Алгоритм A*

Донской Юрий

17 декабря 2022 г.

Оглавление

Глоссарий

Алгоритм поиска пути ищет на графе, начиная с одной (стартовой) точки и исследуя смежные узлы до тех пор, пока не будет достигнут узел назначения (конечный узел). Кроме того, в алгоритмах поиска пути в большинстве случаев заложена также цель найти самый короткий путь.

Шейки Робот был первым мобильным роботом общего назначения, способным рассуждать о своих действиях. В то время как другие роботы должны быть проинструктированы на каждом отдельном этапе выполнения более крупной задачи, Shakey может анализировать команды и самостоятельно разбивать их на основные части.

Взвешенный граф — это граф, у которого рёбрам соответствуют некоторые весовые параметры.

Эвристическая функция — это творческая функция, состоящая в организации избирательного поиска при решении сложных интеллектуальных задач.

Глава 1

Введение

1.1 Определение алгоритма A*

Алгоритм A* (*англ. A star*) — алгоритм поиска пути, который находит во взвешенном графе маршрут наименьшей стоимости от указанной начальной вершины до указанной конечной.

1.2 История

A* был создан как часть проект Шейки, целью которого было создание мобильного робота, который мог бы планировать свои собственные действия. Нильс Нильссон первоначально предложил использовать алгоритм Graph Traverser для планирования пути Шейки. Graph Traverser руководствуется эвристической функцией $h(n)$, расчетным расстоянием от узла n до целевого узла: он полностью игнорирует $g(n)$, расстояние от начального узла до n. Бертрам Рафаэль предложил использовать сумму $g(n) + h(n)$. Питер Харт изобрел концепции, которые мы теперь называем допустимостью и согласованностью эвристических функций. Первоначально A* был разработан для поиска путей с наименьшей стоимостью, когда стоимость пути является суммой его затрат, но было показано, что A* может использоваться для поиска оптимальных путей для любой задачи, удовлетворяющей условиям алгебры затрат.

Исходная статья A* 1968 года содержала теорему, утверждающую, что ни один A* — подобный алгоритм не может расширить меньше узлов, чем A*, если эвристическая функция согласована и правильно выбрано правило A*. Несколько лет спустя было опубликовано «исправление», в котором утверждалось, что согласованность не требуется, но это оказалось ложным в окончательном исследовании Дехтера и Перла оптимальности A* (теперь называемой оптимальной эффективностью), в котором был приведен пример A* с допустимой эвристикой, но не последовательным расширением произвольно большего числа узлов, чем альтернативный A* — подобный алгоритм.

1.3 Применение алгоритма

Алгоритм A* применяют в различных областях от машинного обучения до разработки игр, например, в навигаторах или заклинание "Ясновидение"в игре Skyrim.

Глава 2

Описание алгоритма

2.1 Описание

В процессе работы алгоритма для вершин рассчитывается функция

$$f(v) = g(v) + h(v)$$

- $g(v)$ — наименьшая стоимость пути в v из стартовой вершины,
- $h(v)$ — эвристическое приближение стоимости пути от v до конечной цели.

Фактически, функция $f(v)$ — длина пути до цели, которая складывается из пройденного расстояния $g(v)$ и оставшегося расстояния $h(v)$. Исходя из этого, чем меньше значение $f(v)$, тем раньше мы откроем вершину v, так как через неё мы предположительно достигнем расстояние до цели быстрее всего. Открытые алгоритмом вершины можно хранить в очереди с приоритетом по значению $f(v)$.

2.2 Свойства

Чтобы A* был оптимален, выбранная функция $h(v)$ должна быть допустимой эвристической функцией[1].

Определение: *Говорят, что эвристическая оценка $h(v)$ допустима, если для любой вершины v значение $h(v)$ меньше или равно весу кратчайшего пути от v до цели.*

Допустимая оценка является оптимистичной, потому что она предполагает, что стоимость решения меньше, чем оно есть на самом деле. Второе, более сильное условие — функция $h(v)$ должна быть монотонной.

Определение: *Эвристическая функция $h(v)$ называется монотонной (или преемственной), если для любой вершины v_1 и ее потомка v_2 разность $h(v_1)$ и $h(v_2)$ не превышает фактического веса ребра $c(v_1, v_2)$ от v_1 до v_2, а эвристическая оценка целевого состояния равна нулю.*

Теорема: *Любая монотонная эвристика допустима, однако обратное неверно.*

Доказательство: Пусть $k(v)$ — длина кратчайшего пути из вершины v до цели. Докажем индукцией по числу шагов до цели, что $h(v) \leqslant k(v)$.

Если до цели расстояние 0, то v — цель и $h(v) = 0 \leqslant k(v)$.

Пусть v находится на расстоянии i от цели. Тогда существует потомок v', который находится на кратчайшем пути от v до цели и v' лежит на расстоянии $i - 1$ шагов до цели. Следовательно, $h(v) \leqslant c(v, v') + h(v')$. По предположению, $h(v') \leqslant k(v')$. Следовательно, $h(v) \leqslant c(v, v') + k(v') = k(v)$.

Таким образом, монотонная эвристика $h(v)$ допустима.

Утверждение: *Если $h(v)$ монотонна, то последовательность значений $f(v)$ на любом пути не убывает.*

Доказательство следует из определения монотонности. Пусть v' — потомок v, тогда $g(v') = g(v) + c(v, v')$. Следовательно, $f(v') = g(v') + h(v') = g(v) + c(v, v') + h(v') \geqslant g(v) + h(v) = f(v)$.

Утверждение: *Алгоритм A* является оптимальным, если функция $h(v)$ монотонна.* Последовательность вершин "развёрнутых"во время работы алгоритма находится в неубывающем порядке значений f. Поэтому очередная выбираемая вершина должна представлять собой оптимальное решение, поскольку все дальнейшие узлы будут, по меньшей мере, столь же дорогостоящими.

[1]Эвристическая функция называется допустимой , если она никогда не переоценивает стоимость достижения цели, т. е. оцениваемая ею стоимость достижения цели не превышает минимально возможную стоимость от текущей точки пути

2.3 Примеры эвристик

Поведение алгоритма сильно зависит от того, какая эвристика используется. В свою очередь, выбор эвристики зависит от постановки задачи. Часто A* используется для моделирования перемещения по поверхности, покрытой координатной сеткой.

- Если мы можем перемещаться в четырех направлениях, то в качестве эвристики стоит выбрать манхэттенское расстояние
$$h(v) = |v.x - v'.x| + |v.y - v'.y|$$
- Расстояние Чебышева применяется, когда к четырем направлениям добавляются диагонали:
$$h(v) = max(|v.x - v'.x|, |v.y - v'.y|)$$
- Если передвижение не ограничено сеткой, то можно использовать евклидово расстояние по прямой:
$$h(v) = \sqrt{(v.x - v'.x)^2 + (v.y - v'.y)^2}$$

Также стоит обратить внимание на то как соотносятся $f(v)$ и $h(v)$. Если они измеряются в разных величинах (например, $g(v)$ — это расстояние в километрах, а $h(v)$ — оценка времени пути в часах) A* может выдать некорректный результат.

2.4 Пример

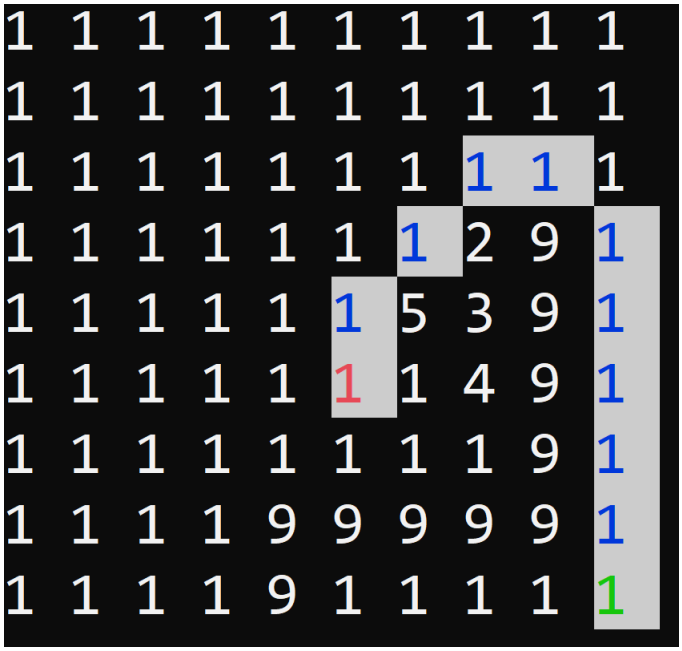

Рис. 2.1: Пример работы алгоритма

Глава 3

Формальная постановка задачи

Поле — прямоугольная область разделенная на квадратные клетки, размер поля определяется $n \times m$ где n — количество клеток по горизонтали, а m — количество клеток по вертикали. Значение клетки $a[i][j]$ равно v, где v — это вес клетки $a[i][j]$, где $0 \leqslant i \leqslant n-1$, $0 \leqslant j \leqslant m-1$.

Исследовать и реализовать алгоритм A* для нахождения наименьшего по весу пути от исходной вершины до конечной вершины (цели) по пересеченной местности.

Ограничения:

- Поле размером $n = 10$ и $m = 9$.
- $0 \leqslant v \leqslant 9999$

Формат входного файла

На вход подается двумерный массив (поле) со значением веса данного поля, координаты исходной точки и координаты конечной точки. Данные хранятся в файле input.txt и формируются пользователем самостоятельно.

Формат выходного файла

Выходной файл должен содержать координаты найденного пути, записанный в столбик и суммарный вес.

Примеры тестов

№	Входной файл (input.txt)	Выходной файл (output.txt)
1	8 9 5 5 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 2 9 1 1 1 1 1 1 1 5 3 9 1 1 1 1 1 1 1 1 4 9 1 1 1 1 1 1 1 1 1 9 1 1 1 1 1 9 9 9 9 9 1 1 1 1 1 9 1 1 1 1 1	Путь найден (8;9) (7;9) (6;9) (5;9) (4;9) (3;9) (2;8) (2;7) (3;6) (4;5) (5;5) Стоимость пути равна 11
2	100 100 5 5 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 2 9 1 1 1 1 1 1 1 5 3 9 1 1 1 1 1 1 1 1 4 9 1 1 1 1 1 1 1 1 1 9 1 1 1 1 1 9 9 9 9 9 1 1 1 1 1 9 1 1 1 1 1	Некорректные данные
3	5 5 5 5 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 2 9 1 1 1 1 1 1 1 5 3 9 1 1 1 1 1 1 1 1 4 9 1 1 1 1 1 1 1 1 1 9 1 1 1 1 1 9 9 9 9 9 1 1 1 1 1 9 1 1 1 1 1	Начало совпадает с концом

Литература

[1] Владимир Моженков - "Алгоритм посика A*"

[2] Wikicsu.ru - "A* search algorithm"

[3] Tproger.ru - "Как реализовать алгоритм A* для поиска пути"

[4] rc74.ru - "Введение в алгоритм A*"

[5] russianblog.com - "Принцип алгоритма планирования пути A *"

[6] wikibooks.org -"Алгоритм поиска A*"

[7] github.io - "Алгоритм A* для новичков"

[8] pmg.org - "Алгоритмы поиска пути."

[9] cryptor.net - "Поисковый алгоритм A-Star (A*)"

[10] itnan.ru - "Введение в алгоритм A*"

[11] hmong.ru - "Алгоритм поиска A*"

[12] spravochnick.ru - "Взвешенный граф"

[13] stackoverfow.com -"Алгоритмы поиска пути A*"

[14] wikipedia.org - "Алгоритм поиска пути"

[15] Лорьер Жан-Луи - "Системы искусственного интеллекта"1991г.

[16] dl.acm.org - "Generalized best-first search strategies and the optimality of A*"

[17] psychology-pedagogy.academic.ru - "Эвристическая функция"

[18] codernet.ru - "Поисковый алгоритм A Star: что это и как эффективно его использовать?"

[19] habr.com - "Введение в алгоритм A*"

[20] academic.ru - "Алгоритм поиска A*"

[21] theory.stanford.edu -"Introduction to A*"

[22] Алекс Барыжков - "Алгоритм поиска пути A стар"

[23] GameDevRu - "Поиск пути в играх. Алгоритм поиска пути A*"

[24] hsrobbob - "Алгоритм поиска A*"

[25] wiki5.ru - "Шейки робот"

[26] tproger.ru - "Евклидова, L1 и Чебышёва — 3 основные метрики, которые пригодятся в Data Science"

[27] Marco Pinter - "К более реалистичному поиску пути"