Generate all unique partitions of an integer - GeeksforGeeks

GeeksforGeeks ⋮ 4-5 minutes ⋮ 12/20/2013

Generate all unique partitions of an integer

Given a positive integer n, generate all possible unique ways to represent n as sum of positive integers.

Examples:

```
Input: n = 2
Output:
2
1 1

Input: n = 3
Output:
3
2 1
1 1 1
Note: 2+1 and 1+2 are considered as duplicates.

Input: n = 4
Output:
4
3 1
2 2
2 1 1
1 1 1 1
```

Solution: We print all partition in sorted order and numbers within a partition are also printed in sorted order (as shown in the above examples). The idea is to get the next partition using the values in the current partition. We store every partition in an array p[]. We initialize p[] as n where n is the input number. In every iteration. we first print p[] and then update p[] to store the next partition. So the main problem is to get the next partition from a given partition.

Steps to get next partition from current partition:
We are given current partition in p[] and its size. We need to update p[] to store next partition. Values in p[] must be sorted in non-increasing order.

1. Find the rightmost non-one value in p[] and store the count of 1's encountered before a non-one value in a variable rem_val (It indicates sum of values on right side to be updated). Let the index of non-one value be k.
2. Decrease the value of p[k] by 1 and increase rem_val by 1. Now there may be two cases:
    - **a) If** p[k] is more than or equal to rem_val. This is a simple case (we have the sorted order in new partition). Put rem_val at p[k+1] and p[0…k+1] is our new partition.
    - **b) Else** (This is a interesting case, take initial p[] as {3, 1, 1, 1}, p[k] is decreased from 3 to 2, rem_val is increased from 3 to 4, the next partition should be {2, 2, 2}).
3. Copy p[k] to next position, increment k and reduce count by p[k] while p[k] is less than rem_val. Finally, put rem_val at p[k+1] and p[0…k+1] is our new partition. This step is like dividing rem_val in terms of p[k] (4 is divided in 2's).

Following is the implementation of above algorithm:

- C++
- Java
- Python3
- C#
- PHP
- Javascript

C++

```
#include<iostream>

using namespace std;

void printArray(int p[], int n)

{

    for (int i = 0; i < n; i++)

    cout << p[i] << " ";

    cout << endl;
```