УДК 519.116+519.688

А.С. Врублевский[1], В.А. Шлык[2]

ВЫЧИСЛЕНИЕ ВЕРШИН ПОЛИТОПОВ РАЗБИЕНИЙ ЧИСЕЛ

Описывается метод генерирования вершин политопов разбиений чисел, с помощью которого авторами были вычислены все вершины и опорные вершины политопов разбиений всех n ≤ 105 и все рюкзачные разбиения n ≤ 165. Метод не требует построения всех разбиений n. Вершины определяются с помощью достаточных и необходимых условий, в трудных случаях применяется известная программа Polymake. Подробно излагаются алгоритм проверки критерия, характеризующего разбиения, являющиеся выпуклыми комбинациями двух других; методика применения двух комбинаторных операций, преобразующих известные вершины в новые вершины, и способ применения программы Polymake для распознавания небольшого (для малых n) числа разбиений, являющихся выпуклыми комбинациями трех и более разбиений. Представляются результаты вычислений и формулируются новые проблемы, к которым приводят полученные данные о числах вершин и опорных вершин политопов разбиений чисел.

Введение

Разбиением натурального числа n [1] называется конечная невозрастающая последовательность натуральных чисел $\rho = (\rho_1, \rho_2, \ldots, \rho_r)$, для которой $\sum_{j=1}^{r} \rho_j = n$. Неформально разбиением положительного целого числа n часто называют его представление в виде суммы положительных целых чисел

$$n = n_1 + n_2 + \ldots + n_k,$$

где $n_1, n_2, \ldots, n_k \in \mathbb{Z}, \quad n_1, n_2, \ldots, n_k > 0,$ и порядок слагаемых не учитывается. Слагаемые $n_1, n_2, \ldots, n_k$ называют частями разбиения.

Основы теории разбиений чисел заложил Л. Эйлер [2]. Существенный вклад в ее развитие внесли многие крупнейшие математики [1, 3]. Разбиения чисел играют важную роль в разных областях математики, в статистической механике и современной теоретической физике. Вторым автором предложен полиэдральный подход в теории разбиений чисел [4, 5]. При его применении каждое разбиение n отождествляется с неотрицательной целочисленной точкой $x = (x_1, x_2, \ldots, x_n) \in \mathrm{R}^n$, являющейся решением уравнения

$$x_1 + 2x_2 + \ldots + nx_n = n.$$

Каждая компонента x_i точки x равна числу вхождений части i в разбиение. Так, разбиение $8 = 1+1+2+4$ отождествляется с точкой (2, 1, 0, 1, 0^4), где 0^4 обозначает последовательность из четырех нулей. То, что x является разбиением n, обозначим $x \vdash n$, через $S(x) = \{i \in \{1, \ldots, n\} \mid x_i > 0\}$ обозначим множество различных частей разбиения $x \vdash n$, через $\mathbb{Z}_+$ – множество неотрицательных целых чисел, а через $[n_1, n_2]$ – отрезок натуральных чисел $\{n_1, \ n_1 + 1, \ldots, n_2\}$.

Политоп $P_n \subset \mathbb{R}^n$ разбиений числа n определяется как выпуклая оболочка множества

$$T_n = \{x \in \mathbb{Z}_+^n \mid x_1 + 2x_2 + \ldots + nx_n = n\},$$

состоящего из всех точек x, соответствующих разбиениям n [4, 5]:

$$P_n = \mathrm{conv}\{x \in \mathbb{R}_+^n \mid x \vdash n\}.$$

Точка x произвольного полиэдра P является его вершиной тогда и только тогда, когда ее нельзя представить в виде выпуклой комбинации $x = \sum_{j=1}^{k} \lambda_j y^j,\ \sum_{j=1}^{k} \lambda_j = 1,\ \lambda_j > 0,$ некоторых других точек $y^j \in P,\ 1 \le j \le k$ [6]. Множество вершин политопа P_n будем обозначать $\mathrm{vert}\, P_n.$

На рис. 1 представлены политопы P_1, P_2, P_3 и проекция политопа $P_4,$ лежащего в пространстве $\mathbb{R}^4,$ на пространство первых трех координат $\mathbb{R}^3.$ Вершина (0, 0, 0, 1) политопа $P_4,$ соответствующая разбиению $4 = 4,$ не изображена. Из рисунка видно, что при $n < 4$ каждое разбиение n является вершиной $P_n,$ однако при $n = 4$ разбиение $(2, 1, 0, 0) \vdash 4$ вершиной P_4 не является.

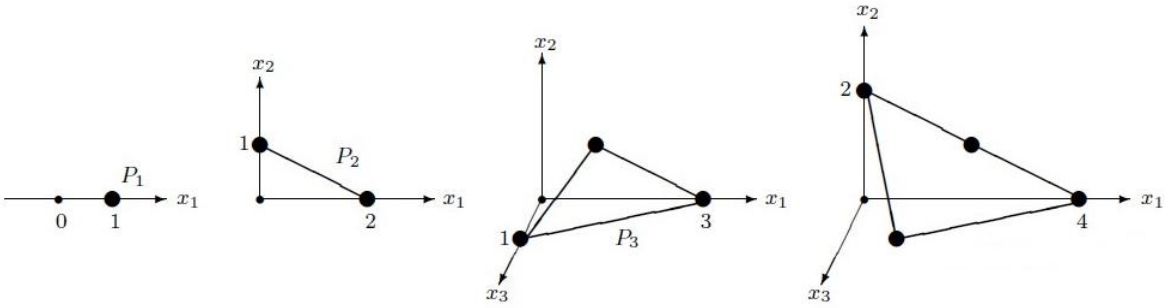

Рис. 1. Политопы P_1, P_2, P_3 и проекция политопа P_4 на пространство $\mathbb{R}^3$

В работе [5] показано, что T_n есть множество всех целочисленных точек политопа P_n и что аффинная размерность P_n равна $n-1.$

Вершины P_n порождают множество всех разбиений $T_n,$ поскольку любое разбиение представляется в виде выпуклой комбинации конечного набора вершин. Другими словами, вершины P_n составляют базис множества T_n относительно операции взятия выпуклой комбинации. Поэтому применение полиэдрального подхода позволяет избежать построения всех разбиений $n,$ ограничившись вершинами $P_n.$ В [7] показано, что вершинный базис множества разбиений может быть уменьшен: нет необходимости знать все вершины $P_n,$ поскольку многие из них можно построить из подмножества вершин, названных опорными, при помощи двух комбинаторных операций (см. разд. 2, условие Д3).

В теории разбиений число разбиений числа n принято обозначать $p(n).$ Обозначим через $v(n)$ число вершин политопа $P_n,$ а через $s(n)$ – число его опорных вершин.

Вычисление значений функции $p(n)$ является предметом постоянного интереса с момента зарождения теории разбиений [1]. Точная формула для $p(n)$ до сих пор неизвестна, однако работы Харди, Рамануджана и Радемахера привели к представлению $p(n)$ в виде суммы быстро сходящегося ряда, причем точное значение $p(n)$ можно получить, взяв малое число членов ряда и округлив сумму до ближайшего целого. Макмагон вычислил значения $p(n)$ для первых двухсот натуральных чисел, используя рекуррентное соотношение, вытекающее из пентагональной теоремы Эйлера и выражающее число разбиений n через числа разбиений меньших натуральных чисел. В частности, он получил, что $p(200) = 3972999029388.$ В [1, 8] приведены таблицы значений $p(n)$ и чисел разбиений из некоторых классов, соответствующих наиболее важным дополнительным ограничениям на части разбиений, для $n \le 100.$ В печатном

виде наиболее полные таблицы значений $p(n)$ для $n \le 400$ представлены в [9]. В онлайн-энциклопедии Слоана «Целочисленные последовательности» можно найти последовательность значений $p(n)$ для $n \le 10\,000$ [10]. Последнее число $p(10\,000)$ содержит 107 знаков:

$$p(10\,000) = 36167251325636293988820471890953695495016030339315650$$
$$422081868605887952568754066420592310556052906916435144.$$

Часто требуется знать не только количество разбиений, но и уметь перечислять их в явном виде. Необходимость иметь полные списки разбиений, перестановок, разбиений множеств, деревьев и других основных комбинаторных структур возникает, например, при тестировании программ и алгоритмов, анализе их вычислительной сложности, а также при проверке всех возможных решений задач. Знание полных списков разбиений важно и для теоретических исследований. Они помогают обнаружить новые свойства, высказать и проверить возникающие предположения или построить опровергающие контрпримеры. Так, анализ списков разбиений показал, что частота появления хотя бы одной двойки в качестве части разбиения n, т. е. $x_2 \ge 1$, возрастает с ростом n. В разбиениях $n = 30$ двойка встречается в 66 % разбиений, а для $n = 90$ – в 78 % разбиений. Доказана сходимость $\dfrac{p(n-2)}{p(n)} \to 1$ при возрастании n [11].

Основные алгоритмы генерирования разбиений чисел, в том числе с ограничениями, которые чаще всего накладывают на величину и (или) число частей, можно найти в [1, 12–15], см. также [10]. Обзор методов генерирования разбиений представлен Д. Кнутом в материалах [16] для четвертого тома его труда «Искусство программирования». Разработка новых алгоритмов генерирования разбиений продолжается и сейчас [17–19].

В настоящей статье описывается разработанный авторами метод вычисления вершин и опорных вершин политопа P_n. С его помощью построены все вершины и опорные вершины политопов P_n для всех $n \le 100$. Вычисленные последовательности чисел $v(n)$ и $s(n)$ включены в энциклопедию «Целочисленные последовательности» [20, 21]. Попутно была продолжена до $n = 165$ вычисленная Эренборгом для $n \le 50$ последовательность чисел рюкзачных разбиений [22].

В разд. 1 статьи изложен алгоритм построения списка $\mathrm{PartList}(n)$ разбиений n, содержащего все вершины политопа P_n и некоторые подозрительные разбиения, о которых в данный момент неизвестно, являются ли они вершинами. Разработанный метод решения проблемы распознавания вершин среди подозрительных разбиений и отсеивания избыточных разбиений, а также основные моменты его программной реализации описаны в разд. 2. Здесь приведены достаточные и (не одновременно) необходимые условия для вершин и охарактеризованы их практическая эффективность и трудоемкость, что обосновывает выбранную стратегию проверок указанных условий. Приведено подробное описание алгоритма проверки наиболее общего и одновременно наиболее трудоемкого условия, характеризующего разбиения, являющиеся выпуклыми комбинациями двух других. Изложена методика применения μ-операций, с помощью которых из известных вершин политопа можно строить новые вершины, а также применения программы Polymake, к которой авторы прибегают для распознавания небольшого (для сравнительно малых n) числа оставшихся разбиений, которые не являются вершинами потому, что они представляют собой выпуклые комбинации более чем двух разбиений. В разд. 3, 4 и 5 изложены соответственно методы построения списков опорных вершин политопа P_n, рюкзачных разбиений и некоторые технические особенности организации вычислений. В разд. 6 изложены результаты проведенных вычислений и сформулированы новые проблемы о структуре множеств разбиений, к которым приводят полученные данные о поведении функций чисел разбиений, вершин и опорных вершин политопов разбиений чисел.

1. Генерирование списка разбиений

Генерирование списка $\mathrm{PartList}(n)$ разбиений каждого числа n производится последовательно по мере роста n, начиная с $n=1,2,3$. При $n=1$ в список безусловно включается единственное разбиение $x=(1)$. При обращении к очередному политопу P_n используются уже построенные списки $\mathrm{VertList}(n-i)$ вершин политопов P_{n-i}, $1\le i\le\lfloor n/2\rfloor$, а последним безусловно включается разбиение $x=(0^{n-1},1)$. Сгенерированный список содержит все вершины P_n, но в нем присутствуют также некоторые избыточные разбиения, не являющиеся вершинами. Разбиения, о которых в текущий момент неизвестно, вершины они или нет, назовем *подозрительными*.

В списках $\mathrm{PartList}(n)$ разбиения хранятся в лексикографическом порядке. Например, четыре разбиения $n=4$ хранятся в $\mathrm{PartList}(4)$ в порядке $x^1=(4,\ 0,\ 0,\ 0)$, $x^2=(2,\ 1,\ 0,\ 0)$, $x^3=(1,\ 0,\ 1,\ 0)$, $x^4=(0,\ 2,\ 0,\ 0)$, где разбиение x^2 избыточное, так как $x^2=\frac{1}{2}(x^1+x^4)$. Такой порядок облегчает выполнение используемых далее операций выбора разбиений из $\mathrm{PartList}(n)$ и построение в дальнейшем списка $\mathrm{SuppList}(n)$ опорных вершин P_n.

Первоначальное построение списка $\mathrm{PartList}(n)$ основано на рекуррентном соотношении

$$\mathrm{vert}\,P_n\subseteq\Big(\biguplus_{i=1}^{\lfloor n/2\rfloor}\Phi_i(\mathrm{vert}\,P_{n-i}[\ge i])\Big)\cup(0^{n-1},\ 1),\qquad(1)$$

где $\uplus$ обозначает объединение непересекающихся множеств; $P_m[\ge i]$ – политоп тех разбиений m, все части которых не меньше i,

$$P_m[\ge i]=\mathrm{conv}\{x\vdash m\,|\,x_j=0,\ j<i\},$$

и Φ_i, $1\le i\le\lfloor n/2\rfloor$, – оператор

$$\Phi_i:\mathbb{R}^{n-i}\to\mathbb{R}^n:\Phi_i(x_1,\ x_2,\ldots,x_{n-i})=(x_1,\ x_2,\ldots,x_{i-1},x_i+1,x_{i+1},\ldots,x_{n-i},0^i).$$

Каждый оператор Φ_i есть композиция сдвига на i вдоль оси x_i и вложения пространства $\mathbb{R}^{n-i}$ в $\mathbb{R}^n$, при котором к каждой точке $\mathbb{R}^{n-i}$ добавляются i нулевых координат $x_{n-i+1}=x_{n-i}=\ldots=x_n=0$. Понятно, что если $y\vdash n-i$, то $x=\Phi_i(y)\vdash n$. Наоборот, если $x\vdash n$, $x\ne(0^{n-1},1)$ и $x_i>0$ для некоторого $1\le i\le\lfloor n/2\rfloor$, то $\Phi_i^{-1}(x)$ есть разбиение числа $n-i$. В работе [5] доказано, что если вершина $x\in\mathrm{vert}\,P_n$ содержит часть $i<n$, то $\Phi_i^{-1}(x)\in\mathrm{vert}\,P_{n-i}$. Поэтому, последовательно включая в $\mathrm{PartList}(n)$ вершины из списков $\mathrm{VertList}(n-i)$, $i=1,\ 2,\ldots,\lfloor n/2\rfloor$, состоящие из частей, не меньших i (см. (1)), получим все вершины P_n в указанном выше порядке. Построенный таким образом список $\mathrm{PartList}(n)$ может содержать некоторые избыточные разбиения, которые вершинами P_n не являются. Для построения списка вершин $\mathrm{VertList}(n)$ их необходимо обнаружить и исключить. Применяемый метод позволяет избежать перебора и проверки всех разбиений n.

2. Распознавание вершин и избыточных разбиений

Отсеивание из PartList(n) избыточных разбиений и построение в результате списка вершин VertList(n) политопа P_n составляют основную проблему вычислений. Для распознавания

вершин P_n и избыточных разбиений в текущем списке PartList(n) применяются доказанные в [5, 23–25] достаточные и необходимые условия для вершин. В сложных случаях программа обращается к созданной в Германии универсальной программе Polymake [26].

Распознавание вершин и избыточных разбиений выполняется в три этапа:

1. Распознавание некоторых вершин и разбиений, являющихся выпуклыми комбинациями двух других.
2. Применение μ-операций (условие Д3, см. далее в описании этапа 1).
3. Применение программы Polymake.

Проверка условия Д3 вынесена в отдельный этап, так как для ее выполнения нужен еще один проход по всем разбиениям из списка, полученного на первом этапе. Выделив из списка вершины и подозрительные разбиения, его можно существенно сократить и повысить скорость дисковых операций.

Этап 1. Распознавание некоторых вершин и разбиений, являющихся выпуклыми комбинациями двух других.

Имеются три достаточных условия для того, чтобы разбиение было вершиной P_n [7, 24, 25, 27]:

Д1. Пусть для множества $\{i_1, i_2,\ldots,i_k\}\subset[1,n]$ уравнение $i_1x_1+i_2x_2+\ldots+i_kx_k=n$ имеет точно одно или два решения в целых положительных числах. Тогда для каждого решения $a_1,a_2,\ldots,a_k$ точка $x\in\mathbb{R}^n$ с ненулевыми компонентами $x_i=a_i$ для $i\in\{i_1, i_2,\ldots,i_k\}$ является вершиной P_n.

Д2. Пусть $\{i_1<i_2<\ldots<i_k\}$ – множество различных частей разбиения $x\vdash n$. Если $x_{i_k}=\lfloor n_k/i_k\rfloor$ для $n_k=n$; $x_{i_{k-1}}=\lfloor n_{k-1}/i_{k-1}\rfloor$ для $n_{k-1}=n_k-x_{i_k}i_k$; …; $x_1=x_{i_1}=\lfloor n_1/i_1\rfloor=n_1$ для $n_1=n_2-x_{i_2}i_2$ и $x_i=0$ для $i\neq i_1, i_2,\ldots,i_k$, то $x=(x_1, x_2,\ldots,x_n)$ – вершина P_n.

Д3. Разбиения, полученные из вершин политопа P_n с помощью определенных ниже μ-операций слияния частей, являются вершинами P_n.

Операция 1. Пусть $x\vdash n$ и пусть $u,v\in S(x)$, $u\neq v$, – две различные части x. Для определенности можно считать, что $x_u\le x_v$. Построить точку $y=\mu_{u,v}(x)\in\mathbb{Z}_+^n$ с компонентами $y_u=0$, $y_v=x_v-x_u$, $y_{u+v}=x_{u+v}+x_u$, $y_j=x_j$ для $j\in[1,n]$, $j\neq u,v,u+v$.

Операция 2. Пусть x – разбиение числа n, в которое некоторая часть $u\in S(x)$ входит более одного раза, т. е. $x_u>1$. Построить точку $y=\mu_u(x)\in\mathbb{Z}_+^n$ с компонентами $y_u=0$, $y_{x_uu}=x_{x_uu}+1$, $y_j=x_j$ для $j\in[1,n]$, $j\neq u,x_uu$.

Для определения в PartList(n) избыточных разбиений имеются следующие необходимые условия [5, 23–25, 27] принадлежности x множеству $\mathrm{vert}P_n$:

Н1. Ни одно целое $k<n$ вида $k=\sum_{i\in S(x)}q_ii$, $q_i\in\mathbb{Z}_+$, $q_i\le x_i$, кроме тривиального случая $k=1\cdot i$, не является частью x.

Н2. $ix_i<k$ для всех i и k, таких, что $1\le i<k\le n$, i делит k и $x_k>0$.

Н3. $ix_i<m-k$ для всех троек индексов i,k,m, таких, что $k<m$, i делит $m-k$ и $x_k,x_m>0$.

Н4. $\prod_{i\in S(x)}(x_i+1)\le n+1$.

Н5. Число различных частей в x не превосходит $\lfloor\log(n+1)\rfloor$, т. е. $|S(x)|\le\lfloor\log(n+1)\rfloor$.

Н6. Если $x \neq (n,\ 0^{n-1})$ и $n > 1$, то x содержит не более $\lceil n/2 \rceil$ частей, т. е. $\sum_{i=1}^{n} x_i \leq \lceil n/2 \rceil$.

Н7. Любые два целочисленных набора $u = \langle u_j;\ j \in S_1 \rangle$, $v = \langle v_k; k \in S_2 \rangle$, $0 < u_j \leq x_j$, $0 < v_k \leq x_k$, соответствующих подмножествам $S_1, S_2 \subset S(x)$, $S_1 \cap S_2 = \varnothing$, удовлетворяют соотношению

$$\sum_{j \in S_1} u_j j \neq \sum_{k \in S_2} v_k k.$$

Как достаточные, так и необходимые условия неравноценны по своей эффективности. Достаточные условия определяют различное число вершин, а одни необходимые условия отсеивают больше подозрительных разбиений, чем другие. Поэтому общее время, затрачиваемое на их проверку, существенно зависит от порядка проверки условий. Экспериментально подобран такой порядок, при котором первыми проверяются либо самые быстрые, либо самые эффективные условия.

От использования трудоемкого условия Д1 в данной версии программы авторы отказались, хотя при проверке разбиений «вручную» в несколько измененных, но трудноформализуемых вариантах оно часто оказывается полезным. Проверка условия Д2 проста и, как показали вычисления, это условие определяет бóльшую часть вершин, содержащихся среди подозрительных разбиений. Условия Н1–Н4 позволяют быстро отсеивать многие разбиения, не являющиеся вершинами политопов P_n. Например, для $x = (1,\ 0,\ 2,\ 0,\ 3,\ 0^{17}) \vdash 22$ имеем $2 \cdot 3 \cdot 4 = 24 > 23$, откуда ввиду Н4 заключаем, что $x \notin \operatorname{vert} P_{22}$. Наиболее результативными оказались необходимые условия Н2 и Н3, причем применение Н2 даже в примитивном варианте

$$x \in \operatorname{vert} P_n,\ i \in S(x),\ q \leq x_i \ \Rightarrow\ qx_i \notin S(x)$$

отсеивает массу подозрительных разбиений. Условия Н4 и Н6, напротив, оказались малопродуктивными.

Условие Н7 выражает доказанный в [23] критерий того, что разбиение является выпуклой комбинацией двух других разбиений того же числа. Условия Н1–Н5 представляют собой его частные случаи. Для ускорения вычислений их было решено проверять первыми, а намного более трудоемкое условие Н7 – последним.

Для малых n сложная задача распознавания разбиений, оставшихся подозрительными после проверки условий Д2 и Н1–Н5, решается достаточно быстро, но уже при $n > 50$ она требует значительных временных затрат. Тем не менее проверять условие Н7 все еще выгоднее, чем обращаться к программе Polymake.

Распознанные вершины помечаются в списке $\mathrm{PartList}(n)$ специальным ключом. Отсеиваемые подозрительные разбиения также не удаляются всякий раз из списка, поскольку удаление записи связано с обращением к внешней памяти и не может быть выполнено быстро. Как и найденные вершины, эти разбиения помечаются специальным признаком, который исключает их из дальнейшей обработки и служит для сбора статистической информации в процессе последующего анализа.

Обработка каждого из включаемых в список $\mathrm{PartList}(n)$ разбиений заканчивается проверкой условия Н7. Алгоритм его проверки описан ниже. После завершения первого этапа в списке $\mathrm{PartList}(n)$ найдены и помечены бóльшая часть вершин и все разбиения , являющиеся выпуклыми комбинациями двух других.

Алгоритм проверки условия Н7.

Вход: подозрительное разбиение $x \vdash n$ из $\mathrm{PartList}(n)$.

Выход: один из двух ответов: «разбиение x не является вершиной P_n» или «разбиение x не является выпуклой комбинацией никаких двух разбиений n».

Для каждой из различных частей разбиения x вводим битовую маску, использование которой облегчит и ускорит обработку x. Например, для разбиения $x = (0,\ 1,\ 2,\ 3,\ 0^{16}) \vdash 20$, состоящего из частей $2,\ 3,\ 3,\ 4,\ 4,\ 4$, используем маски: 001 для части 2, 010 для части 3 и 100 для части 4.

С помощью итерационного процесса строим все возможные комбинации частей разбиения x, соблюдая следующие ограничения. В каждой комбинации любая часть i из x должна повторяться не более x_i раз и комбинации не должны содержать одновременно все различные части x. В комбинациях части упорядочены по возрастанию. Сами комбинации строим по группам: из одной, двух, трех и т. д. частей. Внутри групп комбинации упорядочены лексикографически по возрастанию.

Для каждой комбинации вычисляем:

– сумму, равную сумме всех входящих в нее частей, учитывая повторения;

– битовую маску, являющуюся объединением масок всех входящих в нее различных частей. Ее легко вычислить, применив к маскам частей операцию арифметического OR. Так, маской комбинации <3, 3, 4> является $(010)\mathrm{OR}(100) = (110)$, а ее сумма равна $3+3+4=10$.

Для хранения данных используем четыре массива. В массиве Masks хранятся маски всех различных частей x. В массиве Mas1 хранятся комбинации, состоящие из $k-1$ частей разбиения x (k – номер итерации), а в массиве Mas2 – комбинации, состоящие из k частей. Части могут повторяться, k возрастает от 1. В массиве Sums хранятся пары (сумма, маска), рассчитанные для построенных комбинаций. Sums представляет собой массив, который на языке C++ называют множественным ассоциативным контейнером (std::multimap Sums). «Множественный» означает, что одному значению ключа может соответствовать несколько значений данных. В рассматриваемом случае ключом служит сумма комбинаций, а маски комбинаций выступают в роли данных. Одному значению ключа может соответствовать несколько масок, так как одну и ту же сумму можно получить из различных наборов слагаемых. Интернет-ресурсы (cplusplus.com, cppreference.com) указывают, что в сортированных ассоциативных контейнерах multimap операции поиска, вставки и удаления элементов имеют логарифмическую сложность.

Сам алгоритм состоит из следующих шагов:

Шаг 1. Инициализация: число частей в комбинациях $k = 1$. В Mas1 заносим комбинации, состоящие из одной части разбиения x, т. е. каждую из различных частей разбиения x по одному разу включаем в Mas1. Сумму каждой комбинации (на этапе инициализации сумма равна самой части) и ее битовую маску заносим в массив Sums.

Шаг 2. Итерация $k+1$: переход от построенных на k-й итерации комбинаций, состоящих из k частей, к комбинациям из $k+1$ частей.

К каждой комбинации из массива Mas1 присоединяем по очереди различные не вошедшие в нее части разбиения x. Добавление начинаем с наибольшей из входящих в комбинацию части, так как комбинации с меньшими частями к этому моменту уже построены. Следим за тем, чтобы в новую комбинацию не вошли все различные части x. Новые комбинации из $k+1$ частей добавляем в массив Mas2.

После добавления в Mas2 очередной комбинации c вычисляем ее сумму и маску и выполняем следующую проверку. Если в Sums имеется одна или несколько пар (сумма, маска), для которых сумма равна сумме комбинаций c, а арифметическое AND маски из пары с маской комбинации c равно 0 (это означает, что наборы частей в комбинации c и комбинации, соответствующей рассматриваемой паре из Sums, не пересекаются), то можно сделать следующий вывод: разбиение x не является вершиной P_n. *Конец алгоритма*. В противном случае добавляем сумму и битовую маску комбинации c в Sums и переходим к построению следующей комбинации.

Когда все комбинации из массива Mas1 обработаны и исходя из них в Mas2 построены новые комбинации из $k+1$ частей, очищаем Mas1 и перемещаем в него все комбинации из

Mas2. Массив Mas2 пуст и готов к занесению в него на следующей итерации комбинаций из $k+2$ частей. Если массив Mas1 не пуст, то переходим к итерации $k+2$, в противном случае (когда добавление новой части к каждой комбинации приводит к использованию всех различных частей x) можно сделать следующий вывод: разбиение x не является выпуклой комбинацией никаких двух разбиений n. *Конец алгоритма.*

Проиллюстрируем работу алгоритма, выполнив проверку одной из комбинаций частей разбиения $x=(2,\ 5,\ 3,\ 0,\ 1,\ 0^{21})\vdash 26$. Пусть $k=5$ и на данной итерации построена комбинация из пяти частей <2, 2, 2, 2, 2>. Ее сумма равна $2+2+2+2+2=10$, а маска – 0010. Маска данной комбинации четырехзначная, поскольку в разбиении x только четыре различные части и частям 1, 2, 3, 5 соответствуют маски 0001, 0010, 0100, 1000 соответственно. К моменту построения комбинации x в Sums содержатся пять пар (сумма, маска), соответствующих пяти комбинациям с суммой 10: пара (10, 1110) для комбинации <2, 3, 5>; (10, 1101) для <1, 1, 3, 5>; (10, 1011) для <1, 2, 2, 5>; (10, 0110) для <2, 2, 3, 3> и (10, 0111) для <1, 2, 2, 2, 3>.

Вычисляя арифметическое AND маски 0010 с маской из первой пары (10, 1110), получаем 0010 AND 1110 = 0010, что не равно 0, следовательно, переходим к обработке следующей пары. Арифметическое AND с ее маской дает 0010 AND 1101 = 0000. Таким образом, соответствующие этим парам комбинации <2, 2, 2, 2, 2> и <1, 1, 3, 5> имеют одинаковые суммы, но части данных комбинаций различны и, значит, разбиение $x=(2,\ 5,\ 3,\ 0,\ 1,\ 0^{21})$ не является вершиной P_{26}. *Конец алгоритма.*

Этап 2. Применение μ-операций.

На втором этапе с помощью условия Д3, использующего μ-операции, удается обнаружить некоторые из тех подозрительных разбиений из списка $\mathrm{PartList}(n)$, которые являются вершинами P_n. Применение μ-операций к разбиению реализовано с помощью специальных функций.

Вершины и подозрительные разбиения извлекаются из исходного файла в отдельный файл. Для него запускается функция, реализующая μ-операции из условия Д3. В результате некоторые подозрительные разбиения получают статус «вершина».

Функция, реализующая μ-операции, строит в памяти список вершин, полученных из текущего разбиения-вершины, добавляя их туда в лексикографическом порядке через вспомогательную функцию-сортировщик. Этот список передается другой вспомогательной функции, которая проходит по файлу, находя разбиения из построенного списка и меняя их статус, если они были подозрительными. Затем основная функция переходит к следующей вершине.

Этап 3. Применение программы Polymake.

К началу третьего этапа в $\mathrm{PartList}(n)$ найдены и помечены бóльшая часть вершин и все разбиения, являющиеся выпуклыми комбинациями двух других. Однако среди подозрительных разбиений могут присутствовать такие «трудные» разбиения, которые требуют для своего выпуклого представления трех или более других разбиений. Такие разбиения появляются при $n=15,\ 21,\ 24,\ 27$ и присутствуют далее почти для всех n. Для $n=15$ существует всего одно «трудное» разбиение $15=2\cdot 3+4+5$ с частями 3, 4, 5. Это разбиение – не вершина P_{15}, поскольку является выпуклой комбинацией трех разбиений:

$$(0,\ 0,\ 2,\ 1,\ 1,\ 0^{10})=\frac{1}{3}(0,\ 0,\ 0,\ 0,\ 3,\ 0^{10})+\frac{1}{3}(0,\ 0,\ 1,\ 3,\ 0,\ 0^{10})+\frac{1}{3}(0,\ 0,\ 5,\ 0,\ 0,\ 0^{10}).$$

Число $n=21$ обладает уже тремя разбиениями данного типа: $(1,\ 0,\ 0,\ 1,\ 0,\ 0,\ 1,\ 0,\ 1,\ 0^{12})$, $(0,\ 0,\ 1,\ 0,\ 1,\ 1,\ 1,\ 0^{14})$ и $(0,\ 0,\ 3,\ 0,\ 1,\ 0,\ 1,\ 0^{14})$. С ростом n такие «трудные» разбиения перестают быть исключениями. Для чисел вида $n=21+6k$ и $n=21+3k$ существуют серии разбиений, являющихся выпуклыми комбинациями трех разбиений n [27].

Для распознавания «трудных» разбиений применяем программу Polymake. Polymake не позволяет определить минимальное число разбиений, необходимых для выпуклого представле-

ния. Однако анализ результатов позволил найти разбиения чисел $n = 43$ и $n = 44$, которые требуют четырех разбиений n для своего выпуклого представления.

Задача «*Является ли заданная точка политопа* $P \subset \mathbb{R}^n$ *выпуклой комбинацией* k *точек из* P, *но не является выпуклой комбинацией* m, $m < k$, *точек, где* $k > 2$ *– фиксированное целое*?» ранее не рассматривалась. Характеризация разбиений, представимых через два разбиения, – пока единственный результат в данном направлении. Поэтому для решения этой задачи используется программа Polymake.

Polymake – мощная программа, которая «умеет» вычислять разнообразные характеристики многогранников. Однако ограничиться ее применением для вычисления вершин P_n невозможно по нескольким причинам. Во-первых, для того чтобы с помощью Polymake строить вершины некоторого политопа, его необходимо задать одним из двух способов: указав явно или все фасеты политопа, или набор точек (можно с избытком), выпуклой оболочкой которых он является. Для политопа P_n явно известны только тривиальные фасеты, а нетривиальные охарактеризованы как вершины другого политопа и явно неизвестны [5]. Поэтому нельзя задать политоп P_n первым способом. Второй способ в рассматриваемом случае также непригоден. Число разбиений n имеет порядок

$$p(n) \sim \frac{1}{4n\sqrt{3}} e^{\pi\sqrt{\frac{2}{3}}\sqrt{n}} \qquad (2)$$

и быстро растет [28]. Поэтому перечисление всех разбиений вскоре вынудило бы нас прекратить вычисления, в то время как целью работы является получение информации о вершинах политопа P_n для как можно бóльших значений n. Более того, желательно максимально сократить даже построенный список разбиений $\mathrm{PartList}(n)$, выпуклой оболочкой которых является P_n. Кроме того, в силу своей универсальности программа Polymake выполняется довольно медленно. Она хороша для разовых вычислений, но малопригодна для обработки структур, характеризующих серию многогранников (или политопов), особенно если объем их входных данных быстро возрастает. При вычислении вершин полиэдров Polymake многократно использует алгоритмы линейного программирования, а нам необходимо распознать вершины наиболее эффективным методом, используя их комбинаторные свойства.

Отсеивание с помощью Polymake разбиений, требующих $k > 2$ разбиений, удалось ускорить, воспользовавшись связью между вершинами P_n и вершинами политопов ограниченных разбиений $P_n(M)$, таких, что все их части принадлежат заданному $M \subset \{1, 2, \ldots, n\}$, т. е. $P_n = \mathrm{conv}\{x \in \mathbb{R}_+^n \mid x \vdash n, S(x) \subseteq M\}$.

Эту связь выражает следующее

Утверждение [5]. *Политоп* $P_n(M)$ *является гранью политопа* P_n *и совпадает с пересечением* $P_n \cap E(M)$, *где* $E(M) = \{x \in \mathbb{R}^n : x_i = 0,\ i \notin M\}$.

Следовательно, для того чтобы определить, является ли $x \vdash n$ вершиной P_n, необходимо и достаточно проверить, представляется ли x в виде выпуклой комбинации разбиений $y \vdash n$, $y \in E(M)$. Этот прием позволил сократить время расчета на порядок. Для каждого из оставшихся подозрительных разбиений x выгоднее потратить время на предобработку, выбрав и подав на вход программе Polymake только те разбиения из $\mathrm{PartList}(n)$, которые принадлежат $E(M)$, где $M = S(x)$, чем подавать ей весь список. При этом данное подмножество разбиений, кроме проверяемого, может содержать и другие подозрительные разбиения, статус которых – вершины они или нет – будет определен одновременно со статусом проверяемого разбиения. Для n порядка 100 количество одновременно обрабатываемых таким образом подозрительных разбиений порой превышает 10.

Для реализации описанного приема написаны четыре функции на C++, которые подключаются к программе Polymake и вызываются из встроенного в нее интерпретатора языка Perl. Функции разработаны на языке C++, так как Perl не позволяет считывать байты из файла, как из бинарного потока. При этом упрощается и портирование (Polymake создана на платформе Linux) других, уже написанных на C++ функций, необходимость в которых может возникнуть при использовании вычислений в Polymake. Планируется, например, проверить эффективность применения μ-операций к очередным найденным Polymake вершинам.

Первая функция извлекает из исходного файла в отдельный временный файл очередное подозрительное разбиение и все разбиения, которые принадлежат $E(M)$. В имени временного файла отражен номер позиции этого подозрительного разбиения в исходном файле. Вторая функция из полученного временного файла считывает разбиения в объект Polytope программы Polymake, после чего вызывается встроенная функция этого объекта, определяющая вершины соответствующего политопа. Третья функция обновляет статусы разбиений во временном файле в соответствии с полученным Polymake набором вершин. Четвертая функция объединяет информацию из временного файла с исходным общим файлом.

Обработка подозрительных разбиений в исходном файле продолжается далее до тех пор, пока они не будут исчерпаны. На этом построение списка вершин политопа P_n завершается.

3. Построение списка опорных вершин

Опорные вершины политопа P_n – это такие вершины, которые нельзя получить из других вершин с помощью введенных в [7] и определенных в условии ДЗ μ-операций слияния частей. Их существование следует из того, что применение любой из μ-операций к вершине x приводит к вершине y с меньшим числом частей: $\sum_{i=1}^{n} y_i < \sum_{i=1}^{n} x_i$.

Все вершины P_n можно построить из опорных вершин, рекурсивно применяя к ним операции слияния частей. Все разбиения n можно затем получить из вершин P_n, вычислив все целочисленные точки $\mathbb{R}^n$, являющиеся выпуклыми комбинациями вершин. В последние два десятилетия достигнуты значительные успехи в решении проблемы подсчета и перечисления целочисленных точек полиэдров, представляющей интерес для многих областей математики. Барвинок [29] предложил полиномиальный при фиксированной размерности пространства алгоритм для подсчета таких точек. Метод Барвинка реализован Де Лоерой с соавторами (см. [30] и дальнейшие ссылки в этой работе) в программе LattE, применимой для произвольных рациональных политопов.

Таким образом, зная вершинные базисы множеств разбиений чисел, состоящие из вершин или опорных вершин политопов разбиений, можно достаточно эффективно построить все разбиения относительно малых n

Опорные вершины политопа P_n нетрудно выделить из списка $\mathrm{VertList}(n)$. К каждой вершине $x \in \mathrm{vert}P_n$ последовательно, начиная с начала списка, следует применить следующую процедуру:

1) если вершина x не помечена как опорная, пометить ее;

2) применить к x все возможные операции слияния частей и пометить полученные вершины y как не опорные;

3) перейти к обработке следующей вершины x из $\mathrm{VertList}(n)$.

Порядок расположения вершин в списке $\mathrm{VertList}(n)$ гарантирует, что вершины y расположены в нем после x. Поэтому достаточно выполнить один проход по списку $\mathrm{VertList}(n)$. В результате в $\mathrm{VertList}(n)$ будут помечены все опорные вершины P_n и из них легко составить отдельный список.

4. Построение списка рюкзачных разбиений

Рюкзачные разбиения были введены Эренборгом и Ридди в [31] при исследовании функции Мебиуса частично упорядоченного множества разбиений множества на блоки с ограничениями на размеры. Они назвали так разбиения, все наборы частей которых дают различные суммы. В [27] доказано, что класс рюкзачных разбиений всех чисел совпадает с классом разбиений, не представимых в виде выпуклой комбинации двух разбиений. Все такие разбиения помечены должным образом в списке $\mathrm{PartList}(n)$ в момент окончания проверки условия H7. Число рюкзачных разбиений $k(n)$ связано с числом вершин $v(n)$ политопа P_n равенством

$$k(n) = v(n) + [\text{число разбиений, для выпуклого представления которых необходимы три или более разбиения}],$$

т. е. $k(n) = v(n) + [$число тех разбиений n из $\mathrm{PartList}(n)$, для которых программа Polymake показывает, что они не являются вершинами $P_n]$.

Эренборг и Ридди представили в энциклопедии «Целочисленные последовательности» первые 50 значений функции $k(n)$. Авторам удалось продолжить эту последовательность до $n = 165$ [22].

5. Технические особенности вычислений

Вычисления в программе Polymake выполнялись на виртуальном сервере с двумя процессорами с частотой 1 ГГц каждый под управлением ОС Linux на платформе x86 (i586).

При выполнении программы списки разбиений $\mathrm{PartList}(n)$ хранятся в байтовом представлении. При n порядка 190 файлы достигли размера более 2 Гб и перестали работать стандартные для языка C++ функции для их обработки, реализованные в системных библиотеках ядра ОС Windows. Поэтому пришлось прекратить генерирование разбиений для последующих n. В будущем это препятствие можно обойти, используя 64-разрядные функции. Следует заметить, однако, что предел 2 Гб для размера файлов был достигнут на начальном этапе вычислений, когда генерируемые файлы содержали все разбиения числа n. В дальнейшем авторы оптимизировали алгоритм и не генерировали разбиения n из не вершин, содержащихся в файлах с разбиениями чисел, меньших n. Расчеты показывают, что при такой организации вычислений вполне хватит 2 Гб для $n \le 255$. Однако при больших n критическим окажется размер в 1 байт, отведенный для каждого слагаемого. Для обработки $n > 255$ придется модернизировать программу, отведя на каждое слагаемое по 2 байта.

Основной причиной остановки вычислений при $n = 105$, когда файлы с данными имели еще достаточно небольшой размер, послужило возрастание времени, затрачиваемого программой Polymake на распознавание разбиений, оставшихся после второго этапа процедуры распознавания вершин и избыточных разбиений и не являющихся вершинами P_n. В данный момент некоторый ресурс времени еще остается, но для дальнейшего существенного продвижения по n придется найти новые алгоритмические решения данной проблемы. Резюмируя, можно сказать, что главную проблему составляет распознавание разбиений, представимых в виде выпуклой комбинации более чем двух разбиений.

6. Результаты вычислений

При исследовании политопов разбиений чисел первостепенный интерес представляет поведение функций числа вершин и числа опорных вершин. Пока неясно, как вывести для них точные формулы или хотя бы достаточно точные аппроксимации. Поэтому авторы стремились получить их значения для как можно бóльших n. Диаграмма на рис. 2 демонстрирует рост

функций числа разбиений $p(n)$, числа вершин $v(n)$ и числа опорных вершин $s(n)$ политопов P_n в логарифмической шкале для $n \le 100$.

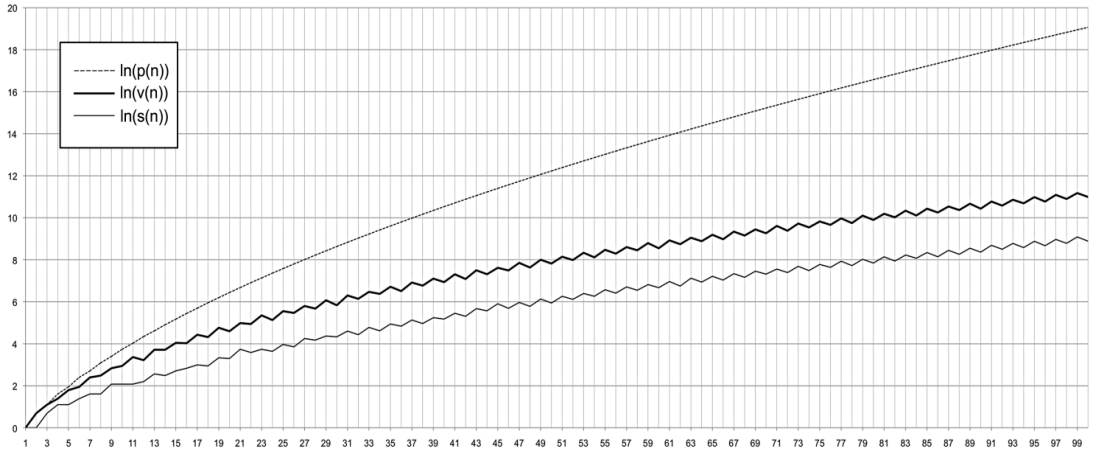

Рис. 2. Диаграмма роста функций $p(n)$, $v(n)$ и $s(n)$

Анализ результатов вычислений приводит к интересным наблюдениям и новым вопросам о структуре множеств разбиений n:

1. Отношения чисел вершин и опорных вершин политопа P_n к числу разбиений n быстро убывают с ростом n. Для $n = 10, 50, 100$ эти числа, соответственно, таковы:

$$p(n) = 42,\ 204226,\ 190569292; \quad v(n) = 17,\ 2488,\ 59294; \quad s(n) = 8,\ 380,\ 7228,$$

а отношения $v(n)/p(n)$ и $s(n)/p(n)$, соответственно, равны $0,45$ и $0,19$; $0,12$ и $0,019$; $0,0031$ и $0,00038$. Легко заметить, что отношение числа опорных вершин к числу всех вершин тоже быстро уменьшается с ростом n. Таким образом, множество всех 190 569 292 разбиений числа $n = 100$ можно построить из множества, содержащего 59 294 вершин политопа P_{100}, и даже всего лишь из 7228 его опорных вершин, что меньше исходного числа разбиений в 330 и более чем в 26 000 раз соответственно. Отметим также, что при $n = 100$ число опорных вершин P_n всего лишь примерно в 72 раза больше самого числа n. При возрастании n от 50 до 100 отношение числа разбиений в PartList, оказавшихся не вершинами P_n, к общему числу разбиений в списке изменяется от 0,52 – 0,64 до 0,63 – 0,73 в зависимости от четности n.

Анализ вычисленных значений показал, что при $n \le 100$ рост числа вершин вполне удовлетворительно описывается экспоненциальной зависимостью от корня из n, что асимптотически совпадает с формулой (2) для $p(n)$. Более точное определение этой зависимости представляет проблему для дальнейших исследований.

2. Из рис. 2 отчетливо видно, что функции $v(n)$ и $s(n)$ растут не монотонно: начиная с n порядка 10, значение $v(n)$ (так же, как и $s(n)$) для нечетного n существенно больше, чем $v(n+1)$. Можно сказать, что графики функций $v(n)$ и $s(n)$ распадаются на два лежащих один под другим подграфика: для четных и нечетных n, причем расстояние между ними экспоненциально возрастает с ростом n. Объяснение такого «пилообразного» роста исследуемых функций может оказаться очень полезным.

3. При более внимательном рассмотрении обнаруживается, что «подграфик» функции $v(n)$ для нечетных n возрастает тоже неравномерно. Значения $v(n)$ для простых n образуют кривую, лежащую выше значений $v(n)$ для составных нечетных n. Более того, создается впечатление, что величины $v(n)$ для составных нечетных n зависят от свойств делимости чис-

ла n. Каковы именно эти свойства, пока неизвестно. Возможно, основную роль играет число делителей n, или величина наименьшего отличного от нуля делителя n, или некий другой параметр. Число $n = 100$ все еще слишком мало, чтобы сделать определенные выводы. Продолжение вычислений вершин политопов разбиений позволит расширить собранную статистику и многое прояснить в этом вопросе. Если высказанное подозрение оправдается, то основным итогом проделанной работы, по мнению авторов, следует считать обнаружение связи между комбинаторно-геометрическими свойствами политопов разбиений чисел с теоретико-числовыми свойствами разбиваемых чисел.

Заключение

Описанные алгоритм и программа разработаны для продолжения исследования полиэдральной структуры множеств разбиений чисел. Авторы вычислили все вершины и опорные вершины политопов разбиений чисел для $n \le 105$ в расчете на то, что знание списков вершин поможет обнаружить их новые свойства. Полученные данные уже сейчас привели к постановке новых проблем о связи числовых характеристик политопов разбиений n с теоретико-числовыми свойствами n. Попутно были вычислены все рюкзачные разбиения чисел $n \le 165$. Вычисленные последовательности чисел вершин и опорных вершин P_n для $n \le 100$ и чисел рюкзачных разбиений для $n \le 165$ представлены в энциклопедии «Целочисленные последовательности» [20–22].

Отметим, что вывод приближенных формул для $p(n)$ потребовал длительных и напряженных усилий многих математиков, а также гениальной интуиции Рамануджана. История теории разбиений подтверждает слова Эйлера, сказанные им в 1750 г. в труде «De Partitione Numerorum»: «Всякий, кто намеревается постичь структуру разбиений чисел, обрекает себя на необъятный труд и должен быть готов страдать от постоянного пребывания в состоянии абсолютного внимания, чтобы не оказаться жертвой чудовищного заблуждения» [16]. Маловероятно, что вершины политопов разбиений подчиняются более простым законам, чем сами разбиения.

Список литературы

1. Эндрюс, Г. Теория разбиений / Г. Эндрюс. – М. : Наука, 1982. – 256 с.

2. Эйлер, Л. Введение в анализ бесконечных / Л. Эйлер. – М. : Гос. изд. физ.-мат. лит., 1961. – Т. 1. – 316 с.

3. Dickson, L.E. History of the Theory of Numbers / L.E. Dickson. – Vol. II : Diophantine Analysis. – Washington : Carnegie Inst., 1919. – 520 p.

4. Шлык, В.А. Политопы разбиений чисел / В.А. Шлык // Весці Нац. акад. навук Беларусі. Сер. фіз.-мат. навук. – 1996. – № 3. – С. 89–92.

5. Shlyk, V.A. Polytopes of partitions of numbers / V.A. Shlyk // European J. Combin. – 2005. – Vol. 26, № 8. – P. 1139–1153.

6. Емеличев, В.А. Многогранники, графы, оптимизация (комбинаторная теория многогранников) / В.А. Емеличев, М.М. Ковалев, М.К. Кравцов. – М. : Наука, 1981. – 341 с.

7. Шлык, В.А. Комбинаторные операции порождения вершин политопа разбиений чисел / В.А. Шлык // Докл. Нац. акад. наук Беларуси. – 2009. – Т. 53, № 6. – С. 27–32.

8. Холл, М. Комбинаторика / М. Холл. – М. : Мир, 1970. – 424 с.

9. Gupta, H. Tables of partitions / H. Gupta, C.E. Gwyther, C.P. Winter // Royal society mathematical tables. – Vol. 4. – Cambridge : University Press, 1958. – 132 p.

10. Sloane, N.J.A. a(n) = number of partitions of n (the partition numbers) [Electronic resource] / N.J.A. Sloane. – 2010. – Mode of access : http://oeis.org/A000041. – Date of access : 11.10.2015.

11. Zoghbi, A. Fast algorithms for generating integer partitions / A. Zoghbi, I. Stojmenovic // Intern. J. Computer Math. – 1998. – Vol. 70. – P. 319–332.

12. Рейнгольд, Э. Комбинаторные алгоритмы: теория и практика / Э. Рейнгольд, Ю. Нивергельт, Н. Део. – М. : Мир, 1980. – 476 с.

13. Nijenhius, A. A method and two algorithms on the theory of partitions / A. Nijenhius, H.S. Wilf // J. Comb. Theory A. – 1975. – Vol. 18. – P. 219–222.

14. Riha, W. Efficient algorithms for doubly and multiply restricted partitions / W. Riha, K.R. James // Algorithm 29. Computing. – 1976. – Vol. 16. – P. 163–168.

15. Constant time generation of integer partitions / K. Yamanaka [et al.] // IEICE Transactions on Fundamentals of Electronics, Communications and Computer Sciences. – 2007. – Vol. E90-A, № 5. – P. 888–895.

16. Knuth, D.E. Generating all partitions. Pre-fascicle 3B of The Art of Computer Programming. A draft of sections 7.2.1.4-5 [Electronic resource] / D.E. Knuth. – 2004. – Mode of access : http://www.cs.utsa.edu/~wagner/knuth/fasc3b.pdf. – Date of access : 11.10.2015.

17. Kelleher, J. Generating all partitions: a comparison of two encodings [Electronic resource] / J. Kelleher, B. O'Sullivan. – 2009. – Mode of access : http://arxiv.org/pdf/0909.2331v1.pdf. – Date of access : 11.10.2015.

18. Opdyke, J.D. A unified approach to algorithms generating unrestricted and restricted integer compositions and integer partitions / J.D. Opdyke // J. Math. Model. Algor. – 2010. – Vol. 9, № 1. – P. 53–97.

19. Stojmenovic, I. Generating all and random instances of a combinatorial object / I. Stojmenovic // Handbook of applied algorithms: solving scientific, engineering, and practical problem. – Hoboken : John Wiley&Sons, 2008. – P. 1–38.

20. Shlyk, V.A. Number of vertices of the integer partition polytope [Electronic resource] / V.A. Shlyk. – 2012. – Mode of access : http://oeis.org/A203898. – Date of access : 11.10.2015.

21. Shlyk, V.A. Number of support partitions-vertices [Electronic resource] / V.A. Shlyk. – 2012. – Mode of access : http://oeis.org/A203899. – Date of access : 11.10.2015.

22. Ehrenborg, R. Number of knapsack partitions of n [Electronic resource] / R. Ehrenborg. – 2005. – Mode of access : http://oeis.org/A108917. – Date of access : 11.10.2015.

23. Шлык, В.А. Критерий представления разбиений чисел в виде выпуклой комбинации двух разбиений / В.А. Шлык // Вестник БГУ. Сер. 1. – 2009. – № 2. – С. 109–114.

24. Shlyk, V.A. Integer partitions from the polyhedral point of view / V.A. Shlyk // Electron. Notes Discrete Math. – 2013. – Vol. 43. – P. 319–327.

25. Shlyk, V.A. Polyhedral approach to integer partitions / V.A. Shlyk // Journal of Combinatorial Mathematics and Combinatorial Computing. – 2014. – Vol. 89. – P. 113–128.

26. Gawrilow, E. Polymake: a framework for analyzing convex polytopes / E. Gawrilow, E.M. Joswig // Polytopes – combinatorics and computation. – Basel : Birkhäuser, 2000. – P. 43–73.

27. Шлык, В.А. О вершинах политопов разбиений чисел / В.А. Шлык // Докл. Нац. акад. наук Беларуси. – 2008. – Т. 52, № 3. – С. 5–10.

28. Handbook of Mathematical Functions / M. Abramowitz, I.A. Stegun [eds]. – Applied Math. Series 55. – Washington : Government Printing Office, 1972. – 1046 p.

29. Barvinok, A.I. Polynomial time algorithm for counting integral points in polyhedra when the dimension is fixed / A.I. Barvinok // Math. Oper. Res. – 1994. – Vol. 19. – P. 769–779.

30. Effective lattice point counting in rational convex polytopes / J.A. De Loera [et al.] // Journal of Symbolic Computation. – 2004. – Vol. 38, № 4. – P. 1273–1302.

31. Ehrenborg, R. The Möbius function of partitions with restricted block sizes / R. Ehrenborg, M.A. Readdy // Adv. in Appl. Math. – 2007. – Vol. 39, № 3. – P. 283–292.

Поступила 07.09.2015

[1]*ИООО «Управляющая компания ”Атлант-М”»,*
Минск, Шаранговича, 22-А
e-mail: alvrub@gmail.com

[2]*Командно-инженерный институт МЧС Республики Беларусь,*
Минск, Машиностроителей, 25
e-mail: v.shlyk@gmail.com

A.S. Vroublevski, V.A. Shlyk

COMPUTING VERTICES OF INTEGER PARTITION POLYTOPES

The paper describes a method of generating vertices of the polytopes of integer partitions that was used by the authors to calculate all vertices and support vertices of the partition polytopes for all $n \leq 105$ and all knapsack partitions of $n \leq 165$. The method avoids generating all partitions of n. The vertices are determined with the help of sufficient and necessary conditions; in the hard cases, the well-known program Polymake is used. Some computational aspects are exposed in more detail. These are the algorithm for checking the criterion that characterizes partitions that are convex combinations of two other partitions; the way of using two combinatorial operations that transform the known vertices to the new ones; and employing the Polymake to recognize a limited number (for small n) of partitions that need three or more other partitions for being convexly expressed. We discuss the computational results on the numbers of vertices and support vertices of the partition polytopes and some appealing problems these results give rise to.