# 1 Программная реализация поиска пути в лабиринте

Программная реализация поиска пути в лабиринте — это интересная и важная задача, которая используется во многих областях, таких как навигация, планирование транспортных маршрутов, сортировка, робототехника и искусственный интеллект. Существует множество алгоритмов, которые можно использовать для решения этой задачи. Наиболее популярные из них приведены ниже.

## Алгоритм Дейкстры

Это алгоритм поиска кратчайшего пути во взвешенном графе с неотрицательными весами ребер. Алгоритм назван в честь голландского ученого Эдсгера Дейкстры, который разработал его в 1956 году.

Алгоритм Дейкстры используется для нахождения кратчайшего пути от одной вершины графа (называемой источником) до всех остальных вершин. За основу алгоритма берется принцип жадного выбора: на каждом шаге выбирается вершина с наименьшим известным расстоянием от источника и проверяются все её соседние вершины. Если расстояние до соседней вершины через текущую вершину оказывается короче, то расстояние обновляется.

Однако этот алгоритм подходит только для графов без отрицательных ребер, так как при наличии отрицательных циклов может возникнуть бесконечный цикл обновления расстояний.

Алгоритм описывается следующим образом:

* В начале алгоритма расстояние для начальной вершины полагается равным нулю, а все остальные расстояния заполняются большим положительным числом (бо́льшим максимального возможного пути в графе).
* Массив флагов заполняется нулями. Затем запускается основной цикл.
* На каждом шаге цикла мы ищем вершину v с минимальным расстоянием и флагом равным нулю. Затем мы устанавливаем в ней флаг в 1 и проверяем все соседние с ней вершины u. Если в них (в u) расстояние больше, чем сумма расстояния до текущей вершины и длины ребра, то уменьшаем его.
* Цикл завершается, когда флаги всех вершин становятся равны 1, либо когда у всех вершин c флагом 0 Последний случай возможен тогда и только тогда, когда граф G несвязный.

Работа алгоритма показана на блок-схеме (Рисунок 1 )

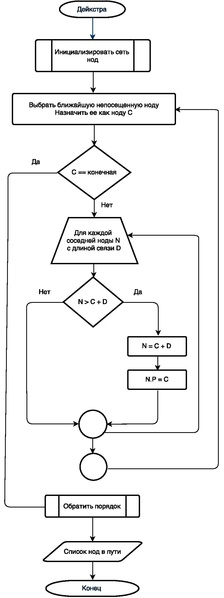


Рисунок 1 – Блок-схема алгоритма Дейкстры.

## 1.2 Алгоритм A\*

Является модификацией алгоритма Дейкстры, который использует эвристику для ускорения процесса поиска. Он работает путем оценки стоимости каждого возможного шага и выбора того, который, как предполагается, приведет к наиболее эффективному пути.

Начните с начальной вершины и установите ее оценочную стоимость равной эвристической оценке расстояния до цели.

Пока есть не посещенные вершины:

Выберите вершину с наименьшей оценочной стоимостью и пометьте ее как посещенную.

Если это целевая вершина, то путь найден.

В противном случае для каждой соседней вершины обновите ее оценочную стоимость, если текущая оценка больше суммы стоимости пути до выбранной вершины, веса ребра между ними и эвристической оценки до цели.

## 1.3 Поиск в ширину (BFS)

Начинается с корневого узла и исследует все соседние узлы на данном уровне перед переходом к узлам следующего уровня. Ключевая идея заключается в том, что мы отслеживаем состояние расширяющегося кольца, которое называется границей. В сетке этот процесс иногда называется заливкой (flood fill), но та же техника применима и для карт без сеток.

Начните с корневого узла и добавьте его в очередь.

Пока очередь не пуста:

Извлеките узел из очереди и пометьте его как посещенный.

Добавьте все не посещенные соседние узлы в очередь.

## 1.4 Поиск в глубину (DFS)

В отличие от BFS, алгоритм DFS идет как можно глубже в граф, пока не достигнет конца, а затем возвращается назад, чтобы исследовать другие ветви.

Начните с корневого узла и добавьте его в стек.

Пока стек не пуст:

Извлеките узел из стека и пометьте его как посещенный.

Добавьте все не посещенные соседние узлы в стек.

## 1.5 Jump Point Search (JPS) –

Является оптимизацией алгоритма поиска A\* для сеток с равномерной стоимостью. Он уменьшает симметрию в процедуре поиска путем обрезки графа, устраняя определенные узлы в сетке на основе предположений, которые можно сделать о соседях текущего узла, при условии, что выполняются определенные условия, относящиеся к сетке

## 1.6 Вывод

Большинство из них основаны на исследовании лабиринта в виде графа.