УДК 519.175

DOI: 10.21779/2542-0321-2020-35-4-13–26

А.М. Магомедов[1], С.А. Лавренченко[2]

Вычислительные средства C# для решения задачи перечисления разбиений прямоугольника

[1] Дагестанский государственный университет; Россия, 367000, г. Махачкала, ул. М. Гаджиева, 43а; magomedtagir1@yandex.ru, yakubovaz@mail.ru;

[2] Российский государственный университет туризма и сервиса; Россия, 141221, Московская обл., Пушкинский р-н, д/п Черкизово, ул. Главная, 99; lawrencenko@hotmail.com

Рассматривается проблема перечисления всех разбиений прямоугольника заданных целочисленных размеров h×w на прямоугольники размеров 1×2, возникшая в 60-е годы XX в. при исследовании учеными-физиками вопросов термодинамики потоков жидкости. Выявление равносильности задаче перечисления всех совершенных паросочетаний в некотором планарном графе привлекло внимание к проблеме со стороны специалистов по дискретной математике. Несмотря на то, что исследованию задачи посвящены десятки работ, интерес к ней не ослабевает.

Приведен краткий обзор известных подходов к решению задачи с анализом их недостатков, в частности «классическая» формула двойного произведения включает действия над числами с плавающей запятой (значения тригонометрических функций и др.), что существенно сужает диапазон значений h и w, для которых возможно точное вычисление искомого значения количества всех возможных разбиений.

Указан способ вычисления коэффициентов рекуррентной формулы, использующей для нахождения a_h (количества разбиений при фиксированном значении параметра w) лишь операции сложения и умножения целых чисел.

Предложен двухшаговый подход к решению. На первом шаге с использованием классов *BigInteger* и *BigFloat* языка программирования C# выполняется трудоемкая работа по точному вычислению необходимого количества начальных членов последовательности a_1, a_2 …, которые в свою очередь предоставляют возможность для вычисления целочисленных коэффициентов для формирования рекуррентной формулы. На втором шаге рекуррентная формула обеспечивает эффективное и точное вычисление последующих членов данной последовательности, используя лишь операции сложения и умножения целых чисел.

В конце статьи сформулированы актуальные подзадачи рассматриваемой проблемы.

Ключевые слова: *прямоугольник, C#, паросочетание графа, перечисление, программное обеспечение.*

Введение

Димерным числом $h \times$-прямоугольника M (прямоугольника с шириной w и высотой h) будем называть количество способов $f(h,w)$ разбиения прямоугольника M на «костяшки домино» – 1×2-прямоугольники, расположенные вертикально или горизонтально. Исследования по проблеме вычисления димерных чисел были начаты в связи с вопросами термодинамики потоков жидкости. Первые результаты опубликованы в 1961 г.

Ниже приведен краткий обзор подходов к вычислению димерного числа: сведение к перечислению совершенных паросочетаний, *классическая формула* двойного

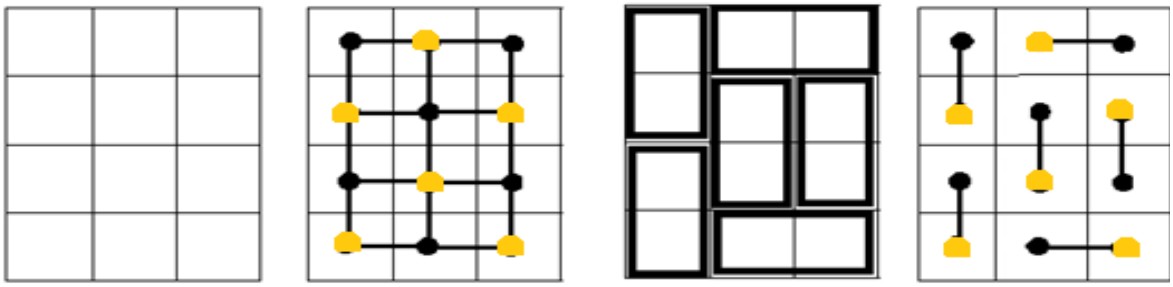
произведения, согласованная система взаимно-рекуррентных формул, динамическое программирование.

Раздел 3 посвящен двухшаговому подходу к решению задачи о димерном числе; предложены вычислительные средства C#, позволяющие получить по классической формуле точные значения димерных чисел при фиксированных (и относительно небольших) значениях параметра *w*; показано, как с помощью вычисленных значений формируется рекуррентная формула для вычисления димерных чисел с использованием операций сложения и умножения целых чисел.

Задача вычисления димерных чисел

Димерные числа и перечисление совершенных паросочетаний

Покажем на примере связь с задачей перечисления совершенных паросочетаний. Задача перечисления разбиений 4×3 – прямоугольника *M*, приведенного на рис. 1*а*, эквивалентна задаче перечисления совершенных паросочетаний в двудольном графе $G = G(M)$, полученном следующим образом: вершины графа – центры клеток исходного прямоугольника, ребра графа соединяют каждую пару вершин, соседних по вертикали или по горизонтали (рис. 1*б*). Очевидно, что каждому разбиению прямоугольника *M* (рис. 1*А*) взаимно-однозначно соответствует некоторое совершенное паросочетание графа *G* (рис. 1*Б*).



а б А Б

Рис. 1. Разбиению прямоугольника *M* (рис. 1*А*) взаимно-однозначно соответствует некоторое совершенное паросочетание (рис. 1*Б*) в плоском графе *G(M)*

Замечание 1. В общем случае задача перечисления совершенных паросочетаний графа #P-полна [1], однако в случае планарного графа – а граф *G*(*M*) , очевидно, планарный – задача разрешима за полиномиальное время.

В [2, с. 406–418] приведено систематическое изложение вопросов использования пфаффианов и определителей в задаче перечисления совершенных паросочетаний, в частности, для доказательства классического результата [4]: «Всякий планарный граф имеет пфаффову ориентацию и такую ориентацию можно построить за полиномиальное время». См. также [3].

Формула для вычисления димерных чисел

В [4] утверждается, что формула

$$f(h,w)=2^{\frac{wh}{2}}\prod_{j=1}^{w}\prod_{k=1}^{h}\left(\cos^2\frac{\pi j}{w+1}+\cos^2\frac{\pi k}{h+1}\right)^{1/4}$$

получена в [5] и [6], в [7] сообщается, что ее независимо получил и Д. Кнут; в [8] формула приведена в виде:

$$f(h,w)=\prod_{j=1}^{[\frac{w}{2}]}\prod_{k=1}^{[\frac{h}{2}]}\left(4\cdot\cos^2\frac{\pi j}{w+1}+4\cdot\cos^2\frac{\pi k}{h+1}\right). \qquad (1)$$

В качестве недостатка этой *классической формулы* укажем на вычисления с числами, представленными в форме с плавающей запятой, с неизбежным накоплением ошибок округления при больших значениях w и/или h (см. примеры в начале статьи [9]).

Системы взаимно-рекуррентных формул

Системы взаимно-рекуррентных формул (в. р. ф.), полученные в [9] с привлечением программного обеспечения, содержат лишь операции сложения и умножения целых чисел и свободны от отмеченного недостатка формулы (1). Однако системы в. р. ф., сгенерированные в [9], весьма громоздки: если при $w = 3$ система в. р. ф. содержит всего две формулы

$$\begin{cases} b_{r-1}=a_{r-2}+b_{r-3}, \\ a_r=a_{r-2}+2b_{r-1} \end{cases} \qquad (2)$$

(a_r – сокращенное обозначение димерного числа $r\times$-прямоугольника, b_r – димерное число фигуры, полученной из $r\times w$-прямоугольника удалением угловой клетки), то при $w = 6$ – уже 10, а при $w = 13$ – 3050 формул.

Пусть $w = 6$. При обозначениях:

$q^{(0)}=(0,0,0,0,0,0)$, $q^{(1)}=(0,0,0,0,1,1)$, $q^{(2)}=(0,0,1,1,1,1)$, $q^{(3)}=(1,0,0,1,0,0)$,
$q^{(4)}=(2,2,1,1,0,0)$, $q^{(5)}=(2,1,1,0,0,0)$, $q^{(6)}=(0,0,1,1,0,0)$, $q^{(7)}=(1,0,0,0,0,1)$,
$q^{(8)}=(2,1,0,0,1,0)$, $q^{(9)}=(1,1,1,0,0,1)$

и $F[n,h]$ – димерное число фигуры, полученной удалением из j-того столбца $h\times$-прямоугольника верхних $h-q_j^{(n)}$ клеток ($j=0,\dots,w-1$); $n=0,1,\dots,9$, система в. р. ф. S, сгенерированная изложенным в [9] методом, первоначально выглядит так:

$F[8,h]=F[7,h-1]+F[8,h-2]$,
$F[9,h]=F[0,h-1]+F[3,h-2]+F[7,h-2]+F[9,h-2]$,
$F[6,h]=F[2,h]+F[4,h]$,
$F[4,h]=F[1,h-1]+F[0,h-2]+F[6,h-2]$,
$F[7,h]=F[9,h]+F[8,h-1]+F[5,h-1]+F[7,h-2]$,
$F[3,h]=F[9,h]+F[1,h-1]+F[3,h-2]$,
$F[5,h]=F[7,h-1]+F[3,h-1]+F[9,h-1]$,
$F[2,h]=F[0,h-1]+F[1,h-1]$,
$F[1,h]=F[2,h]+F[3,h-1]+F[6,h-1]+F[2,h-1]$,
$F[0,h]=F[1,h]+F[5,h]+F[4,h]+F[9,h-1]+F[2,h-1]+F[0,h-2]$.

Если в системе в. р. ф. найдется формула с левой частью $F[j,h]$ и слагаемым $F[i,h]$ в правой части, то будем говорить, что формула $F[i,h]=\cdots$ *предшествует* формуле $F[j,h]=\cdots$. Для организации вычисления димерного числа $F[0,h]$ по системе в. р. ф. требуется:

1) для каждого $i = 0, 1, ...$ вычислить F[i, h] для нескольких начальных значений h;

2) упорядочить систему в. р. ф. таким образом, чтобы соблюдалось условие: если формула F[i, h] =… предшествует формуле F[j, h] = …, то в упорядочении системы в. р. ф. формула F[i, h] =… встречается раньше, нежели формула F[j, h] = … (такое упорядочение назовем *согласованием* системы в. р. ф.).

Пусть в общем случае система в. р. ф. содержит *n* формул. Рассмотрим ациклический орграф $G(S) = (V, E)$, вершины $v_0, v_1, \dots, v_{N-1}$ которого соответствуют формулам системы в. р. ф. *S*, дуги – отношениям предшествования формул: дуга из вершины v_i в вершину v_j проведена тогда и только тогда, когда *i*-тая формула предшествует *j*-той формуле. Вершины графа *G* можно таким образом пометить числами из множества $0, 1, \dots, N-1$, что если $(v_i, v_j) \in E$, то $i < j$ (топологическая сортировка [10, с. 95]). Индуцированное упорядочение системы в. р. ф. и будет искомым согласованием этой системы.

Алгоритм топологической сортировки [10, с. 96]: сначала произвольная вершина с нулевой полустепенью исхода помечается числом N – 1 и удаляется из графа вместе с инцидентными дугами, затем в оставшемся графе произвольная вершина с нулевой полустепенью исхода помечается числом N – 2; описанная процедура повторяется, пока не пометим все вершины.

Приведенная выше система *S* в. р. ф. не согласована. Укажем один из вариантов согласования (ограничиваясь краткой записью левых частей формул): $F[\,8, h] = \cdots$, $F[\,9, h] = \cdots$, $F[\,2, h] = \cdots$, $F[\,4, h] = \cdots$, $F[\,5, h] = \cdots$, $F[\,1, h] = \cdots$, $F[\,3, h] = \cdots$, $F[\,7, h] = \cdots$, $F[\,6, h] = \cdots$, $F[\,0, h] = \cdots$.

Замечание 2. Задача топологической сортировки текстуально близка к задаче поиска ориентированного гамильтонова пути в каждой компоненте ориентированного графа. В общем случае задача о гамильтоновом пути NP–полна [11, с. 249], но для ациклических ориентированных графов задача разрешима за полиномиальное время.

Замечание 3. На первый взгляд, примененный в [9] подход имеет сходство с предложенным в [12], но при ближайшем рассмотрении принципиальное различие подходов становится очевидным.

Метод динамического программирования

При малых *w* и *h* димерные числа $h \times$-прямоугольника могут быть найдены методом динамического программирования [13, с. 52–54] (см. также [14]). Опишем вкратце этот метод.

Пусть Q – фигура, полученная из прямоугольника с шириной *w* удалением из каждого столбца некоторого количества верхних клеток, d_n – высота *n*-ного столбца, $0 \le d_n, n = 1, \dots, w$; Q будем называть *диаграммой*, если $|m - i| \le 1$, где $i = max\,(d_1, \dots, d_w)$, $m = min\,(d_1, \dots, d_w)$; верхнюю границу диаграммы будем называть *ломаной* и обозначать (i, k), где *i – высота* диаграммы *Q*, а *k* (*дескриптор*) – число, записанное в двоичной системе цифрами: $d_1 - m, \dots, d_w - m$.

Замечание 4. Там, где это не вызывает недоразумений, диаграмму с верхней границей (*i, k*) также будем обозначать (*i, k*)/

Дескриптор горизонтальной ломаной равен 0. Если число способов разбиения диаграммы (*i, k*) обозначить через T(*i, k*), то задача о димерном числе $h \times$-прямоугольника заключается в вычислении $T(h, 0)$.

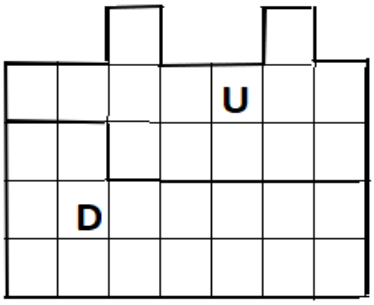


Рис. 2. Диаграмма Q с шириной 7 и высотой 5

На рис. 2 приведена диаграмма Q с шириной 7 и высотой 5. Выделены: 1) верхняя граница (i_1, k_1), где $i_1 = 5$ – высота, $k_1 = 0010010_2 = 18$ – дескриптор; 2) ломаная (i_2, k_2), $i_2 = 3$, $k_2 = 1100000_2 = 192$.

Если рассмотреть рис. 2 в общем случае и число способов разбиения фигур U и D обозначить через $T_u(i_2, k_2)$ и $T_d(i_2, k_2)$ соответственно (и при этом $T_u(i_2, k_2) > 0$, $T_d(i_2, k_2) > 0$), то число способов разбиения диаграммы Q на костяшки домино, каждая из которых находится целиком по одну сторону (выше или ниже) от ломаной (i_2, k_2), равно произведению $T_u(i_2, k_2) \cdot T_d(i_2, k_2)$. В случае, если фигура U допускает разбиение в точности одним способом, произведение равно $T_d(i_2, k_2)$.

Для каждой диаграммы (*i, k*) рассмотрим все диаграммы $(i-1, j)$. Дескрипторы k и j назовем *дружественными*, если существует разбиение диаграммы (i, k), сужение которого над диаграммой (i – 1, j) представляет собой разбиение диаграммы (i – 1, j), и *недружественными* – в противном случае. Дополняя диаграмму (i – 1, j) сверху до диаграмм с высотой *i* различными наборами костяшек домино, допускающих в точности одно разбиение, мы найдем все дескрипторы *k*, дружественные *j*. Множество дескрипторов, дружественных *k*, будем обозначать $D(k)$.

$$T(i, k) = \sum_{j \in D(k)} T(i-1, j). \quad (3)$$

Сформулируем алгоритм:

1. Для каждого $k = 0, 1, \ldots, 2^w - 1$ вычислить множество $D(k)$.

2. $T(0,0) := 1$, т. к. прямоугольник нулевой высоты имеет единственное разбиение (не использующее ни одной костяшки домино!); для $j = 1,2, \ldots, 2^w - 1$ выполнить $T(0, j) := 0$, т. к. соответствующих диаграмм не существует.

3. Для $i = 1, \ldots, h-1$; для $k = 0, 1, \ldots, 2^w - 1$ выполнить

$$T(i, k) := \sum_{j \in D(k)} T(i-1, j).$$

4. $T(h, 0) := \sum_{j \in D(0)} T(h-1, j)$.

Двухшаговый подход к вычислению димерных чисел
Средства многоразрядных вычислений в языке C#

Вернемся к классической формуле (1), полученной рядом авторов еще в 60-е годы XX в., и заметим, что с тех пор не только возросли возможности аппаратного обеспечения компьютеров, но и расширились средства языков программирования. Классы *BigInteger* и *BigFloat* языка C# позволяют значительно увеличить диапазон значений *h* и *w*, для которых удается выполнить точные вычисления по формуле (1).

Класс *BigInteger* из библиотеки *System.Numerics* обеспечивает выполнение вычислений с целыми числами практически неограниченной величины.

Класс *BigFloat* реализован в библиотеке *Extreme.Mathematics* и позволяет выполнять вычисления над вещественными числами с громадным количеством десятичных знаков после запятой. Библиотеку можно подключить через *NuGet* пакеты: https://www.nuget.org/packages/Extreme.Numerics/.

Примеры вычислений, выполненных с использованием этих классов, приведены в Приложениях 1, 2 и 3.

Задача нормализации системы в. р. ф.

В ряде источников показано, что при $w = 2$ (здесь a_r – то же, что в разделе 2.3):

$$a_r = a_{r-1} + a_{r-2}, r \geq 2,$$
$$a_0 = a_1 = 1;$$

другими словами, при $w = 2$ димерные числа – суть числа Фибоначчи, и система в. р. ф. вырождается в одну-единственную рекуррентную формулу с порядком рекурсии, равным 2.

Приведение системы в. р. ф. к эквивалентной рекуррентной формуле будем называть *нормализацией*. Легко показать, что при $w = 3$ система в. р. ф. допускает нормализацию с порядком рекурсии, равным 4. В самом деле, заменив b_{r-1} в первой формуле системы (2) на выражение $\frac{1}{2}(a_r - a_{r-2})$, полученное из второй формулы «непосредственно», а b_{r-3} – на выражение $\frac{1}{2}(a_{r-2} - a_{r-4})$, полученное из второй формулы после понижения индексов на 2, получим:

$$\frac{1}{2}(a_r - a_{r-2}) = a_{r-2} + \frac{1}{2}(a_{r-2} - a_{r-4})$$

или после упрощения:

$$a_r = 4 \cdot a_{r-2} - a_{r-4} \quad (4)$$

(ср. с [12], где для получения формулы (4) привлечен аппарат производящих функций). К формуле (4) мы еще вернемся в следующем разделе.

Замечание 5. Изучению возможностей метода производящих функций для вычисления димерных чисел посвящен ряд работ, например, [15, 16, 17]. В [17] с использованием модификации метода матрицы переноса (подробно об этом методе см. в [18, с. 500–523]) показано также, что при заданном w числа a_r удовлетворяют линейным рекуррентным соотношениям с постоянными целочисленными коэффициентами (см. также [19]).

Сформулируем задачу нормализации при некотором заданном w: найти натуральное R (*порядок рекурсии*) и целочисленные коэффициенты $x_1, x_2, \ldots, x_R$, такие, что

$$a_r = x_1 a_{r-1} + x_2 a_{r-2} + x_3 a_{r-3} + \cdots + x_R a_{r-R} \quad (5)$$

для каждого $r = R + 1,\ R + 2, \ldots$

Пусть выбрано некоторое w и известны димерные числа $a_1, \ldots, a_{2R}$, где по поводу R можно утверждать лишь, что это – «достаточно большое» число. Для определения R и коэффициентов $x_1, x_2, \ldots x_R$ рекуррентной формулы (5) используем следующее рабочее правило: вначале положим $R = w$; если система линейных алгебраических уравнений (с. л. а. у.)

$$\begin{cases} a_{R+1} = x_1 a_R + x_2 a_{R-1} + \cdots + x_R a_1, \\ \ldots\ldots\ldots \\ a_{2R} = x_1 a_{2R-1} + x_2 a_{2R-2} + \cdots + x_R a_R \end{cases}$$

или в матричной записи:

$$\begin{pmatrix} a_R \ \dots \ a_1 \\ \dots \\ a_{2R-1} \dots a_R \end{pmatrix} \begin{pmatrix} x_1 \\ \dots \\ x_R \end{pmatrix} = \begin{pmatrix} a_{R+1} \\ \dots \\ a_{2R} \end{pmatrix} \quad (6)$$

имеет целочисленное решение, то искомые коэффициенты найдены, в противном случае увеличим R на единицу и повторим попытку найти целочисленное решение системы (6).

В результате численных экспериментов получена следующая

Гипотеза. Порядок рекурсии R в (5) можно выбрать равным $2^{\lceil w/2 \rceil}$.

Двухшаговый подход к вычислению димерных чисел

Суть предлагаемого нами двухшагового подхода к вычислению димерных чисел такова. Пусть выбрано некоторое небольшое значение w.

На первом шаге средствами языка C# с применением классов *BigInteger* и *BigFloat* вычислить по формуле (1) значения

$$a_1, a_2, a_3, \dots, a_{2R} \quad (7)$$

(где $R = 2^{\lceil w/2 \rceil}$) и для формирования рекуррентной формулы (5) решить с. л. а. у. (6).

Замечание 6. Разумеется, для вычисления значений (7) можно применить и любой другой из описанных выше способов. Для решения системы (6) рекомендуется применить систему компьютерной математики (с. к. м.) *Wolfram Mathematica*, нетрудно также реализовать на языке C# метод Гаусса последовательного исключения переменных (сложность метода $O(R^3)$),когда с помощью элементарных преобразований система уравнений приводится к равносильной системе треугольного вида, из которой последовательно, начиная с последних (по номеру), находятся все переменные системы.

На втором шаге вычислить димерные числа a_r для произвольного $r = 2R + 1$, $2R + 2$, … по полученной рекуррентной формуле, используя только операции сложения и умножения.

Применим наш подход к рассмотренному выше случаю $w = 3$.

$\boldsymbol{R = 2^{\lceil w/2 \rceil} = 2^{\lceil 1.5 \rceil} = 2^2 = 4}$; далее по формуле (1):

a1 = 0, a2 = 3, a3 = 0, a4 = 11, a5 = 0, a6 = 41, a7 = 0, a8 = 153. (w1)

Дополним (w1) значениями a9 = 0, a10 = 571, избыточными для целей нормализации. Мы планируем использовать ниже значение a10 в качестве «проверочного значения».

Для экономии места выпишем матрицу коэффициентов M в левой части системы (6) и вектор коэффициентов в правой части (6) *построчно*:

M = {{a4, a3, a2, a1}, {a5, a4, a3, a2}, {a6, a5, a4, a3}, {a7, a6, a5, a4}}, (w2)

B = {a5, a6, a7, a8}. (w3)

В действительности мы применили способ записи команд, принятый в с. к. м. *Wolfram Mathematica*. Записи (w1), (w2), (w3), дополненные командой *LinearSolve* [M, B], будем называть сценарием с. к. м. *Wolfram Mathematica* для решения с. л. а. у. (6) или просто *сценарием*. Если наш сценарий поместить в окно с. к. м. *Wolfram Mathematica* и инспирировать командой Shift+Enter его выполнение, то увидим результат: $\{0,4,0,-1\}$, т. е. $x_1 = 0, x_2 = 4, x_3 = 0, x_4 = -1$, что соответствует формуле (4). Первый шаг завершен.

Замечание 7. Непривычное для математического текста написание переменной с индексом (*a10* вместо привычного a_{10} и т. п.) объясняется стремлением придерживаться в данной части статьи написания символов, «как в с. к. м. *Wolfram Mathematica*», поскольку приведенные в Приложениях 1, 2, 3 фрагменты значительных размеров пред-

ставляют практически точные копии текстов из окна данной с. к. м. – как сконструированных авторской программой, написанной на C# для передачи в окно с. к. м. (сценарий), так и сгенерированных самой с. к. м. (вектор коэффициентов x).

На втором шаге последовательно вычислим по формуле (5) значения a9, a10, a11 и т. д., используя только операции умножения и сложения целых чисел. Например,

$$a10 = x_1 \cdot a9 + x_2 \cdot a8 + x_3 \cdot a7 - x_4 \cdot a6 \;\; = \; 0 \cdot a9 + 4 \cdot a8 + 0 \cdot a7 - 1 \cdot a6 = \\ = 4 \cdot 153 - 1 \cdot 41 = 612 - 41 = 571.$$

Таким образом, результат совпал с «проверочным значением».

Для $w = 4$ и $w = 8$ в Приложении 1 приведены как сценарии, так и вычисленные по ним векторы коэффициентов соответствующих рекуррентных формул вида (5). Ввиду громоздкости записи сценария при $w = 13$ в Приложении 2 сценарий изложен в урезанном виде, однако вычисленный по ним вектор коэффициентов рекуррентной формулы (5) приведен полностью. Запись сценария для случая $w = 14$ заняла бы много страниц, поэтому в Приложении 3 мы ограничились вычисленным вектором коэффициентов x.

Замечание 8. Из численных экспериментов по вычислению вектора коэффициентов x для различных w видно, что последний ненулевой элемент вектора x равен –1; если обозначить его позицию в векторе x через N, то $|x_i| = |x_{N-i}|$, $i = 1,2,\ldots$ В Приложениях 1, 2 и 3 в записи вектора x элемент $x_{N/2}$ («центр симметрии») для наглядности выделен полужирным шрифтом. Мы не ставили целью в рамках данной статьи изложить исчерпывающее объяснение этих и других особенностей структуры вектора x (или даже рассмотрение их связи со спецификой системы (6) и с делимостью значения w на 2 или 4).

Заключение

В данной статье мы не уделили достаточного внимания некоторым важным аспектам проблемы вычисления димерных чисел: применению производящих функций (см. например, [15], [16], [18]), исследованию верхних оценок (см., например, [20], [21]) и др.

На наш взгляд, было бы интересно получить непосредственное доказательство существования нормализации (скажем, базирующееся на формуле (1)), а также уточнить или доказать гипотезу п. 3.2 о порядке рекурсии. Из Приложения 2 видно, что при $w = 13$ последние 16 из $2^{\lceil 13/2 \rceil} = 128$ коэффициентов рекуррентной формулы (5) равны нулю; следовательно, для этого случая порядок рекурсии меньше 128.

Интерес представляет также поиск исчерпывающего объяснения особенностей вектора коэффициентов x формулы (5), которые упоминаются в Замечании 8.

Авторы благодарят Т.С. Лугуева за ценные консультации по использованию класса BigFloat.

Работа подготовлена при поддержке отдела математики и информатики ДФИЦ РАН.

Литература

1. *Valiant L.G.* The complexity of computing the permanent // Theoretical Computer Science. – 1979. – Vol. 8 (2). – P. 189–201.

2. *Ловас Л., Пламмер М*. Прикладные задачи теории графов. Теория паросочетаний в математике, физике, химии. – М.: Мир, 1998. – 653 с.

3. *Вялый М.Н*. Пфаффианы или искусство расставлять знаки // Математическое просвещение. Сер. 3. – 2005. – Вып. 9. – С. 129–142.

4. *Matouvsek J*. Thirty-three Miniatures: Mathematical and Algorithmic Applications of Linear Algebra // Amer. Math. Soc. – 2010. – № 182.

5. *Kasteleyn P.W*. The statistic of dimers on a lattice I: The number of dimer arrangements on quadratic lattice // Physica. – 1961. – Vol. 27. – P. 1209–1225.

6. *Temperley H.N.V. and Fisher M.E*. Dimer problem in statistical mechanics – an exact result // Phil. Mag. – 1961. – Vol. 6. – P. 1061–1063.

7. *Klarner D. and Pollack J.* Domino tilings of rectangles with fixed width // Discrete Mathematics. – 1980. – Vol. 32. – P. 45–52.

8. *Нурлигареев Х*. Полоски из домино // Режим доступа: https://elementy.ru/problems/1612/Poloski_iz_domino, режим доступа свободный (дата обращения: 25.06.2020).

9. *Магомедов А.М., Магомедов Т.А., Лавренченко С.А.* Взаимно-рекуррентные формулы для перечисления разбиений прямоугольника // Прикладная дискретная математика. – 2019. – № 46. – С. 108–121.

10. *Swamy M.N., Thulasiraman K.* Graphs, Networks and Algorithms. – New York, Wiley-Inter-science, 1981. – 590 p.

11. *Garey Michael R. and Jonson David S*. Computers and Intractability. – San Francisco: W.H., Freeman and Company, 1979. – 347 p.

12. *Read Ronald C*. A note on tiling rectangles with dominoes // Fib. Q. – 1980. – Vol. 18 (1). – P. 24–27.

13. *Волченков С.Г*. Задача «Паркет» // Информатика и образование. – 1994. – № 3. – С. 52–54.

14. *Караваев А.М*. Метод динамического программирования для подсчета замощений домино на прямоугольной решетке и цилиндре // Актуальные проблемы гуманитарных и естественных наук. – 2013. – № 6 (53). – С. 13–18.

15. *Грэхем Р., Кнут Д., Паташник О.* Конкретная математика. Основание информатики. – М.: Мир, 1998. – 703 с. (Пер. с анг.).

16. *Караваев А.М., Перепечко С.Н*. Задача о димерах на цилиндрах: рекуррентные соотношения и производящие функции // Математическое моделирование. – 2014. – Т. 26, № 11. – С. 18–22.

17. *Faase F.J*. On the number of specific spanning subgraphs of the graphs $G \times P_n$ // Ars Combinatoria. – 1998. – Vol. 49. – P. 129–154.

18. *Stanley R.P*. Enumerative Combinatorics. Vol. 1, Wadsworth & Brooks. – Cole, Monterey, CA, 1986. – 626 p.

19. *Караваев А.М*. Вывод линейного рекуррентного соотношения с постоянными целыми коэффициентами по заданной целочисленной последовательности // Естественные и технические науки. – 2012. – № 5 (61). – С. 22–27.

20. *Perepechko S.N*. Estimation of molecular freedom in the dimer model by the EFM method // Proceedings of the VI international conference "Mathematics, its applications and mathematical education" (MAME-2017). – Ulan-Ude, 2017. – P. 289–294.

21. *Перепечко С.Н.* Простые выражения для оценки параметра молекулярная свобода в задаче о димерах // Вестник ТвГУ. Сер.: Прикладная математика. – 2018. – № 2. – С. 27–47.

Приложение 1. Сценарий и коэффициенты $x_1, \dots, x_R$.для $w = 4$ ($R = 4$) и $w = 8$ ($R = 16$).

$w = 4$

a1 = 1; a2 = 5; a3 = 11; a4 = 36; a5 = 95; a6 = 281; a7 = 781; a8 = 2245;
M = { {a4, a3, a2, a1}, {a5, a4, a3, a2}, {a6, a5, a4, a3}, {a7, a6, a5, a4} }
B = {a5, a6, a7, a8}
LinearSolve[M, B]
x = {1, **5**, 1, -1}

$w = 8$

a1 = 1; a2 = 34; a3 = 153; a4 = 2245; a5 = 14824; a6 = 167089; a7 = 1292697; a8 = 12988816;
a9 = 108435745; a10 = 1031151241; a11 = 8940739824; a12 = 82741005829; a13 = 731164253833;
a14 = 6675498237130; a15 = 59554200469113; a16 = 540061286536921; a17 = 4841110033666048;
a18 = 43752732573098281; a19 = 393139145126822985; a20 = 3547073578562247994;
a21 = 31910388243436817641; a22 = 287665106926232833093; a23 = 2589464895903294456096;
a24 = 23333526083922816720025; a25 = 210103825878043857266833;
a26 = 1892830605678515060701072; a27 = 17046328120997609883612969;
a28 = 153554399246902845860302369; a29 = 1382974514097522648618420280;
a30 = 12457255314954679645007780869; a31 = 112199448394764215277422176953;
a32 = 1010618564986361239515088848178
M = {
{a16, a15, a14, a13, a12, a11, a10, a9, a8, a7, a6, a5, a4, a3, a2, a1},
{a17, a16, a15, a14, a13, a12, a11, a10, a9, a8, a7, a6, a5, a4, a3, a2},
{a18, a17, a16, a15, a14, a13, a12, a11, a10, a9, a8, a7, a6, a5, a4, a3},
{a19, a18, a17, a16, a15, a14, a13, a12, a11, a10, a9, a8, a7, a6, a5, a4},
{a20, a19, a18, a17, a16, a15, a14, a13, a12, a11, a10, a9, a8, a7, a6, a5},
{a21, a20, a19, a18, a17, a16, a15, a14, a13, a12, a11, a10, a9, a8, a7, a6},
{a22, a21, a20, a19, a18, a17, a16, a15, a14, a13, a12, a11, a10, a9, a8, a7},
{a23, a22, a21, a20, a19, a18, a17, a16, a15, a14, a13, a12, a11, a10, a9, a8},
{a24, a23, a22, a21, a20, a19, a18, a17, a16, a15, a14, a13, a12, a11, a10, a9},
{a25, a24, a23, a22, a21, a20, a19, a18, a17, a16, a15, a14, a13, a12, a11, a10},
{a26, a25, a24, a23, a22, a21, a20, a19, a18, a17, a16, a15, a14, a13, a12, a11},
{a27, a26, a25, a24, a23, a22, a21, a20, a19, a18, a17, a16, a15, a14, a13, a12},
{a28, a27, a26, a25, a24, a23, a22, a21, a20, a19, a18, a17, a16, a15, a14, a13},
{a29, a28, a27, a26, a25, a24, a23, a22, a21, a20, a19, a18, a17, a16, a15, a14},
{a30, a29, a28, a27, a26, a25, a24, a23, a22, a21, a20, a19, a18, a17, a16, a15},
{a31, a30, a29, a28, a27, a26, a25, a24, a23, a22, a21, a20, a19, a18, a17, a16}
}
B = {a17, a18, a19, a20, a21, a22, a23, a24, a25, a26, a27, a28, a29, a30, a31, a32}
LinearSolve[M, B]
x = {1, 76, 69, –921, –584, 4019, 829, –7012, 829, 4019, –584, –921, 69, 76, 1, –1}

Приложение 2. W = 13. Сокращенный сценарий и коэффициенты $x_1, \dots, x_{128}$.

a1 = 0; a2 = 377; a3 = 0; a4 = 413351; …; a253 = 0;
a254 =
730569359515397369031028365313309444193874782061102221085674902387923935432979990675588876899534953925938798830339949539847219530439171630727145923951963368549322686109890167429756027300799048204554323059827998337417815952916331934487608323169766170145191630869690716377353696711930616320004240773215109059929792704782965868097449681340392093515537136146360612130448841814992199346868110935256598617;
a255 = 0;
a256 =
102345265667819288741904473152975412260871144504133513499343071674639439201302170 3

463908759575226691837792218983994564957803536908178497517390127035831344765285195999639871940237215457652438517206592379725802631218297481092162810454529131714829332696579717230500263420138525572857652706799821412375832576692563485613095178462366957427605659609089718416729010708239867977279547618836940818936648770788359;

```
M = {
{a128, a127, a126, a125, a124, a123, a122, a121, a120, a119, a118, a117, a116, a115, a114,
a113, a112, a111, a110, a109, a108, a107, a106, a105, a104, a103, a102, a101, a100, a99, a98, a97,
a96,
a95, a94, a93, a92, a91, a90, a89, a88, a87, a86, a85, a84, a83, a82, a81, a80, a79, a78, a77, a76, a75,
a74, a73,
a72, a71, a70, a69, a68, a67, a66, a65, a64, a63, a62, a61, a60, a59, a58, a57, a56, a55, a54, a53, a52,
a51,
a50, a49, a48, a47, a46, a45, a44, a43, a42, a41, a40, a39, a38, a37, a36, a35, a34, a33, a32, a31, a30,
a29, a28,
a27, a26, a25, a24, a23, a22, a21, a20, a19, a18, a17, a16, a15, a14, a13, a12, a11, a10, a9, a8, a7, a6,
a5, a4, a3, a2, a1},
……
{a255, a254, a253, a252, a251, a250, a249, a248, a247, a246, a245, a244, a243, a242, a241,
a240, a239, a238, a237, a236, a235, a234, a233, a232, a231, a230, a229, a228, a227, a226,
a225, a224, a223, a222, a221, a220, a219, a218, a217, a216, a215, a214, a213, a212, a211,
a210, a209, a208, a207, a206, a205, a204, a203, a202, a201, a200, a199, a198, a197, a196,
a195, a194, a193, a192, a191, a190, a189, a188, a187, a186, a185, a184, a183, a182, a181,
a180, a179, a178, a177, a176, a175, a174, a173, a172, a171, a170, a169, a168, a167, a166,
a165, a164, a163, a162, a161, a160, a159, a158, a157, a156, a155, a154, a153, a152, a151,
a150, a149, a148, a147, a146, a145, a144, a143, a142, a141, a140, a139, a138, a137, a136,
a135, a134, a133, a132, a131, a130, a129, a128}
}
B = {a129, a130, a131, a132, a133, a134, a135, a136, a137, a138, a139, a140, a141, a142,
a143, a144, a145, a146, a147, a148, a149, a150, a151, a152, a153, a154, a155, a156, a157,
a158, a159, a160, a161, a162, a163, a164, a165, a166, a167, a168, a169, a170, a171, a172,
a173, a174, a175, a176, a177, a178, a179, a180, a181, a182, a183, a184, a185, a186, a187,
a188, a189, a190, a191, a192, a193, a194, a195, a196, a197, a198, a199, a200, a201, a202,
a203, a204, a205, a206, a207, a208, a209, a210, a211, a212, a213, a214, a215, a216, a217,
a218, a219, a220, a221, a222, a223, a224, a225, a226, a227, a228, a229, a230, a231, a232,
a233, a234, a235, a236, a237, a238, a239, a240, a241, a242, a243, a244, a245, a246, a247,
a248, a249, a250, a251, a252, a253, a254, a255, a256}
LinearSolve[M, B]
x =
{0,
2840, 0,
–2828192, 0, 1385198072, 0,
–393409070536, 0, 71425933414496, 0,
–8826967508184488, 0, 775361571207342328, 0,
–49961396216383420696, 0, 2419007435403764589728, 0,
–8968798175382073798 3992, 0, 2585716854171716078207816, 0,
–5870634246087284384 1090400, 0, 1060935936193104692810763048, 0,
–1540151779039459043056330 6012, 0, 1810234929911127636447237361 84, 0,
–1734494741861771455510557873696, 0, 1362852038696909227472318393928 8, 0,
–8826282753739182325950270477693 6, 0, 4732123717405079935873404235657 92, 0,
–2108097731057783042707603165048904, 0, 7827629872094876528125317595080392, 0,
–2428804831816288590118111674227788 0, 0, 6310901202734510911060342557482627 2, 0,
–1375497033448455246065012942847035 76, 0, 2518085013514117929587772667186058 00, 0,
–3875724565692350873027957546141344 96, 0, 5018863958183873596415205047816215 76, 0,
–5470204866051090053468911806225971 90, 0, 5018863958183873596415205047816215 76, 0,
```

–387572456569235087302795754614134496, 0, 25180850135141179295877726671860580 0, 0,
–137549703344845524606501294284703576, 0, 631090120273451091106034255748262 72, 0,
–242880483181628859011811167422778 80, 0, 782762987209487652812531759508039 2, 0,
–210809773105778304270760316504890 4, 0, 473212371740507993587340423565792, 0,
–882628275373918232595027047769 36, 0, 136285203869690922747231839392 88, 0,
–173449474186177145551055787369 6, 0, 181023492991112763644723736184, 0,
–154015177903945904305633060 12, 0, 106093593619310469281076304 8, 0,
–587063424608728438410904 00, 0, 25857168541717160782078 16, 0,
–8968798175382073798399 2, 0, 2419007435403764589728, 0,
–49961396216383420696, 0, 775361571207342328, 0,
–8826967508184488, 0, 71425933414496, 0,
–393409070536, 0, 1385198072, 0,
–2828192, 0,
2840, 0,
–1,
0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0}.

Приложение 3. $w = 14$. Коэффициенты $x_1, \dots, x_{128}$.

x =
{1,
3976, 12009,
–5087249, –14309061,
3218923168, 5920466231,
–1199408867773, –1007320922928,
289407682932732, 507923996712,
–48012540718366992, 30816562497605028,
5709147070432020240, –6294773468412320948,
–501991838564883720952, 704522446202167612448,
33458286010067039115852, –52498885480919742717352,
–17254478858103660057317 88, 28008227038448039872864 64,
70040884413731774539712728, –11125272668289578867457 1124,
–22701890892672564570257291 04, 33727111986614839292406804 84,
59447059712960616190595476848, –79338279935086463136560036664,
–12696619122064365233793445419 48, 14638172393556042289496834 56272,
22287586381360349352725191618048, –21302306429711074665146579025004,
–32355387197825549469239935891 1128, 24445092111103434392051765615 8464,
39043552981304292844205943223 73444, –21917072528783915972638617617 30376,
–39330834839605559116002017214304456, 14933171288429002851458106239033596,
33198233937969901606743760288904 4856, –71204153167019966670493916849814960,
–23558216524642060238433597134442285 96, 15980932525073138585151813905978406 16,
14097638038759652757851889162574816656, 83898076241813853803853305574072 1044,
–71347016583404790651062399021509813104,
–11964052704303446228209040158420040996, 30621031263440534295779464362144020 8328,
78795368016603651743500579542690100736,
–11174283760467658563043428000496825903 08,
–36797487170393232986904324734150315648 8, 34759495320218789841178303070601558246 12,
13368895674339023809269553459689466819 52,
–92391583618972713948113371144825115595 92,
–39196622929675410337733096904319837036 52,
21032453403595066556316192629527560861360,
94480539151618317389127270964203400921 80,
–41092487000052829019854917067706813303280,

–189256374505025344009750227411195956780 0,
690340255814330220578201948795667503531 40,
317160046938708123383694600802542714813 60,
–998784455715963862710846369335671377304 90,
–446541269504102351710529498914971952735 70,
124595491568723382346496304315750319050 040,
529536132993340060594077070072366703436 90,
–134114604283996242654522198595628392683 110,
–529536132993340060594077070072366703436 90,
124595491568723382346496304315750319050 040,
446541269504102351710529498914971952735 70,
–998784455715963862710846369335671377304 90,
–317160046938708123383694600802542714813 60,
690340255814330220578201948795667503531 40,
189256374505025344009750227411195956780 0,
–410924870000528290198549170677068133032 80,
–944805391516183173891272709642034009218 0,
210324534035950665563161926295275608613 60,
391966229296754103377330969043198370365 2,
–923915836189727139481133711448251155959 2,
–133688956743390238092695534596894668195 2,
347594953202187898411783030706015582461 2, 367974871703932329869043247341503156488,
–111742837604676585630434280004968259030 8,
–787953680166036517435005795426901007 36, 306210312634405342957794643621440208328,
119640527043034462282090401584200409 96,
–713470165834047906510623990215098131 04,
–838980762418138538038533055740721044, 140976380387596527578518891625748166 56,
–159809325250731385851581390597840616,
–235582165246420602384335971344422859 6, 712041531670199666704939168498149 60,
331982339379699016067437602889044856,
–149331712884290028514581062390335 96,
–393308348396055591160020172143044 56, 219170725287839159726386176173037 6,
390435529813042928442059432237344 4,
–244450921111034343920517656158464,
–323553871978255494692399358911128, 213023064297110746651465790250 04,
222875863813603493527251916180 48,
–146381723935560422894968345627 2,
–126966191220643652337934454194 8, 793382799350864631365600366 64,
594470597129606161905954768 48,
–337271119866148392924068048 4,
–227018908926725645702572910 4, 111252726682895788674571124,
700408844137317745397127 28,
–280082270384480398728646 4,
–172544788581036600573178 8, 52498885480919742717352,
33458286010067039115852,
–704522446202167612448,
–501991838564883720952, 6294773468412320948,
57091470704320202 40,
–30816562497605028,
–48012540718366992,
–507923996712,
289407682932732, 1007320922928,
–1199408867773,
–5920466231,

3218923168, 14309061,
–5087249,
–12009,
3976,
–1,
–1}.

Поступила в редакцию 25 августа 2020 г.

UDC 519.175

DOI: 10.21779/2542-0321-2020-35-4-13–26

C# Computing Tools for Solving the Problem of Enumerating Partitions of a Rectangle

A.M. Magomedov[1], S.A. Lawrencenko[2]

[1] Dagestan State University; Russia, 367000, Makhachkala, M. Gadzhiev st., 43a; magomedtagir1@yandex.ru, yakubovaz@mail.ru;

[2] Russian State University of Tourism and Service; Russia, Moscow region, 141221, Pushkino district, Cherkizovo, Glavnaya st., 99; lawrencenko@hotmail.com

We consider the problem of enumerating all partitions of a rectangle with given integer sizes h×w into rectangles with sizes 1×2, which arose in the 60s of the 20th century in the study of thermodynamics of fluid flows by physicists. The identification of equivalence to the problem of enumerating all perfect matchings in a certain planar graph drew attention to the problem from specialists in discrete mathematics. Despite the fact that dozens of papers have been devoted to the study of the problem, interest in it does not weaken.

The article provides a brief overview of common approaches to solving the problem with the analysis of their disadvantages; in particular, the "classical" double product formula involves operations on floating point numbers (values of trigonometric functions, etc.), which significantly narrows the range of values of h and w, for which it is possible to accurately calculate the desired value of the number of possible partitions.

A method is given for calculating the coefficients of a recurrent formula that uses only integer addition and multiplication operations to find a_h (the number of partitions for a fixed value of the parameter w).

A two-step approach is proposed to the solution. At the first step, using the BigInteger and BigFloat classes of the C# programming language, a lot of laborious work is done to accurately calculate the required number of initial terms of the sequence a_1, a_2, ..., which in turn provide an opportunity to calculate integer coefficients for building a recurrence formula. At the second step, the recurrence formula provides an efficient and accurate calculation of the subsequent terms of a given sequence, using only the operations of addition and multiplication of integers.

At the end of the article, the relevant subtasks of the problem are formulated.

Keywords: *rectangle, C#, matching, enumeration, software*.

Received 25 August 2020