Алексей Словеснов

email slovesnov@yandex.ru

сайт http://slovesnov.users.sf.net/?bullscows,russian

Оптимальные алгоритмы в играх быки-коровы и мастермайнд.

Предисловие.

В статье рассматривается построение оптимальных алгоритмов для игр быки-коровы и мастермайнд по двум критериям. Первый - минимизация среднего числа ходов, которое тратится на отгадывание произвольного номера (минимизация средней длины игры). В статье будет доказано, что для игры быки-коровы нельзя создать алгоритм, который угадывал бы все номера не более чем за шесть ходов, при этом есть алгоритмы, которые угадывают любой номер не более чем за семь ходов. Аналогично, для игры мастермайнд будет доказано, что не существует алгоритма, который угадывал бы все номера не более чем за четыре хода, и существуют алгоритмы, которые угадывают все номера не более, чем за пять ходов. Исходя из этого, получается второй критерий оптимизации - минимизация числа номеров, которые угадываются ровно за семь ходов для игры быки-коровы и ровно за пять ходов для игры мастермайнд, при этом остальные номера должны угадываться за меньшее число ходов. Для всех алгоритмов построены деревья, которые можно просматривать, и сделана статистика. Также написано веб приложение, где компьютер отгадывает номера, используя один из оптимальных алгоритмов.

30 января 2017

редакция 3.1

последняя версия статьи доступна по адресу
http://slovesnov.users.sf.net/bullscows/bullscows_ru.pdf

Содержание

1 Введение.

В игре быки-коровы загаданным номером может быть любое четырехзначное число, при этом цифры этого числа не должны повторяться. Цифры могут быть от 0 до 9. Число загаданных номеров равно $10!/6! = 5040$. В игре мастермайнд загаданным может быть любое четырехзначное число с цифрами от 0 до 5, при этом цифры могут повторяться. Число загаданных номеров в этом случае равно $6^4 = 1296$. В общем игры такого вида можно задать тремя параметрами: число позиций, число цифр/цветов, и повторяемость (1 - да или 0 - нет). С этой точки зрения игра быки-коровы имеет параметры (4, 10, 0), а игра мастермайнд имеет параметры (4, 6, 1). Оптимизация алгоритмов для игр быки-коровы и мастермайнд будет сделана по двум направлениям.

Минимизация средней длины игры. Данное направление широко известно и задача по нему решена. Для игры быки-коровы, среднее число ходов равно 26274/5040=5.21 (см. [1] и [2]), при этом достаточно сделать максимум семь ходов. Для игры мастермайнд (см. [3]), среднее число ходов равно 5625/1296=4.34, если число ходов произвольное (достаточно шести), и 5626/1296=4.34, если делать максимум пять ходов. Для этого направления достаточно явно найти алгоритмы с заданным числом ходов. Будут построены три алгоритма. Один для игры быки-коровы, и два для игры мастермайнд: с ограничением в пять ходов и без него.

Минимальное число номеров. Известно, что для игры быки-коровы нельзя придумать алгоритм, в котором любой номер угадывался бы за шесть и менее ходов, в то же время существуют алгоритмы, которые угадывают все номера не более чем за семь ходов. Исходя из этого, получается второе направление оптимизации - минимизировать число номеров, которое угадывается ровно за семь ходов, при этом все остальные номера должны угадываться за шесть ходов или меньше. Аналогично, для игры мастермайнд нет алгоритма, где все номера угадывались бы за четыре и менее ходов, при этом есть алгоритмы, угадывающие все номера не более чем за пять ходов. Таким образом, нужно найти алгоритмы, минимизирующие число номеров, угадываемых за семь ходов, для игры быки-коровы и число номеров, угадываемых за пять ходов, для игры мастермайнд. Статей в интернете, по данному направлению оптимизации, не было найдено, поэтому оценки на число ходов заранее неизвестны. В этом случае был сделан полный перебор и получены следующие оценки: 50 номеров для игры быки-коровы и 539 номеров для игры мастермайнд. Так как найдена минимальная оценка, то также доказано, что нельзя создать алгоритм для игры быки-коровы, который угадывал бы все номера не более чем за шесть ходов, а для игры мастермайнд, нельзя создать алгоритм, который угадывал бы все номера не более чем за четыре хода.

Нахождение минимальной средней длины игры или минимального числа номеров, является чрезвычайно трудоемкой задачей. Попробуем грубо оценить число вариантов для игры быки-коровы в алгоритме полного перебора. Очевидно, что в качестве первого хода, достаточно рассмотреть номер 0123. Это нисколько не ограничивает общности. Ходы со второго по шестой имеют по 5039 вариантов (ход 0123 не используется). После каждого из ходов можно получить максимум 14 ответов, по одному из которых (четыре быка и ноль коров) дальнейший поиск не ведется. Будем считать, что с помощью алгоритма, можно уменьшить среднее число перебираемых вариантов в четыре раза и среднее число ответов также в четыре раза. Получаем оценку числа вариантов для полного перебора $\left(\frac{13}{4}\right)^6 \cdot \left(\frac{5039}{4}\right)^5 \approx 3.7 \times 10^{18}$. Это просто астрономическая цифра даже для современных компьютеров. Таким образом, решить задачу „в лоб“ не получится.

Статья разбита на несколько частей. Сначала вводятся обозначения и дается математическая теория. Далее описываются алгоритмы и компоненты программы. Затем приведены результаты и в самом конце описано построение деревьев ходов, по которым будет получена статистика. Они также будут использоваться для веб приложения, где компьютер будет отгадывать номера, задуманные человеком.

2 Теория.

2.1 Обозначения.

Ответ на номер обозначается - *„число быков.число коров“*. Например, 2.1 означает, что получен ответ два быка и одна корова.

Номер t с ответом r называется ходом, обозначается - (t, r). Иногда, ход будет писаться без скобок и/или без запятой - 0123 0.1.

Множество всех возможных номеров для игры быки-коровы обозначим за $\Omega^b = (0123, 0124 \ldots 9876)$. Для игры мастермайнд за $\Omega^m = (0000, 0001 \ldots 5555)$. Если Ω используется без верхнего индекса, значит имеется в виду оба типа игры.

Множество всех ответов обозначим за $R = (0.0, 0.1 \ldots 4.0)$, оно одинаковое для обоих типов игры.

Последовательность ходов обозначим за $T = (t_1, r_1) \ldots (t_n, r_n)$.

Номера угадываемые ровно за k попыток, будут называться „k-*ходовками*“.

Минимальное число номеров, среди всех алгоритмов, угадываемых ровно за k попыток, после последовательности ходов $T = (t_1, r_1) \ldots (t_n, r_n)$ обозначим за $\Phi_k[T] = \Phi_k[(t_1, r_1) \ldots (t_n, r_n)]$. Если функция Φ_k используется без верхнего индекса, то имеется в виду оба вида игры. Запись Φ_k^b - обозначает минимальную оценку для игры быки-коровы, Φ_k^m - минимальную оценку для мастермайнд.

Минимальную длину игры, среди всех алгоритмов, после последовательности ходов T обозначим за $\Delta[T]$. Предположим, что после последователь-

ности ходов, осталось m номеров. Для того, чтобы работать с целыми числами будем использовать аналогичную функцию равную сумме числа ходов для отгадывания всех оставшихся номеров. $\Lambda[T] = m \times \Delta[T]$. Если функция Λ используется без верхнего индекса, то имеется в виду оба вида игры. Запись Λ^b - обозначает минимальную оценку для игры быки-коровы, Λ^m - минимальную оценку для мастермайнд.

Рассмотрим последовательность ходов $T = (t_1, r_1) \dots (t_n, r_n)$ и номер p. Сумму числа номеров угадываемых ровно за k попыток, после ходов $(t_1, r_1) \dots (t_n, r_n)$ и следующего хода p с любым возможным ответом обозначим $\Phi_k[T(p, *)] = \Phi_k[(t_1, r_1) \dots (t_n, r_n)(p, *)]$.

$$\Phi_k[(t_1, r_1) \dots (t_n, r_n)(p, *)] = \sum_{r \in R} \Phi_k[(t_1, r_1) \dots (t_n, r_n)(p, r)] \tag{1}$$

Аналогично,

$$\Lambda[(t_1, r_1) \dots (t_n, r_n)(p, *)] = \sum_{r \in R} \Lambda[(t_1, r_1) \dots (t_n, r_n)(p, r)] \tag{2}$$

Каждый номер состоит из четырех цифр, каждая из них занимает позицию от 1 до 4. Первая цифра имеет позицию один, вторая два и так далее. Каждая цифра может быть от 0 до 9 (быки-коровы) и от 0 до 5 (мастермайнд). Обозначим множество всевозможных перестановок цифр и позиций от 1 до 4 за Ψ, а элементы этого множества $\psi \in \Psi$.

Результат действия преобразования ψ на номер t, будем обозначать $\psi(t)$.

Результат действия преобразования ψ на множество номеров $T = \{t_1 \dots t_n\}$ будем обозначать множество номеров $\psi(T) = \{\psi(t_1) \dots \psi(t_n)\}$.

Обозначим мощность множества S (число элементов в нем) через $|S|$. Например, $|\Omega^m| = 6^4 = 1296$.

Пустое множество - $\varnothing$.

Цель данной статьи найти $\Phi_7^b[\varnothing]$ для игры быки-коровы и $\Phi_5^m[\varnothing]$ для игры мастермайнд и построить алгоритмы с оптимальными оценками. Так как известно что $\Lambda^b[\varnothing] = 26274$, $\Lambda^m[\varnothing] = 5625$ и $\Lambda^m[\varnothing] = 5626$ если делать максимум пять ходов, то достаточно построить такие алгоритмы.

2.2 Преобразования.

Можно делать любое переобозначение цифр и позиций, которое по сути ничего не меняет, поэтому справедливо следующее утверждение:

Лемма 1 *Пусть есть последовательность ходов* $(t_1, r_1) \dots (t_n, r_n)$, *и преобразование* ψ, *тогда*

$$\Phi_k[(t_1, r_1) \dots (t_n, r_n)] = \Phi_k[(\psi(t_1), r_1) \dots (\psi(t_n), r_n)]$$

$$\Lambda[(t_1, r_1) \dots (t_n, r_n)] = \Lambda[(\psi(t_1), r_1) \dots (\psi(t_n), r_n)]$$

Лемма 2 *[только для игры быки-коровы] Для любых двух номеров* $t_1 \in \Omega^b$ *и* $t_2 \in \Omega^b$ *существует преобразование* ψ *такое что* $\psi(t_1) = t_2$ *и* $\psi(t_2) = t_1$.

Доказательство. Написав, компьютерную программу перебора всех номеров $t_1 \in \Omega^b$ и $t_2 \in \Omega^b$ и преобразований $\psi \in \Psi$, можно убедиться, что утверждение верно.

Пример. Рассмотрим два номера 0123 и 4051. Придумайте преобразование ψ, такое что $\psi(0123) = 4051, \psi(4051) = 0123$.

Решение. Рассмотрим преобразование ψ_1, которое меняет цифры $0 \leftrightarrow\ 1$, $2 \leftrightarrow 5, 3 \leftrightarrow 4$. Номер 0123 переходит в номер $\psi_1(0123) = 1054$, а номер 4051 переходит в $\psi_1(4051) = 3120$. Теперь рассмотрим преобразование позиций ψ_2 $1 \leftrightarrow 4$, цифры в первой и четвертой позиции меняются местами. Применив преобразование ψ_2, получим $\psi_2(\psi_1(0123)) = \psi_2(1054) = 4051$. Аналогично $\psi_2(\psi_1(4051)) = \psi_2(3120) = 0123$. Таким образом композиция $\psi = \psi_2 \circ \psi_1$ дает нужный результат.

Лемма 3 *[только для игры быки-коровы] Для любых двух номеров t_1 и t_2 и любых двух ответов r_1 и r_2*

$$\Phi_k^b[(t_1, r_1)(t_2, r_2)] = \Phi_k^b[(t_1, r_2)(t_2, r_1)]$$

$$\Phi_k^b[(t_1, r_1)(t_2, *)] = \Phi_k^b[(t_2, r_1)(t_1, *)]$$

Доказательство. Из леммы 2 следует, что существует ψ, такое что $\psi(t_1) = t_2, \psi(t_2) = t_1$, тогда, используя лемму 1, получаем что:

$$\Phi_k^b[(t_1, r_1)(t_2, r_2)] = \Phi_k^b[(\psi(t_1), r_1)(\psi(t_2), r_2)] = \Phi_k^b[(t_2, r_1)(t_1, r_2)]$$

Так как порядок ходов неважен, то ходы 1 и 2 можно переставить местами, поэтому

$$\Phi_k^b[(t_2, r_1)(t_1, r_2)] = \Phi_k^b[(t_1, r_2)(t_2, r_1)]$$

Первое доказано. Аналогично получаем, что

$$\Phi_k^b[(t_1, r_1)(t_2, *)] = \sum_r \Phi_k^b[(t_1, r_1)(t_2, r)] = \sum_r \Phi_k^b[(t_2, r_1)(t_1, r)] = \Phi_k^b[(t_2, r_1)(t_1, *)]$$

Утверждение доказано.

Примечание. То же самое верно и для функции Λ^b.

2.3 Классы эквивалентности.

Назовем два номера p и q эквивалентными относительно последовательности номеров $t_1 \dots t_n$ $(p \sim q)$ если $\exists \psi : \psi(p) = q$ и $\psi(t_i) = t_i, 1 \le i \le n$.

Лемма 4 *Отношение $p \sim q$ является отношением эквивалентности.*

Доказательство.

1. Докажем, что $a \sim a$ (рефлексивность).

   Очевидно, достаточно взять тождественное преобразование.

2. Докажем, что если $a \sim b$, то $b \sim a$ (симметричность).

   Так как $a \sim b$, то $\exists \psi : \psi(a) = b, \psi(t_i) = t_i$. У каждого преобразования есть обратное преобразование $\psi^{-1} : \psi^{-1}(b) = a, \psi^{-1}(t_i) = t_i$. Тем самым, симметричность доказана.

3. Докажем, что если $a \sim b$ и $b \sim c$, то $a \sim c$ (транзитивность).

   Так как $a \sim b$, то $\exists \psi_1 : \psi_1(a) = b, \psi_1(t_i) = t_i$. Из того что $b \sim c$, следует что $\exists \psi_2 : \psi_2(b) = c, \psi_2(t_i) = t_i$. Очевидно, что $\psi_2 \circ \psi_1(a) = \psi_2(b) = c$ и $\psi_2 \circ \psi_1(t_i) = \psi_2(t_i) = t_i$. Тем самым, транзитивность доказана.

Лемма 5 *Если два номера p и q принадлежат одному классу эквивалентности, по номерам $t_1 \dots t_n$ то $\forall k$, $\forall r_1 \dots r_n$, $\forall r$*

$$\Phi_k[(t_1, r_1) \cdots (t_n, r_n)(p, r)] = \Phi_k[(t_1, r_1) \cdots (t_n, r_n)(q, r)]$$

$$\Phi_k[(t_1, r_1) \cdots (t_n, r_n)(p, *)] = \Phi_k[(t_1, r_1) \cdots (t_n, r_n)(q, *)]$$

Доказательство. Если $p \sim q$, то $\exists \psi : \psi(p) = q, \psi(t_i) = t_i$ Из леммы 1, получаем что

$$\Phi_k[(t_1, r_1) \cdots (t_n, r_n)(p, r)] = \Phi_k[(\psi(t_1), r_1) \cdots (\psi(t_n), r_n)(\psi(p), r)] =$$

$$= \Phi_k[(t_1, r_1) \cdots (t_n, r_n)(q, r)]$$

Аналогично,

$$\Phi_k[(t_1, r_1) \cdots (t_n, r_n)(p, *)] = \sum_r \Phi_k[(t_1, r_1) \cdots (t_n, r_n)(p, r)] =$$

$$= \sum_r \Phi_k[(\psi(t_1), r_1) \cdots (\psi(t_n), r_n)(\psi(p), r)] = \sum_r \Phi_k[(t_1, r_1) \cdots (t_n, r_n)(q, r)] =$$

$$= \Phi_k[(t_1, r_1) \cdots (t_n, r_n)(q, *)]$$

Утверждение доказано.

Примечание. То же самое верно и для функции Λ.

Так как оценки функций Φ_k и Λ по некой последовательности $t_1 \dots t_n$ будут совпадать для всех эквивалентных номеров по этой последовательности, то вместо перебора всех номеров достаточно взять по одному номеру из каждого класса эквивалентности. Назовем его множеством эквивалентности по последовательности $t_1 \dots t_n$.

Алгоритм построения множества эквивалентности по последовательности $t_1 \dots t_n$. Инициализируем множество $S = (t_1 \dots t_n)$. Затем делаем цикл по всем ходам $p \in \Omega$. Если не существует преобразования ψ такого что, $\psi(t_i) = t_i \ \forall i$ и $\psi(p) \in S$, тогда добавим номер p в множество S. В конце отбросим из множества S номера $t_1 \dots t_n$, так как эти ходы уже сделаны и улучшений не дадут. В результате получится множество эквивалентности S по последовательности $t_1 \dots t_n$.

2.4 Множества номеров для первого - третьего ходов.

2.4.1 Множество первых ходов.

Рассмотрим пустую последовательность ходов $\varnothing$ и построим для нее множество эквивалентности. Для игры быки-коровы оно состоит из одного элемента $S_1^b = (0123)$ и из пяти элементов для игры мастермайнд $S_1^m = (0000, 0001, 0011, 0012, 0123)$. Получаем следующее утверждение:

Лемма 6 *В качестве первого хода достаточно рассмотреть только номера из множества S_1, остальные можно отбросить.*

$$\Phi_k[\varnothing] = \min_{s_1 \in S_1} \Phi_k[(s_1, *)] \quad \Lambda[\varnothing] = \min_{s_1 \in S_1} \Lambda[(s_1, *)]$$

2.4.2 Множества вторых ходов.

Рассмотрим некий элемент из множества $s_1 \in S_1$ и построим для него множество эквивалентности $S_2(s_1)$ по последовательности s_1. После хода s_1 достаточно перебрать в качестве второго хода номера из $S_2(s_1)$. Для игры быки-коровы получим, что

$$S_2^b(0123) = (0124, 0132, 0134, 0145, 0214, 0231, 0234, 0245, 0456,$$

$$1032, 1034, 1045, 1204, 1230, 1234, 1245, 1435, 1456, 4567)$$

Для игры мастермайнд получим пять множеств $S_2^m(s_1), s_1 \in S_1^m$ содержащих суммарно 284 элемента. Получаем следующее утверждение:

Лемма 7 *Если был сделан первый ход $s_1 \in S_1$, то в качестве второго хода достаточно рассмотреть номера из множества $S_2(s_1)$.*

$$\Phi_k[(s_1, r)] = \min_{s_2 \in S_2(s_1)} \Phi_k[(s_1, r)(s_2, *)] \quad \Lambda[(s_1, r)] = \min_{s_2 \in S_2(t)} \Lambda[(s_1, r)(s_2, *)] \quad \forall r$$

2.4.3 Множества третьих ходов.

Проведя аналогичные рассуждения можно получить множества третьих ходов. Для игры мастермайнд их будет слишком много, поэтому они используются только для игры быки-коровы. Все множества $S_3^b(s_1, s_2)$ для игры быки-коровы содержат 7072 элемента. Теперь при переборе ходов с первого по третий нужно перебрать всего 7072 номера, вместо $5039^2 \approx 2.5 \times 10^7$.

2.5 Ускорения алгоритмов.

2.5.1 Порядок ходов.

Известно что, алгоритмы перебора работают быстрее когда лучшие ходы перебираются первыми. В будущем, когда будут построены деревья ходов, то будет получено, что лучшими ходами практически всегда будут номера,

которые удовлетворяют всем предыдущим ходам. Поэтому ходы, для которых не построены множества, предварительно упорядочиваются. Сначала делаются ходы из номеров, удовлетворяющие всем предыдущим ходам, потом все остальные.

2.5.2 Оценки подмножеств.

Лемма 8 *$\forall R' \subset R$, любого номера t и любой последовательности ходов $T = (t_1, r_1) \dots (t_n, r_n)$ справедливо утверждение*

$$\Phi_k[T(t,*)] \geq \sum_{r \in R'} \Phi_k[T(t,r)]$$

в частности

$$\Phi_k[T(t,*)] \geq \Phi_k[T(t,r)] \ \ \forall r \in R$$

Доказательство. Используя формулу 1, получаем что

$$\Phi_k[T(t,*)] = \sum_{r \in R} \Phi_k[T(t,r)] = \sum_{r \in R'} \Phi_k[T(t,r)] + \sum_{r \notin R'} \Phi_k[T(t,r)] \geq \sum_{r \in R'} \Phi_k[T(t,r)]$$

Утверждение доказано.

Примечание. То же самое верно и для функции Λ.

Пусть есть последовательность ходов $T = (t_1, r_1) \dots (t_n, r_n)$, верхняя оценка β и есть некий номер t. Пусть по некоторому набору ответов из $R' \subset R$ было получено, что $\sum\limits_{r \in R'} \Phi_k[T(t,r)] \geq \beta$, тогда этот номер уже не будет лучшим.

2.5.3 Минимально возможная оценка.

Минимизация числа номеров. Если в процессе работы алгоритма удалось найти ход, по которому оценка равна нулю, то сразу возвращаем лучшую оценку 0, так как улучшать дальше некуда.

Минимизация средней длины игры. Пусть множество оставшихся номеров состоит из $|S|$ элементов и делается k-й ход. Если некоторый номер из S делит множество оставшихся номеров на подмножества размером 1, тогда это лучший ход и сумма ходов равна $\Lambda = k + (|S| - 1)(k+1) = |S|(k+1) - 1$.

2.5.4 Быстрая оценка.

Пусть сделана последовательность ходов $T = (t_1, r_1) \dots (t_n, r_n)$. Обозначим за $S(T) = (s_1, s_2 \dots)$ множество номеров, удовлетворяющих всем ходам из последовательности. Когда во множестве S остается мало элементов, в некоторых случаях, можно сразу получить оценку.

Минимизация числа номеров. Если делается седьмой ход для быков-коров, пятый для мастермайнд.

$$\Phi_7^b[T] = \Phi_5^m[T] = \begin{cases} 1 & \text{если } |S(T)| = 1 \\ 5040 & \text{если } |S(T)| > 1 \end{cases}$$

Если делается шестой ход для быков-коров или четвертый для мастермайнд.

$$\Phi_7^b[T] = \Phi_5^m[T] = \begin{cases} & \text{если } |S(T)| = 1 \text{ или } |S(T)| = 2 \text{ или} \\ |S(T)| - 1 & |S(T)| = 3 \text{ и на один из ходов } s_i \\ & \text{два других дают разные ответы} \end{cases}$$

Если делается ход с первого по пятый для быков-коров или с первого по третий для мастермайнд.

$$\Phi_7^b[T] = \Phi_5^m[T] = \begin{cases} & \text{если } |S(T)| = 1 \text{ или } |S(T)| = 2 \text{ или} \\ 0 & |S(T)| = 3 \text{ и на один из ходов } s_i \\ & \text{два других дают разные ответы} \end{cases}$$

Позже для быков-коров будут считаться не только семиходовки но и шестиходовки, пятиходовки и так далее. Для мастермайнд четырехходовки, трехходовки... (см. 3.4). Вышеприведенные формулы можно расширить.

Если делается k-й ход.

$$\Phi_k^b[T] = \Phi_k^m[T] = \begin{cases} 1 & \text{если } |S(T)| = 1 \\ 5040 & \text{если } |S(T)| > 1 \end{cases}$$

Если делается $(k-1)$-й ход.

$$\Phi_k^b[T] = \Phi_k^m[T] = \begin{cases} & \text{если } |S(T)| = 1 \text{ или } |S(T)| = 2 \text{ или} \\ |S(T)| - 1 & |S(T)| = 3 \text{ и на один из ходов } s \in S(T) \\ & \text{два других дают разные ответы} \end{cases}$$

Если делается ход с первого по $(k-2)$-й.

$$\Phi_k^b[T] = \Phi_k^m[T] = \begin{cases} & \text{если } |S(T)| = 1 \text{ или } |S(T)| = 2 \text{ или} \\ 0 & |S(T)| = 3 \text{ и на один из ходов } s \in S(T) \\ & \text{два других дают разные ответы} \end{cases}$$

Средняя длина игры. Если делается k-й ход и $|S(T)| = 1$ или $|S(T)| = 2$ или $|S(T)| = 3$ и на один из ходов $s \in S(T)$ два других дают разные ответы, тогда

$$\Lambda[T] = k + (|S(T)| - 1)(k+1) = |S(T)|(k+1) - 1$$

2.5.5 Ходы разбивающие оставшиеся номера на одну группу.

Пусть сделана последовательность ходов $(t_1, r_1) \dots (t_n, r_n)$. Тогда, очевидно, что если некий номер разобьет множество оставшихся номеров на одну группу, то этот ход будет не лучшим и может быть отброшен.

2.5.6 Алгоритм предпоследнего хода.

Делая предпоследний ход, можно сильно ускорить алгоритм, поскольку мы должны сделать такой ход t, чтобы после него все подмножества $S_i(t)$ содержали максимум по одному элементу. Поэтому как только становится известно, что $\exists i : |S_i(t)| > 1$ - ход t можно отбросить. Как известно, общее число различных ответов равно 14, поэтому если число оставшихся номеров больше 14, тогда невозможно решить все номера за нужное число ходов и можно сразу вернуть максимально возможную оценку. Если осталось ровно 14 номеров, то достаточно перебрать лишь оставшиеся номера. Этот алгоритм всегда дает точную оценку, и используется всегда.

Минимальное число номеров. Сначала в качестве предпоследнего хода переберем номера из множества S. Если ход разбивает множество на группы по одному элементу $|S_1(t)| = \dots = |S_n(t)| = 1$, то сразу же можно сказать, что это лучший ход и оценка равна $|S| - 1$. Затем переберем остальные ходы. Если ход разбивает S на группы по одному элементу, то он лучший и оценка равна $|S|$. Если же так и не оказалось хода, дробящего множество S на группы по одному элементу, значит возвращается наихудшая оценка.

Средняя длина игры. Пусть делается k-й ход, например, для быков-коров $k = 6$ если считаются семиходовки. Сначала в качестве предпоследнего хода переберем номера из S. Если ход разбивает множество S на группы по одному элементу, то он лучший и оценка будет равна $k+(|S|-1)(k+1) = (k+1)|S| - 1$. Затем переберем остальные ходы. Если ход разбивает множество S на группы по одному элементу, то он лучший и оценка равна $(k+1)|S|$. Если же так и не оказалось хода, дробящего множество S на группы по одному элементу, значит возвращается наихудшая оценка.

2.5.7 Отсечения предпредпоследнего хода.

Минимальное число номеров. По аналогии с алгоритмом предпоследнего хода, можно сразу вернуть максимальную оценку если оставшихся номеров больше чем $1+13+13^2 = 183$. Если же оставшихся номеров ровно 183, то достаточно перебрать только оставшиеся номера. Для игры быки-коровы эту оценку можно еще уменьшить. Первый ход может разбить оставшиеся номера максимум на 14 групп, по одной из них, где ответ 4.0, поиск дальше не ведется. По ответу 2.2 можно получить максимум 6 номеров, по ответу 1.3 - 8, по ответу 0.4 - 9. Остается $13 - 3 = 10$ групп, каждая из которых

может иметь максимум 13 номеров. Итого, максимум за 1 ход можно угадать 1 номер, за два хода 13, за три хода $10 \cdot 13 + 6 + 8 + 9 = 153$. Итого $1 + 13 + 153 = 167$ номеров.

Быстрые отсечения. Пусть есть верхняя оценка β, и номер t разбивает оставшиеся номера на множества $S_1 \cdots S_n$. Пусть по некоторым из них $S_i, 1 \leq i \leq m$ уже получены оценки e_i. Множества S_i оцениваются алгоритмом предпоследнего хода и иногда можно прекратить перебор.

Минимальное число номеров. Минимальная оценка на множество S_i алгоритмом последнего слоя равна $|S_i| - 1$. Таким образом, минимальная оценка хода t, после оценки первых m множеств, равна $E(m) = \sum_{i=1}^{m} e_i + \sum_{i=m+1}^{n} (|S_i| - 1)$, если $E(m) \geq \beta$, то оценка этого хода заведомо не лучше β и на этом исследование этого хода можно прекратить. После того, как будет получена еще одна оценка e_{m+1}, можно посчитать $E(m+1)$ и потенциально прекратить перебор.

Средняя длина игры. Пусть предпоследний ход имеет номер $k-1$. Минимальная оценка на множество S_i алгоритмом предпоследнего хода равна $(k+1)|S_i| - 1$. Таким образом, минимальная оценка хода t, после оценки первых m множеств, равна $E(m) = \sum_{i=1}^{m} e_i + \sum_{i=m+1}^{n} ((k+1)|S_i| - 1)$ Если $E(m) \geq \beta$, то оценка этого хода заведомо не лучше β и на этом исследование этого хода можно прекратить. После того как будет получена еще одна оценка e_{m+1}, можно посчитать $E(m+1)$ и потенциально прекратить перебор.

2.5.8 Эквивалентные разбиения.

Рассмотрим два номера p и q. Пусть они оба лежат, либо оба не лежат во множестве оставшихся номеров. Пусть они разбивают разбивают множество оставшихся номеров на одинаковое число подмножеств $S_{p_1} \ldots S_{p_n}$ и $S_{q_1} \ldots S_{q_n}$ соответственно. Рассмотрим минимальный номер mp_i из множества S_{p_i}. Будем считать, что множества S_{p_i} упорядочены по возрастанию минимальных номеров в них $mp_i < mp_{i+1}$. Аналогично считаем и для S_{q_i}. Если окажется, что множества S_{p_i} и S_{q_i} совпадают для всех i, то номера p и q дают одинаковые разбиения и один из них можно отбросить.

Очевидно, что все множества состоящие только из одного элемента дают одинаковую оценку. То же самое верно и для множеств из двух элементов и множеств из трех элементов таких, что ответы по одному из номеров на два оставшихся различны. Поэтому при разбиении номером p, можно посчитать три числа: количество подмножеств размера один n_{p_1}, размера два n_{p_2} и количество специальных множеств из трех элементов n_{p_3}; плюс множества не попавшие ни в одну из категорий $S_{p_1} \ldots S_{p_n}$. Теперь при сравнении

двух номеров p и q нужно сравнить числа n_{p_i} и n_{q_i}, а затем подмножества. В этом случае число отбрасываемых номеров увеличится, так как теперь уже неважно какие конкретно номера содержат множества размера 1, 2 и специальные множества размера 3.

Максимальное число вариантов разбиений равно $|\Omega|$ умноженное на глубину поиска, что не так много, поэтому можно сохранять все разбиения. Для поиска совпадающих разбиений нужно перебрать максимум $|\Omega|$ вариантов, что довольно много. Чтобы избежать этого можно каждому из разбиений сопоставить хеш-код, и искать только по разбиениям с такими же хеш-кодами.

2.5.9 Хеширование.

Пусть нужно найти оценку некоего множества S. Возможно в процессе поиска оптимальных ходов это множество уже оценивалось ранее, по другой ветви дерева перебора. Сохраняя множество и его оценку, можно в будущем не делать поиск, а сразу вернуть оценку. Рассмотрим множество $\Omega = (\omega_1 \cdots \omega_{|\Omega|})$. Множество $S \subset \Omega$ можно задать последовательностью бит длины $|\Omega|$, i-й бит этой последовательности равен 1 если $\omega_i \in S$. Общее число вариантов огромно $2^{|\Omega|}$, поэтому можно сохранять только часть из них. Поиск ведется алгоритмами со следующими вариациями (см. 3).

- Начиная с некоторого хода mde, кроме предпоследнего, используется эвристический алгоритм. Этот параметр определяет тип алгоритма, в том числе и точный алгоритм.
- Перебираются номера только из множества оставшихся номеров.

Для быстрого поиска множества S в хеш-таблице используется тот же механизм, что и при поиске эквивалентных разбиений (используются хеш-коды множеств).

Параметры элемента хеш-таблицы. Для сохранения множества S и его оценки, используются:

- последовательность бит длины $|\Omega|$
- тип алгоритма - mde
- один бит $heuristic$ - сохранялась ли оценка из эвристической функции оценки
- один бит $fromSetOnly$ - делались ли ходы только из множества S
- номер текущего хода n
- флаг оценки (точная оценка или $\geq \beta$)
- оценка подмножества или верхняя оценка β (в зависимости от флага)
- лучший ход (используется для построения дерева)

Проверка хода. Если множество S нашлось в хеш-таблице, то есть совпадают: последовательность бит, тип алгоритма mde, биты $heuristic$ и $fromSetOnly$, то возвращаемое значение зависит от флага оценки.

Флаг оценки - точная оценка. Здесь возможны три варианта.

1. Номер хода совпадает с номером хода из хеш-таблицы. Возвращается оценка из хеш-таблицы.
2. Номер хода n меньше номера хода из хеша n_{hash}. Для критерия минимизации числа номеров возвращается 0. Для средней длины игры возвращается $e_{hash} - |S|(n_{hash} - n)$.
3. Номер хода больше номера из хеша и оценка из хеша больше нуля. Возвращается максимальная оценка, так как за необходимое число ходов отгадать все номера уже не получится.

Флаг оценки - $\geq \beta$. Означает, что при предыдущем поиске обнаружено, что оценка множества S оказалась $\geq \beta$. В этом случае также возможно три варианта.

1. Номер хода совпадает с номером хода из хеш-таблицы и $\beta \leq \beta_{hash}$. Возвращается β.
2. Номер хода n меньше номера хода из хеша n_{hash}. Тогда, только для средней длины игры, если $\beta \leq e_{hash} - |S|(n_{hash} - n)$ возвращается β.
3. Номер хода больше номера из хеша и оценка из хеша больше нуля. Возвращается максимальная оценка, так как за необходимое число ходов отгадать все номера уже не получится.

2.5.10 Таблица ответов.

Поскольку в алгоритме нужно будет часто обращаться к функции ответа одного номера на другой, то для вычисления ответа, можно предварительно выделить память под массив типа char размера $|\Omega|^2$ и до запуска расчетов заполнить его ответами. Использование такой таблицы ускоряет расчеты.

2.6 Неиспользованные отсечения.

2.6.1 Отсутствующие цифры.

Пусть сделана последовательность ходов $(t_1, r_1) \dots (t_n, r_n)$. Обозначим за $S = (s_1, s_2 \dots)$ множество номеров, удовлетворяющих всем ходам из последовательности. Пусть все номера $s_1, s_2 \dots$ не содержат некоторых цифр $D = (d_1, d_2 \dots)$. Будем считать, что отсутствующие цифры упорядочены по возрастанию $d_1 < d_2 < \dots$ Рассмотрим номер p, который включает в

себя одну или несколько отсутствующих цифр. Тогда, если номер p, содержит только отсутствующие цифры, то его можно исключить из перебора - ничего нового он не даст. Если же номер p содержит от одной до трех отсутствующих цифр, то они должны идти строго по возрастанию, иначе этот номер можно отбросить. Поясним сказанное на примере. Пусть в игре быки-коровы была сделана последовательность ходов (0123, 0.1)(1245, 0.0), тогда все номера из множества S не будут содержать цифр $D = (1, 2, 4, 5)$. Рассмотрим номер 4058. Он содержит две отсутствующие цифры 4 и 5. Этот номер отбросить, так как номер 1028, эквивалентен номеру 4058.

2.6.2 Неназванные цифры.

Аналогичные рассуждения можно провести для неназванных цифр. Пусть сделана последовательность ходов $(t_1, r_1) \dots (t_n, r_n)$ и есть множество неназванных цифр $D = (d_1, d_2 \dots)$. Будем считать, что неназванные цифры упорядочены по возрастанию, $d_1 < d_2 < \dots$ Так же, как и для отсутствующих цифр, можно отбросить номера, где неназванные цифры не упорядочены по возрастанию. Рассмотрим пример. Пусть сделаны ходы (0123, 0.1)(1245, 0.2). Множество неназванных цифр $D = (6, 7, 8, 9)$. Номер 7601 можно отбросить, так как номер 6701 ему эквивалентен.

Примечание. Единственное отличие от отсутствующих цифр состоит в том, что нельзя отбросить все номера, содержащие только неназванные цифры. В приведенном примере, среди всех номеров, содержащих только неназванные цифры, необходимо рассмотреть только один номер 6789.

2.6.3 На несколько ходов получены одинаковые ответы.

Рассмотрим последовательность ходов $(t_1, r_1) \cdots (t_n, r_n)$. Разобьем все номера $t_1 \cdots t_n$ на группы с одинаковыми ответами. Множество номеров с ответом $b.c$, где b число быков и c число коров обозначим за $T_{b.c}$.

Пример. Пусть в игре быки-коровы задана последовательность, состоящая из трех ходов, (0123, 0.1)(1234, 0.1)(5678, 0.2). Тогда $T_{0.1} = (0123, 1234), T_{0.2} = (5678)$. Последовательность, заданную выше, можно представить в альтернативной форме $(T_{0.1}, 0.1)(T_{0.2}, 0.2)$.

Лемма 9 *Пусть есть последовательность ходов $(t_1, r_1) \cdots (t_n, r_n)$, имеющая представление в альтернативной форме $(T_{b_1.c_1}, b_1.c_1) \dots (T_{b_l.c_l}, b_l.c_l)$, и два номера p и q. Пусть существует преобразование ψ такое, что все множества номеров $T_{b.c}$ с одинаковыми ответами, переходят сами в себя, $\forall b, \forall c \;\; \psi(T_{b.c}) = T_{b.c}$ и номер p переходит в q $\psi(p) = q$. Тогда, для любого ответа r и любого k*

$$\Phi_k[(t_1, r_1) \cdots (t_n, r_n)(p, r)] = \Phi_k[(t_1, r_1) \cdots (t_n, r_n)(q, r)]$$

Доказательство. Используя лемму 1 и то, что порядок ходов неважен, получаем что

$$\Phi_k[(t_1, r_1) \cdots (t_n, r_n)(p, r)] = \Phi_k[(T_{b_1.c_1}, b_1.c_1) \dots (T_{b_l.c_l}, b_l.c_l)(p, r)] =$$

$$= \Phi_k[(\psi(T_{b_1.c_1}), b_1.c_1) \dots (\psi(T_{b_l.c_l}), b_l.c_l)(\psi(p), r)] =$$

$$= \Phi_k[(T_{b_1.c_1}, b_1.c_1) \dots (T_{b_l.c_l}, b_l.c_l)(q, r)] = \Phi_k[(t_1, r_1) \cdots (t_n, r_n)(q, r)]$$

Утверждение доказано.

Примечание. То же самое верно и для функции Λ.

Лемма 9 является обобщением леммы 1, так как номера t_i уже не обязаны переходить сами в себя при преобразовании ψ. Достаточно лишь того, что все множества номеров с одинаковыми ответами переходят сами в себя. Используя лемму 9, можно еще уменьшить множества третьих ходов, когда на первые два хода получены одинаковые ответы.

3 Алгоритмы.

3.1 Точный алгоритм.

Точный алгоритм всегда будет давать точную оценку, он имеет на входе множество номеров S, удовлетворяющих всем предыдущим ходам, и верхнюю оценку β. Алгоритм делает перебор по всем возможным ходам $t \in \Omega$, после каждого из них множество S разобьется на подмножества с одинаковыми ответами $S_1(t) \dots S_k(t)$, теперь надо найти оценку по всем этим подмножествам, рекурсивно вызвав этот же алгоритм. Оценка хода t будет суммой оценок всех подмножеств $S_1(t) \dots S_k(t)$. Минимум оценки по всем ходам и будет точной оценкой множества S.

3.2 Эвристические алгоритмы.

Эвристические алгоритмы не всегда возвращают точную оценку, но они намного быстрее точного алгоритма, поэтому мы будем последовательно сужать область их использования, постепенно переходя к точному алгоритму. Для поиска минимального числа номеров будет использоваться дробящий алгоритм (алгоритм собственной разработки), для минимальной длины игры - алгоритм Лармауса (описание можно найти в интернете). Любой из этих эвристических алгоритмов можно использовать для обоих типов оптимизации, но дробящий алгоритм дает лучшие оценки для минимального числа номеров, а алгоритм Лармауса - для минимальной длины игры.

Дробящий алгоритм. Пусть есть множество номеров $S = (s_1, s_2 \dots)$, удовлетворяющих всем предыдущим ходам. Будем перебирать всевозможные номера t. Каждый из них разобьет множество S на подмножества с одинаковыми ответами $S_1(t) \dots S_k(t)$. Обозначим за $n_i(t) = |S_i(t)|$, $\sum n_i(t) = |S|$. В дальнейшем будем считать, что подмножества $S_i(t)$ упорядочены по убыванию размера, то есть $n_1(t) \geq n_2(t) \geq \dots$

Пусть для хода t получен набор $n_1(t), n_2(t) \dots$, а для хода p получен набор $n_1(p), n_2(p) \dots$ Будем считать, что ход t лучше чем p, если $n_1(t) < n_1(p)$ или $n_1(t) = n_1(p)$ и $n_2(t) < n_2(p)$ и так далее. Например, ход с набором

$18, 18, 17\ldots$ лучше, чем ход с набором $18, 18, 18\ldots$ Дробящий алгоритм выбирает ход, который лучше всего дробит множество номеров S. Дополнительно используются следующие правила.

- Если найден ход t, такой что $n_1(t) = 1$, то
  1. если $t \in S$ это самый лучший ход.
  2. если $t \notin S$, то дальнейший поиск ведется только по ходам из S.
- Если имеется два хода $t \in S$ и $p \notin S$ с одинаковыми наборами $n_1(t) = n_1(p) \ldots n_k(t) = n_k(p)$, тогда лучшим ходом будем считать ход t. Это означает, что среди ходов с одинаковыми наборами предпочитаем ходы из множества S.

Алгоритм Лармауса. Пусть был сделан ход t и множество номеров S разбилось на подмножества $S_1(t) \ldots S_k(t)$ с размерами $n_i(t) = |S_i(t)|$. Введем функцию принадлежности

$$IN(t, S) = \begin{cases} 1 & \text{если } t \in S \\ 0 & \text{если } t \notin S \end{cases}$$

Тогда лучшим ходом будет тот, который минимизирует функцию

$$F(t) = \sum_{n_i(t)>1} n_i(t) \log(n_i(t)) - 2 \log 2 \times IN(t, S)$$

3.3 Обозначения алгоритмов.

В дальнейшем будут использоваться следующие обозначения алгоритмов.

Минимизация числа номеров.

быки-коровы

1. crush35 - дробящий алгоритм используется для ходов 3-5, для остальных ходов используется точный алгоритм.
2. crush45 - дробящий алгоритм используется для ходов 4, 5.
3. crush5 - дробящий алгоритм используется для хода 5.
4. exact - используется только точный алгоритм.

мастермайнд

1. crush3 - дробящий алгоритм используется для хода 3, для остальных ходов используется точный алгоритм.
2. exact - используется только точный алгоритм.

Минимизация средней длины игры.

быки-коровы

1. avg35 - алгоритм Лармауса используется для ходов 3-5, для остальных ходов используется точный алгоритм.
2. avg45 - алгоритм Лармауса используется для ходов 4, 5.
3. avg5 - алгоритм Лармауса используется для хода 5.
4. exact - используется только точный алгоритм.

мастермайнд

1. avg3 - алгоритм Лармауса используется для хода 3, для остальных ходов используется точный алгоритм.
2. exact - используется только точный алгоритм.

3.4 Подсчет шестиходовок, пятиходовок и так далее.

Рассмотрим игру быки-коровы по критерию минимизации числа номеров. В дальнейшем для различных алгоритмов будут построены деревья и часто будет возникать ситуация, что число семиходовок равно нулю, в этом случае можно найти ход, который минимизирует число шестиходовок, если же и число шестиходовок можно сделать равным нулю, то минимизируются пятиходовки и так далее. Алгоритм оценки написан таким образом, что он может минимизировать не только семиходовки, но и шестиходовки, пятиходовки и так далее. На входе он имеет дополнительный параметр *depthCount* - показывающий, что считать: семиходовки, шестиходовки...

Примечание. Если алгоритм не может угадать все номера за нужное число ходов, то он возвращает максимальную оценку.

Примечание. Точно так же сделано и для игры мастермайнд.

4 Компоненты программы.

Программа состоит из нескольких приложений. Все они написаны на c++, кроме одного, которое написано на javascript и используется для онлайн игры, просмотра деревьев ходов и построения их статистики.

4.1 Приложение под windows - bcw.

Вспомогательные функции.

- Решение с помощью алгоритмов crush35, crush45, avg35, avg45 для быков-коров.

- Решение всеми алгоритмами для игры мастермайнд.
- Построение множеств ходов: ходы 1-3 для быков-коров, ходы 1-2 для мастермайнд.
- Построение деревьев и сохранение их в текстовый файл.
- Загрузка дерева из текстового файла, получение статистики по нему, создание строки дерева для javascript.

4.2 Приложение на javascript.

Игра с человеком через web интерфейс, построение деревьев, их статистика.

- Построение дерева для online просмотра.
- Загрузка и получение статистики для дерева.
- Игра с человеком, используя сохраненное дерево.

4.3 Механизм задач [только быки-коровы].

Для задач, занимающих длительное время, разработан специальный механизм. Вначале создается файл jobs.txt, в котором в каждой строке описаны параметры задачи. Ниже приведен пример файла jobs.txt.

```
solve 0123(0,1)+0145(0,1) 7 4 8 #
solve 0123(0,1)+0134(0,1) 7 4 28 #
solve 0123(0,2)+1234(0,1)+4532(0,1) 7 5 3 #
```

Вначале идет ключевое слово solve, которое обозначает, что эта задача еще не взята для решения. В процессе решения заданий различные процессы берут задачи из файла и заменяют слово solve на taken, чтобы показать, что эта задача уже решается. Затем идет решаемая последовательность ходов с ответами. После этого идут три параметра алгоритма.

- *depthCount* - показывает, что считать: 7 - семиходовки, 6 - шестиходовки и так далее.

  Примечание. Этот параметр не используется при минимизации средней длины игры.

- *mde* - определяет с какого хода начать использовать эвристический алгоритм: 2 - с третьего, 3 - с четвертого... При этом для предпоследнего хода всегда используется точный алгоритм
- β - верхняя оценка бета.

Механизм задач, позволяет параллельно решать различные задания, несколькими нитями. Он реализован таким образом, что возобновляет решение задач после выключения компьютера, таким образом задачи могут считаться несколько дней. Если необходимо выполнить несколько задач, результаты которых не зависят друг от друга, то будет лучше если, первыми будут идти задачи, которые по интуитивным предположениям, будут считаться дольше - это позволяет максимально загрузить процессор. Когда возникает ситуация, что невыполненных заданий уже нет, то все нити, кроме одной, закончили свою работу и ждут. Чем за более короткое время будет выполняться последняя задача тем лучше. Поэтому лучше предварительно, хотя бы на интуитивном уровне, отсортировать задачи по убыванию времени выполнения.

4.4 Приложение executor [только быки-коровы].

Консольное приложение используется для расчета задач, записанных в файле jobs.txt.

4.5 Сервис windows [только быки-коровы].

Запуск нескольких нитей executor'a. Расчеты для алгоритмов crush5, avg5, exact занимают долгое время. В связи с этим был написан системный сервис windows, который автоматически запускался при загрузке компьютера. Поскольку компьютер, где проводились расчеты, имел четыре процессора, то запускалось четыре нити с низким приоритетом, чтобы не мешать остальным приложениям.

5 Результаты.

5.1 Минимизация числа номеров.

5.1.1 Быки-коровы.

Алгоритмы crush35 и crush45. Так как дробящий алгоритм во много раз быстрее точного, запустим сначала алгоритм crush35. Затем, только для случаев, где число семиходовок больше нуля запустим алгоритм crush45 с верхними оценками, полученными от алгоритма crush35. В будущем аналогично поступим для алгоритмов crush5 и exact. Число семиходовок приведено в таблице.

число номеров	первый ход	crush35	crush45
1440	0123 0.1	76	53
1260	0123 0.2	22	10
720	0123 1.1	1	0
480	0123 1.0	0	-
360	0123 0.0	0	-
264	0123 0.3	0	-
216	0123 1.2	0	-
180	0123 2.0	0	-
72	0123 2.1	0	-
24	0123 3.0	0	-
9	0123 0.4	0	-
8	0123 1.3	0	-
6	0123 2.2	0	-
1	0123 4.0	0	-
всего 5040		99	63

Алгоритм crush35 ни в одном из вариантов ответа на первый ход 0123 не вернул оценку 5040, а это означает, что он отгадывает все номера за семь или менее ходов.

Переход от одного алгоритма к другому. Получение оценок для алгоритмов crush5 и exact сделаем в два этапа. Сначала, используя результаты алгоритма crush45, получим оценки для crush5, затем, используя результаты алгоритма crush5, сделаем переход к алгоритму exact. Расчет алгоритмом crush5, а уж тем более алгоритмом exact будет занимать много времени, поэтому решение этими алгоритмами разобьем на несколько шагов. Назовем алгоритм от которого мы переходим prev, а тот, к которому мы переходим - next.

Шаги 1-5. Оценка лучших ходов, полученных алгоритмом prev.

1. Для алгоритма prev находим лучший ход t_1 для первого хода (0123, 0.1).

2. Рассмотрим все ответы r_1 такие что, оценка числа семиходовок алгоритмом prev больше нуля $\Phi_7^{prev}[(0123, 0.1)(t_1, r_1)] > 0$.

3. Для таких ответов r_1 считаем число семиходовок алгоритмом next $e_1(r_1) = \Phi_7^{next}[(0123, 0.1)(t_1, r_1)]$, при этом в качестве верхней оценки β используем оценку от алгоритма prev $\beta = \Phi_7^{prev}[(0123, 0.1)(t_1, r_1)]$.

4. Обозначим $E_1 = \sum_{r_1} e_1(r_1)$

5. Проведя шаги 1-4, для первого хода (0123, 0.2), получим лучший ход t_2 и его оценку E_2.

Таким образом, шаги 1-5 берут лучшие ходы для алгоритма prev и находят для них оценку алгоритмом next.
Шаг 6. Построение таблицы для ходов (0123, 0.1)$(t,$ 0.2).

6. Запускаем алгоритм next по всем вторым ходам t кроме $(t_1, t_2, 0132,$ $0231, 1032, 1230)$ для последовательности (0123, 0.1)$(t,$ 0.2). По ходам t_1, t_2 оценки известны. Также можно предположить, что номера $0132, 0231, 1032, 1230$ уже не будут лучшими ходами, по ним алгоритм next будет запущен в самом конце. При запуске алгоритма next используется верхняя оценка $\beta = \min(E_1, \Phi_7^{prev}[(0123, 0.1)(t, 0.2)])$. Полученную оценку, обозначим за $e_{12}(t) = \Phi_7^{next}[(0123, 0.1)(t, 0.2)]$. Отсортируем ходы по возрастанию $e_{12}(t)$ и выпишем их в таблицу в которую добавим номера t_1 и t_2 с их оценками.

Примечание. Сортируя номера в шаге 6 мы фактически сортируем номера (или задачи) в порядке убывания времени выполнения, тем самым дополнительно ускоряя программу (см. раздел 4.3).
Примечание. Таблицу для ходов (0123, 0.1)$(t,$ 0.2), можно использовать для нахождения оптимального номера, как для первого хода (0123, 0.1), так и для первого хода (0123, 0.2) (см. лемму 3).
Шаги 7-8. Поиск оценки числа семиходовок для первого хода (0123, 0.2).

7. Рассмотрим ход t с минимальной оценкой $e_{12}(t)$. Для него найдем другие ответы r, по которым число семиходовок при использовании алгоритма next, для последовательности $(0123, 0.2)(t, r)$ больше нуля. Для ускорения процесса используем оценки полученные алгоритмом prev. Обозначим за E_2^* сумму всех таких оценок. E_2^* - оценка последовательности $(0123, 0.2)(t, *)$ алгоритмом next. Если $E_2^* < E_2$, то уменьшим лучшую оценку $E_2 = E_2^*$.

8. Для номеров t у которых $e_{12}(t) < E_2$, делаем аналогичную процедуру, как на предыдущем шаге и потенциально снижаем E_2. Результирующая оценка E_2 и будет лучшей оценкой для алгоритма next, после хода (0123, 0.2).

Шаг 9. Поиск оценки числа семиходовок для первого хода (0123, 0.1).

9. То же самое надо сделать для первого хода (0123, 0.1).

Шаг 10. Проверка оставшихся номеров.

10. Убедимся, что ходы $t = (0132, 0231, 1032, 1230)$ не лучшие, для этого запустим алгоритм next для последовательности $(0123, 0.1)(t, 0.1)$ с верхней оценкой $\beta = E_1$ и $(0123, 0.2)(t, 0.2)$ с $\beta = E_2$

Теперь сделаем переход от алгоритма crush45 к алгоритму crush5.

Алгоритм crush5.

Расчет для лучших ходов (шаги 1-5). Рассмотрим лучший номер для первого хода (0123, 0.1), полученный алгоритмом crush45 - это 1245. После него может быть несколько ответов, но только по трем из них число семиходовок больше нуля, по этим случаям запустим алгоритм crush5.

ход 1	ход 2	crush45	crush5
0123 0.1	1245 0.1	41	38
0123 0.1	1245 0.2	10	7
0123 0.1	1245 0.0	2	1
всего		53	46

Аналогично, если первый ход (0123, 0.2), то лучший номер - 1435.

ход 1	ход 2	crush45	crush5
0123 0.2	1435 0.1	9	8
0123 0.2	1435 0.2	1	0
всего		10	8

Теперь у нас есть стартовые оценки на максимальное число семиходовок. После первого хода (0123, 0.1) - $E_1 = 46$, после хода (0123, 0.2) - $E_2 = 8$.

Построение таблицы (шаг 6).

t	(0123, 0.1)	(t, 0.2)
1234		3
1034		4
1204		7
1245		7
1435		8
1045		9
0234		10
1456		10
0245		15
0214		17
0134		18
0456		30
0145		39
0124		41
4567		≥ 46

Лучший ход после (0123, 0.2) (шаги 7-8). Рассмотрим второй ход 1234, только для вторых ответов 0.1 и 0.2 число семиходовок больше нуля.

ход 1	ход 2	crush45	crush5
0123 0.2	1234 0.1	-	3
0123 0.2	1234 0.2	-	1
0123 0.2	1234 1.1	1	0
всего			4

Таким образом если второй ход 1234, то число семиходок равно 4. Из таблицы, построенной на шестом шаге, видно, что рассматривать остальные номера не имеет смысла, так как по ним число семиходовок будет не меньше, чем 4. Таким образом лучший второй ход, для алгоритма crush5 - 1234, число семиходовок - 4.

Лучший ход после (0123, 0.1) (шаг 9).

t	(0123, 0.1)(t, 0.2)	(0123, 0.1)(t, 0.1)	$E_1 = 46$
1234	3	≥ 43	$E_1 \geq 46$
1034	4	≥ 42	$E_1 \geq 46$
1204	7	≥ 39	$E_1 \geq 46$
1245	7	38	$E_1 = 46$
1435	8	≥ 38	$E_1 \geq 46$
1045	9	≥ 37	$E_1 \geq 46$
0234	10	≥ 36	$E_1 \geq 46$
1456	10	≥ 36	$E_1 \geq 46$
0245	15	≥ 31	$E_1 \geq 46$
0214	17	≥ 29	$E_1 \geq 46$
0134	18	≥ 28	$E_1 \geq 46$
0456	30	≥ 16	$E_1 \geq 46$
0145	39	≥ 7	$E_1 \geq 46$
0124	41	≥ 5	$E_1 \geq 46$
4567	≥ 46	$\times$	

После второго хода 1245, полученного как лучший ход алгоритмом crush45, ни один другой номер не дал лучший результат.

Проверка номеров с цифрами 0-3 (шаг 10). В самом конце необходимо убедиться, что номера которые содержат только цифры от 0 до 3, не будут лучшими ходами.

первый и второй ход	crush5
(0123, 0.1)(0132, 0.1)	≥ 46
(0123, 0.1)(0231, 0.1)	≥ 46
(0123, 0.1)(1032, 0.1)	≥ 46
(0123, 0.1)(1230, 0.1)	≥ 46

первый и второй ход	crush5
(0123, 0.2)(0132, 0.2)	≥ 4
(0123, 0.2)(0231, 0.2)	≥ 4
(0123, 0.2)(1032, 0.2)	≥ 4
(0123, 0.2)(1230, 0.2)	≥ 4

Результаты алгоритма crush5. После хода (0123, 0.1) лучший второй ход - 1245, число семиходовок - 46. После хода (0123, 0.2) лучший второй ход - 1234, число семиходовок - 4.

Точный алгоритм.

Расчет для лучших ходов (шаги 1-5). Рассмотрим лучший номер для первого хода (0123, 0.1), полученный алгоритмом crush5 - это 1245. После него может быть несколько ответов, но только по трем из них число семиходовок больше нуля, по этим случаям запустим алгоритм exact.

ход 1	ход 2	crush5	exact
0123 0.1	1245 0.1	38	38
0123 0.1	1245 0.2	7	7
0123 0.1	1245 0.0	1	1
всего		46	46

Аналогично, если первый ход (0123, 0.2), то лучший номер - 1234.

ход 1	ход 2	crush5	exact
0123 0.2	1234 0.1	3	3
0123 0.2	1234 0.2	1	1
всего		4	4

Теперь у нас есть стартовые оценки на максимальное число семиходовок. После первого хода (0123, 0.1) - $E_1 = 46$, после хода (0123, 0.2) - $E_2 = 4$.

Шаги (6-9). Отличия от таблицы алгоритма crush5 выделены в рамочку.

t	(0123, 0.1)(t, 0.2)	(0123, 0.1)(t, 0.1)	$E_1 = 46$	(0123, 0.2)(t, 0.2)
1234	3	≥ 43	$E_1 \geq 46$	1
1034	4	≥ 42	$E_1 \geq 46$	$\times$
1204	7	≥ 39	$E_1 \geq 46$	$\times$
1245	7	38	$E_1 = 46$	$\times$
1435	7	≥ 39	$E_1 \geq 46$	$\times$
1045	9	36	?	$\times$
0234	10	≥ 36	$E_1 \geq 46$	$\times$
1456	10	≥ 36	$E_1 \geq 46$	$\times$
0245	15	≥ 31	$E_1 \geq 46$	$\times$
0214	16	≥ 30	$E_1 \geq 46$	$\times$
0134	18	≥ 28	$E_1 \geq 46$	$\times$
0456	30	≥ 16	$E_1 \geq 46$	$\times$
0124	38	≥ 8	$E_1 \geq 46$	$\times$
0145	38	≥ 8	$E_1 \geq 46$	$\times$
4567	≥ 46	$\times$	$E_1 \geq 46$	$\times$

Номер 1045 нуждается в дополнительном исследовании. Рассмотрим все ответы r по которым оценка последовательности (0123, 0.1)(1045, r) алгоритмом crush5 не равна нулю. Таких ответов три, по двум из них 0.1 и 0.2 оценки уже получены (см. таблицу ниже). Теперь, посчитав оценку последовательности (0123, 0.1)(1045, 0.0) точным алгоритмом, получаем, что ход 1045 тоже будет лучшим, как и номер 1245, после первого хода (0123, 0.1). Таким образом, алгоритм exact дает такую же оценку что и crush5.

ход 1	ход 2	crush5	exact
0123 0.1	1045 0.1	≥ 37	36
0123 0.1	1045 0.2	9	9
0123 0.1	1045 0.0	1	1
всего		≥ 47	46

Проверка номеров с цифрами 0-3 (шаг 10). В самом конце необходимо убедиться, что номера которые содержат только цифры от 0 до 3, не будут лучшими ходами.

первый и второй ход	exact
(0123, 0.1)(0132, 0.1)	≥ 46
(0123, 0.1)(0231, 0.1)	≥ 46
(0123, 0.1)(1032, 0.1)	≥ 46
(0123, 0.1)(1230, 0.1)	≥ 46

первый и второй ход	exact
(0123, 0.2)(0132, 0.2)	≥ 4
(0123, 0.2)(0231, 0.2)	≥ 4
(0123, 0.2)(1032, 0.2)	≥ 4
(0123, 0.2)(1230, 0.2)	≥ 4

Сравнительная таблица результатов алгоритмов.

ход 1	crush35	crush45	crush5	exact
0123 0.1	76	53	46	46
0123 0.2	22	10	4	4
0123 1.1	1	0	0	0
всего	99	63	50	50

Минимизация числа шестиходовок, пятиходовок и так далее. Выше были рассмотрены только ответы на первый ход 0123 при которых число семиходовок больше нуля. Это только два ответа 0.1 и 0.2. Для остальных ответов можно минимизировать число шестиходовок. Если число шестиходовок будет равно нулю, можно минимизировать число пятиходовок и так далее. Задача по поиску минимального числа шестиходовок намного легче основной задачи, так как теперь рассматривается на один ход меньше. Таблица минимизации, представлена ниже, в ней отброшен ответ 4.0.

ход 1	7-ходовки	6-ходовки	5-ходовки	4-ходовки
0123 0.1	46			
0123 0.2	4			
0123 1.1	0	213		
0123 1.0	0	88		
0123 0.0	0	84		
0123 0.3	0	20		
0123 1.2	0	8		
0123 2.0	0	4		
0123 2.1	0	0	28	
0123 3.0	0	0	2	
0123 0.4	0	0	0	6
0123 1.3	0	0	1	
0123 2.2	0	0	0	4

5.1.2 Мастермайнд.

Игра мастермайнд намного проще, чем быки-коровы. Поэтому можно сразу сделать расчет для алгоритмов crush3 и exact, для всех возможных первых ходов. Ниже приведена таблица результатов с числом пятиходовок. Прочерк означает, что за пять или менее ходов при данном первом ходе, все

номера отгадать нельзя.

первый ход	алгоритм crush3	алгоритм exact
0000	-	-
0001	-	695
0011	651	608
0012	569	539
0123	-	583

Таким образом, получается что лучшим первым ходом будет 0012, минимальное число номеров, угадываемых за пять ходов равно 539. Ниже приведена сводная таблица числа пятиходовок по различным ответам после первого хода.

первый ход	crush3	exact
0012 0.1	165	159
0012 1.1	128	121
0012 0.2	124	116
0012 1.0	88	80
0012 2.0	31	30
0012 1.2	19	19
0012 0.0	12	12
0012 0.3	1	1
0012 2.1	1	1
0012 0.4	0	-
0012 1.3	0	-
0012 2.2	0	-
0012 3.0	0	-
0012 4.0	0	-
всего	569	539

5.2 Минимизация средней длины игры.

5.2.1 Быки-коровы.

Задача минимизации средней длины игры намного сложнее задачи минимизации числа номеров, но точная оценка для нее известна (см. [1] и [2]). Таким образом достаточно найти алгоритм, дающий оптимальную оценку. Вначале сделаем расчеты для алгоритмов avg35, avg45. После этого запустим очень быстрый алгоритм, который не использует эвристические алгоритмы, но в качестве ходов 4-6 перебирает только оставшиеся номера (столбец fso). Возьмем результаты точного алгоритма из [2] и запишем их в столбец exact. Если посмотреть таблицу ниже, и сравнить минимальную оценку алгоритмов avg45, fso и точного алгоритма (столбец exact), мы увидим, что оптимальный результат не достигнут только по ответам 0.1, 0.2,

1.1. Поэтому, только для них, запустим алгоритм avg5.

ход 1	avg35	avg45	fso	min(avg45,fso)	exact	avg5
0123 0.0	1808	1807	1806	1806	1806	1806
0123 0.1	8009	7951	7942	7942	7935	7935
0123 0.2	6828	6817	6818	6817	6808	6808
0123 0.3	1284	1268	1269	1268	1268	1268
0123 0.4	32	32	32	32	32	32
0123 1.0	2400	2394	2393	2393	2393	2393
0123 1.1	3731	3717	3716	3716	3712	3712
0123 1.2	1031	1020	1020	1020	1020	1020
0123 1.3	30	30	30	30	30	30
0123 2.0	845	839	839	839	839	839
0123 2.1	314	312	312	312	312	312
0123 2.2	21	21	21	21	21	21
0123 3.0	97	97	97	97	97	97
0123 4.0	1	1	1	1	1	1
всего ходов	26 431	26 306	26 296	26 294	26 274	26 274

Из результирующей таблицы видно, что алгоритм avg5 уже дает точную оценку. Таким образом минимально возможная длина игры достигнута и алгоритм avg5 дает ту же оценку что и точный алгоритм.

5.2.2 Мастермайнд.

Не более пяти ходов. Игра мастермайнд намного проще, чем быки-коровы. Поэтому можно сразу сделать расчет для алгоритмов avg3 и exact, для всех возможных первых ходов, при наложении ограничения - не более пяти ходов. Ниже приведена таблица результатов. Прочерк означает, что за пять или менее ходов при данном первом ходе, все номера отгадать нельзя.

первый ход	avg3	exact
0000	-	-
0001	-	5808
0011	5716	5702
0012	5638	5626
0123	-	5676

Получается что лучшим первым ходом будет 0012. Таким образом, мы получили первый алгоритм - минимизация средней длины игры, при условии, что делается не более пяти ходов. Ниже приведена сводная таблица

длины игры по различным ответам после первого хода.

первый ход	avg3	exact
0012 0.1	1253	1251
0012 1.1	1034	1029
0012 0.2	995	994
0012 1.0	804	800
0012 2.0	445	445
0012 1.2	347	347
0012 0.0	327	327
0012 0.3	171	171
0012 2.1	154	154
0012 3.0	76	76
0012 2.2	15	15
0012 1.3	11	11
0012 0.4	5	5
0012 4.0	1	1
всего	5638	5626

Любое число ходов. Если отбросить ограничение, что можно делать не более пяти ходов, то лучшей оценкой на среднюю длину игры будет 5625/1296=4.03 (см. [3]). Сейчас уже построен алгоритм с длиной игры 5626/1296, лучший ход 0012. Необходимо уменьшить оценку всего на единицу. Предположим, что лучшим ходом, в этом случае, останется ход 0012. Посмотрим, по какому ответу на этот ход, оценка может уменьшится. Очевидно предположить, что это будет по одному из ответов имеющим максимальную среднюю длину игры. По таблице для алгоритмов avg3 и exact такими кандидатами будут ответы 0.1, 1.1, 0.2 и 1.0. Запустим расчет по первым ходам (0012, 0.1), (0012, 1.1), (0012, 0.2), (0012, 1.0). В результате по первому ходу (0012, 1.1) найдена меньшая оценка длины игры 1028/1296. Таким образом, минимально возможная длина игры 5625/1296 достигнута. Любопытно, что в этом случае, в новом и старом алгоритмах совпадают не только первый, но и второй ходы.

6 Дерево ходов.

6.1 Построение дерева на с++.

Для того чтобы создать программу, угадывающую загаданные человеком номера и не использующую длительные расчеты, необходимо предварительно построить дерево ходов и сохранить его в файл. Затем в отдельной программе можно загрузить это дерево и использовать его без всяких расчетов.

6.1.1 Минимизация числа номеров.

Быки-коровы Для игры быки-коровы, при построении дерева, будут минимизироваться не только семиходовки. Если число семиходовок для узла дерева равно нулю, то минимизируется число шестиходовок, если же можно сделать ход, такой что число шестиходовок будет равно нулю - будут минимизироваться пятиходовки и так далее. При этом, для минимизации шестиходовок, пятиходовок и так далее всегда будет использоваться точный алгоритм. Для минимизации же семиходовок, для первых ходов 0123 0.1 и 0123 0.2, используется алгоритм exact.

При создании дерева будут использоваться предварительно рассчитанные первые три хода. В процессе построения, для каждого узла дерева будет сохраняться также дополнительный параметр n. Он будет обозначать с чего начинать считать потомкам. Например, если $n = 5$, то все потомки начинают считать с пятиходовок. Также будем сохранять саму оценку e, которая поможет при расчете в процессе построения дерева. Например, $n = 5$ и $e = 30$, обозначают, что суммарное число пятиходовок у всех потомков равно 30.

Мастермайнд Аналогично, для игры мастермайнд, если число пятиходовок будет равно нулю, то будут минимизироваться четырехходовки и так далее. В этом случае в отличие от быков-коров, где используется предварительно рассчитанные три хода, используются только первые два.

6.1.2 Минимизация средней длины игры.

В процессе построения, для каждого узла дерева сохраним оценку e, которая поможет при расчете в процессе построения дерева. Например, $e = 30$, означает, что суммарное число оценок у всех потомков равно 30. Так же, как и при минимизации числа номеров, используются предварительно рассчитанные первые три хода для игры быки коровы и первые два хода для игры мастермайнд.

6.2 Построение дерева на javascript.

Для online игры, сбора статистики и построения html дерева создана программа на языке javascript. Для описания узла дерева достаточно задать его ход и указатели на дочерние элементы, по всем возможным ответам, кроме ответа 4.0. В языке javascript нет указателей, поэтому каждому элементу дерева присвоим уникальный номер id и вместо указателей на дочерние элементы будем использовать их идентификаторы id. Каждому элементу дерева сопоставим массив оставшихся номеров, чтобы знать по каким ответам необходимо продолжить поиск. Алгоритм построения дерева устроен так, что идентификаторы дочерних элементов вычисляются автоматически, используя массив оставшихся номеров и ход узла. Если один из оставшихся элементов разбивает оставшиеся номера на множества с размерами 1, то ход

этого узла очевиден и вообще не передается. Этому условию автоматически будут удовлетворять все узлы, которые содержат не более двух элементов. Кандидатами в такие ветви дерева могут быть только те, у которых число оставшихся номеров ≤ 14 (14 - число различных ответов). Ниже приведена таблица, где показано число таких узлов для различных алгоритмов.

алгоритм	всего узлов	спец. узлов	остаток %
crushBullsCows	5269	4859	410 8%
avgBullsCows	5117	4711	406 8%
crushMastermind	1349	1259	90 7%
avgMastermind5	1319	1235	84 6%
avgMastermind	1316	1234	82 6%

Можно также дополнительно отбросить узлы которые которые разбиваются на множества размера 1 не только ходами из оставшихся номеров, но и вообще любыми ходами, таких узлов около 1-2%. В этом случае построение дерева на javascript сильно замедляется, поэтому эта идея не используется.

Все дерево задается одной строкой, которая является списком ходов для оставшихся узлов дерева. В строке используются символы от ASCII кода 35 до 127, при этом отбрасывается символ „\“(ASCII код 92), так как для его передачи в строке javascript требуется 2 символа. Таким образом, используется 92-ичная система счисления. Символы с ASCII кодом больше чем 127 не используются, так как они по разному распознаются различными браузерами. Ход - это число от 0 до 5039, которое меньше чем $92 \cdot 92$, поэтому для его передачи нужно два символа или два байта.

Обработка узла дерева. Алгоритм написан таким образом, что к моменту обработки узла у него задан массив оставшихся номеров S. Если один из оставшихся элементов разбивает оставшиеся номера на множества с размерами 1, то вычисляем ход. В противном случае берем лучший ход t из строки дерева. Затем разбиваем множество оставшихся номеров S на подмножества с одинаковыми ответами $S_{b_1c_1} \dots S_{b_nc_n}$ по ходу t, ответ 4.0 не рассматривается. Занумеруем ответы в лексикографическом порядке. Ответу $(b\ c)$ сопоставим число $5b+c$. Таким образом упорядочим множества $S_{b_ic_i}$. Теперь перебирая множества $S_{b_ic_i}$ по порядку, добавляем новые элементы к концу дерева с множествами оставшихся номеров $S_{b_ic_i}$. Новые элементы будут дочерними для данного узла, по ответам $(b_i\ c_i)$.

Построение дерева. Вначале создадим дерево, состоящее из одного корневого элемента. Ход для него берем из строки дерева. Множеством оставшихся номеров будут все возможные номера Ω. Затем переходим к обработке корневого узла дерева, после чего появятся новые не обработанные узлы. Алгоритм работает, до тех пор пока не все узлы обработаны. Из таблицы выше возьмем число узлов, для которых требуется передать лучший ход. Получается, что для всех деревьев потребуется строка размера $(410 + 406 + 90 + 84 + 82) \times 2 = 2144$ байта.

7 История редакций.

Редакция 3.1 30 января 2017

- Изменен размер деревьев для javascript

Редакция 3.0 2 ноября 2014

- Игра быки-коровы решена по критерию минимизации числа номеров, угадываемых ровно за семь ходов.
- Добавлено решение игры mastermind по двум критериям.
- Добавлено построение деревьев/статистика для игры mastermind.
- Добавлены отсечения предпоследнего слоя.
- Добавлены отсечения эквивалентных разбиений.
- Добавлено хеширование.
- Построение html дерева и сбор статистики перенесено в javascript.

Редакция 2.1 5 ноября 2013 (мелкие изменения)

Редакция 2.0 1 ноября 2011

- Статья сильно расширена и дополнена.
- Статья переведена на английский язык.
- Создан сайт проекта на английском языке.

Редакция 1.0 25 декабря 2008

- Первая версия.

Список литературы

[1] John Francis, Strategies for playing MOO, or Bulls and Cows.
`http://slovesnov.users.sf.net/bullscows/bulls_and_cows.pdf`
Оригинальная ссылка `http://www.jfwaf.com/Bulls%20and%20Cows.pdf`

[2] Tetsuro Tanaka, An Optimal MOO Strategy, Game Programming Workshop - Japan.

[3] Kenji Koyama, Tony W. Lai, An Optimal Mastermind Strategy. Journal of Recreational Mathematics, 1993.