Алгоритм A*

Чен Михаил

Содержание

1. ВВЕДЕНИЕ.....3

1.1. Глоссарий.....3

1.2. Определение алгоритма A*.....3

1.3. История.....4

1.4. Применение алгоритма.....4

2. ПОСТАНОВКА ЗАДАЧИ.....4

3. ОПИСАНИЕ АЛГОРИТМА.....4

3.1. Описание.....4

3.2. Свойства.....5

3.3. Примеры эвристик.....6

3.4. Пример работы алгоритма.....6

4. ФОРМАТ ДАННЫХ.....12

5. СРАВНИТЕЛЬНЫЙ АНАЛИЗ АЛГОРИТМА ДЕЙКСТРЫ И A*.....13

5.1. Методология сравнения.....13

5.2. Алгоритм Дейкстры.....13

5.3. Тесты.....13

Тесты (без преград).....13

Тесты (Случайные значения).....14

Тесты (сложный).....15

Тест (лабиринт).....16

Тесты (пересеченная местность).....17

5.4. Вывод.....19

ЗАКЛЮЧЕНИЕ.....19

1. Введение

1.1. Глоссарий

Алгоритм прокладывания кратчайшего 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 - первый 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.

Итерация (лат. iteratio «повторение») — повторение какого-либо действия.

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 $0 \leqslant i \leqslant n - 1$, $0 \leqslant j \leqslant m - 1$.

OSPF (англ. Open Shortest Path First) — протокол динамической маршрутизации, основанный на технологии отслеживания состояния канала (link-state technology) и использующий для нахождения кратчайшего пути алгоритм Дейкстры.

Протокол маршрутизации промежуточных систем (англ. IS-IS) — это протокол внутренних шлюзов (IGP), стандартизированный ISO и использующийся в основном в крупных сетях провайдеров услуг. IS-IS может также использоваться в корпоративных сетях особо крупного масштаба. IS-IS — это протокол маршрутизации на основе состояния каналов. Он обеспечивает быструю сходимость и отличную масштабируемость. Как и все протоколы на основе состояния каналов, IS-IS очень экономно использует пропускную способность сетей.

Местность — в ландшафтоведении — морфологическая единица ландшафта, природно-территориальный комплекс более высокого ранга, чем урочище.

Урочище — в широком смысле, любой географический объект или ориентир, о названии которого договорились («уреклись») люди.

Пересеченная местность — местность изрезанная чем либо, речками, канавами, пролесками, гребнями и прочим.

1.2. Определение алгоритма A*

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

1.3. 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 $g(n)$, рас- стояние 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.

1.4. Применение алгоритма

Алгоритм A* применяют в различных областях от навигационных карт до машинного обучения, например, в 2gis

2. Постановка задачи

Изучить и реализовать алгоритм A* для нахождения наименьшего по весу пути от исходной вершины до конечной вершины (цели) по пересеченной местности. Провести сравнительный анализ Алгоритма Дейкстры и Алгоритма A*.

3. Описание алгоритма

3.1. 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 $f(v)$.

3.2. Свойства

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

Утверждение 1: 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.

Утверждение 2: 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 3: 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 4: 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.

3.3. 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 можно 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:

$$h(v) = \sqrt{(\mathbf{v.x} - \mathbf{v'.x})^2 + (\mathbf{v.y} - \mathbf{v'.y})^2}$$

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.

3.4. Пример работы алгоритма

Процесс прокладывания наименьшего по стоимости пути продемонстрирован на рисунках 3.1 - 3.12. Задается взвешенное поле, стартовая и конечная вершины. *Рис. 3.1*

Рис. 3.1: Исходное поле

Стартовая вершина отмечается как посещенная. *Рис. 3.2*

Рис. 3.2: Шаг 1

Расчитывается эвристическая оценка пути F для не посещенных соседних вершин. *Рис. 3.3*

Рис. 3.3: Определение эвристикой оценки пути

Текущей вершиной становится вершина с наименьшей эвристической оценкой. *Рис. 3.4*

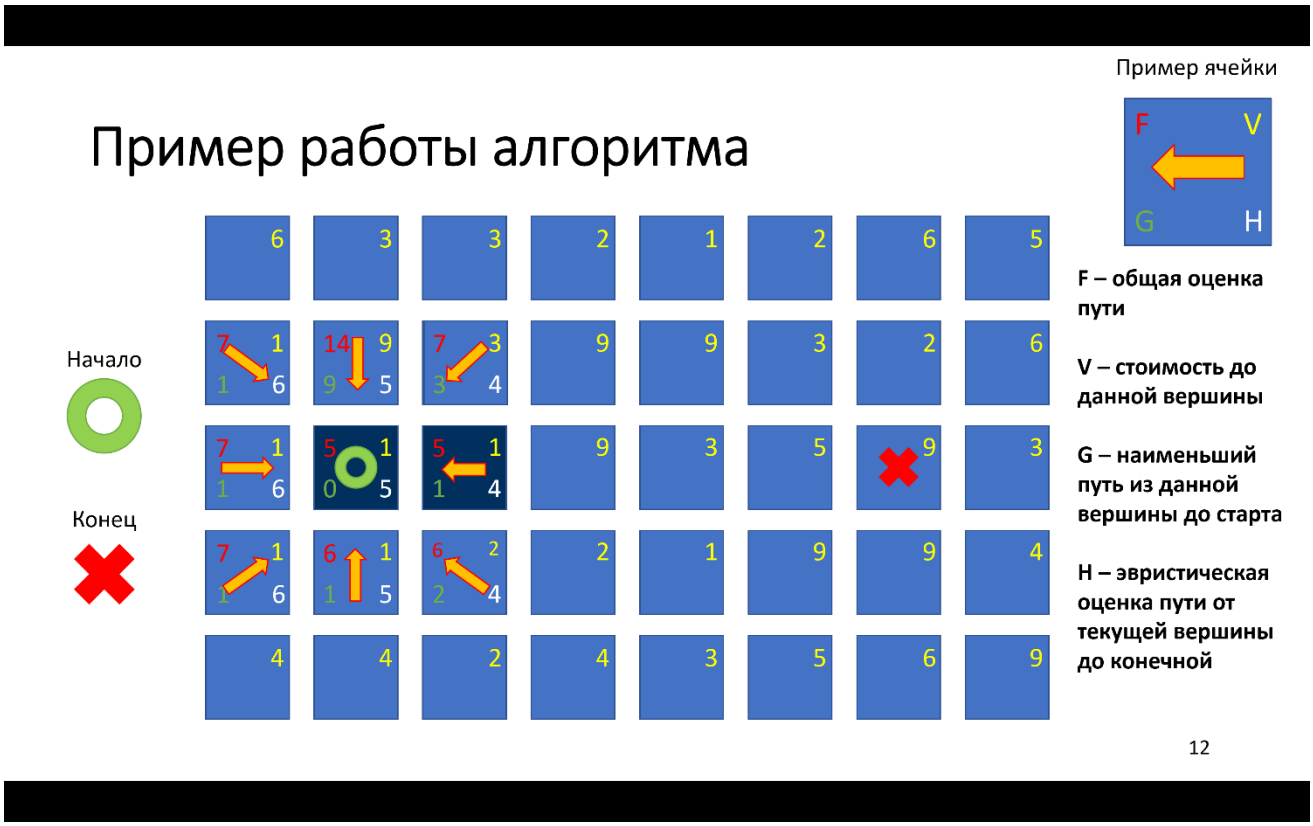

Рис. 3.4: Шаг 2

Алгоритм повторяется до тех пор, пока текущая вершина не будет совпадать с конечной. *Рис. 3.5, Рис. 3.6, Рис. 3.7, Рис. 3.8, Рис. 3.9, Рис. 3.10, Рис. 3.11, Рис.3.12*

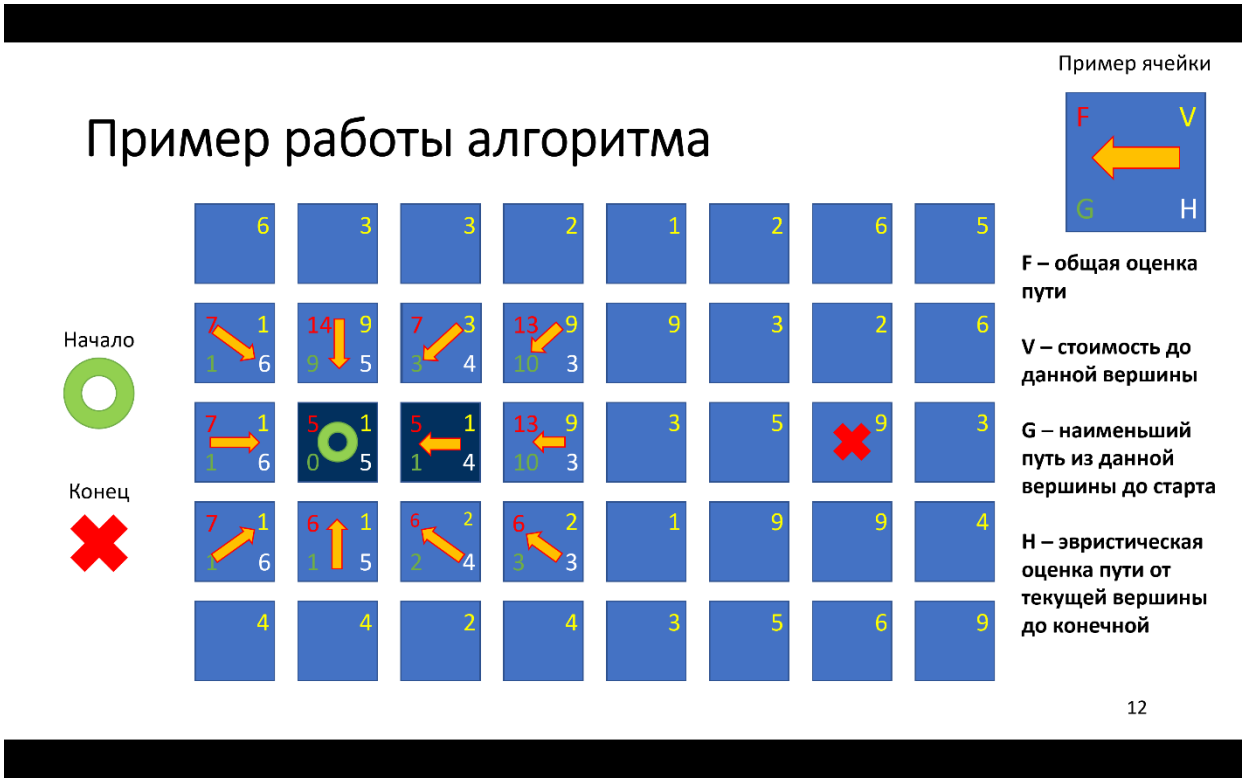

Рис. 3.5: Определение эвристикой оценки пути

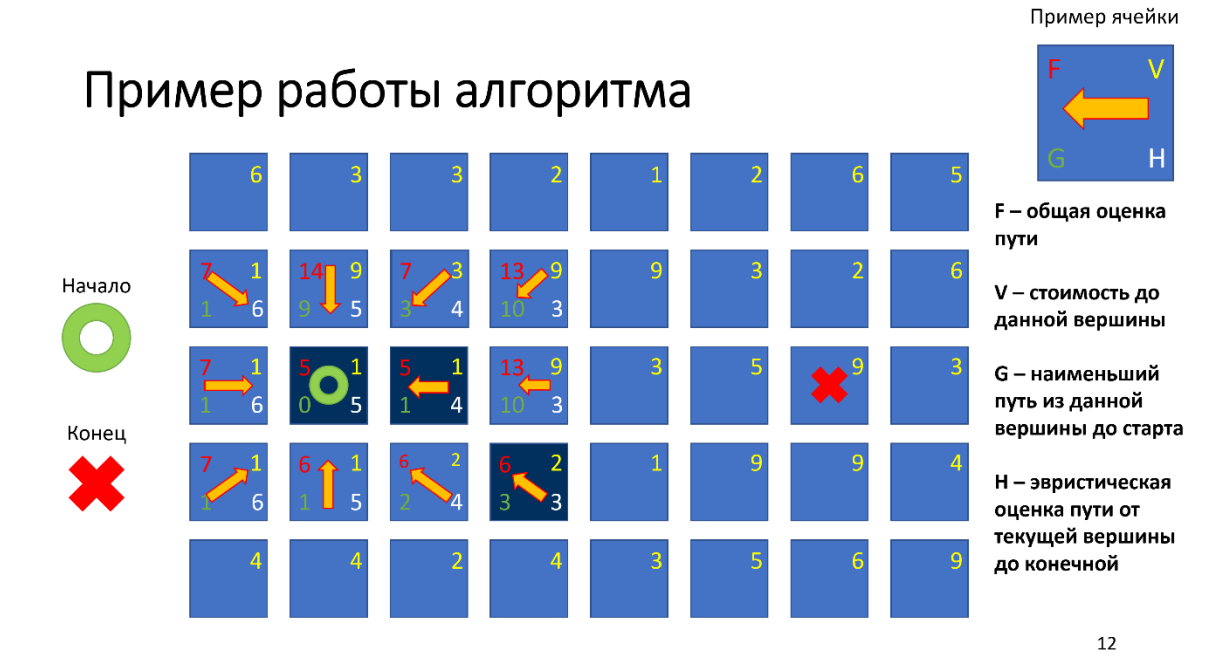

Рис. 3.6: Шаг 3

Рис. 3.7: Определение эвристикой оценки пути

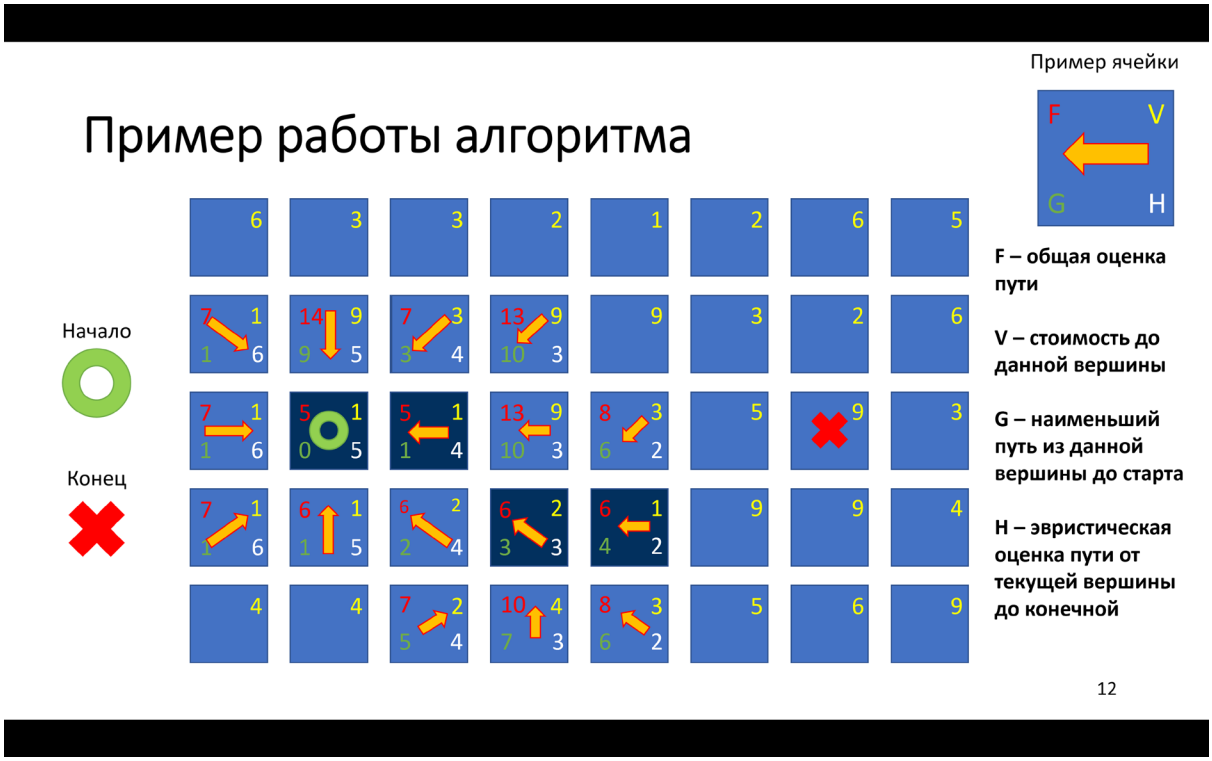

Рис. 3.8: Шаг 4

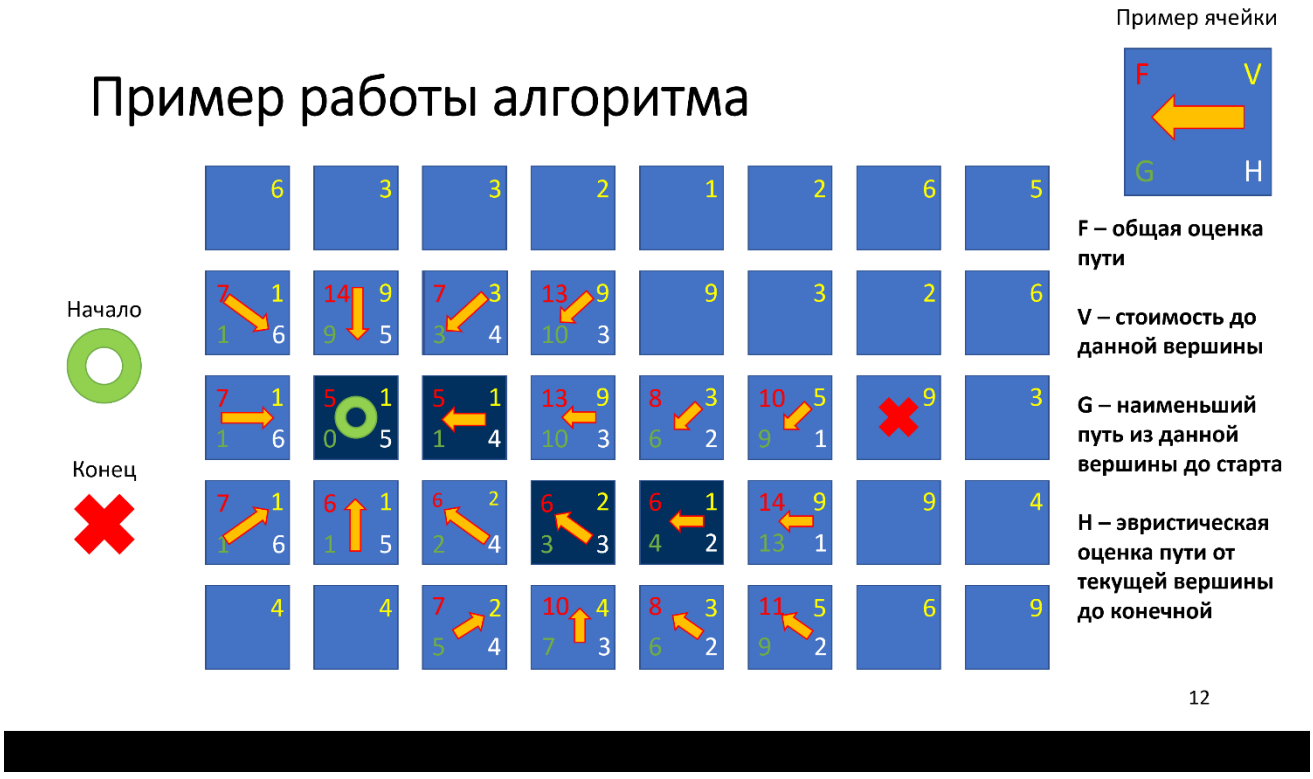

Рис. 3.9: Определение эвристикой оценки пути

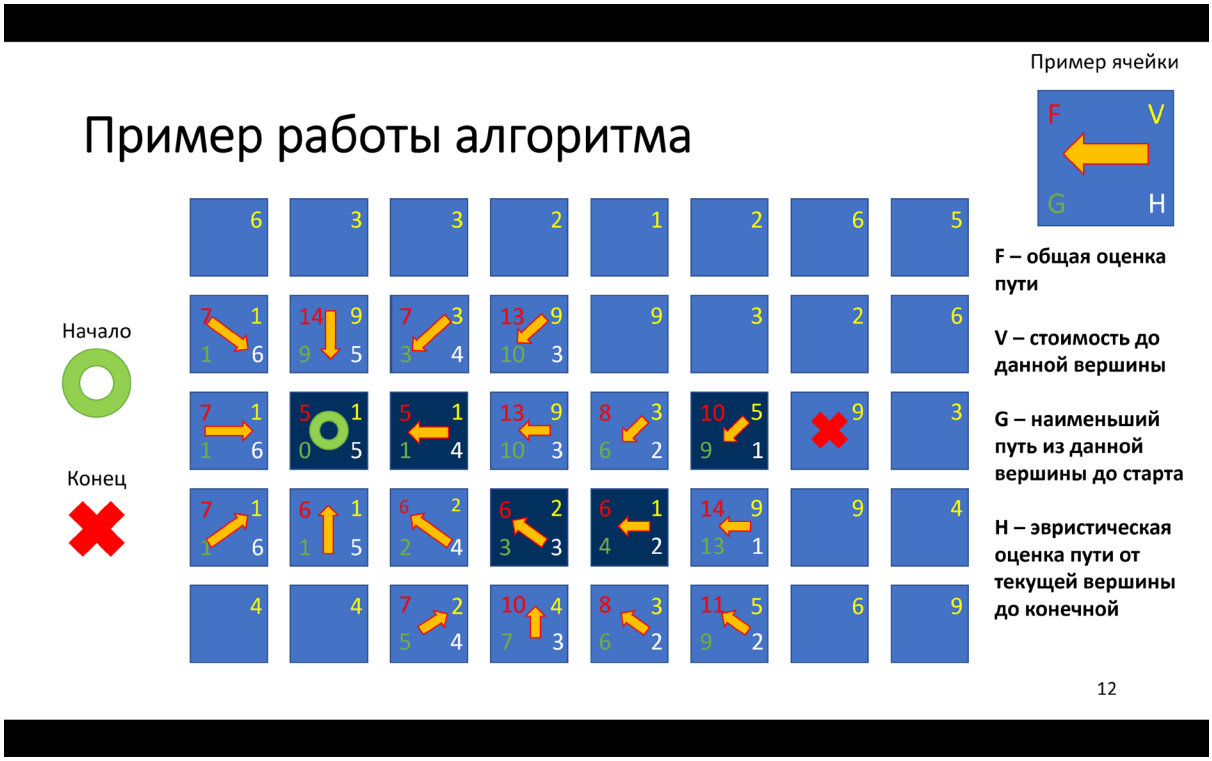

Рис. 3.10: Шаг 5

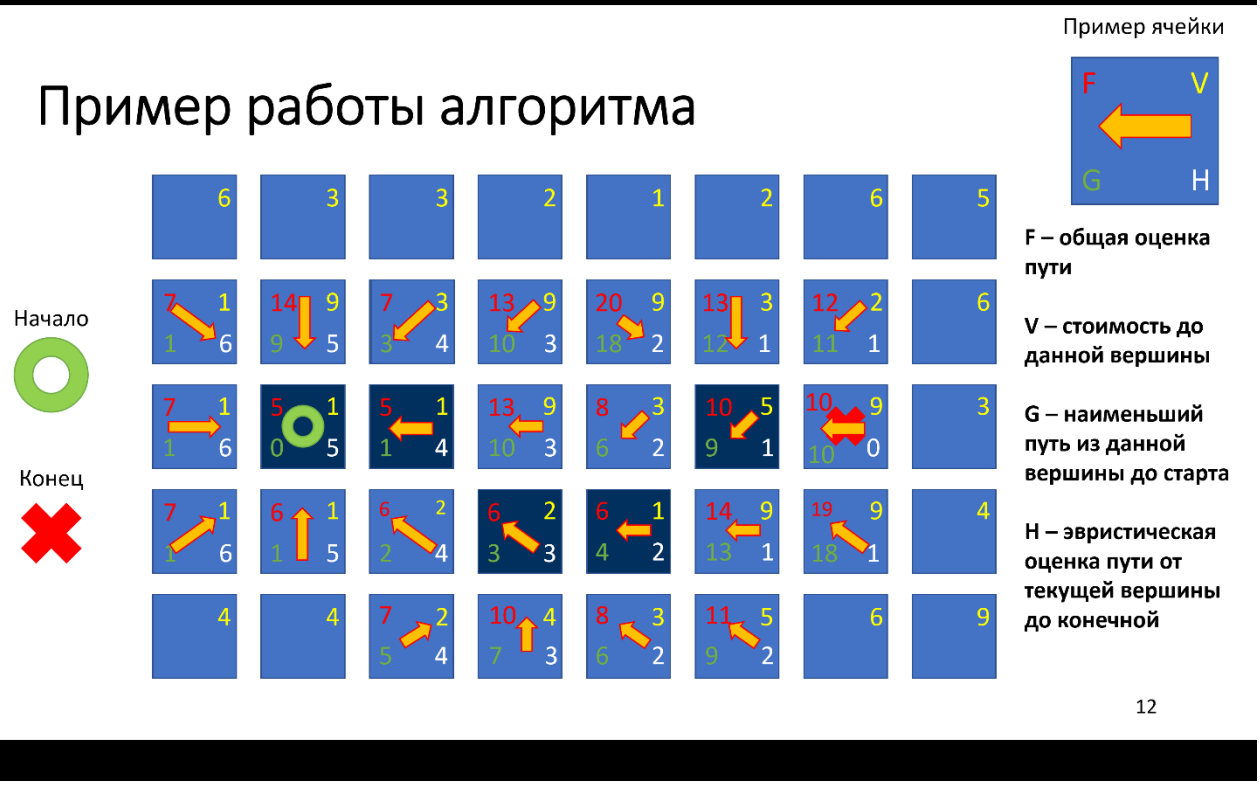

Рис. 3.11: Определение эвристикой оценки пути

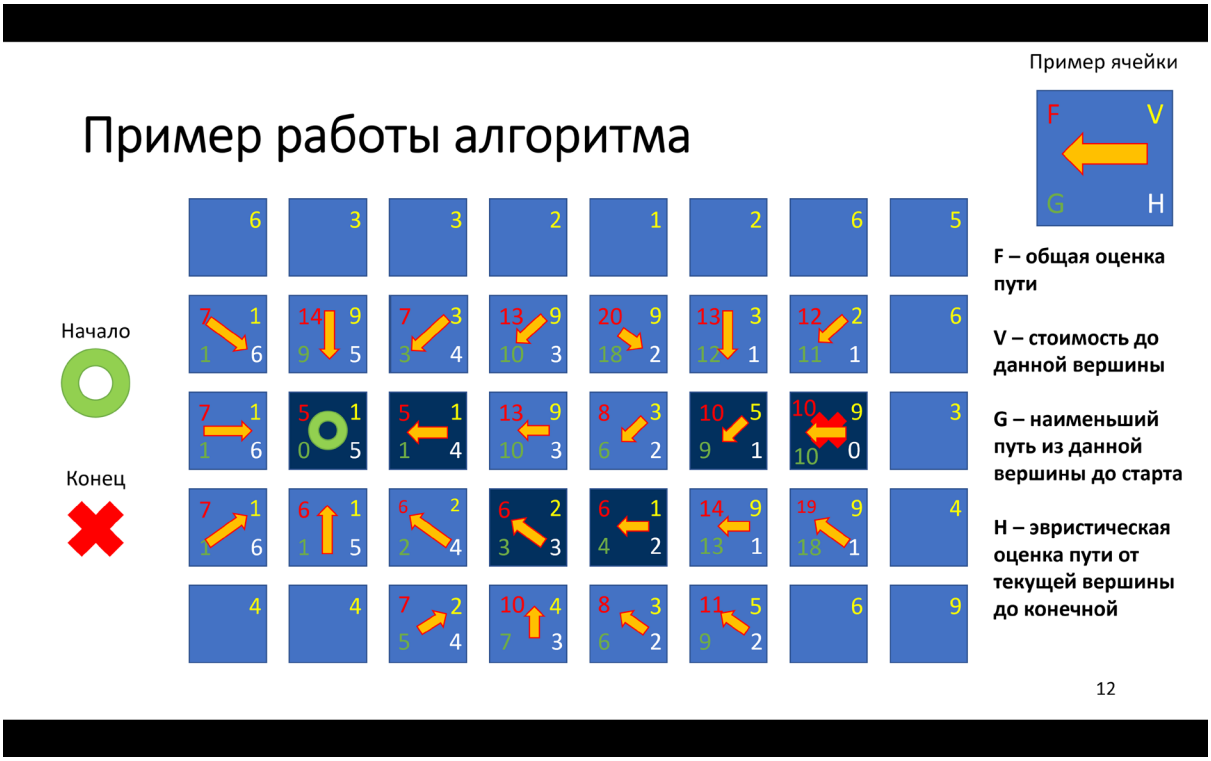

Рис. 3.12: Маршрут проложен

4. Формат данных

Ограничения:

Поле размером $n \leqslant 10000$ и $m \leqslant 10000$.

$0 \leqslant v \leqslant 1000000$

Формат входного файла

На вход подаются размер двумерного массива, координаты исходной точки и координаты конечной точки и двумерный массив (поле) со значениями веса данного поля. 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)**
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 2 9 1 1 1 1 1 1 5 3 9 1 1 1 1 1 1 1 4 9 1 1 1 1 1 1 1 1 9 1 1 1 1 9 9 9 9 9 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

5. Сравнительный анализ Алгоритма Дейкстры и A*

5.1. Методология сравнения

Алгоритм A* будет сравнен с алгоритмом Дейкстры. Тесты учитываются исходя из размеров поля, сложности рельефа поля, сложности маршрута и удаленности начальной и конечной вершин. Так же будут проводится по 2 контрольных теста для независимости от технических помех.

5.2. Алгоритм Дейкстры

Алгоритм Дейкстры — алгоритм на графах, изобретённый нидерландским учёным Эдсгером Дейкстрой в 1959 году. Находит кратчайшие наименьшего по стоимости пути от одной из вершин графа до всех остальных. Алгоритм работает только для графов без рёбер отрицательного веса. Алгоритм широко применяется в программировании, например, его используют протоколы маршрутизации OSPF и IS-IS.

5.3. Тесты

Тесты (без преград)

На вход подаются: поле размером 10000*10000, заполненный 1, а также стартовая вершина (0;0) и конечная вершина (9999;9999).

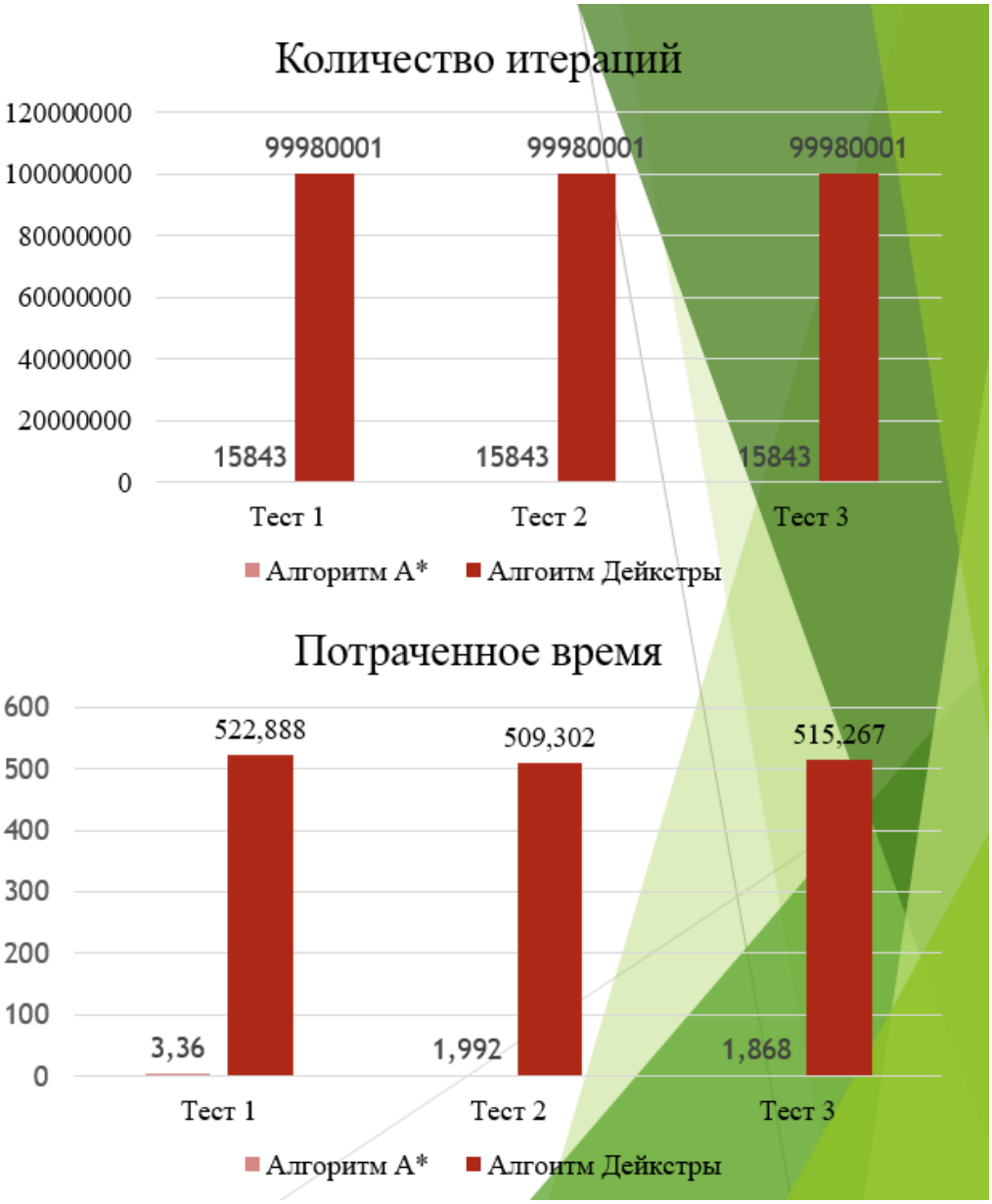

Рис. 5.1: Тест (без преград)

Исходя из исследования видно, что в алгоритме A* меньше итераций чем в алгоритме Дейкстры на 630933% и быстрее на 21368%. Это обусловлено тем, что алгоритму Дейкстры необходимо проверить все вершины, а алгоритм A* проверяет то, что находится в сторону конечной вершины.

Тесты (Случайные значения)

На вход подаются: поле размером 10000*10000, заполненный 1000000, а также стартовая вершина (0;0) и конечная вершина (9999;9999).

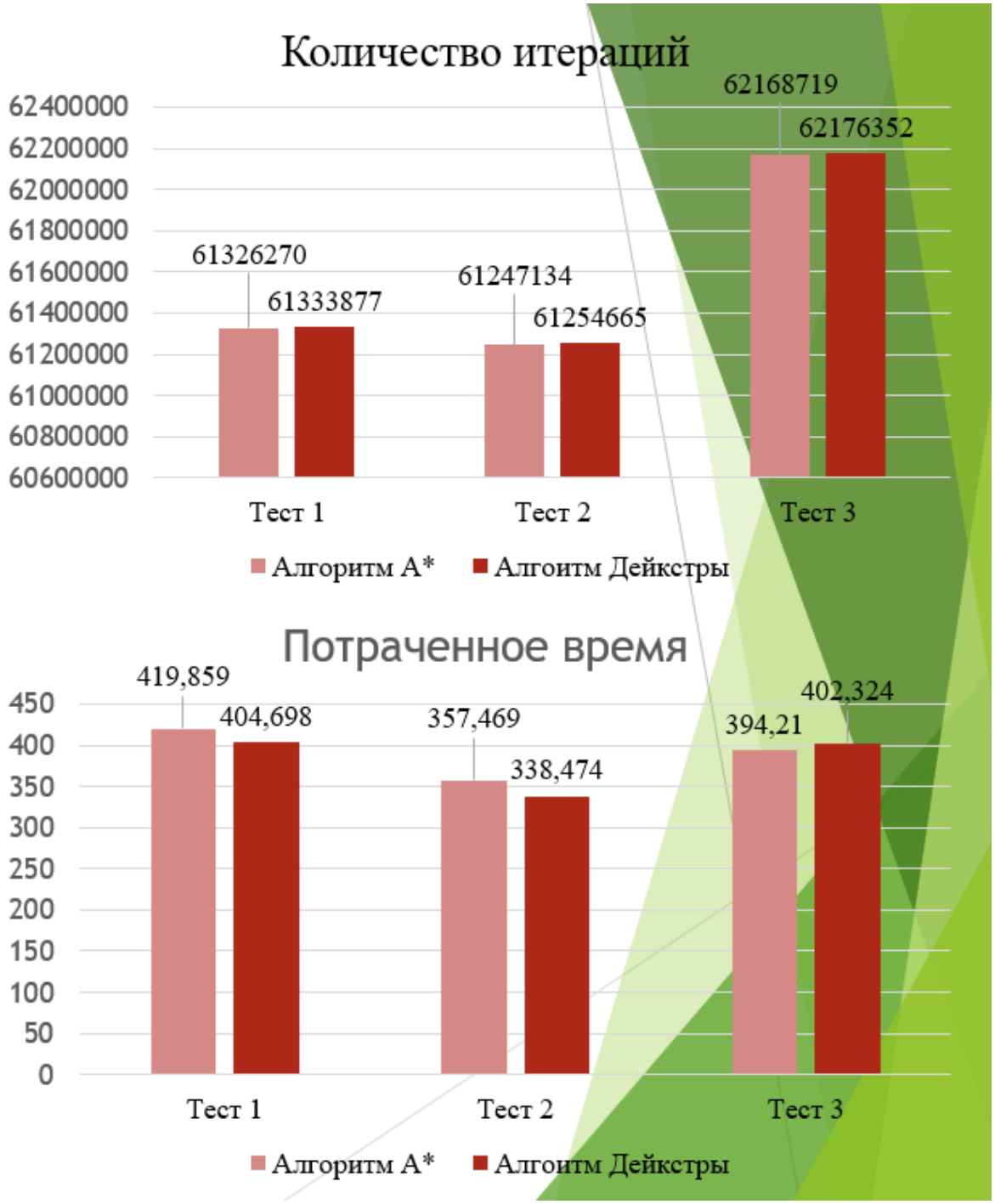

Рис. 5.2: Тест (случайные значения)

В среднем разница не велика и может ссылаться на погрешность. Результат зависит от сложности поля.

Тесты (сложный)

На вход подаются: поле размером 10000*10000, заполненный 1, а также стартовая вершина (0;0) и конечная вершина (9999;9999).

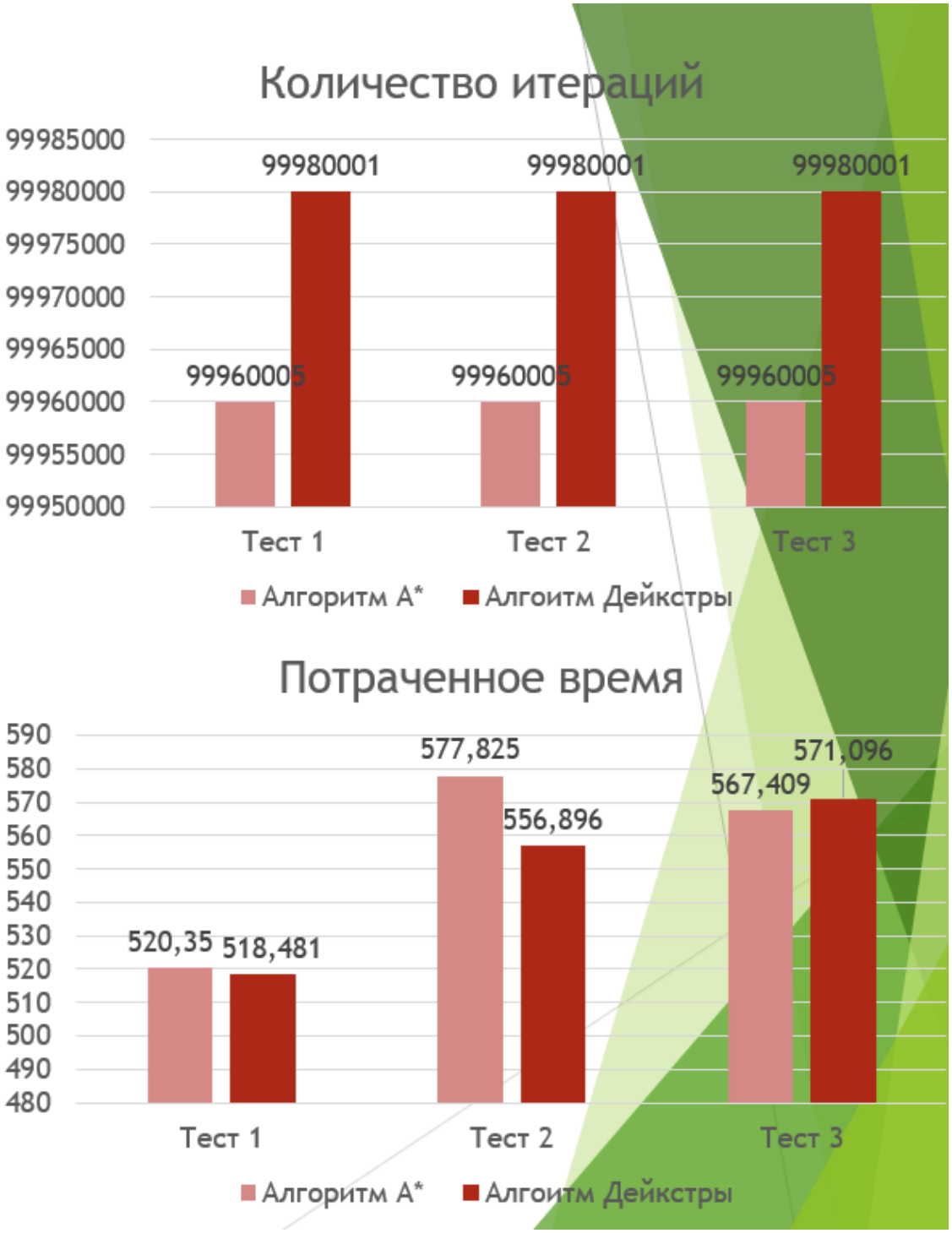

Рис. 5.3: Тест (труднопроходимый)

В среднем разница не велика и может ссылаться на погрешность. Результат обусловлен тем, что Алгоритм А*, в поисках кратчайшего пути, посещает все вершины как алгоритм Дейкстры.

Тест (лабиринт)

На вход подаются: поле размером 20*20, поле представляет собой лабиринт, а также стартовая вершина (1;1) и конечная вершина (18;18).

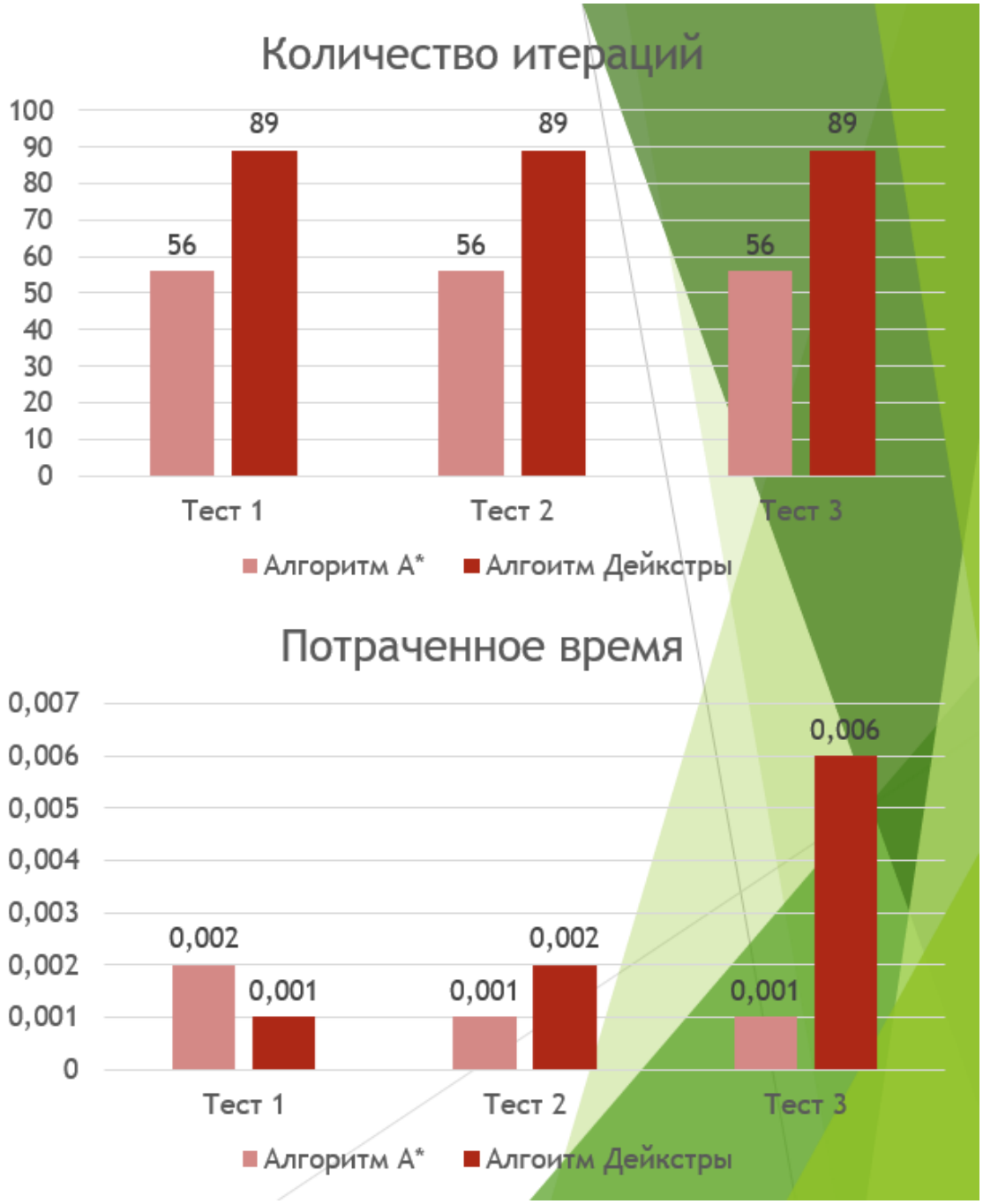

Рис. 5.4: Тест (лабиринт)

Исходя из исследования видно, что в алгоритме А* меньше итераций чем в алгоритме Дейкстры на 38% и быстрее на 65%. Результат зависит от сложности лабиринта. Если Алгоритм А* сможет найти выход быстро, то Алгоритм А* будет лучше, чем Алгоритм Дейкстры.

Тесты (пересеченная местность)

На вход подаются: поле размером 20*20, поле представляет собой пересеченную местность, а также стартовая вершина (0;0) и конечная вершина (29;29).

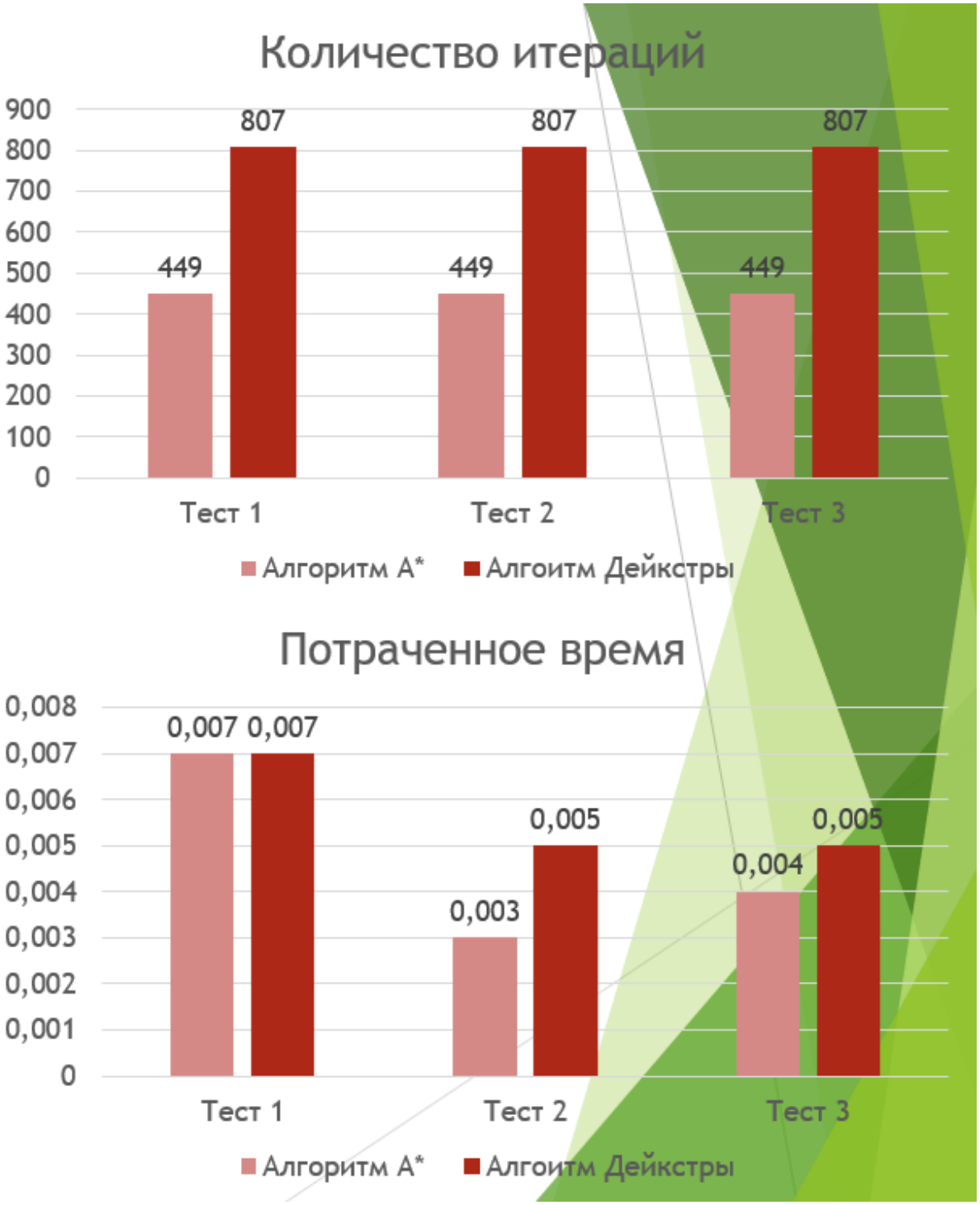

Рис. 5.5: Тест (пересеченная местность

Исходя из исследования видно, что в алгоритме A* меньше итераций чем в алгоритме Дейкстры на 44% и быстрее на 17%. Результат обусловлен тем, что алгоритм A* стремится к конечной точке, когда алгоритм Дейкстры посещает все доступные вершины

5.4. Вывод

Если требуется посещение всех вершин, то предпочтительнее будет использовать алгоритм Дейкстры, но если требуется найти путь к одной вершине, то предпочтительнее будет использовать алгоритм A*.

Заключение

В ходе данной работы, изучен и реализован на языке "c++"алгоритм A* с визуализатором для полей до значений 10000*10000 со значениями весов до 1000000. Проведено тестирование на 37 тестах, выполнено исследование на сравнительный анализ с алгоритмом Дейкстры.

Список литературы

[1] https://ru.wikipedia.org/wiki/A*
[2] https://habr.com/ru/post/331192/
[3]https://ru.wikipedia.org/wiki/%D0%9F%D0%BE%D0%B8%D1%81%D0%BA_%D0%BF%D1%83%D1%82%D0%B8
[4] https://habr.com/ru/post/444828/
[5] https://dtf.ru/gamedev/709133-poisk-puti-ili-kak-vragi-v-igrah-nahodyat-dorogu
[6] https://www.youtube.com/watch?v=gCclsviUeUk&ab_channel=GameDevRu
[7]https://www.youtube.com/watch?v=AsEC2TJZ3JY&ab_channel=VolodyaMozhenkov
[8]https://ru.wikipedia.org/wiki/%D0%90%D0%BB%D0%B3%D0%BE%D1%80%D0%B8%D1%82%D0%BC_%D0%94%D0%B5%D0%B9%D0%BA%D1%81%D1%82%D1%80%D1%8B
[9] https://blog.skillfactory.ru/glossary/algoritm-dejkstry/
[10] https://habr.com/ru/post/111361/
[11]https://neerc.ifmo.ru/wiki/index.php?title=%D0%90%D0%BB%D0%B3%D0%BE%D1%80%D0%B8%D1%82%D0%BC_%D0%94%D0%B5%D0%B9%D0%BA%D1%81%D1%82%D1%80%D1%8B
[12] https://ru.wikipedia.org/wiki/OSPF
[13] https://ru.wikipedia.org/wiki/IS-IS
[14] https://ru.wikipedia.org/wiki/Shakey