# Практика

## Описание задания

Написать алгоритм, ищущий решение в игре 8 (см. пункт 1.3).

Алгоритм находит решение путем перебора состояний. Каждое состояние представляет из себя конкретное состояние в игре (в нашем случае плиток в порядке принадлежности их квадрат).

**Необходимо** реализовать функцию построения случайного допустимого состояния (состояния являются допустимыми, если из них можно достигнуть цели см. пункт 1.2)

**Необходимо** реализовать программу поиска последовательности действий, которые необходимо совершить для приведения начального состояния к целевому состоянию (см. пункт 1.3).

**Необходимо** реализовать поиск в A\* посредством списка (периферии), добавляя в очередь новые, потенциально достижимые состояния (см. пункт 1.4).

**Необходимо** каждое состояние описать с помощью структуры узла дерева (см. пункт 1.5). (можно не указывать Parent-Node)

**Необходимо** выдать результат алгоритма в виде последовательности действий, приводящих к решению, если оно существует. Выдать особый результат в случае неудачи (пустой список действий не подойдет).

## Генерация начального состояния

Можно показать, что ровно половину из всех возможных начальных положений игры в 8 невозможно привести к целевому состоянию.

Пусть плитка с числом  расположена до (если считать слева направо и сверху вниз) плиток с числами, меньшими ; тогда обозначим . В частности, если после плитки с числом {\displaystyle i}  нет плиток с числами, меньшими , то . Также введём число – номер ряда пустой клетки (начиная с 1). {\displaystyle i}

Если сумма нечетна, то решения головоломки существует (8 заменяется на 15 в случае игры в пятнашки).

Итак, генерируем случайное состояние, если оно не удовлетворяет условию выше, генерируем заново, пока не получим подходящее начальное состояние.

### Постановка задачи игры с перемещением плиток

Поле игры в 8 состоящая из коробки 3 х 3 квадрата с восемью пронумерованными плитками и одним пустым пространством. Цель состоит в том, чтобы достичь определенного целевого состояния. Мы договоримся на таком целевом состоянии:



**Состояния** – описание состояния определяет местонахождение каждой из восьми плиток и пустого участка на игральной доске из девяти квадратов.

**Начальное состояние** – в качестве начального может быть определено любое состояние. Обратите внимание, что в этом мире свойство четности делит пространство состояний на две части: любая заданная цель может быть достигнута точно из половины возможных начальных состояний (см. пункт 1.2).

**Действия** – в то время как в физическом мире головоломка состоит из плиток, перемещаемых в пределах границ головоломки, простейший способ из описания действий в соответствующем клеточном мире - представить, что перемещается пустое поле. Соответствующие этому случаю действия вполне типичны: *left* (влево), *right* (вправо), *up* (вверх) и *down* (вниз). Если пустое поле окажется на краю сетки или в углу, не все действия будут применимы.

**Функция определения преемника** – функция, отображающая текущее состояние, в множество состояний, представляющих результат перемещения пустой плитки.

**Целевое состояние** - решение определено как состояние, когда все плитки расположены по возрастанию их номеров (см. пункт 1.3).

**Функция стоимости** - каждое перемещение плитки стоит 1 балл.

### Поиск A\*

Функция оценки (оценивает дистанцию до целевого состояния):

,

где – это стоимость пути от начального состояния до узла п, а — это расчетная стоимость кратчайшего пути от узла до целевого состояния.

= Path-Cost (см. пункт 1.5)

= оценочная стоимость наилучшего пути, который идет через узел к цели.

0) В список состояний добавляются все доступные состояния, которые можно достичь из начального состояния.

1) Из списка доступных состояний (периферии см пункт 1.6) выбирается состояние с наименьшим значением .

2) Посещается состояние с наименьшим значением , назовем состояние .

3 Добавляем в список доступных состояний (периферии см пункт 1.6) все состояния достижимые из (кроме того, откуда мы пришли).

4) Переходим к пункту 1).

**Условия остановки:**

1) на шаге 2) состояние оказалось целевым. Завершаем поиск. Возвращаем список последовательности действий, которые привели к решению.

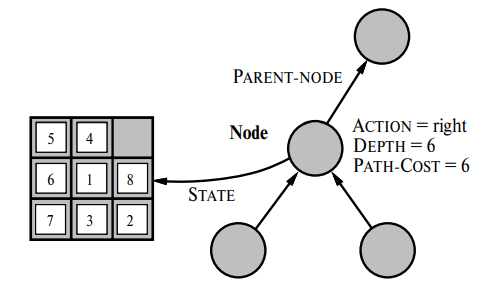
2) на шаге 1) если список доступных состояний пуст, заканчиваем поиск. Достигнуть целевого состояния невозможно.

## Описание структуры узла дерева

Существует множество способов представления узлов, но здесь предполагается, что узел представляет собой структуру данных с пятью компонентами, которые описаны ниже.

* State. Состояние в пространстве состояний, которому соответствует данный узел.
* Parent-Node. Узел в дереве поиска, применявшийся для формирования данного узла (родительский узел).
* Action. Действие, которое было применено к родительскому узлу для формирования данного узла.
* Path-Cost. Стоимость пути (от начального состояния до данного узла), показанного с помощью указателей родительских узлов, которую принято обозначать как g(n).
* Depth. Количество этапов пути от начального состояния, называемое также глубиной.

Необходимо учитывать различие между узлами и состояниями. Узел — это учетная структура данных, применяемая для представления дерева поиска, а состояние соответствует конфигурации мира. Поэтому узлы лежат на конкретных путях, которые определены с помощью указателей Parent-Node, а состояния - нет. Кроме того, два разных узла могут включать одно и то же состояние мира, если это состояние формируется с помощью двух различных путей поиска. Структура данных узла показана на рисунке.



## Создание периферии

Необходимо также представить коллекцию узлов, которые были сформированы, но еще не развернуты; такая коллекция называется **периферией**. Каждый элемент периферии представляет собой **листовой узел**, т.е. узел, не имеющий преемников в дереве.

Простейшим представлением периферии может служить множество узлов. Тогда стратегия поиска должна быть выражена в виде функции, которая выбирает определенным образом из этого множества следующий узел, подлежащий развертыванию.

## Примеры допустимых эвристик для игры в 8.

= количество плиток, стоящих не на своем месте.

= сумма расстояний всех плиток от их целевых позиций. Поскольку плитки не могут передвигаться по диагонали, это расстояние представляет собой сумму горизонтальных и вертикальных расстояний.

Если захотите более сильную эвристику, пишите)