ПРАВИТЕЛЬСТВО РОССИЙСКОЙ ФЕДЕРАЦИИ
ФЕДЕРАЛЬНОЕ ГОСУДАРСТВЕННОЕ АВТОНОМНОЕ
ОБРАЗОВАТЕЛЬНОЕ УЧРЕЖДЕНИЕ
ВЫСШЕГО ПРОФЕССИОНАЛЬНОГО ОБРАЗОВАНИЯ
«НАЦИОНАЛЬНЫЙ ИССЛЕДОВАТЕЛЬСКИЙ УНИВЕРСИТЕТ
“ВЫСШАЯ ШКОЛА ЭКОНОМИКИ”»

Московский институт электроники и математики
Департамент компьютерной инженерии
Вдовкин Василий Алексеевич

Разработка программы для поиска кратчайшего пути на карте с учетом препятствий

Междисциплинарная курсовая работа по направлению подготовки
09.03.01 «Информатика и вычислительная техника»
студент группы № ИВТ-11
(образовательная программа «Информатика и вычислительная техника»).

Научный руководитель
Ассистент Романов Александр Юрьевич

Москва 2015

Содержание

1 Аннотация

1.1 Русский

1.1.1 Основная цель

Исследование алгоритмов, позволяющих определить кратчайший путь между двумя вершинами в графе. Их практическое применение на примере карты с препятствиями.

1.1.2 Задачи

Реализация программы на любом языке программирования, визуализирующей работу рассматриваемых алгоритмов поиска кратчайшего пути.

1.1.3 Результаты

Реализована программа на языке JavaScript, позволяющая:

- Найти кратчайший путь между двумя точками на карте, если он существует.
- Оценить время работы алгоритма, количество обработанных вершин, существование оптимального пути и его длину.
- Изучить и сравнить алгоритм A* и алгоритм Дейкстры на основе визуализации их работы.
- Изменять размер карты и её топологию.
- Выбирать начальную и конечную точку, если они не являются препятствиями.

1.1.4 Рекомендации

- Имеется начальная карта с размещёнными препятствиями, начальной точкой и конечной точкой пути.
- Начальная и конечная точки должны находятся внутри карты.
- Конечная точка должна быть пустой.

1.2 English

1.2.1 The main objective

Исследование алгоритмов, позволяющих определить оптимальный путь между двумя вершинами в графе.

1.2.2 Tasks

Реализация программы на любом языке программирования, визуализирующей работу рассматриваемых алгоритмов поиска оптимального пути.

1.2.3 Results

Реализована программа на языке JavaScript, позволяющая:

- Найти кратчайший путь между двумя точками на карте, если он существует.
- Оценить время работы алгоритма, количество обработанных вершин, существование оптимального пути и его длину.
- Изучить и сравнить алгоритм A* и алгоритм Дейкстры на основе визуализации их работы.
- Изменять размер карты и её топологию.
- Выбирать начальную и конечную точку, если они не являются препятствиями.

1.2.4 Recomendations

- Имеется начальная карта с размещёнными препятствиями, начальной точкой и конечной точкой пути.
- Начальная и конечная точки должны находятся внутри карты.
- Конечная точка должна быть пустой.

2 Введение

Поиск пути является одной из важнейших задач в теории графов. Решение данной задачи имеет широкое практическое применение в современных технологиях. В любой сфере разработки, в которой рабочее пространство можно представить в виде графа, реализация поиска пути является одним из основных аспектов.

Представление сети дорог ориентированным графом с положительными весами и возможностью изменения веса ребёр позволяет разработчикам картографических сервисов решить данную задачу с учетом расположения физических объектов, длины дорог, их типа, проходимости, наличия пробок. Алгоритмы маршрутизации, с помощью которых информация находит свой путь от одного устройства к другому, также основываются на теории графов и задаче поиска пути. Создание искусственного интелекта во многих играх не обходится без поиска пути между объектами на игровой карте.

Первые алгоритмы, позволяющие оптимально решить данную задачу, начали появляться в конце 50-ых — начале 60-ых годов XX века. Один из самых известных, алгоритм Дейкстры, был изобретён в 1959 и стал основой для многих последующих алгоритмов поиска пути. В 1968 году появилось его улучшение: алгоритм A*, который также приобрёл широкую популярность за счёт более быстрой работы в условиях определённости конечной точки.

Необходимость ускорения и оптимизации работы алгоритмов поиска пути вызвало появление огромного количество их реализаций с различными структурами хранения вершин.

3 Основная часть

3.1 Алгоритмы

На сегодняшний день существует большое количество алгоритмов поиска пути, в этой работе будут рассматриваться алгоритм Дейкстры (англ. Dijkstra's algorithm) и A* (англ. A star). Алгоритмы применимы во взвешенном графе, все рёбра которого неотрицательны. Примером такого графа является сеть дорог или сетка.

3.1.1 Алгорим Дейкстры

Данный алгоритм находит кратчайший путь от стартовой вершины до любой другой в графе. Определённость конечной вершины во время работы алгоритма не требуется. Введём обозначения:

- $G = (V, E)$ — взвешенный ориентированный граф, где V — множество вершин, а E — множество рёбер.
- s — исходная вершина, v — текущая вершина, для которой кратчайший путь уже расчитан, u — вершина, имеющая общее ребро с текущей.
- $\pi[v]$ — родительская вершина.
- S — множество уже обработанных вершин.
- $w(u, v)$ — весовая функция, возвращает вес ребра uv.
- $g[u]$ — максимальная оценка кратчайшего пути из s в u.
- $g[v]$ — длина кратчайшего пути из s в v.

Входные данные: $G = (V, E), s$.

Выходные данные: $G = (V, E), \forall v \in V : g[v], \pi[v]$.

Опишем алгоритм с помощью псевдокода [1].

Путь от стратовой клетки до самой себя равен нулю, поэтому в первой строчке мы приравниваем $g[s]$ к нулю, родителя $\pi[s]$ у данной клетки нет.

В строчках 2-4, происходит инициализация параметров всех вершин, кроме стартовой. Для удобства считаем $g[v]$ бесконечно большим числом.

Далее, вводятся два множества Q и S. Первое множество содержит обрабатываемые вершины, а второе те, для которых кратчайший путь уже

Алгоритм 1: Псевдокод алгоритма Дейкстры

1 $g[s] \leftarrow 0, \pi[s] \leftarrow nil$
2 **for** $u \in V, u \neq s$ **do**
3     $g[u] \leftarrow \infty$
4     $\pi[u] \leftarrow nil$
5 $Q \leftarrow \{s\}$
6 $S \leftarrow \emptyset$
7 **while** $Q \neq \emptyset$ **do**
8     $v \leftarrow Extract - Min(Q, g[v])$
9     $S \leftarrow S \cup \{v\}$
10   **for** $v \notin S, uv \in E$ **do**
11   **if** $g[u] > g[v] + w(u, v)$ **then**
12             $g[u] \leftarrow d[v] + w(u, v)$
13             $\pi[u] \leftarrow v$
14             $Q \leftarrow Q \cup \{v\}$

расчитан.

Пока множество Q не пустое, из него будет переноситься в множество S вершина с наименьшим $g[v]$.

Далее, в строках 12-14, для каждой смежной с ней вершиной u, не входящей в множество S, пременяется основной метод алгоритма — релаксация. Эта техника заключается в хранении для каждой вершины u максимальной оценки кратчайшего пути $g[u]$ из s в u. Если из вершины u кратчайший путь $g[u]$ больше при проходе через её родителя, то производится пересчёт кратчайшего пути в u через вершину-родителя v, то есть значение $g[u]$ уменьшается до $g[v]+w(u, v)$. Вершина u добавляется в множество Q. Если Q является очередью, то повторно добавлять u в очередь нельзя.

После окончания работы алгоритма у каждой вершины, из которой возможен путь к стартовой, будет определён родитель $\pi[v]$ и длина кратчайшего пути $g[v]$. Чтобы пройти по вершинам, составляющим кратчайший путь необходимо двигаться от дочерней вершины к родительской, пока она не станет равна стартовой. Если же $\pi[v]$ неопределена, то найти кратчайший путь из вершины s в v нельзя. Такое возможно только в случае несвязанного графа.

Так как вершина после попадания в множество S больше не обрабатывается, то количество итерации цикла while будет равно V в случае связанного графа и меньше V в случае несвязанного графа.

3.1.2 Алгоритм A*

Данный алгоритм находит кратчайший путь от стартовой вершины s до заданной a.

Алгоритм эквивалентен алгоритму Дейкстры с добавлением эвристики и раннего выхода при достижении заданной вершины. Допустимая эвристическая оценка $h[u]$ — примерное расстояние от вершины u до заданной a, непереоценивающая реальное расстояние, тогда $f[v] = g[v] + h[v]$ — примерное расстояние от s до a.

Отличия A* от алгоритма Дейкстры показаны на псевдокоде [2]. Так же показана работа функции pathTo, позволяющая получить список вершин, по которым проходит кратчайший путь.

Входные данные: $G = (V, E), s, a$.

Выходные данные: $G = (V, E), g[a], \pi[a]$.

Тепрь текущая вершина v выбирается по минимальному значению $f[v]$, вычисляемому в 22 строчке для всех клеток, кроме стартовой, в множестве Q. Очевидно, что при $h[v] = 0$ A* превращается в алгоритм Дейкстры.

В строчках 16-17 реализован ранний выход.

Алгоритм 2: Псевдокод алгоритма A*

```
function pathTo(v)
    path ← ∅
    while ∃π[v] do
        path ← path ∪ {v}
        v ← pi[v]
    return reverse(path)
g[s] ← 0, π[s] ← nil, f[s] = 0 + h[s]
for u ∈ V, u ≠ s do
    g[u] ← ∞
    π[u] ← nil
Q ← {s}
S ← ∅
while Q ≠ ∅ do
    v ← Extract-Min(Q,f[v])
    S ← S ∪ {v}
    if v = a then
        exit pathTo(v)
    for v ∉ S, uv ∈ E do
        if g[u] > g[v] + w(u, v) then
            g[u] ← d[v] + w(u, v)
            π[u] ← v
            f[v] ← g[v] + h[v]
            Q ← Q ∪ {v}
```

3.2 Реализация

3.2.1 Список используемых технологий

3.2.2 Выбора языка программирования и библиотек

3.2.3 Особенности реализации алгоритмов

3.2.4 Пример работы алгоритмов

3.2.5 Реализация интерфейса

4 Заключение

Список литературы

[1] Кормен, Т., Лейзерсон, Ч., Ривест, Р. Алгоритмы: построение и анализ = Introduction to Algorithms. — 1-е. — М.: МЦНМО, 2000. — 960 с. — ISBN 5-900916-37-5.