# Алгоритмы

Шашки – игра с конечным числом состояний. Это означает, что, обладая бесконечными вычислительными ресурсами, мы смогли бы найти решение этой игры – то есть такую стратегию, следуя которой всегда можно было бы выиграть или, по крайней мере, сыграть вничью. Каждая позиция pos в шашках - это либо победа белых, либо победа черных, либо ничья. Определим функцию ***f:P→Z***, которая ставит в соответствие позиции ***pos*** из множества всевозможных позиций ***P*** целое число, отражающее «выгодность» этой позиции для текущего игрока***.*** Такая функция называется *оценочной функцией.*Если бы у нас был бесконечно быстрый компьютер, мы могли бы вычислить ее следующим образом:

* присвоим всем финальным позициям значения –1, 0, 1, в зависимости от исхода игры;
* применим рекурсивное правило

где **pos** соответствует ходу текущего игрока, **pos‘** соответствует ходу противника, а  **pos→pos‘** обозначает все допустимые ходы из позиции **pos**.

Чем больше значение функции, тем выгодней позиция. Таким образом, ставя каждому ходу в соответствие его оценку, мы бы могли однозначно определить лучший ход.

Количество [возможных позиций](http://en.wikipedia.org/wiki/Shannon_number) в шашках составляет около 5\*1020. Вычисления такого масштаба выполнить практически невозможно (канадским ученым потребовалось 200 компьютеров и 20 лет вычислений). Следовательно, находить наилучший ход нужно другим способом.

### Алгоритм полного перебора

Самым простым способом получить если не лучший, то хотя бы осмысленный ход, является метод полного перебора с возвратом в соединении с рекурсией. Если сгенерировать все возможные ходы из данной позиции для белых, затем для каждого такого хода повторить те же действия для черных, потом снова для белых и т.д., получится простой и понятный рекурсивный алгоритм, основанный на переборе позиций. Пример дерева игры, возникающего таким образом, представлен на рис.х,х1.

Однако, как уже было сказано, рассмотреть все возможные варианты до конца невозможно, поэтому приходится ограничивать глубину рекурсии. Пусть на каждом шаге известна текущая глубина рекурсии – параметр ***depth***, первоначально равный максимальной заданной глубине. Тогда если ***depth***≠0, то рекурсия будет вызываться с параметром ***depth-1***, в противном случае вычисления следует прекратить и вызвать *оценочную функцию[[1]](#footnote-2)*.

Блок-схема алгоритма полного перебора представлена на рис.х.

### Оценочная функция

### Пусть P - множество всевозможных позиций на доске. Функция f:P→Z, ставящая в соответствие некоторой позиции из множества P целое число, отражающее «выгодность» этой позиции для текущего игрока, называется оценочной функцией.

### Простейшая оценочная функция считает шашки на доске, т.е. суммирует вес всех белых шашек и вычитает из полученного результата сумму всех черных шашек. Если шашек поровну, функция вернет 0.

### Установим вес шашек следующим образом:

* простая шашка – 50;
* дамка – 350.

Таким образом, простая оценочная функция при расстановке шашек как на рис. для белого игрока вернет число 250, для черного игрока -250.

Более сложная оценочная функция учитывает ценность полей доски, которая определяется следующими принципами:

### надо стараться по возможности подвигаться вперед, т.к. чем ближе к последней горизонтали, тем сила полей больше;

### поскольку ценность шашек a1 и h2 является наименьшей, то их надо по возможности быстро ввести в игру;

### учитывая, что с полей c5 и f4 удобно препятствовать развитию сил противника, следует бороться за овладением центральными полями c5 и f4, стремиться к захвату центра;

### следует придерживаться принципа пропорционального, равномерного распределения шашек по обоим флангам;

### шашки с полей c1, e1, g1 лучше без особой надобности не сдвигать, чтобы не открывать поля последнего ряда;

### дамке по возможности стоит контролировать главную диагональ, т.к. она будет препятствовать продвижению противника в дамки.

Согласно этим правилам, ценность полей доски была обозначена следующим образом (рис.х):

На практике оказалось, что бот, использующий простую оценочную функцию, проигрывает боту, использующему более сложную оценочную функцию.

### Alpha-Beta с отсечениями

Рассмотрим основной алгоритм оптимизации перебора - alpha-beta c отсечениями. Суть его в том, что для получения оценки такой же точности, как и при полном переборе, совершенно не обязательно просматривать все варианты. Для определения отсекаемых вариантов не требуется знать особенностей данной игры.

В рекурсивной функции мы должны ввести две новые переменные — максимум для белых (alpha) и максимум для черных (beta). При начальном вызове обе эти величины равны минимально возможному значению. Если в какой-то позиции, например, для черных, мы получили результат, превышающий максимум для черных, достигнутый до этого, мы увеличиваем этот максимум (beta). Теперь рассмотрим некоторую строку игры. Представим, что где-то в глубине дерева перебора мы обнаружили, что максимумы белых и черных сравнялись. Допустим, мы просчитываем позицию для белых. Если мы продолжим перебирать в данной позиции, то максимум для белых может еще увеличиться, а может остаться прежним, но он уже сравнялся с максимумом черных. Это значит, что когда программа поднимется на уровень выше (рекурсивно), результат не будет записан, т. к. он не превышает максимума для черных в этой позиции. Также это значит, что в данной позиции для белых мы можем прекратить перебор и вернуть полученный результат досрочно. Дальше считать нет смысла.

Проиллюстрируем вышесказанное на примере (рис.х). Пусть на каком-либо шаге рекурсии максимум для белых (alpha) равен 4. Сделаем какой-либо ход и просчитаем возможные ходы для черных (опустимся на уровень ниже). Пусть при дальнейших вычислениях для одного из ходов результат равен 1 (beta=1). Если впоследствии найдется ход, оценка которого будет меньше 1, то он будет записан в beta, но не будет записан в alpha, и наоборот, если найдется ход, оценка которого будет больше 1, то он не будет записан даже в beta, следовательно, и не будет записан в alpha. Получается, дальнейшие вычисления бессмысленны.

На практике результаты показали, что…

### Форсированные варианты

1. См … [↑](#footnote-ref-2)