- сертификация и декларирование соответствия;
- испытания.

### 3 СУТЬ АЛГОРИТМА

Алгоритм включает в себя функцию реализующую алгоритм кратчайшего пути из одной вершины для графа, представленного матрицей. Создается выходной массив, который будет содержать кратчайший путь. Инициализируются все расстояния как бесконечность и `stpSet[]` как `false`. Расстояние исходной вершины от самой себя всегда равно 0. Затем происходит поиск кратчайшего пути для всех вершин. Выбирается вершина минимального расстояния из набора еще не обработанных вершин. Запускается функция нахождения минимального расстояния, при нахождении таковой, функция возвращает этот индекс минимального элемента. Помечается выбранную вершину как обработанная. Обновление значения расстояния смежных вершин выбранной вершины. Выполняется условие обновления `dist[v]` только в том случае, если его нет в `sptSet`, есть ребро от `u` до `v` и общий вес пути от `src` до `v` через `u` меньше текущего значения `dist[v]`. После всех итераций запускается функция вывода построенного массива расстояний.

#### 4 БЛОК-СХЕМА И РЕАЛИЗАЦИЯ ПРОГРАММЫ



## Рисунок 4.1 – Блок-схема алгоритма

Далее представлен код выполненной программы

```
#include <limits.h>
#include <stdlib.h>
#include <stdio.h>
#include <stdbool.h>

int minDistance(int m, int dist[], bool sptSet[]) //функция
нахождения минимального расстояния
{
    // Инициализация минимального значения
    int min = INT_MAX, min_index = 0;

    for (int v = 0; v < m; v++)
        if (sptSet[v] == false && dist[v] <= min)
            (min = dist[v]), min_index = v;

    return min_index;
}

// Вспомогательная функция для печати построенного массива
расстояний
void printSolution(int m, int dist[])
{
    printf("Vertex \t\t Distance from Source\n");

    char *x = "ABCDE";
    for (int i = 0; i < m; i++)
        printf("A-%c \t\t %d\n", x[i], dist[i]);
}

// Функция, реализующая алгоритм кратчайшего пути из одной
вершины.
// для графа, представленного матрицей
void dijkstra(int n, int m, int **graph, int src)
```

```

{
    int dist[n]; // выходной массив. dist[i] будет содержать
кратчайший путь
    // расстояние от источника до i

    bool sptSet[n]; /*sptSet[i] будет истинным, если вершина i
включена в дерево
    кратчайших путей или кратчайшее расстояние от src до i
завершено*/

    // Инициализация все расстояния как бесконечность и sptSet[]
как false
    for (int i = 0; i < n; i++)
        dist[i] = INT_MAX, sptSet[i] = false;

    // Расстояние исходной вершины от самой себя всегда равно 0
    dist[src] = 0;

    // Поиск кратчайшего пути для всех вершин
    for (int count = 0; count < n - 1; count++) {
        // Выбираем вершину минимального расстояния из набора еще
не обработанных вершин. u всегда равно src в первой итерации.
        int u = minDistance(m, dist, sptSet);

        // Помечается выбранную вершину как обработанную
        sptSet[u] = true;
        // Обновление значения расстояния смежных вершин
выбранной вершины.
        for (int v = 0; v < m; v++)

            /* Обновляется dist[v] только в том случае, если его
нет в sptSet, есть ребро от u до v и общий вес пути от src до v
через u меньше текущего значения dist[v]*/
            if (!sptSet[v] && graph[u][v] && dist[u] != INT_MAX
                && dist[u] + graph[u][v] < dist[v])
                dist[v] = dist[u] + graph[u][v];
    }
}

```

```

        // вывод построенного массива расстояний
        printSolution(m, dist);
        /*{0,2,1000000,2,1000000},
           {2,0,4,1000000,1000000},
           {1000000,4,0,2,1},
           {2,1000000,2,0,7},
           {1000000,1000000,1,7,0}*/
    }

    // A-0,B-1,C-2,D-3,E-4
    int main() {
        int n, m;
        int **matrix; // указатель на указатель на строку элементов
        printf("Введите количество строк: ");
        scanf("%d", &n);
        printf("Введите количество столбцов: ");
        scanf("%d", &m);
        // Выделение памяти под указатели на строки
        matrix = (int**)malloc(n * sizeof(int*));
        // Ввод элементов массива
        for (int i = 0; i<n; i++) // цикл по строкам
        {
            // Выделение памяти под хранение строк
            matrix[i] = (int*)malloc(m * sizeof(int));
            for (int j = 0; j<m; j++) // цикл по столбцам
            {
                printf("matrix[%d][%d] = ", i, j);
                scanf("%d", &matrix[i][j]);
            }
        }
        // Вывод элементов массива
        for (int i = 0; i < n; i++) // цикл по строкам
        {
            for (int j = 0; j < m; j++) // цикл по столбцам
            {

```

```

        printf("%5d ", matrix[i][j]); // 5 знакомест под
элемент массива
    }
    printf("\n");
}

dijkstra(n, m, matrix, 0); // 0 - вершина которую надо найти
A-0, B-1, C-2, D-3, E-4
for (int i = 0; i < n; i++) // цикл по строкам
    free(matrix[i]); // освобождение памяти под строку
free(matrix);
return 0;
}

```

Пример выполнения задачи программой:

1 character | < 0 changes > Uncommitted Changes

Зведите количество строк: 5  
Зведите количество столбцов: 5  
matrix[0][0] = 12342  
matrix[0][1] = 231  
matrix[0][2] = 12  
matrix[0][3] = 445  
matrix[0][4] = 22344  
matrix[1][0] = 422  
matrix[1][1] = 123  
matrix[1][2] = 344  
matrix[1][3] = 456653  
matrix[1][4] = 2345  
matrix[2][0] = 211  
matrix[2][1] = 2345  
matrix[2][2] = 3222  
matrix[2][3] = 585  
matrix[2][4] = 4303  
matrix[3][0] = 493932  
matrix[3][1] = 43034  
matrix[3][2] = 43034  
matrix[3][3] = 43034  
matrix[3][4] = 43034


All Output ↕

Filter




1 character | < 0 changes > Uncommitted Changes

```
matrix[3][3] = 47574
matrix[3][4] = 3345
matrix[4][0] = 332
matrix[4][1] = 21
matrix[4][2] = 12345
matrix[4][3] = 4556
matrix[4][4] = 644
12342 231 12 445 22344
 422 123 344 456653 2345
 211 2345 3222 585 4303
 493 43834 449 49594 3345
 332 21 12345 4556 644
Vertex Distance from Source
A 0
B 231
C 12
D 445
E 2576
Program ended with exit code: 0
```

All Output 

1 character | < 0 changes > Uncommitted Changes

```
Зведите количество строк: 5
Зведите количество столбцов: 5
matrix[0][0] = 0
matrix[0][1] = 2
matrix[0][2] = 1000000
matrix[0][3] = 2
matrix[0][4] = 1000000
matrix[1][0] = 2
matrix[1][1] =
}
matrix[1][2] = 4
matrix[1][3] = 1000000
matrix[1][4] = 1000000
matrix[2][0] = 1000000
matrix[2][1] = 4
matrix[2][2] = 0
matrix[2][3] = 2
matrix[2][4] = 1
App Store
All Output 
```

```
matrix[3][3] = 0
matrix[3][4] = 7
matrix[4][0] = 1000000
matrix[4][1] = 1000000
matrix[4][2] = 1
matrix[4][3] = 7
matrix[4][4] = 0
      0      2 1000000      2 1000000
      2      0      4 1000000 1000000
1000000      4      0      2      1
      2 1000000      2      0      7
1000000 1000000      1      7      0
/vertex      Distance from Source
A-A          0
A-B          2
A-C          4
A-D          2
A-E          5
Program ended with exit code: 0
```

All Output ↕

Filter

## **Заключение**

Данный алгоритм пошагово перебирает все вершины графа и назначает им метки, которые являются известным минимальным расстоянием от вершины источника до конкретной вершины.

Данный алгоритм «физичен до мозга костей», например, с помощью него можно описать как молния ищет свой путь до точки.