

The partitions Package

October 4, 2007

Type Package

Title Additive partitions of integers

Version 1.7-4

Date 2007-08-20

Author Robin K. S. Hankin

Depends polynom

Maintainer Robin K. S. Hankin <r.hankin@noc.soton.ac.uk>

Description Additive partitions of integers. Enumerates the partitions, unequal partitions, and restricted partitions of an integer; the three corresponding partition functions are also given. Set partitions are now included.

License GPL

R topics documented:

P	2
as.matrix.partition	3
conjugate	4
partitions-package	5
parts	6
print.partition	7
setparts	8
summary.partition	10
Index	12

P

*Number of partitions of an integer***Description**

Given an integer, `P ()` returns the number of additive partitions, `Q ()` returns the number of unequal partitions, and `R ()` returns the number of restricted partitions. Function `S ()` returns the number of block partitions.

Usage

```
P(n, give = FALSE)
Q(n, give = FALSE)
R(m, n, include.zero = FALSE)
S(f, n = NULL, include.fewer = FALSE)
```

Arguments

<code>n</code>	Integer whose partition number is desired. In function <code>S ()</code> , the default of <code>NULL</code> means to return the number of partitions of any size
<code>m</code>	In function <code>R ()</code> , the order of the decomposition
<code>give</code>	Boolean, with default <code>FALSE</code> meaning to return just <code>P (n)</code> or <code>Q (n)</code> and <code>TRUE</code> meaning to return <code>P (1:n)</code> or <code>Q (1:n)</code> (this option takes no extra computation)
<code>include.zero</code>	In <code>restrictedparts ()</code> , Boolean with default <code>FALSE</code> meaning to count only partitions of <code>n</code> into <i>exactly</i> <code>m</code> parts; and <code>TRUE</code> meaning to include partitions of <code>n</code> into <i>at most</i> <code>m</code> parts (because parts of zero are included)
<code>include.fewer</code>	In function <code>blockparts ()</code> , Boolean with default <code>FALSE</code> meaning to return partitions into <i>exactly</i> <code>n</code> and <code>TRUE</code> meaning to return partitions into <i>at most</i> <code>n</code>
<code>f</code>	In function <code>S ()</code> , the stack vector

Details

Functions `P ()` and `Q ()` use Euler's recursion formula. Function `R ()` enumerates the partitions using Hindenburg's method (see Andrews) and counts them until the recursion bottoms out.

Function `S ()` finds the coefficient of x^n in the generating function $\prod_{i=1}^L \sum_{j=0}^{f_i} x^j$, where `L` is the length of `f`, using the **polynom** package.

All these functions return a double.

Note

Functions `P ()` and `Q ()` use `unsigned long long` integers, a type which is system-dependent. For me, `P ()` works for `n` equal to or less than 416, and `Q ()` works for `n` less than or equal to 792. YMMV; none of the methods test for overflow, so use with care!

Author(s)

Robin K. S. Hankin; `S()` is due to an anonymous JSS referee

Examples

```
P(10, give=TRUE)
Q(10, give=TRUE)
R(10, 20, include.zero=FALSE)
R(10, 20, include.zero=TRUE)

S(1:4, 5)
```

`as.matrix.partition`
Coerces partitions to matrices

Description

Coerces partitions objects to matrices

Usage

```
as.matrix.partition(x, ...)
```

Arguments

<code>x</code>	Object to be coerced
<code>...</code>	Further arguments

Author(s)

Robin K. S. Hankin

Examples

```
as.matrix(parts(5))
```

 conjugate

Conjugate partitions and Durfee squares

Description

Given a partition, provide its conjugate or Durfee square

Usage

```
conjugate(x)
durfee(x)
```

Arguments

`x` Either a vector describing a partition, in standard form (ie nonincreasing); or a matrix whose columns are partitions in standard form

Details

Conjugation is described in Andrews, and (eg) Hardy and Wright.

Essentially, `conjugate()` carries out R idiom `rev(cumsum(table(factor(a[a>0], levels=max(a):1))))`, but faster.

The “Durfee square” of a partition is defined on page 281 of Hardy and Wright. It is the largest square of nodes contained in the partition’s Ferrers graph. Function `durfee()` returns the side of the Durfee square which Andrews denotes $d(\lambda)$. It is equivalent to R idiom `function(a){sum(a>=1:length(a))}`, but faster.

Value

Returns either a partition in standard form, or a matrix whose columns are partitions in standard form.

Note

If argument `x` is not nonincreasing, all bets are off: these functions will not work and will silently return garbage. Caveat emptor! (output from `blockparts()` is not necessarily non-increasing)

Author(s)

Robin K. S. Hankin

Examples

```
parts(5)
conjugate(parts(5))

restrictedparts(6,4)
conjugate(restrictedparts(6,4))

durfee(10:1)
```

partitions-package *Integer partitions*

Description

Routines to enumerate all partitions of an integer; includes restricted and unequal partitions.

Details

This package comprises eight functions: `P()`, `Q()`, `R()`, and `S()` give the number of partitions, unequal partitions, restricted partitions, and block partitions of an integer.

Functions `parts()`, `diffparts()`, `restrictedparts()`, and `blockparts()` enumerate these partitions.

Function `conjugate()` gives the conjugate of a partition and function `durfee()` gives the size of the Durfee square.

NB the emphasis in this package is terse, efficient C code. This means that there is a minimum of argument checking. For example, function `conjugate()` assumes that the partition is in standard form (ie nonincreasing); supplying a vector in nonstandard form will result in garbage being returned silently. Note that a block partition is not necessarily in standard form.

Author(s)

Robin K. S. Hankin

References

- G. E. Andrews 1998 *The Theory of Partitions*, Cambridge University Press
- M. Abramowitz and I. A. Stegun 1965. *Handbook of Mathematical Functions*, New York: Dover
- G. H. Hardy and E. M. Wright 1985 *An introduction to the theory of numbers*, Clarendon Press: Oxford (fifth edition)
- R. K. S. Hankin 2006. *Additive integer partitions in R*. Journal of Statistical Software, Code Snippets, 16(1)
- R. K. S. Hankin 2007. *Urn sampling without replacement: enumerative combinatorics in R*. Journal of Statistical Software, Code Snippets, 17(1)

Examples

```
parts(5)
diffparts(9)
restrictedparts(15,10)
P(10,give=TRUE)
Q(10,give=TRUE)
R(5,10)
```

 parts

Enumerate the partitions of an integer

Description

Given an integer, return a matrix whose columns enumerate its partitions.

Function `parts()` returns the unrestricted partitions; function `diffparts()` returns the unequal partitions; function `restrictedparts()` returns the restricted partitions; function `blockparts()` returns the partitions subject to specified maxima; and function `compositions()` returns all compositions of the argument.

Usage

```
parts(n)
diffparts(n)
restrictedparts(n, m, include.zero=TRUE, decreasing=TRUE)
blockparts(f, n=NULL, include.fewer=FALSE)
compositions(n, m=NULL, include.zero=TRUE)
```

Arguments

<code>n</code>	Integer to be partitioned. In function <code>blockparts()</code> , the default of <code>NULL</code> means to return all partitions of any size
<code>m</code>	In <code>restrictedparts()</code> , the order of the partition
<code>include.zero</code>	In functions <code>restrictedparts()</code> and <code>compositions()</code> , Boolean with default <code>FALSE</code> meaning to include only partitions of n into <i>exactly</i> m parts; and <code>TRUE</code> meaning to include partitions of n into <i>at most</i> m parts (because zero parts are included)
<code>include.fewer</code>	In function <code>blockparts()</code> , Boolean with default <code>FALSE</code> meaning to return vectors whose sum is <i>exactly</i> n and <code>TRUE</code> meaning to return partitions whose sum is <i>at most</i> n
<code>decreasing</code>	In <code>restrictedparts()</code> , Boolean with default <code>TRUE</code> meaning to return partitions whose parts are in decreasing order and <code>FALSE</code> meaning to return partitions in lexicographical order, as appearing in Hindenburg's algorithm. Note that setting to <code>decreasing</code> to <code>FALSE</code> has the effect of making <code>conjugate()</code> return garbage
<code>f</code>	In function <code>blockparts()</code> , a vector of strictly positive integers that gives the maximal number of blocks; see details

Details

Function `parts()` uses the algorithm in Andrews. Function `diffparts()` uses a very similar algorithm that I have not seen elsewhere. These functions behave strangely if given an argument of zero.

Function `restrictedparts()` uses the algorithm in Andrews, originally due to Hindenburg. For partitions into at most m parts, the same Hindenburg's algorithm is used but with a start vector of `c(rep(0, m-1), n)`.

Function `blockparts()` enumerates the compositions of an integer subject to a maximum criterion: given vector $y = (y_1, \dots, y_n)$ all sets of $a = (a_1, \dots, a_n)$ satisfying $\sum_{i=1}^p a_i = n$ subject to $0 < a_i \leq y_i$ for all i are given in lexicographical order. If argument `y` includes zero elements, these are treated consistently (ie a position with zero capacity).

If `n` takes its default value of `NULL`, then $\sum_{i=1}^p a_i = n$ is removed (the numbers may sum to anything). Note that these solutions are not necessarily in standard form, so functions `durfee()` and `conjugate()` may fail.

Function `compositions()` returns all 2^n ways of partitioning an integer; thus `4+1+1` is distinct from `1+4+1` or `1+1+4`. This function is different from all the others in the package in that it is written in `R`; it is not clear that `C` would be any faster.

Author(s)

Robin K. S. Hankin

References

G. E. Andrews. "The Theory of Partitions", Cambridge University Press, 1998

Examples

```
parts(5)
diffparts(10)
restrictedparts(9, 4)
restrictedparts(9, 4, FALSE)
restrictedparts(9, 4, decreasing=TRUE)

blockparts(1:4)
blockparts(1:4, 3)
blockparts(1:4, 3, include.fewer=TRUE)

compositions(3)
```

Description

A print method for partition objects and summary partition objects, including various configurable options

Usage

```
print.partition(x, mat = getOption("matrixlike"), h =
getOption("horiz"), ...)
print.summary.partition(x, ...)
```

Arguments

x	Object to be printed: an object of class either <code>partition</code> or <code>summary.partition</code>
mat	Boolean, with <code>TRUE</code> meaning to print like a matrix, and any other value meaning to print without column names (which usually results in more compact appearance)
h	Boolean, with <code>TRUE</code> meaning to print as is, and any other value meaning to print the transpose
...	Further arguments provided for compatibility

Author(s)

Robin K. S. Hankin

Examples

```
print(parts(5))
summary(parts(7))
```

setparts

Set partitions

Description

Enumeration of set partitions

Usage

```
setparts(x)
```

Arguments

x	If a vector of length 1, the size of the set to be partitioned. If a vector of length greater than 1, return all equivalence relations with equivalence classes with sizes of the elements of x. If a matrix, return all equivalence classes with sizes of the columns of x
---	---

Details

A *partition* of a set $S = \{1, \dots, n\}$ is a family of sets T_1, \dots, T_k satisfying

- $i \neq j \longrightarrow T_i \cap T_j = \emptyset$
- $\cup_{i=1}^k T_k = S$
- $T_i \neq \emptyset$ for $i = 1, \dots, k$

The induced *equivalence relation* has $i \sim j$ if and only if i and j belong to the same partition.

There are exactly fifteen ways to partition a set of four elements:

```
(1234)
(123)(4), (124)(3), (134)(2), (234)(1)
(12)(34), (13)(24), (14)(23)
(12)(3)(4), (13)(2)(4), (23)(1)(4), (24)(1)(3), (34)(1)(2)
(1)(2)(3)(4)
```

Note that (12)(3)(4) is the same partition as, for example, (3)(4)(21) as the equivalence relation is the same.

Consider partitions of a set S of five elements (named 1, 2, 3, 4, 5) with sizes 2,2,1. These may be enumerated as follows:

```
> u <- c(2, 2, 1)
> setparts(u)

[1,] 1 1 1 1 1 1 1 1 1 1 1 1 3 3 3
[2,] 2 2 3 1 1 1 2 2 3 2 2 3 1 1 1
[3,] 3 2 2 3 2 2 1 1 1 3 2 2 2 1 2
[4,] 2 3 2 2 3 2 3 2 2 1 1 1 2 2 1
[5,] 1 1 1 2 2 3 2 3 2 2 3 2 1 2 2
```

See how each column has two 1s, two 2s and one 3. This is because the first and second classes have size two, and the third has size one.

The first partition, $x=c(1, 2, 3, 2, 1)$, is read “class 1 contains elements 1 and 5 (because the first and fifth element of x is 1); class 2 contains elements 2 and 4 (because the second and fourth element of x is 2); and class 3 contains element 3 (because the third element of x is 3)”. Formally, class i has elements $\text{which}(x==u[i])$.

Value

Returns a matrix each of whose columns show a set partition; an object of class `c("set_partition", "partition")`. Type `?print.partition` to see how to change the options for printing.

Author(s)

Luke G. West (C++) and Robin K. S. Hankin (R)

References

- R. K. S. Hankin 2006. *Additive integer partitions in R*. Journal of Statistical Software, Code Snippets 16(1)
- R. K. S. Hankin 2007. *Urn sampling without replacement: enumerative combinatorics in R*. Journal of Statistical Software, Code Snippets 17(1)

See Also

`parts`, `print.partition`

Examples

```

setparts(4)                # all partitions of a set of 4 elements

setparts(c(3,3,2))        # all partitions of a set of 8 elements
                           # into sets of sizes 3,3,2.

jj <- restrictedparts(5,3)
setparts(jj)              # partitions of a set of 5 elements into
                           # at most 3 sets

setparts(conjugate(jj))   # partitions of a set of 5 elements into
                           # sets not exceeding 3 elements

setparts(diffparts(5))    # partitions of a set of 5 elements into
                           # sets of different sizes

```

`summary.partition` *Provides a summary of a partition*

Description

Provides a summary of an object of class `partition`: usually the first and last few partitions (columns)

Usage

```
summary.partition(object, ...)
```

Arguments

<code>object</code>	Partition
<code>...</code>	Further arguments; see details section below

Details

The ellipsis arguments are used to pass how many columns at the start and the end of the matrix are selected; this defaults to 10.

The function is designed to behave as expected: if there is an argument named “n”, then this is used. If there is no such argument, the first one is used.

Value

A summary object is a list, comprising three elements:

<code>shortened</code>	Boolean, with <code>TRUE</code> meaning that the middle section of the matrix is omitted, and <code>FALSE</code> meaning that the entire matrix is returned because <code>n</code> is too big
<code>n</code>	Number of columns to return at the start and the end of the matrix
<code>out</code>	Matrix returned: just the first and last <code>n</code> columns (if <code>shortened</code> is <code>TRUE</code>), or the whole matrix if not

Author(s)

Robin K. S. Hankin

Examples

```
summary(parts(7))
```

```
summary(parts(11), 3)
```

Index

*Topic **math**

- as.matrix.partition, 3
- conjugate, 3
- P, 1
- parts, 5
- print.partition, 7
- setparts, 8
- summary.partition, 10

*Topic **package**

- partitions-package, 4

as.matrix.partition, 3

blockparts (*parts*), 5

compositions (*parts*), 5

conjugate, 3

diffparts (*parts*), 5

Durfee (*conjugate*), 3

durfee (*conjugate*), 3

P, 1

partitions (*partitions-package*), 4

partitions-package, 4

parts, 5, 9

print.partition, 7, 9

print.summary.partition
(*print.partition*), 7

Q(*P*), 1

R(*P*), 1

restrictedparts (*parts*), 5

S(*P*), 1

setparts, 8

summary.partition, 10