**Практическая работа №2**

**Оглавление**

[Практическая работа № 1. Поиск в ширину 3](#_Toc153871711)

[Содержание пояснительной записки 13](#_Toc153871712)

[Используемое программное обеспечение 14](#_Toc153871713)

[Список литературы 15](#_Toc153871714)

Практическая работа № 2.  
объектно-ориентированное  
программирование.

алгоритм Поиск в ширину

**Цель практической работы**

Закрепление теоретических знаний по разработке поисковых алгоритмов в рамках объектно-ориентированного программирования.

**Постановка задачи**

**Граф** — математическая абстракция реальной системы любой природы, объекты которой обладают парными связями. Граф как математический объект есть совокупность двух множеств — множества самих объектов, называемого множеством вершин, и множества их парных связей, называемого множеством рёбер. Элемент множества рёбер есть пара элементов множества вершин.

Графы находят широкое применение в современной науке и технике. Они используются и в естественных науках (физике и химии) и в социальных науках (например, социологии), но наибольших масштабов применение графов получило в информатике и сетевых технологиях.

В качестве простейшего примера из жизни можно привести схему перелётов определённой авиакомпании, которая моделируется графом, где вершинами графа являются города, а рёбрами — рейсы, соединяющие пары городов. Дерево каталогов в компьютере также является графом: диски, папки и файлы являются вершинами, а рёбра показывают вложенность файлов и папок в папки и диски[1]. Строение Википедии моделируется ориентированным графом, в котором статьи — вершины графа, а гиперссылки — дуги (тематическая карта).

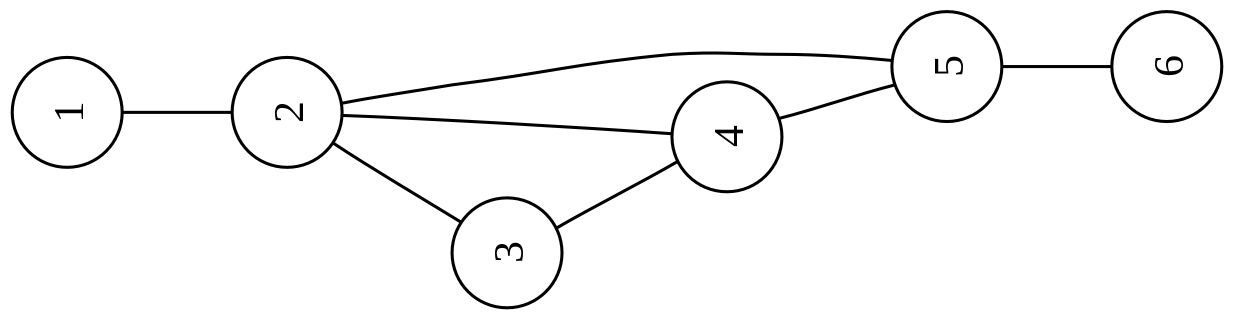


Рисунок 1 – Пример неориентированного графа

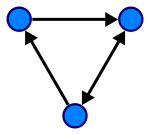


Рисунок 2 – Пример ориентированного графа

Дополнительная информация по графам: <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)>

<https://habr.com/ru/articles/564594/>

**Что такое обход графа?**

Простыми словами, обход графа — это переход от одной его вершины к другой в поисках свойств связей этих вершин. Связи (линии, соединяющие вершины) называются направлениями, путями, гранями или ребрами графа. Вершины графа также именуются узлами.

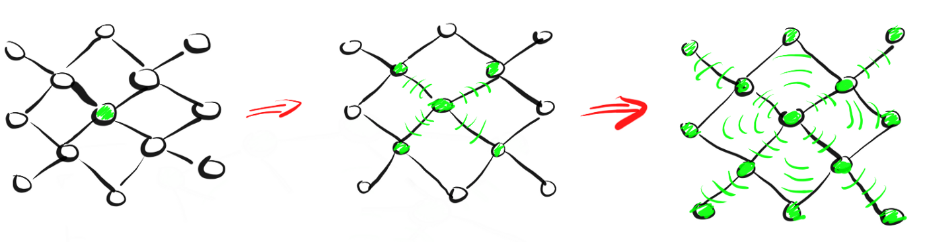
Двумя основными алгоритмами обхода графа являются поиск в глубину (Depth-First Search, DFS) и поиск в ширину (Breadth-First Search, BFS).

**Поиск в глубину**

DFS следует концепции «погружайся глубже, головой вперед» («go deep, head first»). Идея заключается в том, что мы двигаемся от начальной вершины (точки, места) в определенном направлении (по определенному пути) до тех пор, пока не достигнем конца пути или пункта назначения (искомой вершины). Если мы достигли конца пути, но он не является пунктом назначения, то мы возвращаемся назад (к точке разветвления или расхождения путей) и идем по другому маршруту.

**Поиск в ширину** (англ. Breadth-First Search, BFS) позволяет вычислить кратчайшие расстояния (в терминах количества рёбер) от выделенной вершины ориентированного графа до всех остальных вершин, и/или построить корневое направленное дерево, расстояния в котором совпадают с расстояниями в исходном графе. Кроме того, поиск в ширину позволяет решать задачу проверки достижимости (существуют ли пути между вершиной источником и остальными вершинами графа). Впервые алгоритм поиска в ширину описан в работах Мураи Ли.

Алгоритм основан на обходе вершин графа "по слоям". На каждом шаге есть множество "передовых" вершин, для смежных к которым производится проверка, относятся ли они к еще не посещенным. Все еще не посещенные вершины добавляются в новое множество "передовых" вершин, обрабатываемых на следующем шаге. Изначально в множество "передовых" вершин входит только вершина-источник, от которой и начинается обход.



**Рассмотрим пример** поиска некоторого объекта (определенной ячейки) в двумерном лабиринте, с координатами ячеек x, y. Псевдокод алгоритма выглядит следующим образом:

1. Создаем словарь d для хранения посещенных ячеек. У словаря ключ – координаты ячеек (x,y), значение – расстояние (r) до ячейки.
2. Создаем очередь (Queue), для хранение x, y, r, которые мы хотим посетить (r необходим т.к. мы можем попасть в заданную ячейку несколькими путями, в таком случае нужно выбирать меньший r).
3. Помещаем начальную точку в очередь со значениями x0, y0, 0.
4. В цикле пока очередь не пуста:
   1. Достаем текущий объект из очереди: xi, yi, ri.
   2. Если ячейка – искомая, завершаем поиск.
   3. Если ячейка есть в словаре, то необходимо проверить расстояние и если новое расстояние меньше, то обновить расстояние в словаре. Перейти на шаг 4.1.
   4. Поместить ячейку в словарь со значениями: xi, yi, ri.
   5. В цикле по ячейкам смежным к текущей ячейке (куда можно перейти) добавить координаты соседних ячеек xj, yj в очередь и расстояние, равное текущему расстоянию, увеличенному на единицу rj = ri + 1.

В результате мы получаем координаты ближайшуй искомой ячейки и расстояние до нее.

Данный алгоритм не сохраняет путь до ячейки, однако может быть модифицирован для сохранения пути (подумать самостоятельно).

**Задание на практическую работу**

1. Реализовать алгоритм поиска в ширину и определить ближайшую ячейку (относительно бота), содержащую объект, который может поднять бот.
2. Модифицировать алгоритм для сохранения информации о всех исследованных ячейках, сохранить кратчайшие пути до ячеек.
3. Реализовать алгоритм перевода координат ячеек в движения бота (PlayerAction).
4. Сохранить информацию о найденных путях в текстовый файл. Каждый путь сохранять в формате: [Целевая ячейка]; [Тип объекта]; [Шаг 1]; [Шаг 2]; … [Шаг N].
5. 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»