Эвристические алгоритмы и распределённые вычисления, **1,** 4 (2014) 42–67

УДК 519.816

Восстановление турнира по множеству полустепеней выхода вершин *

А. Ивани
Венгрия, Будапешт,
Университет им. Л. Этвёша
(Eötvös Loránd University)
email: tony@inf.elte.hu

Я. Элек
Венгрия, Будапешт,
Университет им. Л. Этвёша
(Eötvös Loránd University)
email: elekjani@caesar.elte.hu

Аннотация. Множество результатов (или множество счёта) турнира (турнирного орграфа) – это множество полустепеней выхода его вершин. В 1978 г. Рейд опубликовал гипотезу о том, что для любого множества (различных) неотрицательных целых чисел D существует турнир T, множеством результатов которого является D. Рейд проверил эту гипотезу для турниров, состоящих из 1, 2 и 3 вершин.

В 1986 г. Хагер опубликовал конструктивное доказательство гипотезы для 4 и 5 вершин. В 1989 г. Яо представил арифметическое доказательство гипотезы, однако общий полиномиально-временной алгоритм неизвестен до сих пор.

В 2013 г. одним из авторов данной статьи опубликован полиномиально-временной алгоритм, восстанавливающий турнирный орграф по множество результатов в случае, все элементы этого множества меньше 7. В 2014 г. авторами настоящей статьи эта граница была увеличена до 9.

В этой статье мы приводим описание и анализ алгоритмов Hole-Map, Hole-Pairs, Hole-Max и Hole-Shift и используем их для увеличения границы существования полиномиально-временного алгоритма для рассматриваемой проблемы: эти алгоритмы дают возможность восстановления *любого* турнирного орграфа в том случае, когда все элементов множества результатов меньше 12.

Мы также приводим результаты работы переборных (brute force) алгоритмов Sequencing и Diophantine и разработанных с их использованием аппроксимационных алгоритмов Shiftening и Hole.

Поскольку существуют быстрые (квадратичные) алгоритмы построения турниров, соответствующие заданным *последовательностям* результатов, разработанному нами алгоритму достаточно построить за полиномиальное время подобную (подходящую) последовательность на основе заданного *множества* результатов – что и делается в разработанных авторами алгоритмах.

Ключевые слова и фразы: турнир; полустепень выхода (результат) вершины; множество полустепеней выхода (множество результатов); аппроксимационный алгоритм; гипотеза Рейда.

1 Основные определения

Мы будем использовать следующие определения, согласованные с [1]. Рассматривается ориентированный граф (V, E), не имеющий петель. *Турнир* – это некоторый полный орграф. (т.е. орграф, не имеющий петель, у которого между любой парой вершин имеется ровно одна дуга).

Для некоторой его дуги $(u, v) \in E$ будем говорить, что u *доминирует над* v. Ориентированный граф (в том числе турнир) $F = (E, V)$ называется *транзитивным*, если из условий $(u, v) \in E$ и $(v, w) \in E$ следует $(u, w) \in E$. *Порядком турнира* T называется число его вершин. Турнир порядка n будет также называться n-*турниром*.

Результат (или *полустепень выхода*)[1] вершины v турнира T – это число вершин, над которыми v доминирует. Это значение обозначается $d_T^+(v)$, или, короче, $d(v)$.

Последовательность результатов некоторого n-турнира T – это упорядоченная n-ка $(s_1, s_2, \ldots, s_n)$, где s_i – результат вершины v_i, $1 \le i \le n$, причём

$$s_1 \le s_2 \le \cdots \le s_n.$$

Множество результатов некоторого n-турнира T – это упорядоченная m-ка $D = (d_1, d_2, \ldots, d_m)$ *различных* результатов вершин турнира T, где

$$d_1 < d_2 < \cdots < d_m.$$

[1] Русская терминология не устоялась. Мы предлагаем термин «результат» – вместо иногда применяемого термина «счёт». При этом не требующий дополнительных пояснений термин «полустепень выхода» представляется громоздким – особенно в применяемых ниже конструкциях: «последовательность счёта», «множество полустепень выхода» и т.п.; ниже мы не будем приводить возможные синонимы.

Соответственно, название статьи кратко может быть сформулировано следующим образом: «Восстановление турнира по таблице результатов». *(Прим. перев.)*

Если последовательностью результатов турнира является S, а множеством результатов – D, то будем говорить, что S *генерирует* D, либо что D *соответствует* S. Последовательность результатов некоторого турнира будем называть *возможной*. Поскольку существуют быстрые (квадратичные) алгоритмы восстановления n-турниров, соответствующих заданной возможной последовательности результатов, мы в данной статье рассматриваем только алгоритмы восстановления возможных последовательностей результатов (по заданному множеству результатов).

На рис. 1 показан пример турнира с последовательностью результатов $S = (0,\ 2,\ 2,\ 2)$ и множеством результатов $D = \{0,\ 2\}$.

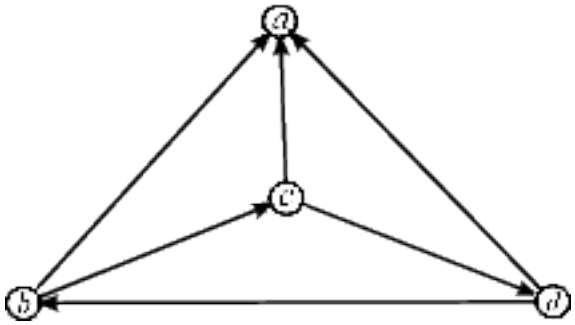

Рис. 1: Пример турнира.

Отметим, что те же самые понятия (результат вершины, последовательность и множество результатов) совершенно аналогично определяются для произвольных ориентированных графов. Ниже кроме терминологии теории графов мы будем использовать «обычные игровые» термины – «выигрыш», «проигрыш» и т.п.

2 Введение

В 1976 г. Шартран (Chartrand), Лесняк (Lesniak) и Робертс (Roberts) доказали, что для любого конечного множества S неотрицательных целых чисел существует ориентированный граф, множеством результатов которого является S – см. [2]. Следующие описываемые нами результаты относятся к n-турнирам.

Теорема Ландау (Landau) [3] позволяет за линейное время проверять потенциальные последовательности результатов – т.е. определять, существует ли некоторый турнир с заданной последовательностью:

Теорема 1 ([3, 4]) *Некоторая неубывающая последовательность неотрицательных целых чисел* $S = (s_1, s_2, \ldots, s_n)$ *является последовательностью результатов некоторого турнира тогда и только тогда, когда*

$$\sum_{i=1}^{k} s_i \geq \frac{k(k-1)}{2}, \quad 1 \leq k \leq n,$$

причём в случае $k = n$ *выполнено равенство.* □

Однако восстановление турнира по некоторому допустимому множеству результатов – это более сложная задача, чем восстановление турнира по некоторой допустимой *последовательности* результатов. Поэтому весьма необычной была гипотеза, выдвинутая Рейдом (Reid) в 1978 г., см. [5]: если $m \geq 1$, $D = \{d_1, d_2, \ldots, d_m\}$ – множество неотрицательных целых чисел, то существует некоторый турнир, множеством результатов которого является D. В той статье Рейд приводит доказательство для множеств, содержащих 1, 2 и 3 элемента, и, кроме того, для множеств, являющихся арифметическими или геометрическими прогрессиями. В 1986 г. Хагер (Hager, [6]) доказал выдвинутую Рейдом гипотезу для $m = 4$ и $m = 5$.

Полностью гипотеза Рейда была доказана в 1989 г. Яо (Yao):

Теорема 2 ([7]) *Если* $m \geq 1$, $\mathcal{D} = \{d_1, d_2, \ldots, d_m\}$ – *множество неотрицательных целых чисел, то существует некоторый турнир, множеством результатов которого является* D. □

Однако доказательство Яо использует несколько неконструктивных утверждений, доказанных в теоретической арифметике, – и, следовательно, доказывает *только существование* соответствующего турнира, но не способ его построения.

В 1983 г. Вэйлэнд (Wayland, [8]) опубликовал достаточное условие того, чтобы некоторое множество неотрицательных целых чисел являлось множеством результатов т.н. двустороннего турнира. Этот результат был улучшен в том же 1983 г. Пе́тровичем (Petrovič, [9]) – он получил необходимое и достаточное условие этого факта.

В 2006 г. Пирзада (Pirzada) и Наикоо (Naikoo) опубликовали конструктивное доказательство *частного случая* гипотезы Рейда:

Теорема 3 ([10]) *Если* $s_1, s_2, \ldots, s_m$ – *неотрицательные целые числа, причём* $s_1 < s_2 < \cdots < s_m$, *то существует некоторое* $n \geq m$,

и для него – некоторый n*-турнир* T *с множеством результатов*

$$D = \left\{ d_1 = s_1,\ d_2 = \sum_{i=1}^{2} s_i,\ \ldots,\ d_m = \sum_{i=1}^{m} s_i \right\}. \quad \square$$

Ранее в [11, 12] мы приводили полиномиальные аппроксимационные алгоритмы Balancing, Shortening и Shiftening, основанные на теореме 4. В настоящей статье мы описываем новые алгоритмы Shortening и Hole и с помощью вычислительных экспериментов доказываем теорему 2 для множеств, все элементы которых меньше 9. При этом все наши доказательства – конструктивные, причём описываемые нами алгоритмы восстановления турнира работают за полиномиальное время. [2]

В данной статье мы также приводим описание и анализ алгоритмов Hole-Map, Hole-Pairs, Hole-Max и Hole-Shift и используем их для увеличения границы существования полиномиальновременно́го алгоритма для рассматриваемой проблемы: эти алгоритмы дают возможность восстановления *любого* турнирного орграфа в том случае, когда $d_m < 9$.

Мы также применяем переборные (brute force) алгоритмы Sequencing и Diophantine (см. также [11]). На их основе разработаны аппроксимационные алгоритмы Shiftening и Hole.

3 Вспомогательные утверждения

В этом разделе мы рассмотрим 3 леммы, каждая из которых является обобщением теоремы 2. Всюду ниже $a^{<b>}$ обозначает мультимножество (набор), в котором элемент a повторен b раз. (При этом мы обычно не будем делать различий между мультимножествами и последовательностями.) Такая форма записи последовательности результатов называется *степенно́й формой*.

Лемма 1 *Если* $d_1 \geq 1$*, то множество результатов* $D = \{d_1\}$ *реализуемо – причём единственной последовательностью результатов* $S = d_1^{<2d_1+1>}$.

[2] Отметим также, что в 2006 г. в [13] одним из авторов данной статьи было доказано, что обобщение теоремы Яо неверно для k-турниров (в которых каждая пара вершин соединена k дугами, $k \geq 2$).

Доказательство. Если $|S| = n$ и последовательность результатов S соответствует множеству результатов D, то сумма элементов последовательности S равна, с одной стороны, nd_1, а с другой стороны $n(n-1)/2$; отсюда получаем $n = 2d_1 + 1$.

Подобный турнир реализуем, например, следующим образом. Игрок P_i выигрывает матчи у игроков $P_{i+1}, \ldots, P_{i+(n-1)/2}$ (индексы выбираются по модулю n) и проигрывает оставшимся игрокам. □

Лемма 2 *Если некоторая последовательность результатов*

$$S = (s_1, s_2, \ldots, s_n)$$

соответствует множеству результатов

$$D = \{d_1, d_2, \ldots, d_m\},$$

то $n \geq d_m + 1$.

Доказательство. Если результат вершины v равен d_m, то v выигрывает у d_m различных вершин. □

Лемма 3 *Если* $m \geq 2$, *и последовательность* $S = (s_1, s_2, \ldots, s_n)$ *соответствует множеству результатов* $D = \{d_1, d_2, \ldots, d_m\}$, *то*

$$2d_1 + 2 \leq n \leq 2d_m, \tag{1}$$

и при этом оба неравенства являются точными (т.е. неулучшаемыми).

Доказательство. Каждый элемент множества D должен присутствовать в S. Поэтому среднее арифметическое результатов больше, чем d_1, и меньше, чем d_m. С другой стороны, каждый из n-турниров содержит $B_n = \binom{n}{2}$ дуг, поэтому среднее арифметическое их результатов равно $B_n/n = (n-1)/2$, поэтому

$$d_1 < \frac{n-1}{2} < d_m, \tag{2}$$

откуда следует (1).

В частности, если $k \geq 0$ и $D = \{k, k+1\}$, то, согласно (2), равенство $n = 2k + 2$ влечёт точность обеих оценок. □

Следующая теорема доказана Ивани (Iványi), Луцем (Lucz), Гомбошем (Gombos) и Матýшкой (Matuszka) в [11]. Она представляет собой развитие теоремы 2 и основана на леммах 1, 2 и 3.

Теорема 4 *Если* $m \geq 1$, *и* $D = \{d_1, d_2, \ldots, d_m\}$ – *неубывающее упорядоченное множество неотрицательных целых чисел, то*

- *существует некоторый турнир* T, *последовательностью результатов которого является* S, *а множеством результатов* – D;
- *если* $m = 1$, *то* $S = s_1^{<2d_1+1>}$;
- *если* $m \geq 2$, *то*

$$\max(d_m + 1, 2d_1 + 2) \leq n \leq 2d_m; \tag{3}$$

- *оценки* (3) *являются точными.*

Доказательство. Это утверждение является следствием приведённых выше лемм – подробнее см. [11]. □

Согласно [4, стр. 180], мы можем сформулировать гипотезу Рейда как некоторое арифметическое утверждение – т.е. без использования терминологии теории графов. Итак, пусть $D = \{d_1, d_2, \ldots, d_m\}$ является неубывающим упорядоченным множеством неотрицательных целых чисел. Гипотеза заключается в том, что существуют положительные «степени» $x_1, x_2, \ldots, x_m$, такие что последовательность

$$S = (d_1^{<x_1>}, d_2^{<x_2>}, \ldots, d_m^{<x_m>})$$

является последовательностью результатов некоторого $(\sum_{i=1}^{m} x_i)$-турнира. С помощью теоремы Ландау можно весьма несложно показать, что гипотеза Рейда эквивалентна следующему утверждению ([7, 14]): для каждого множества $D = \{d_1, \ldots, d_m\}$, обладающего свойством $0 \leq d_1 < d_2 < \cdots < d_m$, существуют положительные целые числа $x_1, \ldots, x_m$, такие что

$$\sum_{i=1}^{k} x_i d_i \geq \binom{\sum_{i=1}^{k} x_i}{2}, \quad \text{для } k = 1, \ldots, m-1,$$

и

$$\sum_{i=1}^{m} x_i d_i = \binom{\sum_{i=1}^{m} x_i}{2}.$$

Кяо Ли, комментируя доказательство Яо в [15], пишет:

> *Доказательство Яо – первое доказательство этой гипотезы, но я не думаю, что оно последнее. Я надеюсь, что в ближайшем будущем появятся новые доказательства – причём более короткие и более простые.*

... Однако *конструктивного* доказательства нет до сих пор.

В описываемых далее алгоритмах рассматриваются только множества результатов, не содержащие 0.[3] Основой этого подхода является следующая лемма.

Лемма 4 *Пусть* $m \geq 2$. *Последовательность*

$$S = \left(s_1^{<e_1>}, s_2^{<e_2>}, \ldots, s_n^{<e_m>}\right)$$

является последовательностью результатов, соответствующей множеству результатов

$$D = \{0, d_2, d_3, \ldots, d_m\}$$

тогда и только тогда, когда последовательность

$$S' = ((s_2 - 1)^{<e_2>}, (s_3 - 1)^{<e_3>}, \ldots, (s_n - 1)^{<e_n>})$$

является некоторой последовательность результатов, соответствующей множеству

$$D' = \{d_2 - 1, d_3 - 1, \ldots, d_m - 1\}.$$

Доказательство. Если S является последовательностью результатов, соответствующей множеству D, то $s_1 = 0$ и $e_1 = 1$ – т.е. все остальные игроки выигрывают у игрока, имеющего результат $s_1 = 0$; поэтому S' соответствует множеству D'.

Если S' не соответствует D', то мы добавляем к множеству D' новое значение $d_1 = 0$, увеличиваем степень остальных результатов на 1, получая множество D, которое не соответствует последовательности S. □

[3] Можно несложно показать, что общность при этом не ограничивается. *(Прим. ред.)*

4 Восстановление множества результатов турнира: Hole-подобные алгоритмы

Hole-подобные алгоритмы основаны на следующей идее. Рассматривая неубывающую последовательность результатов $s = (0, 1, \ldots, d_m)$, мы постепенно удаляем из неё элементы, отсутствующие в требуемом множестве результатов. При этом во время обработки последовательности s разрешается временно изменять её наибольший элемент.

Будем говорить, что во множестве результатов $D = \{d_1, d_2, \ldots, d_m\}$ имеется k-*дыра* перед элементом d_i $(1 \le i \le m)$, если:

- $d_1 = k + 1$,
- либо $2 \le i \le m$ и $d_i - d_{i-1} = k$.

В первом случае дыру будем называть *внешней*, а во втором – *внутренней*. Отсутствующие в D элементы будем называть *элементами дыры*, а соседние с граничными для дыры элементы D будем называть её *нижней и верхней границами*. Во время работы с множеством результатов D будем называть дыру *активной*, если элементы этой дыры присутствуют в s; в противном случае дыру будем называть *пассивной*. (В начале работы каждого из описываемых далее алгоритмов все дыры являются активными.) В Hole-подобных алгоритмах в процессе восстановления множества D мы постепенно переделываем активные дыры в пассивные; и восстановление множества D завершается тогда, когда D будет содержать только пассивные дыры.

Мы начнём рассмотрение Hole-подобных алгоритмов с алгоритма hole-map («карта дыр»), после чего рассмотрим алгоритмы Hole-pairs («пары дыр»), Hole-Max («максимальная дыра») и Hoile-Shift («сдвиг дыр»).

4.1 Алгоритм Hole-Map («карта дыр»)

Карта дыр множества результатов $D = \{d_1, \ldots, d_m\}$ – это двумерный массив $H(D) = H[1\,..\,m, 1\,..\,d_m]$ (размерности $m \times d_m$), где j-й столбец матрицы $H(D)$ описывает активные дыры размерности j; а именно, элемент $H[i, j]$ $(1 \le j \le N[i])$ даёт стартовый адрес i-й

дыры размерности j множества D (если таковая существует – иначе значение $H[i, j]$ не определено).

Следующий алгоритм Hole-Map генерирует карту дыр $H(D)$, а также т.н. «вектор частоты дыр» (the hole frequency vector) множества D; для последнего в $N[0]$ заносится общее число активных дыр этого множества, а в $N[i]$ $(1 \le i \le m)$ – число его активных i-дыр. [4]

Алгоритм Hole-Map(m, D)

Вход:
$m = m(D)$ $(m \ge 1)$ – число элементов множества D;
$D = \{d_1, \ldots, d_m\}$ – множество результатов из m элементов.

Рабочие переменные:
i, j – переменные циклов.

Выход:
$H([1\,..\,m, 1\,..\,d_m]$ – карта дыр множества D;
$N[0\,..\,m] = N(D)$ – вектор частоты дыр множества D.

Примечание.
D, m и c – глобальные константы;
$H(D)$ – глобальный динамически изменяющийся вектор.

Описание алгоритма.

```
// Строки 01–02: инициализация
01 for i = 0 to d_m
02     N[i] = 0
// Строки 03–06: это внешняя дыра?
03 if d_1 > 0
04    N[0] = 1
05    N[d_1] = 1
06    H[1, d_1] = 0
// Строки 07–11: исследование внутренних дыр
07 for j = 1 to m − 1
08     if d_{j+1} − d_j > 1
09         N[d_{j+1} − d_j − 1] = N[d_{j+1} − d_j − 1] + 1
10         N[0] = N[0] + 1
11         H[N[d_{j+1} − d_j − 1, d_{j+1} − d_j − 1] = d_{j+1} + 1
// Строки 12–13: во входных данных дыр нет
12 if N[0] = 0
```

[4] Применяемый далее псевдокод согласован с [16]. В этой же монографии приведены также используемые нами обозначения, связанные с временно́й оценкой сложности алгоритмов.

13 **return** *'во входных данных дыр нет'*
// Строки 14: окончание работы алгоритма
14 **return** H, N

Рассмотрим пример – использование алгоритма Hole-map для множества результатов $D = \{2, 4, 6, 7\}$. Размер вектора N при этом равен 7. Таблица 1 содержит результат работы алгоритма – вектор $H(D)$.

номер/размер	1	2	3	4	5	6	7
1-я дыра	2	0	−−	−−	−−	−−	−−
2-я дыра	4	−−	−−	−−	−−	−−	−−
3-я дыра	−−	−−	−−	−−	−−	−−	−−
4-я дыра	−−	−−	−−	−−	−−	−−	−−

Таблица 1: Карта дыр $H(D)$ множества результатов $D = \{2, 4, 6, 7\}$.

4.2 Алгоритм Hole-Pairs («пары дыр»)

Будем говорить, что две i-дыры формируют *пару дыр*. При этом ту дыру, которая при построении множества D появляется ранее, будем называть *нижней дырой пары*, а другую – *верхней*.

Описываемые нами алгоритмы не изменяют результаты матчей между игроками, один из которых входит в какую-либо активную дыру [5], а другой – не входит в неё. Описываемые нами алгоритмы начинают работать с некоторыми стартовой последовательности результатов – соответствующей некоторому турниру, в котором каждый из игроков, входящих в какую-либо активную дыру, обязательно выигрывает у игроков, результаты которых меньше нижней границы этой дыры, и обязательно проигрывает игрокам, результаты которых больше верхней её границы. Мы будем называть такие последовательности *дыро-транзитивными* – и, согласно отмеченному выше ограничению на алгоритмы, это свойство сохраняется в процессе работы.

Описываемый нами далее алгоритм Hole-Pairs формирует максимально возможные количества пар i-дыр множества D для всех возможных i ($1 \le i \le d_m$; в наших обозначениях эти значения суть $\lfloor N[i]/2 \rfloor$) и удаляет элементы дыры из обрабатываемой последовательности.

[5] Такое возможно *в процессе работы* алгоритмов. *(Прим. ред.)*

Алгоритм Hole-Pairs(N)

Вход:
множество D и соответствующий ему вектор частоты дыр $N(D)$.

Глобальные переменные:
$D = \{d_1, \ldots, d_m\}$ – множество результатов;
H – карта дыр множества D;
$s = (0, 1, \ldots, d_m)$ – начальная транзитивная последовательность результатов;

Рабочие переменные:
i, j – переменные циклов.

Выход:
$t = (t_0, t_1, \ldots, t_{d_m})$ – сгенерированная последовательность результатов;
$M(D)$ – изменённый вектор частоты дыр множества D.

Описание алгоритма.

```
// Строки 01–03: инициализация
01 for i = 0 to d_m
02     M[i] = N[i]
03     t[i] = i
// Строки 04–11: обработка пар дыр
04 for i = d_m downto 1
05     while M[i] > 1
// Строки 06–07: обработка верхней дыры
06             for j = H[M[i], i] to H[M[i], i] + i − 1
07                 t[j] = t[H[M[i], i]]
// Строки 08–09: обработка нижней дыры
08             for j = H[M[i] − 1, i] to H[M[i] − 1, i] + i − 1
09                 t[j] = t[H[M[i − 1] + i, i]]
// Строки 10: изменение значения M[i]
10             M[i] = M[i] − 2
// Строка 11: окончание работы алгоритма
11 return t, M
```

Рассмотрим пример работы алгоритма. Пусть входными данными алгоритма Hole-Pairs является карта дыр $H(D)$ множества результатов $D = \{2, 4, 6, 7\}$, которое рассматривалось выше. Тогда алгоритм начинает работать с транзитивной последовательности результатов $t = (0, 1, \ldots, 7)$, и, обрабатывая 1-дыры, соответствующие

элементам 3 и 5, выдаёт на выходе сокращённую последовательность $t = (0, 1, 2, 4^3, 6, 7)$. Число активных дыр множества D уменьшается до $M[0] = 1$.

Несложно показать, что время выполнения алгоритма Hole-Pairs для всех входов может быть записано как $\Theta(d_m)$.

4.3 Алгоритм Hole-Max («максимальная дыра»)

Если $M[0](D) > 0$, то имеется по крайней мере одна активная дыра и мы можем с помощью описанного далее простого алгоритма Hole-Max удалить одну из наибольших дыру: а именно, если наибольшая дыра – c-дыра, то мы можем добавить к последовательности t элементы $d_m + 1$, $d_m + 2$, ..., $d_m + c$, после чего уменьшение этих элементов на d_m позволяет увеличить элементы рассматриваемой c-дыры до значения её верхней границы.

Алгоритм Hole-Max(M, t)

Вход:
вектор M(D), где M[0] является количеством активных дыр множества D, а каждое M[i] ($1 \le i \le d_m$) является числом активных i-дыр этого множества;
$t = (t_0, t_1, \ldots, t_{d_m})$ – последовательность, полученная алгоритмом Hole-Pairs.

Глобальные переменные:
D – множество результатов;
H – карта дыр множества D.

Рабочие переменные:
i, j, k – переменные циклов.

Выход:
$u = (u_0, u_1, \ldots, u_{d_m+c})$ – изменённая последовательность результатов;
O[0](D) – изменённое количество активных дыр множества D;
O[i](D) ($1 \le i \le d_m$) – изменённое количество активных i-дыр в D;
c – размер наибольшей активной дыры в t.

Описание алгоритма.

```
// Строки 01–02: инициализация
01 for j = 0 to d_m
```

```
02      O[j] = M[j]
// Строки 03–14: удаление пар дыр
03 c = d_m
// Строки 04–05: поиск наибольшей дыры
04 while M[c] = 0
05          c = c − 1
// Строки 06–09: добавление к u наибольшего элемента
06 for j = 0 to d_m
07      u_j = t_j
08 for j = d_m + 1 to d_m + c
09      u_{d_m+j} = d_m + j
10 for k = 1 to c
// Строка 11: обработка нижней дыры
11      u_{H[1,c]+k} = u_{H[1,c]+c}
// Строка 12: обработка верхней дыры
12      u_{H[1,c]+k} = u_{H[1,c]+c}
// Строки 13–14: изменение числа активных дыр
13 O[i] = O[i] − 1
14 O[0] = O[0] − 1
// Строки 15-16: активные дыры отсутствуют
15 if O[0] = 0
16      return 'активные дыры отсутствуют'
// Строка 17: возврат результатов и окончание работы алгоритма
17 return c, u, O
```

Продолжим рассмотрение предыдущего примера. Пусть, как и ранее, $D = \{2, 4, 6, 7\}$, а входом алгоритма Hole-Max является последовательность $t(D)$; тогда алгоритм Hole-Max уменьшает оставшуюся дыру, и мы получаем соответствующую D последовательность результатов $u = (2^3, 4^3, 6, 7^3)$.

4.4 Алгоритм Hole-Shift («сдвиг дыр»)

После применения алгоритма Hole-Pairs для каждого возможного i имеется по крайней мере одна i-дыра; иными словами, $0 \le O[i] \le 1$ для каждого i, такого что $1 \le i \le d_m$. Описанный в предыдущем подразделе алгоритм Hole-Max устранял наибольшую дыру путём добавления c результатов (а именно – $d_m + 1, \ldots, d_m + c$) ко входной последовательности t и уменьшению этих значений на d_m. А в описываемом далее алгоритме Hole-Shift мы пытаемся её удалить

иным способом.

Мы будем использовать степенну́ю форму $w = (w_1^{<e_1>}, \ldots, w_n^{<e_n>})$ записи входной последовательности u. Игрока, имеющего u_j очков, назовём T_j. Некоторый элемент w_k, входящий в w, будем называть игроком, *отказывающимся от побед*, в случае, если его степень e_k больше 1. Если w_k является верхней границей активной дыры и $j < k$, то игрок T_k может передать $e_k - 1$ очков игрокам, входящим в рассматриваемую активную дыру – для увеличения их очков на w_k. Если $2 \leq j \leq q$, $e_j > 1$ и $w_j - w_{j-1} = 1$, то $e_j - 1$ значений из общего числа e_j результатов w_j могут быть уменьшены на 1. Эти $e_j - 1$ очков мы будем называть *свободными очками*. Заранее отметим, что описываемый нами алгоритм Hole-Shift развивает эту идею на общий случай – когда $w_j - w_{j-1} > 1$.

Списком дыр назовём пару векторов, состоящую из *вектора начальных значений* $b(D) = (b_1, \ldots,, b_{O[0]})$ и *вектора длин* $l(D) = (l_1, \ldots, l_{O[0]})$; при этом b содержит индексы (номера) начальных значений активных дыр в порядке возрастания, а l – длины соответствующих дыр. Например, если $D = \{2, 5, 9\}$, то $b(d) = (0, 3, 6)$ и $l(D) = (2, 2, 3)$.

Предлагаемый далее алгоритм Hole-Shift находит максимальную дыру в рассматриваемой последовательности результатов и пытается переместить эту дыру – путём передачи очков от выбранного нами отказывающимся от побед игрока игрокам, находящимся в рассматриваемой активной дыре. [6]

Алгоритм Hole-Shift(P)

Вход:
вектор $O(D)$, где $O[0]$ является количеством активных дыр множества D, а каждое $O[i]$ $(1 \leq i \leq d_m)$ является числом активных i-дыр этого множества;
V – сумма длин активных дыр множества D;
$u = (u_0, u_1, \ldots, u_{d_m+c})$ – последовательность, получаемая в результате работы алгоритма Hole-Max.

Глобальные переменные:
D – множество результатов;
H – карта дыр множества D;
c – длина наибольшей дыры, удалённой в процессе работы алгорит-

[6] Ещё раз отметим, что такая ситуация возможна *в процессе работы* алгоритма. *(Прим. ред.)*

ма Hole-Max.

Рабочие переменные:
$b(D) = (b_1, \ldots, , b_{O[0]})$ – вектор начальных значений множества D;
$l(D) = (l_1, \ldots, l_{NO[0]})$ – вектор длин для текущего множества D;
$w(D) = (w_1^{<e_1>}, \ldots, w_q^{<e_q>})$ – степенна́я форма записи u;
q – длина степенно́й формы записи u;
g – размер максимальной текущей активной дыры;
a – индекс (номер) обрабатываемой максимальной активной дыры;
вектор $b = (b_1, \ldots, b_{O[0]})$: элемент b_i является стартовым индексом i-й дыры ($1 \le i \le O[0]$);
вектор $l = (l_1, \ldots, l_{O[0]})$: l_i является длиной i-й дыры ($1 \le i \le O[0]$);
$h = O[0]$ – количество активных дыр;
s – разность между двумя (обрабатываемыми) соседними результатами в последовательности w;
r – число очков, необходимых для того, чтобы в исследуемой активной дыре переместить очки к её верхней границе;
i, j, k – переменные циклов.

Выход:
$v = (v_1, \ldots, v_{d_m+c+1})$ – изменённая последовательность результатов;
$P[0](D)$ – число активных дыр множества D;
$P[i](D)$ ($1 \le i \le m$ – число активных i-дыр этого множества D.

Описание алгоритма.

```
// Строки 01–05: инициализация
01 for i = 0 to d_m
02     P[i] = O[i]
03 for i = 1 to d_m + c
04     v[i] = u[i]
// Строки 06–14: вычисление степенной формы u
06 w_1 = u_1
07 e_1 = 1
08 q = 1
09 for k = 2 to d_m + c
10     if u_k = u_{k-1}
11         e_k = e_k + 1
12     else q = q + 1
13         w_q = u_k
14         e_q = 1
// Строки 15–24: вычисление b и l
15     if d_1 > 0 и u_1 = 0
```

```
16        b_1 = 0
17        l_1 = d_1
18        h = 1
19      else h = 0
20          for i = 1 to m + c − 1
21              if d_{i+1} − d_i ≥ 2
22                  h = h + 1
23                  b_h = d_i + 1
24                  l_h = d_{i+1} − d_i − 1
// Строка 25: выполняем, пока имеется активная дыра
25 while P[0] > 0
// Строки 26–31: поиск наибольшей активной дыры
26        b = d_m
27        a = 1
28        for i = 1 to h
29            if l_i > l_a
30                a = i
31                b = l_a
// Строки 32: число требуемых очков
32        r = b(b − 1)/2
33        while r > 0
// Строки 34–36: поиск начального числа очков
34                j = 1
35                while w_j < b_a + l_a and e_j = 1
                        and w_j − w_{j−1} > r and j ≤ q
36                    j = j + 1
// Строки 37–38: неполный результат работы
// (к результатам необходимо применить другие алгоритмы)
37                    if j > q
38                    return 'Hole-Shift недостаточен'
39                y = ⌊r/(w_j − w_{j−1}
40                r = min(e_j − 1, y)(w_j − w_{j−1}))
41                e_j = e_j − min(e_j − 1, y)
42        for i = 1 to l_a
43            w_{j−1} + 1 = w_{j−1} + l_a
44        q = q − l_a
45 return P, w, e
```

Время выполнения алгоритма Hole-Shift для всех входов может быть записано как $\Theta(d_m)$.

5 Восстановление множества результатов турнира: алгоритм Prefix-Deletion

Ранее в [11, 12] мы описали алгоритм Shortening. Его частью является удаление из рассматриваемого множества результатов стартового нулевого элементов и уменьшения оставшихся элементов на 1.

В настоящей статье мы обобщаем эту идею. Описанные ранее алгоритмы были неспособны восстановить множество результатов $D = \{1, 7, 10\}$. Согласно теореме Ландау каждое соответствующее множество результатов должно содержать 1, 2 либо 3 элемента, равных 1. Если существуют ровно 3 элемента, равных 1, то каждый из соответствующих игроков набирает ровно 1 очко в микротурнире между ними, поэтому временно их удаляя, мы получаем множество результатов $D' = \{4, 7\}$, соответствующее решению $s(D') = (4^6, 7^6)$; следовательно, окончательным ответом является $s(D) = (1^3, 7^6, 10^6)$.

В общем случае мы должны исследовать ещё и такие ситуации, когда соответствующая последовательность содержит 1 или 2 элемента, равных 1. Итак, на основе теоремы Ландау возникает такая естественная идея расширения наших алгоритмов на общий случай (т.е. когда $1 \leq k \leq d_1$): в начале работы строящиеся соответствующие последовательности содержат p элементов 1, где $1 \leq p \leq 2d_1 + 1$.

При этом, поскольку $d_m < 12$, достаточно рассмотреть случай $k = 1$ и $p = 3$; для него мы при описании нашего алгоритма будем использовать приведённый ранее также в [11, 12] вспомогательный алгоритм Sequence-Base – дающий ответ для любого множества результатов, для которого выполнено условие $d_m < 12$. Входом этого вспомогательного алгоритма в нашей ситуации является «урезанное» множество D (а именно – $D' = \{d_2 - 3, \ldots, d_{m+c-1} - 3\}$), а выходом – соответствующая множеству D' последовательность результатов ($w = (w_1, \ldots, w_y)$, а также её длина y, заранее неизвестная).

Алгоритм Prefix-Deletion(Q, W)

Вход:
вектор $Q(D)$, где $Q[0]$ является количеством активных дыр множества D, а каждое $Q[i]$ ($1 \leq i \leq d_m$) является числом активных i-дыр этого множества;

$v = (v_0, v_1, \ldots, v_{d_m+c})$ – последовательность, получаемая в результате работы алгоритма Hole-Max.

Рабочие переменные:
$b(D) = (b_1, \ldots, , b_{O[0]})$ – вектор начальных значений множества D;
$l(D) = (l_1, \ldots, l_{NO[0]})$ – вектор длин для текущего множества D;
$w(D) = (w_1^{<e_1>}, \ldots, w_q^{<e_q>})$ – степенна́я форма записи u;
i, j, k – переменные циклов.

Выход:
$x = (x_1, \ldots, x_{3+y})$ – последовательность результатов, соответствующая D.

Описание алгоритма.

```
// Строки 01–06: инициализация
01 R = Q
02 for i = 1 to d_m
03     R[i] = Q[i]
04     w[i] = v[i]
05 for i = d_m + 1 to d_m + b
06     w[i] = w[i]
// Строки 07–08: неполный результат работы
// (к результатам необходимо применить другие алгоритмы)
07 if m < 4 or d_1 ≠ 1 or d_2 < 3
08    'Prefix-Deletion недостаточен'
09 for i = 1 to m + c − 1
10     d'_i = d_{i+1} − 3
// Строка 11: вызов Sequence-Base
11 Sequence-Base(D')
// Строка 12: восстановленная последовательность
// Строка 12: начинается 3 элементами, равными 1
12 x_1 = x_2 = x_3 = 1
// Строки 13–14: вычисление дальнейших элементов
// последовательности x
13 for k = 4 to y + 3
14     x_k = w_{k−3}
// Строка 15: окончание работы алгоритма
15 return x
```

Можно показать, что время выполнения алгоритма Hole-Pairs для всех входов может быть записано как $\Omega(d_m)$.

6 Некоторые результаты вычислительных экспериментов

Алгоритм Balancing восстанавливает любое множество результатов, для которого $d_m < 6$, но не может восстановить, например, множества результатов $\{1,3,6\}$ и $\{1,2,3,5,6\}$ (см. [11]). Для этих входных множеств даёт решения алгоритм Shortening – соответственно $(1^3,3,6^5)$ и $(1^2,2,3^2,5,6)$. Однако оба последних алгоритма не могут восстановить множества результатов $(1,2,3,5,7)$ и $(1,2,3,4,6,7)$. Для первого из этих множеств алгоритм Shiftening даёт последовательность $(1^2,2,3^2,5,7^3)$, а для второго – $(1,2,3,4^2,6^4,7)$, см. [12].

Алгоритмы Balancing, Shortening и Shiftening восстанавливают большинство из рассматривавшихся нами множеств результатов, для которых $d_m \le 8$. Исключениями являются только множества $\{1,2,3,5,7,8\}$ и $\{1,2,3,4,6,7,8\}$. Во втором случае алгоритм Hole-Pairs выдаёт ответ $(1^2,2,3^2,4,6,7,8)$, а в первом случае алгоритм Hole-Max даёт последовательность результатов $(1^2,2,3^2,5^2,7,8^2)$.

Если $d_m = 9$, то для трёх первых алгоритмов «критическими» множествами являются $\{2,4,5,6,7,8,9\}$, $\{1,2,5,6,7,8,9\}$, а также $\{1,2,4,7,8,9\}$. Для всех этих множеств задача восстановления решается с помощью алгоритма Hole-Max: соответствующие выходные последовательности суть: в первом случае $(2^3,4^2,5,6,7,8,9^2)$, во втором – $(1^2,2,5^2,6,7,8,9^2)$, а в третьем – $(1^2,2^2,4,7^3,8,9^3)$.

Если $d_m = 10$, то имеется 18 последовательностей, которые не удаётся восстановить при использовании только алгоритмов Balancing, Shortening и Shiftening. В таблице 2 приведена работа указанных алгоритмов (P = Hole-Pairs, M = Hole-Max, S = Hole-Shift, X = Prefix-Shift и R = Recursive-Shift) для всех таких последовательностей. Таблица также содержит наиболее короткое решение, найденное путём применения переборного алгоритма Diophantine, описанного в [11]. Отметим, что последний алгоритм в качестве вспомогательного использует алгоритм, описанный Д. Кнутом в [17, разд. 7.2.1.4].

А если $d_m = 11$, то существуют 72 последовательности, которые не удаётся восстановить при использовании тех же самых алгоритмов Balancing, Shortening и Shiftening. Большинство из этих множеств могут быть восстановлены путём добавления к этим трём

n	D	s	Алгоритмы	DIOPHANTINE
1	$\{2,4,5,6,7,8,9,10\}$	$(3,2,1,1,1,1,2,2)$	M + S	$4,1,1,1,1,1,2,1$
2	$\{2,3,5,7,9,10\}$	$(3,1,2,2,2,4)$	P + M + S	$3,2,2,2,1,1$
3	$\{1,7,10\}$	$(3,6,6)$	X	same
4	$\{1,4,5,7,9,10\}$	$(2,3,1,2,2,4)$	P + M + S	$3,1,3,2,1,1$
5	$\{1,3,6,8,9,10\}$	$(2,2,3,2,1,4)$	P + M + S	$1,5,2,1,1,1$
6	$\{1,3,6,7,8,10\}$	$(2,2,3,1,1,5)$	P + M + S	$3,1,4,1,1,1$
7	$\{1,3,4,7,9,10\}$	$(2,2,1,3,2,4)$	P + M + S	$3,1,4,1,1,1$
8	$\{1,3,4,5,8,10\}$	$(2,2,1,1,3,5)$	P + M + S	$2,2,2,1,3,1$
9	$\{1,2,5,7,9,10\}$	$(2,1,3,2,2,4)$	P + M + S	$2,1,5,1,1,1$
10	$\{1,2,5,6,7,8,10\}$	$(2,1,3,1,1,2,3)$	P + M	$2,2,2,1,1,2,1$
11	$\{1,2,4,8,9,10\}$	$(2,2,1,4,1,4)$	P + M	$1,3,2,4,1,1$
12	$\{1,2,4,6,9,10\}$	$(2,1,2,2,3,4)$	P + M	$2,1,2,4,1,1$
13	$\{1,2,3,7,8,10\}$	$(1,2,3,7,8,10)$	P + M	$1,1,4,2,2,1$
14	$\{1,2,3,5,8,10\}$	$(2,1,1,2,3,5)$	P + M	$1,2,2,2,3,1$
15	$\{1,2,3,5,8,9,10\}$	$(2,1,2,1,3,1,3)$	P + M	$1,2,1,4,1,1,1$
16	$\{1,2,3,5,7,10\}$	$(2,1,1,2,2,6)$	P + M + S	$2,1,1,2,4,1$
17	$\{1,2,3,5,6,9,10\}$	$(2,1,2,1,1,3,3)$	P + M	$2,1,1,1,4,1,1$
18	$\{1,2,3,4,6,7,10\}$	$(2,1,1,2,1,1,5)$	P + M	$2,1,1,1,1,4,1$

Таблица 2: Работа алгоритмов восстановления для «критических» множеств результатов, заканчивающихся на $d_m = 10$.

основным алгоритмам только двух новых HOLE-подобных (самых простых – P = HOLE-PAIRS и M = HOLE-MAX). Таблица 3 аналогична предыдущей таблице 2, но содержит только те примеры (30 множеств), для восстановления которых нужен по крайней мере один алгоритм из следующих: HOLE-SHIFT, PREFIX-SHIFT и RECURSIVE-SHIFT.

Представляет интерес анализ длин «критических» множеств. Согласно лемме 3, для длины n последовательностей результатов, соответствующих множествам $D = \{d_1, d_2, \ldots, d_m\}$, выполнена оценка $2d_1 + 2 \leq n \leq 2d_m$; как видно из изложенного выше, в нашем случае эти границы достижимы.

7 Заключение

Проверяя все необходимые множества результатов описанными выше полиномиально-временны́ми аппроксимационными алгоритмами, мы доказываем теорему 2 для всех множеств результатов, максимальный элемент которых меньше 12. При этом наше доказательство является конструктивным, поскольку мы генерируем последовательности, соответствующие заданным множествам результатов.

Список спецификаций и псевдокоды рассматриваемых нами ап-

n	D	s	Алгоритмы	DIOPHANTINE
1	$\{3,5,6,7,8,9,10,11\}$	$5,1,1,2,1,1,1,1$	M + S	same
2	$\{3,4,5,6,7,10,11\}$	$4,1,1,1,1,6,1$	M + S	$4,2,1,1,2,1,1$
3	$\{2,4,5,6,7,8,9,10,11\}$	$3,2,1,1,1,1,1,2,2$	M + S	$4,1,1,1,1,1,1,2,1$
4	$\{2,3,5,9,10,11\}$	$3,1,2,4,1,5$	M + S	$1,3,5,1,1,1$
5	$\{2,3,5,7,9,10,11\}$	$3,1,2,2,2,1,4$	P + M + S	$3,2,2,2,1,1,1$
6	$\{2,3,4,8,9,11\}$	$3,1,1,4,1,6$	M + S	$3,1,3,2,2,1$
7	$\{2,3,4,5,6,8,9,10,11\}$	$3,1,1,1,1,2,1,2,2$	M + S	$3,1,2,1,1,1,1,1,1$
8	$\{2,3,4,5,6,7,8,10,11\}$	$3,1,1,1,1,1,1,3,2$	M + S	$3,1,1,1,2,1,1,1,1$
9	$\{2,3,4,5,6,7,8,9,11\}$	$1,2,3,1,1,1,1,1,1$	R	$3,1,1,1,1,2,1,1,1$
10	$\{1,7,11\}$	$3,2,14$	X	$3,1,3,2,1,1,1$
10	$\{1,4,5,7,9,10,11\}$	$2,3,1,2,2,1,4$	P + M + S	$3,1,3,2,1,1,1$
11	$\{1,3,5,6,7,10,11\}$	$2,2,2,1,1,3,4$	P + M + S	$3,1,3,2,1,1,1$
12	$\{1,3,4,5,8,10,11\}$	$2,2,1,1,3,2,4$	P + M + S	$2,2,2,1,3,1,1$
13	$\{1,2,6,8,9,11\}$	$2,1,4,3,1,4$	P + M + S	$2,2,4,2,1,1$
14	$\{1,2,5,9,10,11\}$	$2,1,3,4,1,5$	P + M + S	$2,2,3,4,1,1$
15	$\{1,2,4,8,10,11\}$	$2,1,2,5,2,3$	P + M + S	$1,2,4,3,1,1$
16	$\{1,2,4,8,9,11\}$	$2,1,2,5,1,4$	P + M + S	$1,2,4,2,2,1$
17	$\{1,2,4,7,9,10,11\}$	$2,1,2,3,2,1,4$	P + M + S	$1,3,2,3,1,1,1$
18	$\{1,2,4,6,9,10,11\}$	$2,1,2,2,3,1,4$	P + M + S	$2,1,2,4,1,1,1$
19	$\{1,2,4,5,7,10,11\}$	$2,1,2,1,2,3,4$	P + M + S	$2,1,2,1,4,1,1$
20	$\{1,2,3,7,8,9,10,11\}$	$2,1,1,4,1,1,2,3$	M + S	$2,1,2,3,1,2,1,1$
21	$\{1,2,3,6,10,11\}$	$2,1,1,3,4,5$	M + S	$1,1,2,6,1,1$
22	$\{1,2,3,5,8,9,10,11\}$	$2,1,2,1,3,1,3$	P + M + S	$1,2,1,4,1,1,1,1$
23	$\{1,2,3,5,7,11\}$	$1,1,1,5,2,3$	R	$1,2,1,1,6,1$
24	$\{1,2,3,5,7,10,11\}$	$2,1,1,2,2,3,4$	P + M + S	$1,2,3,5,7,10,11$
25	$\{1,2,3,5,7,8,11\}$	$2,1,1,2,2,1,6$	P + M + S	$2,1,1,2,2,3,1$
26	$\{1,2,3,5,6,9,11\}$	$2,1,1,2,1,3,5$	P + M + S	$2,1,1,3,1,3,1$
27	$\{1,2,3,5,6,8,11\}$	$2,1,1,2,1,2,6$	P + M + S	$2,1,1,2,1,4,1$
28	$\{1,2,3,4,9,10,11\}$	$2,1,1,1,5,2,4$	M + S	$1,1,2,3,4,1,1$
29	$\{1,2,3,4,5,9,10,11\}$	$2,1,1,1,1,4,2,3$	P + M	$2,1,1,1,2,4,1,1$
30	$\{1,2,3,4,5,6,7,8,11\}$	$2,1,1,1,1,3,2,3$	R	$1,1,2,1,1,1,1,3,1$

Таблица 3: Работа алгоритмов восстановления для «критических» множеств результатов, заканчивающихся на $d_m = 11$.

проксимационных алгоритмов, а также генерируемые ими последовательности результатов можно найти в [18].

8 Благодарности

Авторы выражают признательность профессору Золтану Каше (Zoltán Kása, Sapientia Hungarian University of Transylvania, Romania), а также неизвестным им рецензентам за полезные замечания и корректировку.

Список литературы

1. Gross J. L., Yellen J., Zhang P. Handbook of Graph Theory. – CRC Press, Boca Raton, FL, 2014. – 1633 p. ⇒43
2. Chartrand G., Lesniak L., Roberts J. Degree sets for digraphs // Periodica Mathematica Hungarica, Vol. 7, No. 1 (1976) 77–85. ⇒44
3. Landau H. H. On dominance relations and the structure of animal societies // Bulletin of Mathematical Biophysics, Vol. 15 (1953) 143–148. ⇒44, 45
4. Reid K. B. Tournaments: Scores, kings, generalizations and special topics // Congressus Numerantium, Vol. 115 (1996) 171–211. ⇒45, 48
5. Reid K. B. Score sets for tournaments // Congressus Numerantium, Vol. 21 (1978) 607–618. ⇒45
6. Hager M. On score sets for tournaments // Discrete Mathematics, Vol. 58 (1986) 25–34. ⇒45
7. Yao T. X. On Reid conjecture of score sets for tournaments // Chinese Science Bulletin, Vol. 34, No. 10 (1989) 804–808. ⇒ 45, 48
8. Wayland K. Bipartite score sets // Canadian Mathematical Bulletin, Vol. 26, No. 3 (1983) 273–279. ⇒45
9. Petrović V. On bipartite score sets // Zbornik radova Prirodno-matematičkog Fakulteta Ser. Mat., Universitat u Novom Sadu, Vol. 13 (1983) 297–303. ⇒45
10. Pirzada S., Naikoo T. A. On score sets in tournaments // Vietnam Journal of Mathematics, Vol. 34, No. 2 (2006) 157–161. ⇒45
11. Iványi A., Lucz L., Gombos G., Matuszka T. Score sets in multitournaments // I. Mathematical results, Annales Universitatis Scientiarum Budapestinensis de Rolando Eötvös nominatae, Sectio Computatorica, Vol. 40 (2013) 307–320. ⇒46, 47, 48, 59, 61
12. Iványi A., Elek J. Degree sets of tournaments // Studia Univ. Babeş-Bolyai, Informatica, Vol. 59, No. 1 (2014) 150–164. ⇒46, 59, 61
13. Iványi A., Phong B. M. On the unicity of the score sets of multitournaments // In: Fifth Conference on Mathematics and Computer Science (Debrecen, June 9–12, 2004), University of Debrecen, 2004, 117–126. ⇒46
14. Pirzada S., Iványi A., Khan M. A. Score sets and kings // In: Algorithms of Informatics, Iványi A., ed., Mondat, Vác, 2013, 1337–1389. ⇒48

15. Li Q. Some results and problems in graph theory // New York Academy of Science, Vol. 576, No. 1 (1989) 336–343. ⇒48
16. Кормен Т. Х., Лейзерсон Ч. И., Ривест Р. Л., Штайн К. Алгоритмы: построение и анализ, 2-е издание: Пер. с англ. М.: Издательский дом Вильямс, 2005. – 1296 с. ⇒51
17. Кнут Д. Е. Искусство программирования, том 4, А. Комбинаторные алгоритмы, часть 1: Пер. с англ. М.: Издательский дом Вильямс, 2013. – 960 с. ⇒61
18. Elek J. Programs and Tables. [Электронный ресурс] – Режим доступа: `http://elekjani.web.elte.hu` ⇒63

Reconstruction of score sets

A. Iványi J. Elek
Hungary, Budapest, Eötvös Loránd University
email: tony@inf.elte.hu,elekjani@caesar.elte.hu

Abstract. The score set of a tournament is defined as the set of its different outdegrees. In 1978, Reid published the conjecture that for any set of nonnegative integers D exists a tournament T whose degree set is D. Reid proved the conjecture for tournaments containing 1, 2, and 3 vertices.

In 1986, Hager published a constructive proof of the conjecture for 4 and 5 vertices. In 1989, Yao presented an arithmetical proof of the conjecture, but general polynomial construction algorithm is not known.

In 2013, the first author of this paper published polynomial time algorithms which reconstruct the score sets containing only elements less than 7. In 2014, we improved this bound to 9.

In this paper we present and analyze new algorithms Hole-Map, Hole-Pairs, Hole-Max and Hole-Shift and using them improve the above bound to 12, giving a constructive partial proof of Reid's conjecture.

We also described exact brute force algorithms Sequencing и Diophantine and based on them approximation algorithms Shiftening и Hole.

Since there exist quick (quadratic) algorithms to construct n-tournaments corresponding to a given score sequence, our algorithms construct only a suitable score sequence.

Keywords and phrases: tournament; score set; score sequence; approximation algorithms; Reid conjecture.

References

1. Gross J. L., Yellen J., Zhang P. (2014), Handbook of Graph Theory. – CRC Press, Boca Raton, FL, 1633 p.

2. Chartrand G., Lesniak L., Roberts J. Degree sets for digraphs // Periodica Mathematica Hungarica, Vol. 7, No. 1 (1976) 77–85.

3. Landau H. H. On dominance relations and the structure of animal societies // Bulletin of Mathematical Biophysics, Vol. 15 (1953) 143–148.

4. Reid K. B. Tournaments: Scores, kings, generalizations and special topics // Congressus Numerantium, Vol. 115 (1996) 171–211.

5. Reid K. B. Score sets for tournaments // Congressus Numerantium, Vol. 21 (1978) 607–618.

6. Hager M. On score sets for tournaments // Discrete Mathematics, Vol. 58 (1986) 25–34.

7. Yao T. X. On Reid conjecture of score sets for tournaments // Chinese Science Bulletin, Vol. 34, No. 10 (1989) 804–808.

8. Wayland K. Bipartite score sets // Canadian Mathematical Bulletin, Vol. 26, No. 3 (1983) 273–279.

9. Petrović V. On bipartite score sets // Zbornik radova Prirodno-matematičkog Fakulteta Ser. Mat., Universitat u Novom Sadu, Vol. 13 (1983) 297–303.

10. Pirzada S., Naikoo T. A. On score sets in tournaments // Vietnam Journal of Mathematics, Vol. 34, No. 2 (2006) 157–161.

11. Iványi A., Lucz L., Gombos G., Matuszka T. Score sets in multitournaments // I. Mathematical results, Annales Universitatis Scientiarum Budapestinensis de Rolando Eötvös nominatae, Sectio Computatorica, Vol. 40 (2013) 307–320.

12. Iványi A., Elek J. Degree sets of tournaments // Studia Univ. Babeş-Bolyai, Informatica, Vol. 59, No. 1 (2014) 150–164.

13. Iványi A., Phong B. M. On the unicity of the score sets of multitournaments // In: Fifth Conference on Mathematics and Computer Science (Debrecen, June 9–12, 2004), University of Debrecen, 2004, 117–126.

14. Pirzada S., Iványi A., Khan M. A. Score sets and kings // In: Algorithms of Informatics, Iványi A., ed., Mondat, Vác, 2013, 1337–1389.

15. Li Q. Some results and problems in graph theory // New York Academy of Science, Vol. 576, No. 1 (1989) 336–343.

16. Cormen T. H., Leiserson Ch. E., Rivest R. L., Stein C. Introduction to Algorithms (Third edition). – The MIT Press, Cambridge, 2009. – 1312 p.

17. Knuth D. E. The Art of Computer Programming, Volume 4A, Combinatorial Algorithms: Part 1 – Addison Wesley, Upper Saddle River, NJ, 2011. – 933 p.

18. Elek J. Programs and Tables. [Electronic resource] – Access mode: `http://elekjani.web.elte.hu`