PARCIAL PRACTICO

PRESENTADO POR:

JUAN CAMILO BAZURTO ARIAS

PRESENTADO A:

SEBASTIAN CAMILO MARTINEZ REYES

ESCUELA COLOMBIANA DE INGENIERÍA JULIO GARAVITO

ALGORITMOS Y ESTRUCTURAS DE DATOS

PROGRAMA DE INGENIERÍA DE SISTEMAS

BOGOTÁ D.C.

2021 – 1

1. **Contexto**

A. ***Power Sum:*** Hallar el número de formas en que un número entero dado, ***X***, puede expresarse como la suma n de las potencias de números naturales únicos.

B. ***Problema de Sumas:*** Dada una cadena s formada únicamente por dígitos, es posible insertar el signo de la suma cero o más veces entre los dígitos de s para construir una expresión. Por ejemplo, para la cadena s= 12345 es posible insertar los símbolos de diferentes maneras generando las siguientes expresiones:

1 + 2 + 3 + 4 + 5 evalúa a 10

12 + 3 + 4 + 5 evalúa a 24

12 + 34 + 5 evalúa a 41

123 + 4 + 5 evalúa a 132, entre otras.

Dado un numero n estamos interesados en encontrar cual es la mínima cantidad de símbolos de la suma que deben ser insertados en la cadena s para que evalúa a n. Están permitidas aquellas expresiones que contengan números con ceros a la izquierda por ejemplo 1 + 01, por supuesto cadenas como las siguientes no están permitidas 1++1 o +1+1.

C. ***Cupido:*** En Britania, las parejas reales tienen una prohibición, la altura de la princesa debe ser estrictamente menor a la del príncipe. Esto quiere decir que una mujer no se puede casar con un hombre mas bajo o de la misma estatura que ella. Adicionalmente a esto, la tradición dice que entre mas cercana sea la altura de la pareja de esposos más felices serán en su vida matrimonia. Al príncipe Lolouch no le gustan esas reglas , pero mientras su revolución toma fuerza, debes ayudar al príncipe a determinar cual es la mejor pareja para él.

D. ***Ultra Quick-Sort:*** El siguiente algoritmo procesa una secuencia de n enteros distintos, intercambiando dos secuencias adyacentes de elementos hasta que la secuencia se encuentra ordenada de forma ascendente. Para la secuencia de enteros: 9 1 0 5 4 Ultra Quick-Sort produce la salida 0 1 4 5 9. Su tarea es determinar cuantas operaciones de intercambio Ultra Quick-Sort necesita para ordenar una secuencia dada.

2. Requisitos

2.1. Especificación

2.1.1. Entrada

A. Esta función recibe como parámetro de entrada un entero ***X*** el cual es mayor o igual que 1 y menor o igual que 1000, como segundo parámetro de entrada un entero ***n*** el cual debe ser mayor o igual que 2 y menor o igual que 10.

B. La primera línea de la entrada contiene ´únicamente un número T que representa el número de casos de prueba. Cada una de las siguientes T líneas contiene una cadena s no vacía, compuesta de dígitos minúsculas (’0’-’9’) , cuya longitud es menor o igual a 10,seguida de un espacio y el número n ($0 \leq 100$)

C. La primera línea es la cantidad de princesas, la siguiente línea tendrá N valores que son las alturas de las princesas. Después viene una nueva línea con un número Q, este valor son la cantidad de consulta s que se harán. En la siguiente línea habrá Q números, donde cada uno es una altura hipotética del príncipe Lolouch.

D. La entrada consiste en varios casos de prueba. Cada caso comienza con una línea que contiene un entero n < 500,000 . Cada una de las n lineas siguientes tendrá un entero a, que corresponderá al K − esimo elemento de la secuencia. La entrada termina cuando n = 0 (Esta secuencia no debe ser procesada).

2.1.2. Salida

A. Esta función retorna un único número entero, el cual es el número de combinaciones posibles calculadas.

B. Por cada cadena s de la entrada, imprima la mínima cantidad de símbolos ‘+’(comillas por claridad) que deben se r insertados en la expresión para que esta evalúe a n, en caso de que sea imposible imprima -1.

C. Para cada consulta Q imprimir la altura de la princesa más alta que mide menos que Lolouch, si no existe una princesa con esa particularidad , imprimía X.

D. Por cada secuencia dada, aplique el algoritmo de Merge-Sort para encontrar el m´ını m on´umero de intercambi o s necesarios para orden ar l a secue n ci a .Imprima el m´ı n i m o número de intercambios.

3. Diseño

3.1. Estrategia

3.1.1. Descripción general

A. Para resolver este problema utilizamos una estrategia incremental donde planteamos un ***numN*** el cual es la potencia de ***k*** y ***n***, donde ***k*** es un entero que inicia desde 1. Se plantean los casos base cuando ***X*** sea igual a ***numN***, donde la función retornaría 1, dado que sea calculado una forma de poder expresar ***X*** en potencias de $\boldsymbol{k^n}$, para el segundo caso base se tiene que, si ***numN*** es mayor a ***X***, entonces no hay forma de expresar a ***X*** en potencias, por lo que retornamos 0. Para el caso inductivo planteamos la suma de el llamado de dos ***powerSum***, por la izquierda tenemos una función que recibe como parámetros ***X – numN, n, k+1***, y por la derecha una función que recibe como paramtros ***X, n, k+1,*** esto dado que tenemos que calcular las potencias cuando tome y no tome a ***X***.

B. Se plantea el caso base cuando total es igual a n, total es la suma total de los elementos en la secuencia. Para el caso inductivo cuando total es mayor a n, aquí total va a ser la suma de los elementos de la secuencia eliminando el ultimo elemento de la secuencia mientras que total sea mayor a n.

C. Se plantea el caso base cuando la cantidad de princesas es igual a cero, aquí retornamos menos infinito dado que ya se han validado todas las princesas. Para el caso inductivo se plantea que cuando la princesa de la mitad de la lista es mayor o igual en altura a la del príncipe quiere decir que las princesas que son válidas están hacia la izquierda de la lista, si la princesa de la mitad es de menor altura a la del príncipe esto quiere decir que la princesa con mayor altura, pero menor a la del príncipe esta hacia la derecha de la princesa de la mitad.

D. Para resolver el problema inicialice una variable global ***contador*** igual a cero, según el código de merge-sort cada vez que s1_element > s2_element, se agregara s2_element al resultado, esto quiere decir que la diferencia entre la longitud de s1 y s1_index, es el numero de "desplazamientos" que he hecho.

3.2. Casos de Prueba

A.

Como casos de prueba se tomaron los siguientes casos:

- 10
  2
- 100
  2
- 100
  3

Los resultados de estos casos fueron los siguientes:

- 1
- 3
- 1

B.

Como casos de prueba se tomaron los siguientes casos:

- 5
  99999 45
  1110 3
  0123456789 48
  99999 1000
  382834 100

Los resultados de estos casos fueron los siguientes:

- 4
  3
  8

-1

2

C.

Como casos de prueba se tomaron los siguientes casos:

- 5

  1 2 3 4 5
- 2

  4 1

Los resultados de estos casos fueron los siguientes:

- 3
- X

D.

Como casos de prueba se tomaron los siguientes casos:

- 5

  9

  1

  0

  5

  4

- 3

  1

  2

  3

Los resultados de estos casos fueron los siguientes:

- 6
- 0

4. Análisis

4.1. Temporal

A. Esta función tiene un costo de O(n^k): exponencial, dado que se evaluarán los números dos veces cuando se tomen y cuando no.
B. Esta función tiene un costo de O(n): lineal, dado que en el peor de los casos la secuencia o el numero sean muy grandes.
C. Esta función tiene un costo de O(log n): logarítmica, dado que, dada una secuencia va a validar la mitad y luego la mitad de la mitad, y así hasta llegar a que la longitud de la secuencia sea 0, esto quiere decir que será n dividido 2^k, es decir, log(n).
D. Esta función tiene un costo de O(nlog(n)):, dado que tiene la misma complejidad de merge-sort.