Integer partitions

Definitions, enumeration and constrained generation

Amirhossein Mahmoudi



$$7 = 4 + 2 + 1$$

Integer partitions · Definitions & generation

What is an integer partition?

A partition of a positive integer m is a sum of positive integers, ignoring their order.

$$m = a_1 + a_2 + \cdots + a_N$$
$$a_i \geq 1$$

Symbols

m · total being partitioned

N · number of parts

a_i · size of part i

Parts may be written in any order.

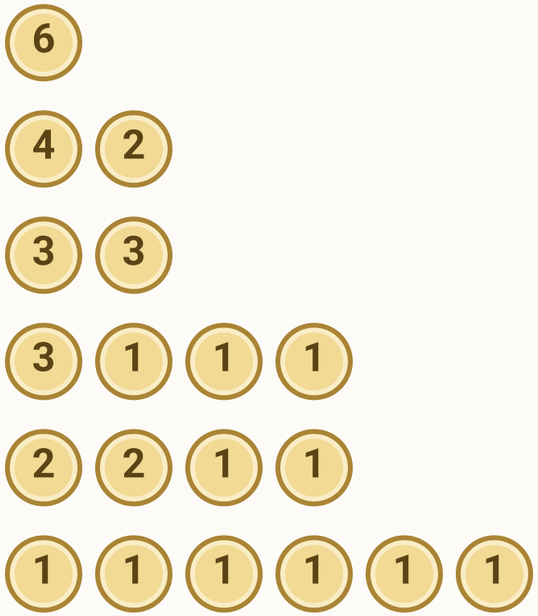
Example: a partition of 7

$$a = [4, 2, 1], N = 3$$
$$7 = 4 + 2 + 1$$

Think of each part as a coin value.

Order is irrelevant. Repeated parts and [7] are allowed.

Partitions of 6: eleven distinct answers



$$6 = 6$$

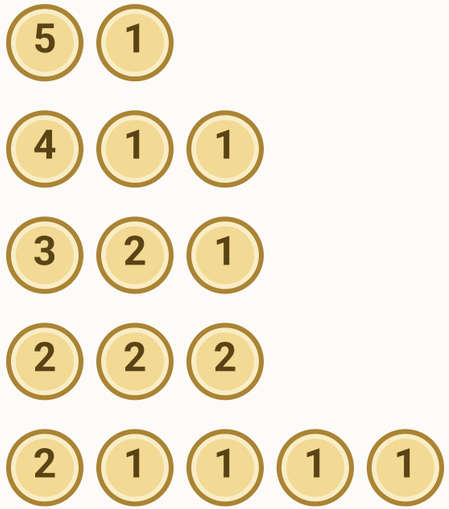
$$6 = 4 + 2$$

$$6 = 3 + 3$$

$$6 = 3 + 1 + 1 + 1$$

$$6 = 2 + 2 + 1 + 1$$

$$6 = 1 + 1 + 1 + 1 + 1 + 1$$



$$6 = 5 + 1$$

$$6 = 4 + 1 + 1$$

$$6 = 3 + 2 + 1$$

$$6 = 2 + 2 + 2$$

$$6 = 2 + 1 + 1 + 1 + 1$$

$$P(6) = 11$$

Partitions ignore order; compositions retain it

Partition · unordered positive parts



The same parts (coin values): one partition.

$$4 + 2 \equiv 2 + 4$$

Composition · ordered positive parts

$$(4, 2) \neq (2, 4)$$

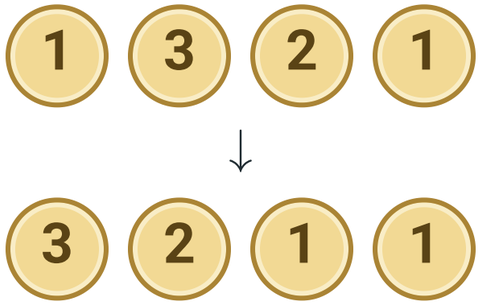
Part 4 then part 2, or part 2 then part 4: two sequences.

That is a different question.

Standard form: write parts in nonincreasing order

Standard form: write the parts in nonincreasing order, from largest to smallest, allowing equal parts.

Same partition, before and after sorting



Definition of standard form

$$a_1 \geq a_2 \geq \cdots \geq a_N \geq 1$$

Each part is at least as large as the next.

Equal parts are allowed: this is not strictly decreasing order.

Sorting changes only the written order—not the partition. The generator will preserve this form as it builds each answer.

Integer partitions · Definitions & generation

The number of integer partitions grows rapidly

P(m) counts the partitions of the integer m.

$$P\left(m\right) \sim \frac{e^{\pi\sqrt{\frac{2m}{3}}}}{4m\sqrt{3}}$$

Faster than any polynomial; slower than c^m for every fixed $c > 1$.

The symbol ∼ means the ratio tends to 1 as m grows—not exact equality.

$$P\left(1\right), P\left(2\right), P\left(3\right), P\left(4\right), P\left(5\right), P\left(6\right) = 1, 2, 3, 5, 7, 11$$

No simple elementary closed-form formula for P(m) is known. However, its asymptotic growth is known and is shown above. Exact computation is still possible.

3.1 · Generate all partitions: goal and idea

Goal: given a positive integer m, generate every partition of m exactly once, in standard form.

How will we generate them?

1. Try each allowed value for the next part.
2. Recursively partition the remaining sum.
3. Limit later parts to the value just chosen, preserving standard form.

When the remainder is zero, output the completed partition.

State used to implement this idea

$$\mathrm{Rec}\,(\,r, B, N\,)$$

Partition-variable legend

`m` Target sum / amount	`r` Remaining sum
`B` Next-part upper bound	`N` Selected-part count
`a[1..N]` Selected parts / coin values	`i` Candidate next part

Invariant: selected-part sum + r = m. Start with r = m, B = m, N = 0.

3.1 · Enumerate partitions by recursively selecting parts

Meaning of Rec(r, B, N): generate every completion of the current prefix a[1..N].

Try every positive next part allowed by both the bound and the remainder:

$$1 \leq i \leq \min\left(B, r\right)$$

Store a[N+1] = i; subtract i from the remainder, set the next bound to i, and add one to the part count.

$$\mathrm{Rec}\left(r - i, i, N + 1\right)$$

Base case: if r = 0, output only a[1..N] and return.

$$\mathrm{Rec}\left(m, m, 0\right)$$

Start: empty prefix, full sum remaining. **Invariant:** selected-part sum + r = m.

```
procedure Rec(r, B, N):
    if r == 0:
        output a[1..N]
        return
    for i = 1 to min(B, r):
        a[N + 1] = i
        Rec(r - i, i, N + 1)

Rec(m, m, 0)
```

Partition-variable legend

- `m` Target sum / amount
- `r` Remaining sum
- `B` Next-part upper bound
- `N` Selected-part count
- `a[1..N]` Selected parts / coin values
- `i` Candidate next part

Integer partitions · Definitions & generation

Recursive enumeration of 4: five output partitions

Input: m = 4. Output: five partitions, including the single part [4].



$4 = 1 + 1 + 1 + 1$



$4 = 2 + 1 + 1$



$4 = 2 + 2$



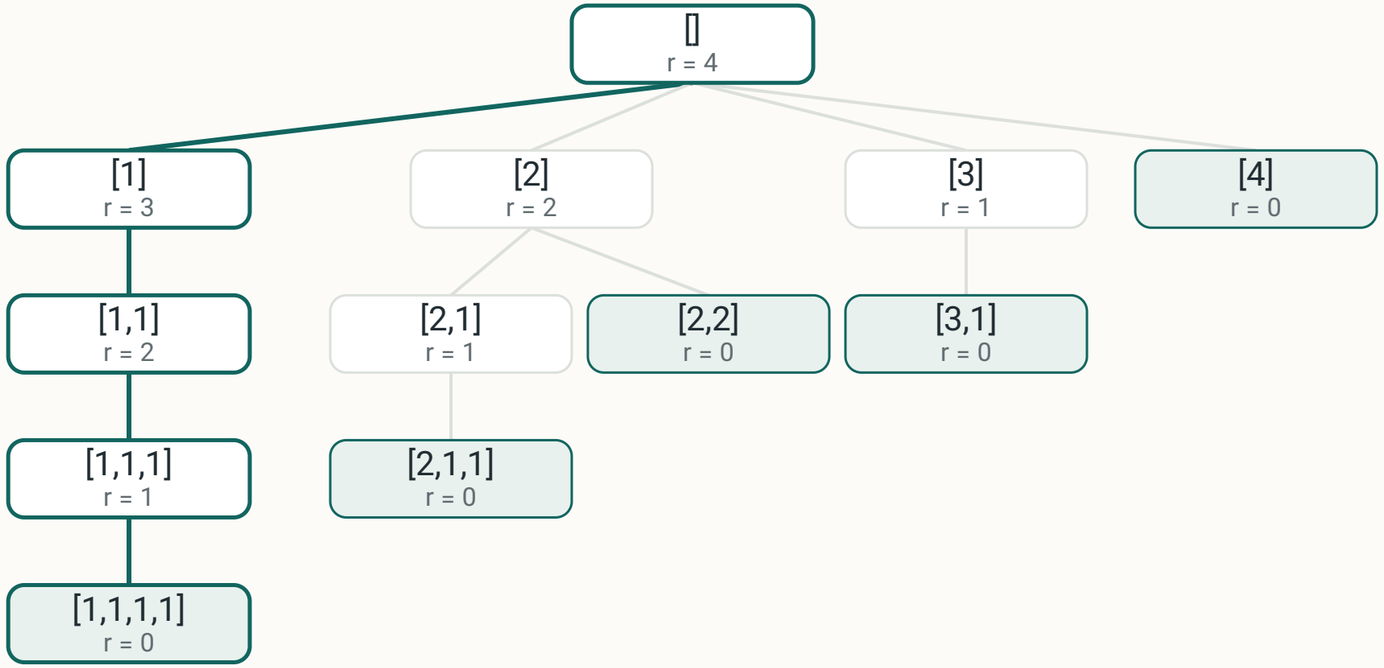
$4 = 3 + 1$



$4 = 4$

3.1 · From the recursion tree to the time bound

Example: every call from Rec(4,4,0). Nodes show the chosen prefix and remainder; green leaves are outputs.



Count leaves and path lengths

V counts calls (nodes): here V=12, with P(4)=5 leaves.

The teal path makes 4 choices. Generally, depth is at most m because every chosen part is positive.

Every node lies on a root-to-leaf path. Summing all path lengths gives an upper bound on the number of nodes.

$$V \leq 1 + mP(m)$$

$$T(m) = O(mP(m))$$

Search takes O(V); output writing adds at most mP(m) parts. These are upper bounds, not exact operation counts.

Integer partitions · Definitions & generation

Correct enumeration: valid, complete and duplicate–free

Validity

Selected-part sum + remainder = m.

Output only at remainder zero.

Completeness

Try every legal next part.

Follow any standard-form partition through its choices.

Uniqueness

Parts remain nonincreasing.

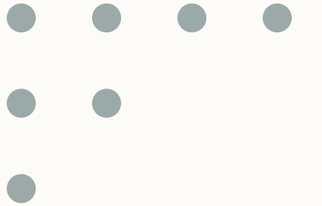
One standard vector has one search path.

Each selected part reduces the remainder by at least one, so every branch terminates.

Ferrers–Young diagrams and conjugation

Original partition

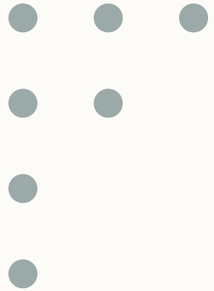
$a = [4, 2, 1]$





Conjugate partition

$a^* = [3, 2, 1, 1]$





Part = coin value; partition = a coin collection. The dots form a Ferrers–Young diagram.

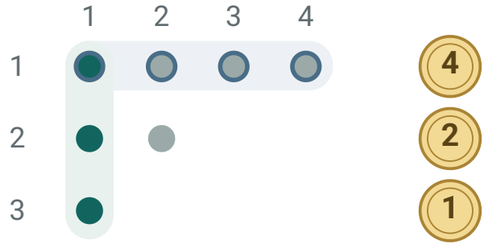
Each row contains one dot per unit of a part. Both diagrams contain seven dots.

Why does column 1 count the parts?

Every part is positive and every row starts at the left: one first-column dot per row.

Input: 3 parts, so 3 rows

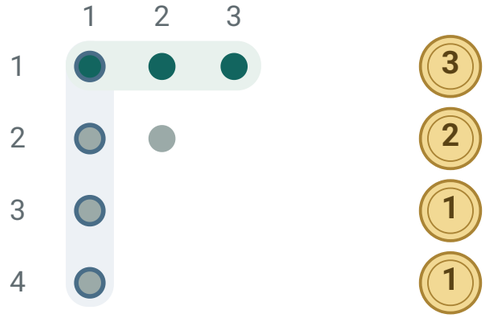
$$a = [\,4, 2, 1\,]$$



One coin is one part; three coins make three rows.

Conjugate: largest part is 3

$$b = [\,3, 2, 1, 1\,]$$



The three first-column dots become the first conjugate row.

Theorem 3.1: partitions of m with exactly n parts and partitions of m with largest part n are equinumerous.

Reason: conjugation is a reversible bijection. Exchanging rows and columns twice restores the original partition.

Integer partitions · Definitions & generation

What is bⱼ? Count one column.

$$b_j = \left| \left\{ i : a_i \geq j \right\} \right|$$

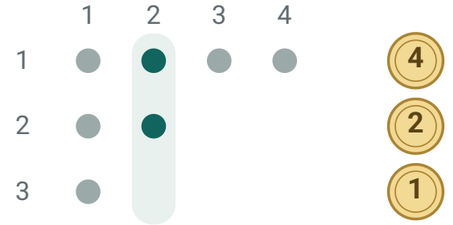
New part j = how many input rows reach column j.

Every symbol, in plain words

a_i	Input part i (coin value); a_1 is largest.
i	Input row index
N	Number of input parts / rows
j	Column index being counted
b_j	Conjugate part j: height / new coin value
$\lvert S \rvert$	Number of elements in set S
$\text{len}\,(\,b\,)$	Number of conjugate parts

Example: count column 2

$$a = [\,4, 2, 1\,], j = 2$$



$$b_2 = \left| \{\,1, 2\,\} \right| = 2$$

Rows 1 and 2 reach column 2; row 3 does not.

$b_1 = N = 3$ Every row reaches column 1.

$\text{len}\,(\,b\,) = a_1 = 4$ Each column gives one new part.

3.2 · Conjugate a partition using column heights

Goal: conjugate [4, 2, 1] into [3, 2, 1, 1], preserving sum 7.

```
procedure ConjPartition(a[1..N]):
    for j = 1 to a[1]:
        b[j] = 1
    nPrime = a[1]
    for i = 2 to N:
        for j = 1 to a[i]:
            b[j] = b[j] + 1
    return b[1..nPrime]
```

Partition-variable legend

- `a[i]` Input part i / coin value
- `b[j]` Conjugate part j / column height
- `j` Column index
- `N` Number of input parts
- `i` Input row index
- `nPrime` Output-part count: a[1]

Input: a positive partition already in nonincreasing standard form.

Partitions of 4: largest part versus number of parts

Largest part exactly 2

$[2,2]$	2 parts
$[2,1,1]$	3 parts

Every part is at most 2, and at least one part equals 2. Thus [1,1,1,1] does not qualify.

Exactly 2 parts

$[2,2]$	maximum 2
$[3,1]$	maximum 3

We require two parts; the largest part need not equal 2.

$$[2,1,1] \leftrightarrow [3,1]$$

Conjugation maps the largest-part family to the part-count family, so we can reuse the generator.

Count by the number of parts

P(m,n) counts partitions of m into exactly n parts (n coins in the analogy).

$$P(m) = \sum_{n=1}^{m} P(m,n)$$

Every partition has one definite number of parts. These disjoint groups cover all partitions.

$$P(6) = 1 + 3 + 3 + 2 + 1 + 1 = 11$$

The six terms correspond to counts 1 through 6.

Boundary values for positive m

$$P(m,1) = P(m,m) = 1 \qquad P(m,0) = 0, P(0,0) = 1$$

Exact counts for two and three parts

Exactly two parts

$$P(m,2) = \left\lfloor \frac{m}{2} \right\rfloor$$

Example: P(6,2) = 3: [5,1], [4,2], [3,3].

Exactly three parts

$$P(m,3) = \left\lfloor \frac{m^2+4}{12} \right\rfloor$$

Example: P(6,3) = 3: [4,1,1], [3,2,1], [2,2,2].

The floor ⌊x⌋ is the greatest integer no larger than x.

For fixed n, as m grows:

$$P(m,n) = \Theta\left(m^{n-1} \right)$$

Conjugation gives the same count for largest part n (largest coin value n).

Exact counting: one example, two groups

Example: count partitions of 6 into exactly 3 parts. Every partition belongs to exactly one group below.

$$P(\,m,n\,) = P(\,m-1,n-1\,) + P(\,m-n,n\,)$$

Group A: at least one part is 1

Remove just one part equal to 1: both the sum and the part count decrease by one.

$$[\,4,1,1\,] \to [\,4,1\,]$$
$$[\,3,2,1\,] \to [\,3,2\,]$$
$$P(\,5,2\,) = 2$$

Group B: every part is at least 2

Subtract 1 from each of the three parts: the sum decreases by 3; the part count stays 3.

$$[\,2,2,2\,] \to [\,1,1,1\,]$$
$$P(\,3,3\,) = 1$$

$$P(\,6,3\,) = P(\,5,2\,) + P(\,3,3\,) = 2 + 1 = 3$$

There is one empty partition: the empty list, with no parts and sum zero. The 1 counts arrangements, not parts.

$$P(\,0,0\,) = 1$$

Removing 1 from [1] leaves the empty list: one valid completion. Impossible cases have count zero.

$$P(\,1,1\,) = P(\,0,0\,) + P(\,0,1\,) = 1 + 0 = 1$$

3.1 vs 3.3: only initialization changes

3.1

Unrestricted start

All partitions of total `m`

Precondition: `m ≥ 1`

```
GenerateAll(m):
    Rec(m,m,0)
```

Example `m=7, n=3`: `(r,B,N) = (7,7,0)`

3.3

Changed initialization

Largest part (coin value) exactly `n`

Precondition: `1 ≤ n ≤ m`

```
GenerateLargest(m,n):
    a[1] = n
    Rec(m-n,n,1)
```

Example `m=7, n=3`: `(r,B,N) = (4,3,1)`; first choose `a[1] = 3`

Shared procedure

UNCHANGED

```
Rec(r,B,N):
    if r == 0:
        output a[1..N]
    else:
        for i = 1..min(B,r):
            a[N+1] = i
            Rec(r-i,i,N+1)
```

Variable legend

- **`m`** target sum / amount
- **`n`** required largest part / coin value
- **`r`** remaining sum
- **`B`** next-part upper bound
- **`N`** chosen-part count
- **`a`** array of parts (coin values)
- **`i`** candidate next part

Exactly, not at most: `Rec(m,n,0)` is insufficient: it can omit the part `n`.

3.3 · Enumerate partitions with largest part 3

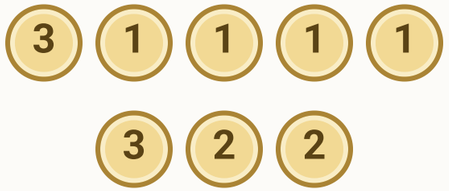
Input: m = 7, n = 3. Output: four partitions with largest part exactly 3.

```
a[1] = 3
Rec(4, 3, 1)
```

Partition-variable legend

- `m` Target integer
- `n` Required largest part
- `a[1] = n` Preselect the largest part n
- `r = m − n` Remaining sum
- `B = n` Later parts cannot exceed n
- `N = 1` One part is already selected

Preselect a[1] = n to guarantee a part equal to n. Bound all later parts by n: the maximum is exactly n, not merely at most n.





3.4 · Enumerate partitions with exactly 3 parts

Goal: partitions of m into exactly n parts; example m = 7, n = 3.

```
procedure RecPartition2(r, B, N):
    if r == 0:
        b = ConjPartition(a[1..N])
        output b
    else:
        for i = 1 to min(B, r):
            a[N+1] = i
            RecPartition2(r-i, i, N+1)

a[1] = n
RecPartition2(m-n, n, 1)
```

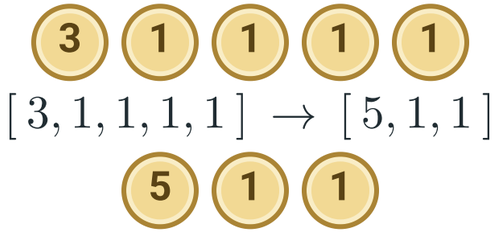
For 1 ≤ n ≤ m: initialize as in 3.3; at each leaf output the conjugate, not a.

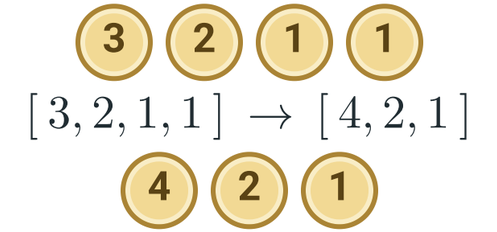
Partition-variable legend

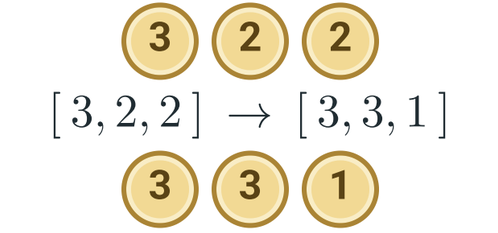
- `m` Original target sum
- `n` Required output-part count; 1 ≤ n ≤ m
- `r` Remaining sum of the intermediate partition
- `B` Next-part upper bound
- `N` Selected intermediate-part count
- `a[1..N]` Intermediate prefix; starts with n
- `i` Candidate next part
- `b` Conjugate output: exactly n parts

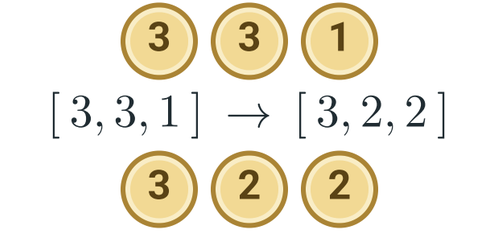
Conjugation pairs two constrained partition families

Sum 7 throughout: largest part 3 → exactly 3 parts (coin count 3).









Integer partitions · Definitions & generation

Why these bounds? Algorithms 3.1–3.4

P(m) counts all partitions; P(m,n) counts partitions with exactly n parts.

Algorithm / task	Where the bound comes from	Time	Space
3.1 · All partitions	At most m parts per path and P(m) leaves. Total path lengths bound both search and output writing.	$O(mP(m))$	$O(m)$
3.2 · One conjugate	Initialize a_1 entries, then make m-a_1 increments: m operations altogether. The output array has a_1 entries.	$\Theta(m)$	$\Theta(a_1)$
3.3 · Largest part n	Same search as 3.1, but only P(m,n) leaves; conjugation gives this family count.	$O(mP(m,n))$	$O(m)$
3.4 · Exactly n parts	Search from 3.3, plus one Θ(m) conjugation for each of its P(m,n) outputs.	$O(mP(m,n))$	$O(m)$

Assume unit-cost arithmetic and streamed outputs. Space includes the active stack and current output, not storage of all answers. O-bounds are upper bounds, not constant-delay claims.

From target to answers: combine the algorithms

Target: generate every partition of 7 into exactly 3 parts.

Step 1 · 3.3 reuses the recursion of 3.1

Preselect part 3; partition the remaining 4 with bound 3.

$$a_1 = 3, \mathrm{Rec}\ (\ 4, 3, 1\ )$$

Step 2 · Apply 3.2 to each output

Conjugate each intermediate partition whose largest part is 3; the result has exactly 3 parts.

$$b = \mathrm{ConjPartition}\ (\ a\ )$$

Algorithm 3.4 is this combination: at each leaf of the 3.3 search, output the conjugate instead of a.

$$[\,3,1,1,1,1\,] \to [\,5,1,1\,]$$

$$[\,3,2,2\,] \to [\,3,3,1\,]$$

$$[\,3,2,1,1\,] \to [\,4,2,1\,]$$

$$[\,3,3,1\,] \to [\,3,2,2\,]$$

Left: all largest-part-3 partitions. Right: all exactly-3-part partitions. The bijection preserves completeness and uniqueness.

Kreher & Stinson, *Combinatorial Algorithms: Generation, Enumeration, and Search*, 1999.