# DOCUMENTO TÉCNICO

**Integrantes:**

* Diego Cardenas
* Zayra Gutiérrez
* Felipe Calvache

# Solución al Algoritmo PowerSum

# Requisitos

## Especificación

## Resumen del problema

El problema se basa encontrar todas las combinaciones posibles que tiene un numero sea X, la combinaciones se basan en la sumatoria de n numero elevado a la potencia K el cual tiene que dar exactamente al dado, ejemplo Para las combinaciones para hallar 10 a una potencia de 2 seria 12+32 La suma de las variables tiene que dar exactamente X entonces 1 + 9 = 10 por tanto es una combinación y esta es la única posible, por esto el algoritmo retorna la cantidad de combinaciones que satisfacen exactamente a X.

## Entrada:

La primera línea contiene un número entero el cual sería X.

La segunda línea contiene un número entero el cual sería N.

## Salida:

Retorna la cantidad e posibles combinaciones que satisfacen a X

# Diseño

## Estrategia

La solución más optima fue usando recursividad, lo cual comienza primero analizando los datos que me dan y encontrar la solución, debido a esto solicitamos los datas con la entrada estándar y la función PowerSum recibe los parámetros x,k,n a los que corresponde a el valor de x, la potencia n-sima, la cantidad de combinaciones encontradas, se crea una variables la cual es pnum lo que corresponde a la variable n elevada a la potencia k después compara caso por caso si ejemplo x es igual a 0 retorna 1 debido a que solo hay una combinación que lo satisface después si el numero x ingresado es menor a 0 entonces no hay combinación que lo satisfaga en los reales y retorna 0, al final si no se cumple ningún caso entonces retorna la función PowerSum con los parámetros x- pnum y la constante k y se le suma a n + 1 y todo esto se le suma recurrentemente a PowerSum con los parámetros x y la constante k y se le suma a n + 1 y al final devuelve toda la cantidad de posibilidades posibles que satisfagan X.

## Algoritmo

Se realiza un algoritmo recursivo el cual es el mas optimo para este problema

from sys import stdin  
def PowerSum(x,k,n): # costos pasos  
 pnum = n\*\*k # 1 1  
 if x == 0: # 1 n  
 return 1 # 1 n  
 if x < 0 or pnum > x: # 1 n  
 return 0 # 1 n  
 return PowerSum(x-pnum, k, n+1) + PowerSum(x, k, n+1) # 1 1  
  
def main(): # costos pasos  
 line = stdin.readline().strip() # 1 1  
 while line: # 1 n  
 x = int(line) # 1 n-1  
 k = int(stdin.readline().strip()) # 1 n-1  
 print(PowerSum(x, k, 1)) # 1 n-1  
 line = stdin.readline().strip() # 1 n-1  
if \_\_name\_\_ == '\_\_main\_\_':  
 main()

## Invariantes

### Invariante #1:

Devuelve la cantidad de combinaciones posibles de X elevado a K

* Iniciación: sin problemas y de manera optima el cual no pretende ser complejos.
* Estabilidad: Procedural mente no posee problemas debido a que posee lo establecido para cada caso.
* Terminación: termina cuando supera la cantidad de casos.

### Invariante #2

Recibe los datos e imprime el resultado

* Iniciación: Recibe dos datos tanto x como k
* Estabilidad: debido a que estos datos se envían directamente a PowerSum solamente. imprime el resultado de PowerSum.
* terminación: Termina cuando se ingresa algo vacío.

## Casos prueba

|  |  |  |
| --- | --- | --- |
| Entrada | Justificación | Salida |
| 30  1 | Verificar la respuesta con x = 30 y n = 1 | 296 |
| 40  4 | Verificar la respuesta con x = 40 y n = 4 | 0 |
| 65  3 | Verificar la respuesta con x = 65 y n = 3 | 1 |
| 0  0 | Verificar la respuesta cuando los dos valores son 0 | 1 |

## Fuentes

/Arena\_2\_c