Pairwise Distinct Summands

The first time I heard about this problem is from Mohamed Anis Mani, a computer science teacher.

The problem description

Task: The goal of this problem is to represent a given positive integer **n** as a sum of as many pairwise distinct positive integers as possible. That is, to find the **maximum k** such that **n** can be written as

a(1) + a(2) + ⋯ + a(k) where a(1) ,…,a(k) are positive integers and **a(i) != a(j)** for all 1 ≤ i < j ≤ k.

Input Format: The input consists of a single integer **n**.

Constraints: 1 ≤ **n** ≤ 10^9

Output Format: In the first line, output the **maximum number k** such that n can be represented as a **sum of k pairwise distinct positive integers**. In the second line, output k pairwise distinct positive integers that sum up to n (if there are many such representations, output any of them).

Time Limits. C: 1 sec, C++: 1 sec, Java: 1.5 sec, Python: 5 sec. C#: 1.5 sec, Haskell: 2 sec, JavaScript: 3 sec, Ruby: 3 sec, Scala: 3 sec.

Memory Limit: 512 Mb

Sample 1	**Sample 2**	**Sample 3**
Input:	Input:	Input:
6	8	2
Output:	Output:	Output:
3	3	1
1 2 3	1 2 5	2

What is this about?

This an integers partition problem: https://en.wikipedia.org/wiki/Partition_(number_theory)

A partition of n into exactly k parts is an unordered sum of n that uses exactly k positive integers is denoted: $P(n,k)$

Examples:

- $P(5,2)=2$, because:

  $5=1+4$

  $5=2+3$

  There are only ***two*** sums of 5 that can be formed by using two positive integers.

- $P(5,3)=2$, because:

  $5=1+1+3$

  $5=1+2+2$

  There are only ***two*** sums of 5 that can be formed by using three positive integers.

A partition of n into exactly k parts is an unordered sum of n that uses exactly k positive distinct integers is denoted: $P^d(n,k)$

Examples:

- $P^d(5,2)=2$, because:

  $5=1+4$

  $5=2+3$

  There are only ***two*** sums of 5 that can be formed by using ***two*** distinct positive integers.

- $P^d(5,3)=0$, because there are no sums of 5 that can be formed by using three distinct positive integers.

- $P^d(8,3)=1$, because

  $8=1+2+5$

  There are only ***one*** sums of 8 that can be formed by using ***three*** distinct positive integers.

In our problem, we're looking for the maximum k that gives us at least one partition. Also, we must find the distinct positive integers that their sum gives n .

My solution

This solution is inspired from: https://stackoverflow.com/a/38255116

n can be written as $n=1+2+3+....+i+r$, where $i<r<n$

Example:

12 = 1 + 2 + 3 + 6

Here i = 3, and r = 6

Let's figure out how to compute i?

$$n=1+2+3+....+i+r$$
$$n=\frac{i\times(i+1)}{2}+r$$

The sum: $1+2+2+....+i=\frac{i\times(i+1)}{2}<n$

$$\frac{i^2+i}{2}<n$$

$$i^2+i-2n<0$$

let's resolve the equation: $i^2+i-2n=0$

$i=\frac{\sqrt{(1+8n)}-1}{2}$, this solution give us: $i^2+i-2n=0$

but we search an integer that give us: $i^2+i-2n<0$

So, the solution is the integer just before $i=\frac{\sqrt{(1+8n)}-1}{2}$

which is: $i=\lfloor\frac{\sqrt{(1+8n)}-1}{2}\rfloor-1$

Examples:

- $n=8$

  $$i=\lfloor\frac{\sqrt{(1+8\times8)}-1}{2}\rfloor-1=3-1=2$$

  $$s=\frac{2\times3}{2}=3$$ (s is the sum from 1 to 2 = $1+2$)

  $$r=8-s=8-3=5$$

  So, the maximum number of positive integers that we can sum is ***three*** (that's our k) which are 1 , 2 and 5 .

- $n=2$

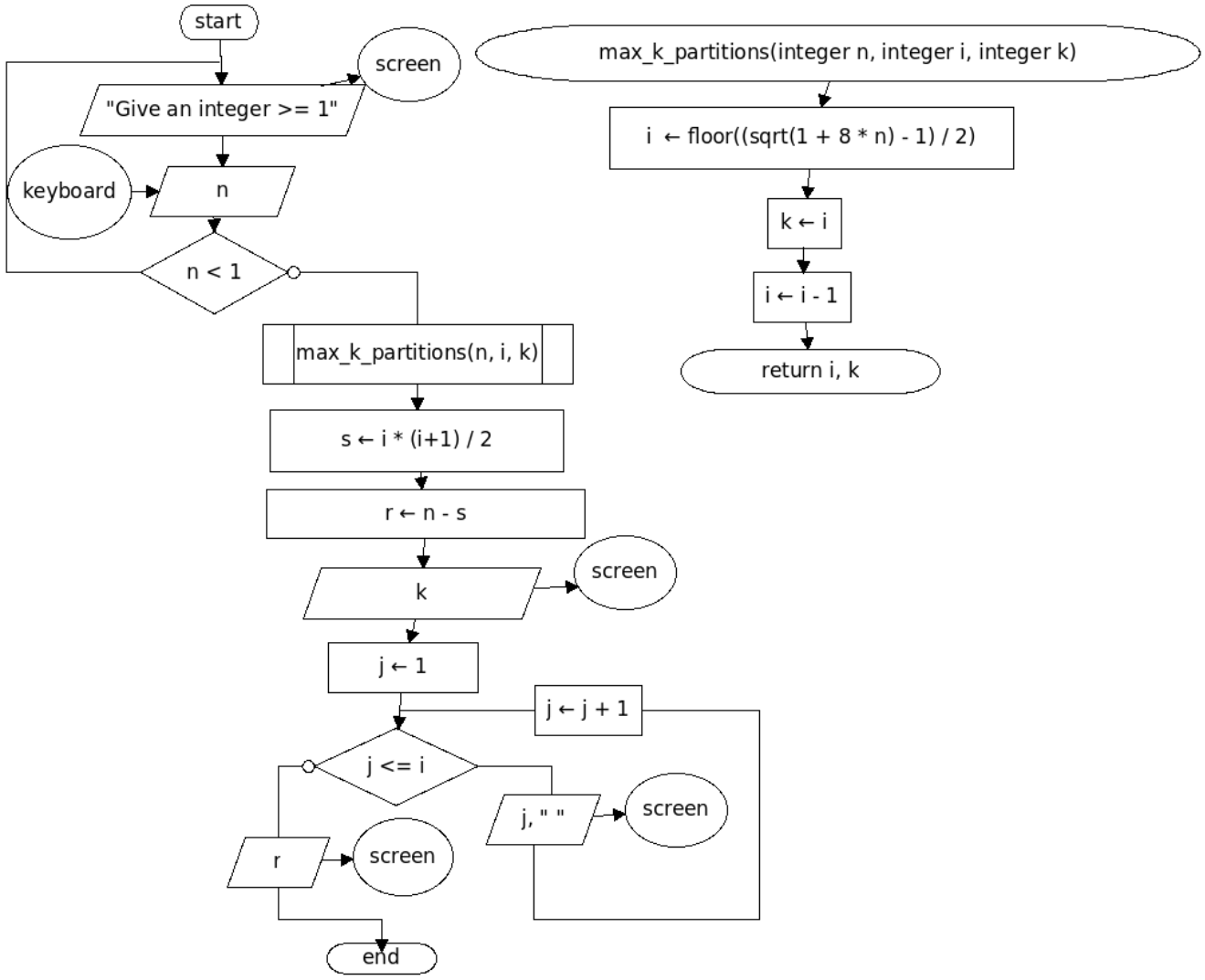
  $$i=\lfloor\frac{\sqrt{(1+8\times2)}-1}{2}\rfloor-1=1-1=0$$

  $$s=\frac{0\times1}{2}=0$$

  $$r=2-s=2-0=2$$

  So, the maximum number of positive integers that we can sum is ***one*** (that's our k) which is 2

Flowchart


start
screen
"Give an integer >= 1"
keyboard
n
n < 1
max_k_partitions(n, i, k)
s ← i * (i+1) / 2
r ← n - s
k
screen
j ← 1
j ← j + 1
j <= i
j, " "
screen
r
screen
end
max_k_partitions(integer n, integer i, integer k)
i ← floor((sqrt(1 + 8 * n) - 1) / 2)
k ← i
i ← i - 1
return i, k

C++ code

```
#include <bits/stdc++.h>

typedef unsigned long long int  ulli;

// Calculate the maximum number (k) such that (n) can be represented
// as a sum of k pairwise distinct positive integers.
// Time complexity: O(1)
void max_k_partition(ulli n, ulli & i, ulli & k) {
    // Find (i), such as (i) is the largest positive number...
    i = floor((sqrt(1 + 8 * n) - 1) / 2);

    // Compute k
    k = i;

    // ... that i < r
    i--;
}
```

```
int main() {
    ulli n;
    do {
        std::cout << "n (>=1): ";
        std::cin >> n;
    }while (n < 1);

    ulli i, k;
    max_k_partition(n, i, k);

    // Compute the sum (1..i) (s)
    ulli s = i * (i + 1) / 2;

    // Compute the remainder (r)
    ulli r = n - s;

    // Print k
    std::cout << k << '\n';

    // Print all numbers from 1 to i
    for (ulli j = 1 ; j <= i ; ++j)
        std::cout << j << ' ';

    // Print r
    std::cout << r;

    std::cout << '\n';

    return 0;
}
```

Other solution

I found a full description of another solution here:

https://medium.com/competitive/pairwise-distinct-summands-9ef4e8686b17