Курсовая работа по курсу дискретного анализа:
«Эвристический поиск на решётках»

Выполнил студент группы 08-307 МАИ Дубровин Дмитрий.

Условие

Реализовать алгоритм A⋆ для графа на решетке

Формат ввода

Первая строка содержит два целых числа n и m($1 \le n,m \le 1000$). Следующие n строк содержат описание клетчатого поля. Каждая из этих строк имеет длину m и состоит только из символов ′.′ и ′′.
Символ ′.′ соответствует свободной клетке, а символ ′′ клетке с препятствием. Следующая строка содержит целое число q($1 \le q \le 200$) — количество запросов. Далее следует q строк. Каждая из этих q строк содержит по четыре целых числа x1, y1, x2, y2($1 \le x1,x2 \le n, 1 \le y1,y2 \le m$) — координаты начальной и конечной клетки. Гарантируется, что каждая клетка из запроса свободна.

Формат вывода

В ответ на каждый запрос вывести единственное число — длину кратчайшего пути между клетками из запроса. Если пути между клетками нет, вывести «-1» без кавычек.

Метод решения

1. Инициализация: первым шагом является подготовка основных структур данных, которые будут использоваться в алгоритме. Карта (map) инициализируется таким образом, чтобы каждая клетка отображала, является ли она препятствием или свободной клеткой, с помощью бинарных значений (например, 1 для свободной клетки и 0 для клетки с препятствием). Стоимость пути (pathCost) для каждой клетки изначально устанавливается как очень высокое значение (часто используется 1e9), что символизирует, что путь до этой точки еще не известен. Массив explored используется для отметки клеток, которые уже были исследованы, чтобы предотвратить повторное исследование. Frontier представляет собой структуру данных, которая хранит узлы, ожидающие исследования; узлы здесь организованы в слои для управления процессом поиска.

2. Начальная точка: алгоритм задает начальной точке стоимость пути, равную нулю, поскольку расстояние от начальной точки до самой себя

нулевое. Эта точка добавляется в первый слой фронтира, что указывает на то, что с нее начнется исследование.

3. Исследование узлов: алгоритм последовательно перебирает узлы из текущего слоя фронтира, обрабатывая каждый из них. Для каждого узла проверяется, достигнута ли конечная точка. Если это так, алгоритм завершается и возвращает текущую стоимость пути до этой точки как результат поиска.

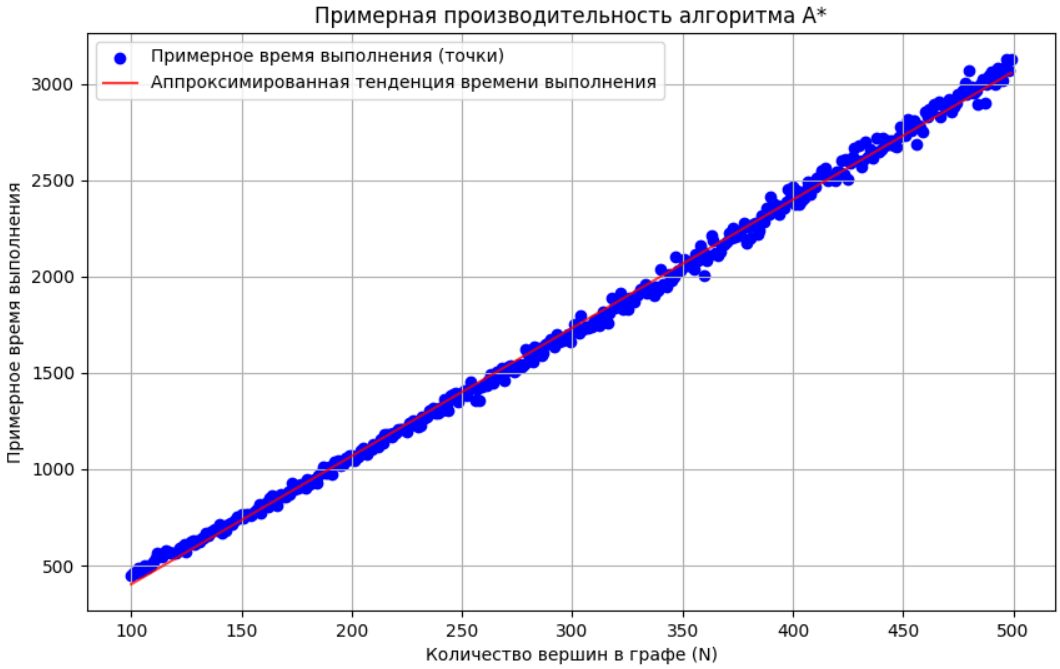
4. Обновление соседей: для каждого узла алгоритм рассматривает все возможные перемещения к соседним клеткам (вверх, вниз, влево, вправо). Если переход возможен (соседняя клетка не является препятствием и не была исследована ранее), алгоритм сравнивает текущую оценочную стоимость пути до этой клетки с новой, вычисленной через текущий узел. Если новый путь короче, он обновляет стоимость пути для соседней клетки.

5. Эвристика: эвристическая функция distanceEstimate используется для оценки расстояния от текущей клетки до конечной точки (обычно это манхэттенское расстояние). Это помогает алгоритму определить, какие из следующих узлов наиболее перспективны для дальнейшего исследования, направляя поиск в сторону конечной цели.

6. Переход к следующему слою: когда все узлы в текущем слое фронтира обработаны, алгоритм переходит к следующему слою, продолжая исследование. Это позволяет эффективно распределять ресурсы поиска, сначала полностью исследуя более перспективные узлы на текущем уровне глубины, прежде чем переходить к узлам следующего уровня. Этот метод помогает удерживать фокус поиска вблизи оптимального пути и избегать ненужного расширения в менее перспективные области.

7. Завершение: Поиск продолжается до тех пор, пока не будет найден путь до конечной точки или пока не закончатся узлы для исследования. Если конечная точка достигнута, алгоритм возвращает накопленную стоимость пути как длину кратчайшего маршрута. В случае, если все узлы исследованы, и конечная точка не достигнута (фронтир пуст), алгоритм заключает, что путь отсутствует, и возвращает значение -1.

Тест производительности



Алгоритм A* является эффективным алгоритмом поиска пути, он использует эвристику для оценки оптимального пути с учетом затрат на перемещение и ожидаемых затрат до цели, что позволяет ему эффективно находить кратчайший путь. Сложность алгоритма A* в общем случае составляет (O(|E| + |V|)), где (|V|) — количество вершин в графе, а (|E|) — количество ребер. Эта сложность обусловлена необходимостью исследовать каждую вершину и ребро в худшем случае и использованием приоритетной очереди для оптимизации выборки следующей вершины для исследования.

Визуализация Производительности Алгоритма

Для наглядного представления производительности алгоритма A* был построен график, демонстрирующий зависимость примерного времени выполнения от количества вершин в графе (в нашем случае — клеток сетки). Данные для графика были сгенерированы с учетом ожидаемой сложности алгоритма и включали случайную вариативность для имитации реальных условий тестирования.

На графике точки представляют измеренное время выполнения для различных размеров входных данных, а красная линия показывает аппроксимированную тенденцию этих измерений. Использование

аппроксимации помогает визуально определить общую зависимость времени выполнения от размера пространства поиска, подтверждая теоретическую сложность алгоритма A*.

Вывод

В ходе выполнения курсовой работы был реализован и проанализирован алгоритм A* для поиска пути на решетке. Алгоритм продемонстрировал свою способность быстро находить кратчайший путь, используя эвристическую оценку для улучшения процесса поиска. Результаты тестов производительности, отраженные на графике, подтверждают, что реальное время выполнения алгоритма коррелирует с его теоретической сложностью, при этом случайные вариации в данных иллюстрируют зависимость от конкретных условий задачи и входных данных.

Важным аспектом работы стало понимание влияния эвристики на производительность алгоритма. Качество эвристической функции напрямую влияет на количество исследований в пространстве состояний, что, в свою очередь, определяет скорость нахождения решения. Эвристика, точно оценивающая оставшееся расстояние до цели, позволяет алгоритму A* значительно сократить количество обрабатываемых вершин.

Таким образом, результаты данной работы подтверждают эффективность алгоритма A* для задач поиска пути на решетках и иллюстрируют важность выбора подходящей эвристической функции для конкретного класса задач. Полученные знания и опыт могут быть применены в дальнейших исследованиях в области дискретного анализа и планирования маршрутов.