ПРИМЕНЕНИЕ ГРАФОВ В КОМПЬЮТЕРНОМ МОДЕЛИРОВАНИИ ЛАБИРИНТА И ПОИСКА КРАТЧАЙШЕГО ПУТИ В НЕМ

Автор: Киров Евгений, студент 3 курса.
Руководитель: Кульман Татьяна Николаевна, доцент кафедры Информационных технологий, к.т.н.
Образовательное учреждение: ГОУ Московской области Международный университет природы, общества и человека «Дубна», филиал «Протвино», г. Протвино.

GRAPHS APPLICATION IN A LABYRINTH COMPUTER SIMULATION AND SEARCH OF THE SHORTEST PATH IN LABYRINTH

Kirov E.

Графы – сложные математические структуры, которые помогают решать различные задачи в экономике, транспорте, химии, электронике и т.д., а также позволяют упростить постановку многих задач. В данной работе рассматривается программа приведение лабиринта к представлению в виде графа и поиска оптимального пути в нём.

В работе будут использованы следующие структуры данных [1]:

Граф – как основа представления лабиринта;

Стек – структура данных, в которой доступ к элементам организован по принципу LIFO (англ. last in – first out, «последним пришёл – первым вышел»). Стек необходим для записи кратчайшего пути;

Очередь – структура данных с дисциплиной доступа к элементам «первый пришёл – первый вышел» (FIFO, First In – First Out). Очередь применяется для реализации поиска в ширину.

Рассматриваемая программа содержит несколько этапов:

1. Генерация лабиринта.
2. Представление лабиринта в виде графа.
3. Обход графа с целью нахождения кратчайшего пути.

На первом этапе программа осуществляет генерацию лабиринта случайным образом по методу Прима. Для визуального построения лабиринта и применения алгоритма Прима первоначально лабиринт представляется в виде набора из структур, моделирующих клетки лабиринта (см. Рис. 1). Для осуществления поиска по лабиринту удобно его представить в виде графа, в отношении которого можно применить поиск в ширину. Матрица смежности графа генерируется на основе матрицы лабиринта, и передается алгоритму поиска в ширину.

Для описания лабиринта создан класс Labirinth (см. Рис.2), в который включены элементы лабиринта на этапе его построения, а также элементы, необходимые для поиска пути. Представление Area – обычный двухмерный массив, из структур объединяющих координаты клеток лабиринта и их атрибуты. Представление Graph – неориентированный граф в виде матрицы смежности, формируемый из представления Area.

Сначала производится инициализация, т.е. создаётся заготовка, в которой каждая клетка замкнута. На первом шаге присваиваем случайной клетке атрибут INSIDE. На втором шаге присваиваем соседним с ней клеткам атрибут BORDER. Далее осуществляем генерацию. Пока атрибут хотя бы одной клетки равен BORDER, выбираем случайную клетку [X][Y], атрибут которой равен BORDER и присваиваем ей атрибут INSIDE. Изменим на BORDER атрибут всех соседних с [X][Y] клеток, если их

атрибут равен OUTSIDE. Из всех соседних с [X][Y] клеток, атрибут которых равен INSIDE, выбираем случайную клетку и разрушаем стену между этой клеткой и клеткой [X][Y]. На этом генерация лабиринта завершена.

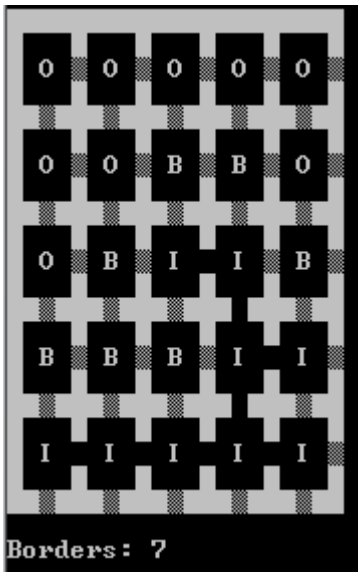

Рис.1 Набор из структур метода Прима

```
class Labyrinth
{       int Height;
        int Width;

        struct Cell
    {
                int state;
                int y;
                int x;
                bool left_wall;
                bool top_wall;
                bool right_wall;
                bool bottom_wall;
        };
vector < vector<Cell> > Area;
vector < vector<int> > Graph;    и т.д.
```

Рис. 2 Фрагмент описание класса Labyrinth с включением описание клетки - struct Cell

Существует вариант запуска программы пошаговой «отрисовки» построения лабиринта, т. е. выбор клетки, затем постановка границ, выбор следующей клетки, создание прохода и т. д.

На втором этапе происходит представление лабиринта в виде графа. Это необходимо для передачи его на обработку алгоритму обхода в ширину. Сначала присваиваем всем вершинам графа нулевые начальные значения. Затем производим отметку связей, основываясь на представлении Area.

Нумерация вершин графа осуществляется построчно: Num_Vertex = X + Y * Width (где X и Y координаты клетки лабиринта). Граф неориентированный, поэтому (i, j) и (j, i) обозначают одно и то же ребро. Граф хранится в виде матрицы смежности.

На третьем этапе выполняется обход графа с помощью алгоритма обхода в ширину. В качестве входных данных алгоритм принимаем матрицу смежности из этапа два. Алгоритм обхода в ширину служит для установления кратчайшего пути между двумя узлами графа. Далее опишем работу алгоритма в программе.

Создадим очередь, назовем ее Q, в которую будут помещаться рассматриваемые вершины. Создадим также булевский массив used[], в котором для каждой вершины будем отмечать, была ли она посещена. Изначально в очередь помещается только одна выбранная нами вершина, допустим это – Begin_vertex и used[begin_vertex]=true, для всех остальных вершин used[]=false. До тех пор, пока очередь непуста, из ее «головы» берётся одна вершина и просматриваются все возможные ребра, исходящие из этой вершины, если же какие-то из этих вершин еще не просмотрены, – они перемещаются в конец очереди. В итоге, когда очередь опустеет, алгоритм «обойдет» все достижимые из Q вершины, причем до каждой дойдет кратчайшим путем.

Далее происходит восстановление пути до искомой вершины. В алгоритме восстановления пути задействован вектор.

Для демонстрации процесса поиска создадим лабиринт. Пусть это будет лабиринт размером 5x5. Как было описано выше, при создании лабиринта применяется

алгоритм Прима. На рисунках 3 и 4 приводится лабиринт в том виде, в котором его выводит программа, а так же в виде графа.

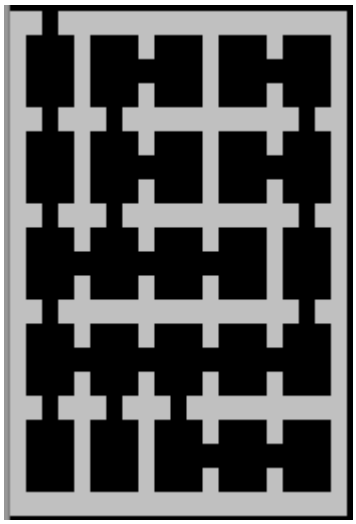

Рис. 3 Представление лабиринта программой

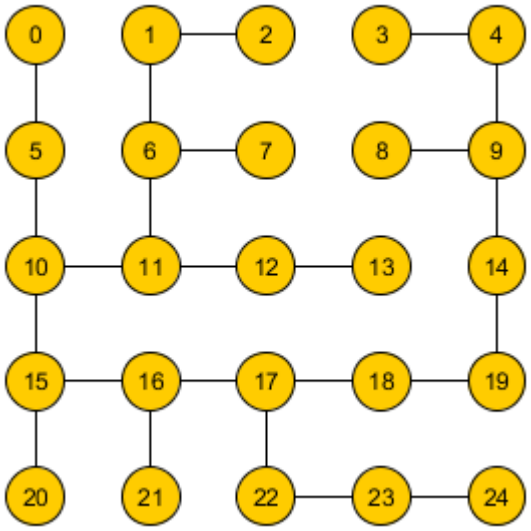

Рис. 4 Представление лабиринта в виде графа

На рис. 3 вершины графа – это клетки лабиринта, а рёбра – это проходы в лабиринте. Таким образом, лабиринт представлен в виде графа, следовательно, теперь можно применить алгоритмы поиска кратчайшего пути, таким образом, вся сила алгоритмов работы с графами здесь уместна и более того, эффективна.

Результат работы программы для рассматриваемого лабиринта показан на Рис. 5. В качестве выхода из лабиринта задана ячейка (4;4).

Как мы можем убедиться, программа осуществила обход наиболее оптимальным путем, и вывела номера ячеек этого кратчайшего пути (см. Рис. 5) (ячейки нумеруются слева направо построчно).

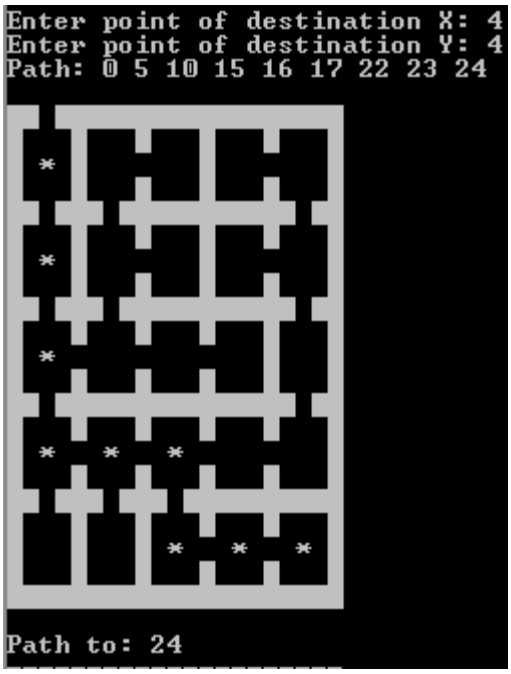

Рис. 5 Результат работы программы

Программа написана на языке программирования C++ с использованием библиотеки STL (Standard Template Library).

Библиографический список

1. Кормен Т., Лейзерсон Ч., Ривест Р. Алгоритмы: построение и анализ. М.: МЦНМО, 2001.