**Практическая работа №3**

**Оглавление**

[Практическая работа № 3. объектно ориентированное программирование. 3](#_Toc154064157)

[Содержание пояснительной записки 7](#_Toc154064158)

[Используемое программное обеспечение 8](#_Toc154064159)

[Список литературы 9](#_Toc154064160)

Практическая работа № 3.  
объектно ориентированное  
программирование.

**Цель практической работы**

Закрепление теоретических знаний по разработке объектно-ориентированных программ.

**Постановка задачи**

Алгоритм Дейкстры — это алгоритм на графах, который был разработан Эдсгером Дейкстрой в 1959 году. Он позволяет вычислить кратчайший путь от одной изначально выбранной вершины до всех остальных вершин в графе с неотрицательными весами ребер. Алгоритм примечателен тем, что он гарантирует нахождение оптимального решения.

Аналогия для понимания: представьте, что вы находитесь в центре города и хотите найти кратчайший путь до всех достопримечательностей, при этом зная расстояние между каждым из объектов. Алгоритм Дейкстры позволит вам оптимально спланировать маршрут.

Прежде чем переходить к коду, важно понять некоторые основные понятия:

* **Граф** — это набор вершин (или узлов) и ребер, соединяющих их.
* **Вершина** — это отдельная точка в графе.
* **Ребро** — это соединение между двумя вершинами.
* **Вес ребра** — это числовое значение, которое может представлять, например, длину пути или его стоимость.
* **Путь** — это последовательность вершин, по которым можно пройти от одной точки к другой.
* **Кратчайший путь** — это путь с наименьшей суммарной длиной (или весом) ребер.

Алгоритм Дейкстры работает следующим образом:

1. Назначается начальная вершина, и для неё расстояние до себя устанавливается равным нулю, а до всех остальных вершин — бесконечность.
2. Выбирается вершина с наименьшим временным расстоянием, которое еще не было посещено.
3. Обновляются расстояния до соседних вершин.
4. Вершина помечается как посещенная.
5. Шаги 2-4 повторяются, пока не будут посещены все вершины.

Перед тем как начать кодирование, необходимо создать классы для представления графа, ребер и вершин. Вот пример таких классов в C#:

public class Graph

{

public List<Vertex> Vertices { get; } = [];

}

public class Vertex

{

public string Name { get; }

public IEnumerable<Edge> Edges => edges;

private List<Edge> edges { get; }

public Vertex(string name)

{

Name = name;

edges = new List<Edge>();

}

public void AddEdgeTo(Vertex to, int weight)

{

edges.Add(new Edge(this, to, weight));

}

public override string ToString()

{

return $"{Name}";

}

}

public class Edge

{

public Vertex From { get; }

public Vertex To { get; }

public int Weight { get; }

public Edge(Vertex from, Vertex to, int weight)

{

From = from;

To = to;

Weight = weight;

}

public override string ToString()

{

return $"{From} {To}";

}

}

После создания необходимых классов можно переходить к реализации самого алгоритма. Вот пример функции алгоритма Дейкстры на C#:

public class Graph

{

public List<Vertex> Vertices { get; } = [];

public Dictionary<Vertex, int> DijkstraAlgorithm(Vertex source, out Dictionary<Vertex, Vertex> path)

{

var distances = Vertices.ToDictionary(v => v, v => int.MaxValue);

path = new Dictionary<Vertex, Vertex>();

var notVisited = new HashSet<Vertex>(Vertices);

distances[source] = 0;

while (notVisited.Any())

{

var nearestVertex = notVisited.OrderBy(v => distances[v]).FirstOrDefault();

notVisited.Remove(nearestVertex);

foreach (var edge in nearestVertex.Edges)

{

var neighbor = edge.To;

if (notVisited.Contains(neighbor))

{

var currentDistance = distances[nearestVertex] + edge.Weight;

if (currentDistance < distances[neighbor])

{

distances[neighbor] = currentDistance;

path[neighbor] = nearestVertex;

}

}

}

}

return distances;

}

}

Алгоритм Дейкстры имеет временную сложность O(V^2), где V — количество вершин в графе. Однако с использованием очереди с приоритетами (например, кучи) его можно улучшить до O((V + E) log V), где E — количество ребер.

Для оптимизации кода выше, можно использовать SortedSet вместо простого HashSet, что позволит избавиться от сортировки на каждом шаге, так как SortedSet поддерживает порядок элементов.

Создадим граф и найдем кратчайшие пути от вершины A до всех остальных:

using WideSearch;

var vertexA = new Vertex("A");

var vertexB = new Vertex("B");

var vertexC = new Vertex("C");

var vertexD = new Vertex("D");

// Добавление остальных вершин...

var graph = new Graph();

graph.Vertices.AddRange(new[] { vertexA, vertexB, vertexC, vertexD });

vertexA.AddEdgeTo(vertexB, 1);

vertexB.AddEdgeTo(vertexC, 3);

vertexA.AddEdgeTo(vertexD, 1);

vertexD.AddEdgeTo(vertexC, 1);

// Добавление остальных ребер...

var shortestPaths = graph.DijkstraAlgorithm(vertexA, out Dictionary<Vertex, Vertex> pathPrevious);

foreach (var p in shortestPaths)

{

Console.WriteLine($"Кратчайший путь до {p.Key.Name}: {p.Value}");

Console.Write($"Предыдущие узлы: ");

Vertex? current = p.Key;

while (true)

{

if (!pathPrevious.ContainsKey(current))

break;

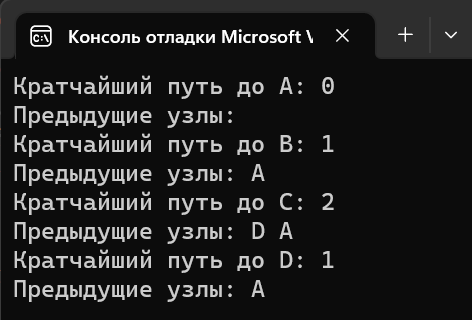
current = pathPrevious[current];

Console.Write($"{current.Name} ");

}

Console.WriteLine();

}



**Задание на практическую работу**

1. Модифицировать алгоритм поиска с учетом «полезности» точек пути. «Полезные» объекты должны иметь меньший вес, в таком случае путь через них будет самый «дешевый», приоритетный. Весовые коэффициенты задать произвольно.
2. Разработать метод для поиска точки, к которой должен двигаться бот. Выбор целевой точки должен быть логически обоснован.
3. Модифицировать алгоритм поиска с учетом ограничения на дальность поиска (максимальное количество ячеек).
4. Определить оптимальную область поиска, для которой алгоритм будет успевать найти пути быстрее, чем закончится ход.
5. Реализовать управление ботом для движения по найденному пути.
6. Защита работы включает демонстрацию работы программы на подготовленном примере.

# Содержание пояснительной записки

1. Постановка задачи. Приводится теоретический материал, использованный при написании приложения.

2. Формулировка задания и вариант. Приводится задание на лабораторную работу и вариант этого задания.

3. Описание выполняемых действий. Необходимо привести описание последовательности разработки программы, реализации используемых методов, алгоритмов, блок-схем.

4. Анализ результатов. Привести анализ входных и выходных данных. Показать результаты выполнения программного кода. Предоставить скриншоты обработки тестовых примеров. Сделать выводы.

5. Листинг программы. Привести листинг разработанного программного кода, содержание файлов входных и выходных данных.

# Используемое программное обеспечение

1. Среда программирования MS Visual Studio Community 2022 (Свободно распространяемое программное обеспечение (в учебных целях));
2. Microsoft Office Standard 2007 (Open License: 42267924);
3. Open Office (Свободно распространяемое программное обеспечение).
4. Браузер (Свободно распространяемое программное обеспечение).

# Список литературы

* + - 1. Мейер Б. Объектно-ориентированное программирование и программная инженерия [Электронный ресурс]/ Мейер Б. – Электрон. текстовые данные. – М.: Интернет-Университет Информационных Технологий (ИНТУИТ), 2016. – 285 c.
      2. Биллиг, В. A. Основы объектного программирования на С# (C# 3.0, Visual Studio 2008) [Электронный ресурс] : учебное пособие / В. A. Биллиг. — Электрон. текстовые данные. — Москва, Саратов : Интернет-Университет Информационных Технологий (ИНТУИТ), Вузовское образование, 2017. — 583 c. — 978-5-4487-0145-0. — Режим доступа: http://www.iprbookshop.ru/72339.html
      3. Павловская, Т. А. Программирование на языке высокого уровня C# [Электронный ресурс] / Т. А. Павловская. — 2-е изд. — Электрон. текстовые данные. — М. : Интернет-Университет Информационных Технологий (ИНТУИТ), 2016. — 245 c. — 2227-8397. — Режим доступа: http://www.iprbookshop.ru/73713.html
      4. Агапов, В. П. Основы программирования на языке С# [Электронный ресурс] : учебное пособие / В. П. Агапов. — Электрон. текстовые данные. — М. : Московский государственный строительный университет, ЭБС АСВ, 2012. — 128 c. — 978-5-7264-0576-6. — Режим доступа: http://www.iprbookshop.ru/16366.html
      5. Медведев, М. А. Программирование на СИ# [Электронный ресурс] : учебное пособие / М. А. Медведев, А. Н. Медведев ; под ред. А. В. Присяжный. — Электрон. текстовые данные. — Екатеринбург : Уральский федеральный университет, ЭБС АСВ, 2015. — 64 c. — 978-5-7996-1561-1. — Режим доступа: http://www.iprbookshop.ru/69667.html
      6. Казанский А.А. Объектно-ориентированное программирование на языке Microsoft Visual С# в среде разработки Microsoft Visual Studio 2008 и .NET Framework. 4.3 [Электронный ресурс]: учебное пособие и практикум/ Казанский А.А.— Электрон. текстовые данные.— М.: Московский государственный строительный университет, ЭБС АСВ, 2011.— 180 c
      7. Уйманова Н.А. Основы объектно-ориентированного программирования [Электронный ресурс]: практикум/ Уйманова Н.А., Таспаева М.Г.— Электрон. текстовые данные.— Оренбург: Оренбургский государственный университет, ЭБС АСВ, 2017.— 156 c.— Режим доступа: http://www.iprbookshop.ru/78808.html.— ЭБС «IPRbooks»
      8. Новиков П.В. Объектно-ориентированное программирование [Электронный ресурс]: учебно-методическое пособие к лабораторным работам/ Новиков П.В.— Электрон. текстовые данные.— Саратов: Вузовское образование, 2017.— 124 c.— Режим доступа: http://www.iprbookshop.ru/64650.html.— ЭБС «IPRbooks»