Записки научных
семинаров ПОМИ
Том 402, 2012 г.

**А. П. Давыдов**

# ОЦЕНКИ СЛОЖНОСТИ АЛГОРИТМА ГРИГОРЬЕВА ДЛЯ РЕШЕНИЯ ТРОПИЧЕСКИХ ЛИНЕЙНЫХ СИСТЕМ

## §1. Введение

**Определение.** Тропическое (min-plus) *полукольцо — это множество вещественных чисел, снабженное операциями тропического сложения и умножения:*

- $x \oplus y = \min(x, y)$,
- $x \odot y = x + y$.

Аппарат тропической математики используется в алгебраической геометрии [5] и криптографии [4]. При этом возникает множество задач, аналогичных задачам классической алгебры. Одна из центральных таких задач, задача о разрешимости линейных систем, и будет рассмотрена в данной статье. В разделе 2 будет дана постановка задачи, в третьем — описан алгоритм Григорьева и известные оценки на время его работы. Четвертый раздел посвящён улучшению верхней оценки алгоритма Григорьева до $O(\log M \cdot m \cdot n^2 \cdot \binom{m+n}{n})$, а в пятом показана нижняя оценка, доказывающая его неполиномиальность.

## §2. Постановка задачи

**Определение.** Тропическая линейная система — *это прямоугольная матрица $m \times n$.* Решение тропической линейной системы — *это такая строка из $n$ элементов, что при прибавлении ее к каждой из строк матрицы получится строка, в которой не будет строгого минимума, т.е. минимальный элемент в каждой строке должен встречаться хотя бы два раза. Тропическая линейная система называется* разрешимой, *если существует строка, являющаяся решением данной системы* [3].

*Ключевые слова*: тропические линейные системы, алгоритм Григорьева.

Так, например для матрицы

$$\begin{pmatrix} 0 & 1 & 2 & 3 \\ 0 & 0 & 1 & 3 \\ 1 & 0 & 0 & 2 \\ 0 & 1 & 0 & 2 \end{pmatrix}$$

решением будет строка

$$\begin{pmatrix} 0 & 0 & -1 & -3 \end{pmatrix}$$

В данной статье мы будем рассматривать задачу о разрешимости *целочисленных* тропических линейных систем. Хотя в частных случаях (например, на квадратных матрицах [2]) данная задача может быть решена эффективно, не известно ни одного алгоритма, для которого доказано полиномиальное время работы в худшем случае. Известно, что эта задача лежит в пересечении классов $NP$ и $coNP$ [3].

**Предложение 2.1.** *Класс разрешимых тропических линейных систем инвариантен относительно прибавления произвольной константы ко всем числам в одной строке или в одном столбце. Более того, из решения системы после подобного преобразования можно получить решение исходной системы, прибавив к найденному решению разность первых строк матрицы до и после преобразования.*

Из этого предложения сразу же получается следующее замечание.

**Замечание.** Не умаляя общности, можно считать, что все числа в матрице системы неотрицательны.

Далее будут использоваться следующие обозначения:

- $n(A)$ — количество строк матрицы $A$,
- $m(A)$ — количество столбцов матрицы $A$,
- $M(A)$ — максимальное число в матрице $A$,
- $R(A)$ — множество строк матрцы $A$.

Для простоты, если понятно, о какой матрице идет речь, параметр в этих обозначениях будет опускаться.

## §3. Алгоритм Григорьева

Один из сравнительно новых алгоритмов для решения тропических линейных систем — это алгоритм Григорьева. Ключевая особенность этого алгоритма заключается в том, что в работе [3], в которой был

предложен этот алгоритм, сразу же была показана как оценка, полиномиально зависящая от размеров матрицы (но при этом полиномиальная от $M$, а не от $\log M$), так и оценка, полиномиально зависящая от $\log M$ (но неполиномиально от размеров матрицы).

Это означает, что любая серия контрпримеров, на которой будет достигаться неполиномиальное время работы, должна состоять из матриц неограниченного размера с неограниченно большими элементами. В то же время, все известные контрпримеры ко всем остальным алгоритмам состояли из матриц фиксированного размера. Например, трудным примером для алгоритма Акиан, Гобера, Гутермана [1] будет серия матриц вида [6]

$$\begin{pmatrix} 0 & 0 & M \\ 0 & 1 & M \end{pmatrix}$$

с неограниченно растущим $M$.

Приведем описание алгоритма Григорьева.

Для начала заметим, что ввиду предложения 2.1 можно искать не решение матрицы, а серию преобразований из добавления константы ко всем элементам в строке или столбце, которая приведет исходную матрицу к матрице, решением которой является нулевая строка. Далее решением системы мы будем называть именно такую матрицу.

Для того чтобы решить систему размера $m \times n$, решим систему размера $m \times (n-1)$, получаемую из первой системы удалением первой строки. С этого момента будем считать, что во всех строках матрицы, кроме, возможно, первой, строгих минимумов нет.

Далее определим операцию *спуска* матрицы следующим образом.

(1) Заведем множество столбцов $J$. Изначально в нем будет только один столбец — тот, в котором достигается минимум в первой строке.
(2) Если $J$ содержит все столбцы, то спуск невозможен, и алгоритм завершает работу.
(3) Если найдется строка, в которой лишь один минимум достигается на столбцах, не лежащих в $J$, то добавим столбец с этим минимумом в $J$ и вернемся к шагу 2.
(4) Вычтем из всех столбцов, не лежащих в $J$, максимальное число, для которого в каждой строке, все минимальные элементы которой лежат в $J$, эти элементы останутся минимальными и после вычитания.

**Замечание.** Заметим, что хотя алгоритм спуска содержит некоторую недетерминированность в порядке прибавления столбцов к множеству $J$, в пункте 4 множество $J$ определено однозначно, т.к. максимальное по включению множество, построенное по правилам 1–3, единственно.

**Предложение 3.1.** *Если изначально строгого минимума во всех строках, кроме первой, не было, то и после спуска его не будет.*

Затем преобразуем матрицу в соответствии со следующим алгоритмом.

(1) Если в первой строке нет строгого минимума, то задача решена.
(2) Иначе — спустить матрицу. Если это невозможно, то решения нет, иначе перейти к пункту 1.

Для ясности продемонстрируем работу алгоритма на матрице из примера:

$$\begin{pmatrix} 0 & 1 & 2 & 3 \\ 0 & 0 & 1 & 3 \\ 1 & 0 & 0 & 2 \\ 0 & 1 & 0 & 2 \end{pmatrix}$$

Выберем множество $J$, последовательно добавляя столбцы:

$$\begin{pmatrix} \mathbf{0} & 1 & 2 & 3 \\ \mathbf{0} & 0 & 1 & 3 \\ \mathbf{1} & 0 & 0 & 2 \\ \mathbf{0} & 1 & 0 & 2 \end{pmatrix}$$

$$\begin{pmatrix} \mathbf{0} & 1 & \mathbf{2} & 3 \\ \mathbf{0} & 0 & \mathbf{1} & 3 \\ \mathbf{1} & 0 & \mathbf{0} & 2 \\ \mathbf{0} & 1 & \mathbf{0} & 2 \end{pmatrix}$$

$$\begin{pmatrix} \mathbf{0} & \mathbf{1} & \mathbf{2} & 3 \\ \mathbf{0} & \mathbf{0} & \mathbf{1} & 3 \\ \mathbf{1} & \mathbf{0} & \mathbf{0} & 2 \\ \mathbf{0} & \mathbf{1} & \mathbf{0} & 2 \end{pmatrix}$$

Теперь множество $J$ максимально, и можно производить спуск. Из последнего столбца вычитается 2, т.к. если вычесть хотя бы 3, то минимальные элементы в третьей и четвертой строках перестанут быть минимальными.

$$\begin{pmatrix} \mathbf{0} & \mathbf{1} & \mathbf{2} & 3 \\ \mathbf{0} & \mathbf{0} & \mathbf{1} & 3 \\ \mathbf{1} & \mathbf{0} & \mathbf{0} & 2 \\ \mathbf{0} & \mathbf{1} & \mathbf{0} & 2 \end{pmatrix}$$

$$-$$

$$\begin{pmatrix} 0 & 0 & 0 & \mathbf{2} \end{pmatrix}$$

$$=$$

$$\begin{pmatrix} 0 & 1 & 2 & 1 \\ 0 & 0 & 1 & 1 \\ 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \end{pmatrix}$$

Снова выберем множество $J$:

$$\begin{pmatrix} \mathbf{0} & 1 & 2 & 1 \\ \mathbf{0} & 0 & 1 & 1 \\ \mathbf{1} & 0 & 0 & 0 \\ \mathbf{0} & 1 & 0 & 0 \end{pmatrix}$$

$$\begin{pmatrix} \mathbf{0} & \mathbf{1} & 2 & 1 \\ \mathbf{0} & \mathbf{0} & 1 & 1 \\ \mathbf{1} & \mathbf{0} & 0 & 0 \\ \mathbf{0} & \mathbf{1} & 0 & 0 \end{pmatrix}$$

Множество $J$ — максимально, производится спуск. В последней и предпоследней строке минимальные во множестве $J$ перестанут быть минимальными, но это допускается, т.к. в этих строках есть минимальные элементы вне множества $J$.

$$\begin{pmatrix} \mathbf{0} & \mathbf{1} & 2 & 1 \\ \mathbf{0} & \mathbf{0} & 1 & 1 \\ \mathbf{1} & \mathbf{0} & 0 & 0 \\ \mathbf{0} & \mathbf{1} & 0 & 0 \end{pmatrix}$$

$$
\begin{pmatrix} 0 & 0 & \mathbf{1} & \mathbf{1} \end{pmatrix}
$$

$$
=
$$

$$
\begin{pmatrix} 0 & 1 & 1 & 0 \\ 0 & 0 & 0 & 0 \\ 1 & 0 & -1 & -1 \\ 0 & 1 & -1 & -1 \end{pmatrix}
$$

В каждой строке по два минимума, значит, ответ получен:

$$
\begin{pmatrix} 0 & 1 & 1 & 0 \\ 0 & 0 & 0 & 0 \\ 1 & 0 & -1 & -1 \\ 0 & 1 & -1 & -1 \end{pmatrix}
$$

При желании можно получить и решение-строку:

$$
\begin{pmatrix} 0 & 1 & 1 & 0 \end{pmatrix}
$$

$$
\begin{pmatrix} 0 & 1 & 2 & 3 \end{pmatrix}
$$

$$
=
$$

$$
\begin{pmatrix} 0 & 0 & -1 & -3 \end{pmatrix}
$$

В [3] показано, что временная сложность данного алгоритма составляет $O(M \cdot \log M \cdot m^2 \cdot n^2)$. Позже также была показана оценка $O(2^{mn} \cdot \log M)$.

## §4. Улучшенная верхняя оценка

**Теорема 4.1** (Верхняя оценка)**.** *Алгоритм Григорьева завершает работу за $O(n \cdot \binom{m+n}{n})$ операций спуска, что дает оценку на сложность алгоритма в $O(\log M \cdot m \cdot n^2 \cdot \binom{m+n}{n})$.*

Чтобы доказать эту теорему, введем следующее определение.

**Определение.** *Назовем* расстоянием *до строки матрицы минимальную итерацию, на которой она может быть добавлена к множеству $J$ в операции спуска матрицы. В случае, если строка не может быть*

*добавлена к множеству $J$, расстояние до нее положим равным бесконечности. Здесь и далее мы будем говорить, что строка содержится в множестве столбцов $J$, если все столбцы, ы которых достигаются ее минимумы содержатся в множестве $J$.*

Обозначим расстояние до строки $r$ через $d_0(r)$, а расстояние до строки $r$ после $i$-го спуска алгоритма Григорьева — через $d_i(r)$. Обозначим мультимножество расстояний для всех строк матрицы $A$ через $D_0(A)$, а соответствующее мультимножество после $i$-го спуска алгоритма Григорьева — через $D_i(A)$. Также будем использовать обозначения $d(r)$ и $D(A)$ для $d_0(r)$ и $D_0(A)$ соответственно.

Добавим расстояния в матрицу из примера:

| Матрица | | | | $d = d_0$ | $d_1$ |
|---|---|---|---|---|---|
| 0 | 1 | 2 | 3 | 0 | 0 |
| 0 | 0 | 1 | 3 | 1 | 1 |
| 1 | 0 | 0 | 2 | 2 | $\infty$ |
| 0 | 1 | 0 | 2 | 1 | $\infty$ |

Для расстояний, определенных вышеописанным образом, верна следующая теорема.

**Теорема 4.2.** *Для любой матрицы $A$ выполняются следующие свойства.*

(1) $\forall_{n \in D(A)} (|\{x < n | x \in D(A)\}| \geq n - 1)$.

(2) *После операции спуска алгоритма Григорьева расстояние до хотя бы одной из строк увеличится.*

(3) *Обозначим через $r_i$ строку с минимальным $d_{i+1}(r)$ среди всех строк, для которых $d_i(r) \neq d_{i+1}(r)$. Тогда*

$$\forall_{r \in R(A)} ((d_i(r) < d_i(r_i)) \quad \Rightarrow \quad (d_{i+1}(r) = d_i(r)), \tag{1}$$

$$\forall_{r \in R(A)} ((d_i(r) \geq d_i(r_i)) \quad \Rightarrow \quad (d_{i+1}(r) \geq d_{i+1}(r_i)). \tag{2}$$

**Доказательство.** (1) Допустим, что данное утверждение неверно для расстояния $n$, соответствующего столбцу $r$. Рассмотрим операцию спуска, при которой столбец $r$ добавляется к множеству $J$ на итерации $n$. На этой итерации размер множества $J$ равен $n-1$, значит, найдется $n-1$ столбец, расстояние до которых меньше $n$.

(2) Пусть $r$ — строка с минимальным расстоянием после спуска среди всех строк, у которых после операции спуска появился

новый минимум. Несложно заметить, что расстояние до нее должно увеличиться.

(3) Чтобы получить свойство (1) достаточно заметить, что невозможно уменьшить расстояние до строки, не добавив в $J$ столбцов с новыми минимумами. Как мы уже отмечали в пункте 2, расстояние до строки, у которой появился новый минимум, и новое расстояние до которой минимально, изменилось. Свойство (2) следует по тем же соображениям, ведь если в оптимальном спуске для данной строки строка с добавленным минимумом не используется, то расстояние не уменьшится (раньше можно было использовать тот же спуск), в противном же случае расстояние не может быть меньше, чем расстояние до строки с добавленным минимумом.

□

Используя теорему 4.2, несложно заметить, что если записать элементы $D$ в порядке возрастания, то подобная запись после спуска будет лексикографически больше записи до спуска, откуда и следует ограничение на число спусков (оно будет не больше, чем число отсортированных последовательностей длины $n$ из чисел от 1 до $m$). Таким образом, теорема 4.1 доказана. Одновременно с автором аналогичная нижняя оценка была получена В. Подольским.

## §5. Контрпример

Посмотрев на оценки сложности в частях 3 и 4, можно заметить, что если зафиксировать хотя бы одну из трех размерностей задачи ($M$, $m$, или $n$), то алгоритм Григорьева будет работать полиномиальное время. Этим и объясняется трудность конструкции контрпримера.

**Теорема 5.1** (Нижняя оценка)**.** *Существует последовательность матриц (с неограниченным количеством строк и столбцов), на которой алгоритм Григорьева работает, в худшем случае, за время* $\Omega(n^{\frac{m}{6}} \log M)$. *При этом* $M = poly(n^{\frac{m}{6}})$.

**Доказательство.** Построим последовательность матриц $S_{k,i}$ со следующими свойствами:

- количество столбцов матрицы не зависит от $i$ и равна $6k + C$, где $C$ — константа;

- количество строк матрицы линейно по $i$ при фиксированном $k$;
- максимальное число в матрице полиномиально по $i^k$;
- алгоритм Григорьева на матрице $S_{k,i}$ делает $i^k$ спусков.

Эту последовательность мы будем строить индукционно, используя в конструкции для $S_{k,i}$ матрицу $S_{k-1,i}$. Не умаляя общности, будем считать, что все элементы $S_{k-1,i}$ неотрицательны, и минимальные элементы в каждой строке — нули.

Чуть ниже будет приведена сама конструкция, далее мы проиллюстрируем контрпример несколькими шагами алгоритма, а затем приведем подробные объяснения и аккуратные оценки.

Основная идея конструкции заключается в следующем: мы приведем конструкцию подматрицы (будем называть ее *гаджет*), которая имеет константное количество строк и позволяет перезапустить алгоритм на $S_{k-1,i}$ заново. В конструкции используются следующие обозначения: $L(A)$ — сумма всех чисел, на которые спустит части матрицы $A$ алгоритм Григорьева; в нашей конструкции $L$ без параметров обозначает $L(S_{k-1,i})$; $a_j$ — сумма спусков $j$-го столбца матрицы $S_{k-1,i}$. Обозначение "$\infty$" введено лишь для упрощения понимания и используется везде, где точное значение элемента не принципиально, если оно достаточно велико; в данном случае "достаточно велики" будут числа, большие $L \cdot (2i+1)$. Через $S'_{k-1,i}$ обозначим матрицу-решение для системы $S_{k-1,i}$ (в том виде, к которому ее приводит алгоритм Григорьева). Также для простоты заметим, что, поскольку у каждой строки значение в последнем столбце равно ее значению в предпоследнем столбце, то если в строке минимум оказывается в этих столбцах, то далее она никак на работу алгоритма не повлияет, и ее можно не рассматривать (такие строки мы будем отмечать косыми крестиками $\times$).

Подробная конструкция контрпримера изображена в таблице 1. В конструкции в таблице предполагается, что количество столбцов матрицы $S_{k-1,i}$ равно пяти, обощение для большего коичества столбцов тривиально. Предполагается, что если в некоторой области стоит число, то вся область заполнена этим числом. Для того, чтобы легче было выделить границы клеток таблицы, по краям таблицы в каждой строке и столбце стоит символ "$\bullet$".

Сначала алгоритм будет работать так же, как и на матрице $S_{k-1,i}$ (см. таблицу 6). Затем первый гаджет "перезапускает" процесс сначала (см. таблицу 6). Важное замечание: строки матрицы $S_{k-1,i}$ в это время не могут оказаться решающими для присоединения столбца: если столбец $j$ не присоединен с помощью гаджета, то это значит, что из третьего и четвертого столбцов вычтено ровно столько же, сколько и из столбца $j$, а все элементы матрицы $S_{k-1,i}$ неотрицательны (поэтому если минимум достигается в столбце $j$, то он также будет и в третьем и четвертом столбцах). Посмотрим внимательно на третью таблицу. Как мы видим, прибавив к строкам с третьей по девятую $L$, мы получим исходную ситуацию с числом гаджетов на единицу меньше, а такое прибавление не влияет на работу алгоритма.

Заметим, что из нашей конструкции следуют следующие рекурентные формулы:

$$\begin{aligned} n(S_{k,i}) =& n(S_{k-1,i}) + (m(S_{k-1,i}) + 2) \cdot i + 4, \\ m(S_{k,i}) =& m(S_{k-1,i}) + 6, \\ T(S_{k,i}) =& i \cdot T(S_{k-1,i}), \\ L(S_{k,i}) =& L(S_{k-1,i}) \cdot (2i + 1), \end{aligned}$$

где через $T$ обозначено количество спусков. Взяв в качестве $S_{0,i}$ нулевую матрицу из одной строки и двух столбцов, мы получим оценки, требуемые в теореме 5.1. □

Заметим, что у контрпримера появится тривиальное решение, если все "бесконечные элементы" одинаковы. Поэтому "бесконечные элементы" следует считать разными числами, превосходящими $L \cdot (2i+1)$.

## §6. Заключение

В этой работе мы доказали неполиномиальность алгоритма Григорьева. Дальнейшие исследования могут быть направлены как на разработку полиномиального алгоритма для решения тропических линейных систем, так и на сведение задачи к одной из задач из класса $NP \cap coNP$, которая считалась бы трудной.

**Благодарности.** Автор благодарен С. И. Николенко и Д. Ю. Григорьеву за плодотворные обсуждения результатов и доказательств. Работа была поддержана грантом РФФИ 11-01-12135-офи-м-2011.

Таблица 1. Контрпример

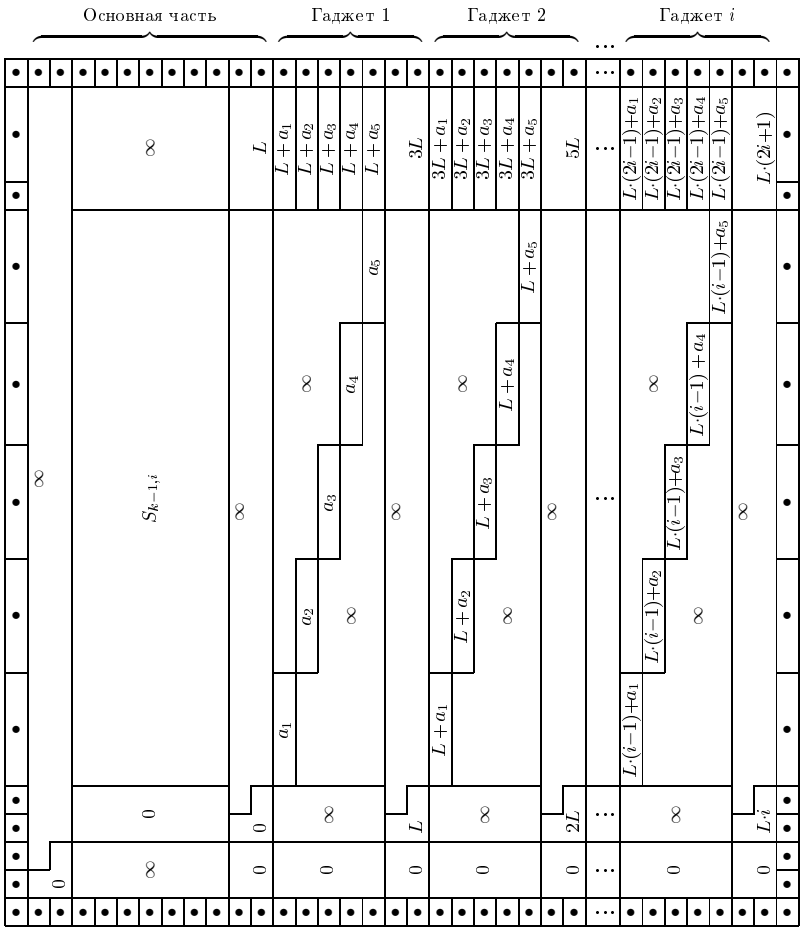


Таблица 2. Начало работы

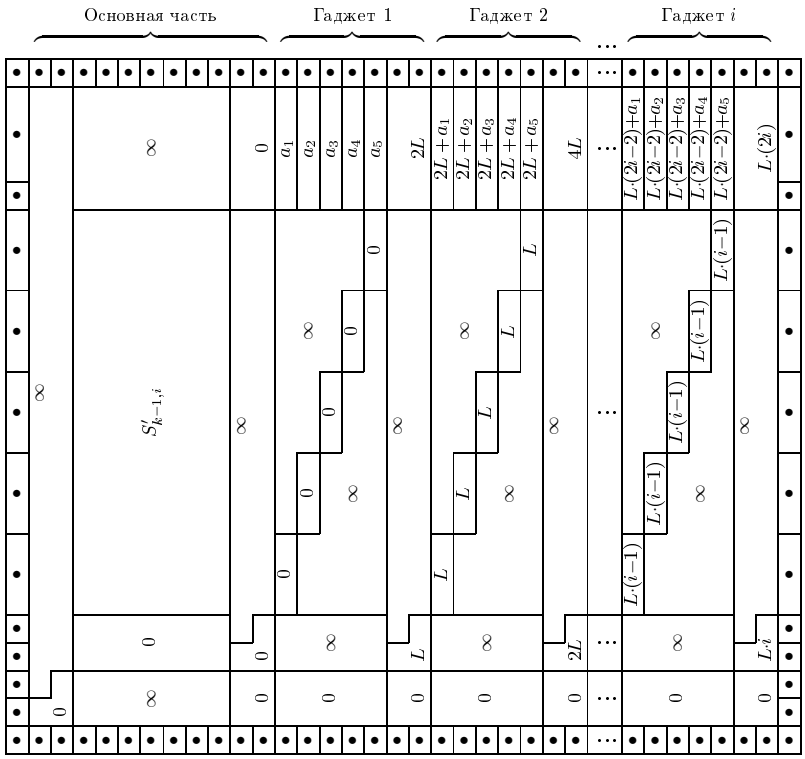
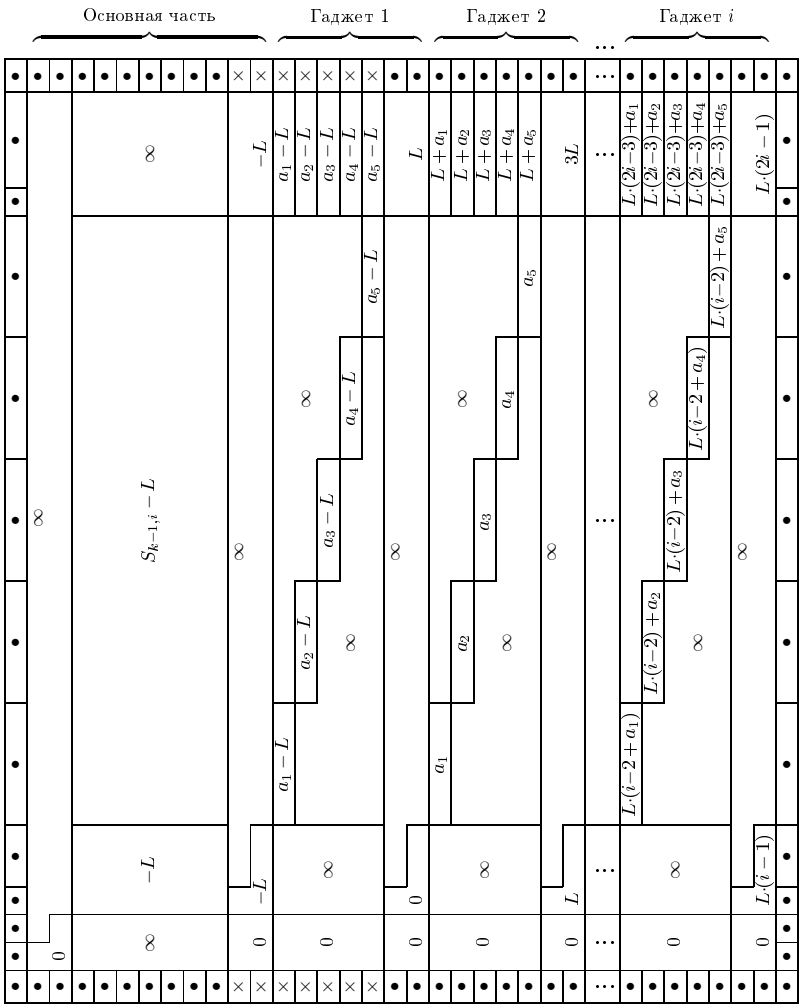


Таблица 3. Использован первый гаджет



## Литература

1. M. Akian, S. Gaubert, and A. Guterman. The correspondence between tropical convexity and mean payoff games. In *Proceedings of the 19th Intern. Symp. Math. Theory of Networks and Systems, Budapest*, pages 1295–1302, 2010.
2. P. Butkovic and F. Hevery. A condition for the strong regularity of matrices in the minimax algebra. *Discr. Appl. Math*, 11:209–222, 1985.
3. Dima Grigoriev. Complexity of solving tropical linear systems. Preprint MPIM 2010-60, Bonn, to appear in *Computational Complexity*
4. Dima Grigoriev and Vladimir Shpilrain. Tropical cryptography. Preprint MPIM 2011-11, Bonn.
5. B. Huber and B. Sturmfels. A polyhedral method for solving sparse polynomial systems. *Math. of Computation* 64:1541вАУ1555, 1995.
6. V. Podolskii. Личное сообщение.

Davydow A. P. Upper and lower bounds for Grigoriev's algorithm for solving integral tropical linear systems.

We investigate an algorithm for solving integral tropical linear systems proposed by Dima Grigoriev in 2010, We give the first nonpolynominal lower bound on time complexity of the algorithm, and also improve known upper bound.

Академический Университет РАН
ул. Хлопина, 8, корп. 3,
Санкт-Петербург 194021,
Россия
*E-mail*: adavydow@yandex.ru

Поступило 3 сентября 2012 г.