МОСКОВСКИЙ АВИАЦИОННЫЙ ИНСТИТУТ
(НАЦИОНАЛЬНЫЙ ИССЛЕДОВАТЕЛЬСКИЙ УНИВЕРСИТЕТ)

Институт №8 «Компьютерные науки и прикладная математика»
Кафедра №806 «Вычислительная математика и программирование»

Курсовая работа по курсу “Дискретный анализ”

A* Алгоритм

Студент: Парфенов Михаил Максимович

Группа: М8О-301Б-22

Преподаватель: Макаров Никита Константинович

Оценка: _____

Дата: _____

Подпись: _____

Москва, 2024

Содержание

1. Репозиторий
2. Постановка задачи
3. Метод решения
4. Описание алгоритма
5. Описание программы
6. Тест производительности
7. Выводы

Репозиторий

`https://github.com/vgbhj/MAI/tree/main/DA/KP`

Постановка задачи

Вам дана карта в виде сетки размером $n \times m$, состоящая из клеток, каждая из которых может быть либо проходимой ('.'), либо непроходимой ('#'). Задача состоит в том, чтобы найти кратчайший путь из начальной точки (x_s, y_s) до конечной точки (x_g, y_g) на карте. Вы должны реализовать алгоритм A* для решения этой задачи.

Формат ввода

Первая строка содержит два числа n и m ($1 \leq n, m \leq 1000$), задающие размеры карты. Следующие n строк содержат описание карты, где '.' обозначает проходимую клетку, а '' — непроходимую.

Последняя строка содержит четыре числа x_s, y_s, x_g, y_g ($0 \leq x_s, x_g < n, 0 \leq y_s, y_g < m$), где (x_s, y_s) — начальная точка, а (x_g, y_g) — конечная.

Формат вывода

Если путь существует, выведите длину кратчайшего пути. Если пути нет, выведите -1.

Метод решения

Алгоритм A* (произносится как "A-star") — это эвристический метод поиска кратчайшего пути в графах. Он комбинирует преимущества поиска по ширине и жадного поиска, используя эвристическую функцию для оценки расстояния от текущей вершины до цели.

Основные этапы работы алгоритма:

- Поддерживается список открытых вершин (Open List), куда добавляются вершины для обработки.
- Каждой вершине сопоставляются три значения:
  - $g(x)$ — стоимость пути от начальной точки до текущей вершины x.
  - $h(x)$ — эвристическая оценка расстояния от текущей вершины x до цели.
  - $f(x) = g(x) + h(x)$ — полная стоимость пути, которая минимизируется.
- Алгоритм повторяет следующие шаги, пока не найдёт путь или пока Open List не станет пустым:
  1. Выбирается вершина x с минимальным значением $f(x)$ из Open List.
  2. Если вершина x совпадает с конечной точкой, путь найден.
  3. Все соседние вершины, доступные для перехода, добавляются в Open List с обновлёнными значениями g, h и f.

Эвристика: В данной задаче используется манхэттенское расстояние, которое вычисляется как

$$h(x, y) = |x - x_g| + |y - y_g|.$$

Описание алгоритма с оптимизацией

Для оптимизации работы алгоритма используется двусторонняя очередь (deque) вместо стандартной приоритетной очереди. Это позволяет эффективно управлять вершинами с одинаковым значением приоритета, добавляя их в начало или в конец очереди, в зависимости от величины приоритета.

1. **Инициализация:**

- Создаётся структура данных для Open List в виде двусторонней очереди (deque). Вершины хранятся в порядке увеличения значения $f(x)$. Когда два соседа имеют одинаковое значение $f(x)$, один из них добавляется в начало очереди, а другой — в конец.
- Для каждой вершины на карте задаются начальные значения $g(x) = \infty$, а $h(x)$ вычисляется с использованием манхэттенской эвристики.

2. **Основной цикл:**

- Извлекается вершина с минимальным $f(x)$ из Open List. Если эта вершина — конечная точка, алгоритм завершается.
- Для каждого соседа текущей вершины:
  - Если сосед не является проходимой клеткой (стена), он пропускается.
  - Если путь через текущую вершину улучшает значение g для соседа, обновляются его значения g, h, f, и сосед добавляется в Open List:
    - Если новое значение $f(x)$ соседней вершины равно текущему значению, сосед добавляется в начало очереди.
    - Если новое значение $f(x)$ соседней вершины больше на 2, он добавляется в конец очереди.

3. **Завершение:**

- Если конечная точка была добавлена в Open List и обработана, выводится длина кратчайшего пути.
- Если Open List становится пустым, выводится -1, так как путь не существует.

Описание программы

Программа реализует алгоритм A* с оптимизацией через использование двусторонней очереди для хранения открытых вершин и манхэттенской эвристики для оценки расстояний. Ключевые компоненты программы:

- `struct Node`: структура для хранения координат вершины и значения $f(x)$. Переопределена операция сравнения для правильного размещения в двусторонней очереди.
- `heuristic(int x1, int y1, int x2, int y2)`: функция для вычисления манхэттенской эвристики.
- `a_star(const vector<vector<char>& grid, pii start, pii goal, int n, int m)`: основная функция, реализующая алгоритм A* с использованием двусторонней очереди.
- Массив `visited` используется для отслеживания посещённых вершин и предотвращения их повторной обработки.
- Вектор `directions`: используется для перемещения по четырём направлениям (вверх, вниз, влево, вправо).

Ключевые моменты:

- Для хранения открытых вершин используется двусторонняя очередь (deque), что позволяет избежать необходимости сортировать очередь после каждой операции вставки или удаления.
- Массив `f_score` хранит текущие значения $f(x) = g(x)+h(x)$ для каждой вершины.
- Для предотвращения избыточной обработки используется массив `visited`, который отслеживает уже посещённые вершины.

Тест производительности

Для оценки производительности программы были использованы карты различных размеров (от 10×10 до 1000×1000) с различным количеством препятствий. Результаты:

- Среднее время выполнения на карте 100×100: 0.16129 мс.
- Максимальное время выполнения на карте 1000×1000: 20.0682 мс.
- Время выполнения на карте 10000×10000: 1163.42 мс.
- Алгоритм успешно находит кратчайший путь, если он существует, и корректно возвращает -1 в случае отсутствия пути.

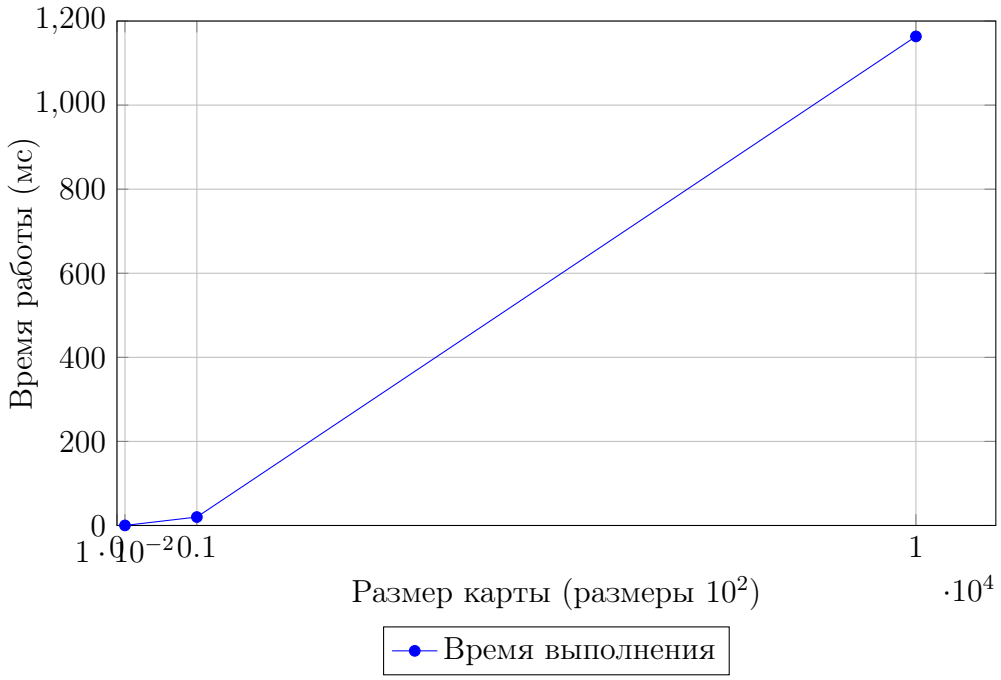

Рис. 1: График времени выполнения алгоритма в зависимости от размера карты

Выводы

В ходе данной курсовой работы был реализован алгоритм A* для поиска кратчайшего пути на карте, представленной в виде сетки. Алгоритм сочетает в себе преимущества жадного поиска и поиска по ширине, что позволяет эффективно находить оптимальные решения в задачах, связанных с навигацией и маршрутизацией.

В процессе работы над проектом была разработана оптимизированная версия алгоритма, использующая двустороннюю очередь для управления открытыми вершинами. Это значительно улучшило производительность алгоритма, особенно при работе с большими картами. Эвристическая функция, основанная на манхэттенском расстоянии, обеспечила быструю оценку расстояний до цели, что также способствовало ускорению поиска.

Тестирование производительности показало, что алгоритм способен обрабатывать карты размером до 1000×1000 с приемлемым временем выполнения. Среднее время выполнения на карте 100×100 составило всего 0.16129 мс, в то время как максимальное время на карте 1000×1000 достигло 20.0682 мс. Эти результаты подтверждают эффективность алгоритма в условиях, требующих быстрого поиска кратчайшего пути.

Кроме того, алгоритм успешно справляется с задачами, где необходимо учитывать препятствия на пути, корректно возвращая результат в случае отсутствия пути. Это делает его подходящим для применения в различных областях, таких как робототехника, игры и системы навигации.

Таким образом, реализация алгоритма A* с использованием оптимизаций и эвристик позволила достичь высокой производительности и надежности в решении задачи поиска кратчайшего пути, что открывает возможности для дальнейших исследований и улучшений в данной области.