Generalized algorithm

- arguments:
  - value to partition
  - max partition summands
  - set of partition summands
- initialization
  - set the number of summands - n sub s
    - value of partition if 0
    - else max partition summands
  - create array of values for summands
    - set of partition summands as argument
    - otherwise 1 ... num summands otherwise
  - initialize coefficients array of size n sub s to 0
  - initialize remainder array of size n sub s to 0
    - set `remainder[n sub s - 1] = x`
    - set `remainder[n sub s] = x`
  - store the value to partition, x
  - initialize count to 0
  - set level, l, to n sub s - 1
  - set coefficient temporary, c sub t, to 0
  - set remainder temporary, r sub i, to x

```
x = n
ns = num_summands ? num_summands else x
summands = [1 ... n]
coeff = [0 ... 0]
remainder = [0 ... x , x]
count = 0
level = ns - 1
c sub t = 0
r sub i = x
```

1. if the current level is greater than or equal to the number of summands return the number of summands

```
if level is >= ns return ns
```

1. set the remainder at level i to r sub i (the value to partition to start) set the coefficient at level i to c sub i (0 to start)

```
remainder[level] = r sub i // x to start
coeff[level] = c sub i // 0 to start
```

1. if the current level is not 0

    - if the remainder at the current level (index of last summand to start) is greater than 0
        - set coefficient at index level - 1 to 0
        - set remainder at level - 1 = remainder at level
        - decrement level
        - repeat 3.

2. if the current level is 0

    - if remainder at current level is not 0
        - calculate rmdr / summand for level, set to var d (number of times summand goes into remainder)
        - rmdr at level = rmdr at level - d * summand at level (remainder is summand X (floor (remainder / summand))) i.e. r - (floor(r / x))
        - coefficient at level = d (i.e. remainder / summand)

3. if remainder at current level is 0

    - increment count
    - go to ... with level

4. increment level

    - if level is >= number of summands go to ... with level number of summands

5. remainder at level equals remainder - value (i.e. reduce remainder by on value ot current index

    - increment coefficient at level
    - go to 3.

6. level equals current level (from 5.) or number of summands (from 6.)

    - set coefficients and remainders up to current level equal to 0
    - increment level
    - r sub i = remainder at level - value at level
    - c sub i = coefficient at level + 1

- if the current level is less than the number to partition go to 1.