LABORATORIO # 2

PRESENTADO POR:

MIGUEL ANGEL SALAMANCA

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**
2. 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.
3. Los problemas en Ciencias de la Computación a menudo se clasifican como pertenecientes a una determinada clase de problemas (por ejemplo, NP, Insoluble, Recursivo). En este problema se analizará una propiedad de un algoritmo cuya clasificación no se conoce para todas las entradas posibles.

Considere el siguiente algoritmo:

1. entrada n

2. imprimir n

3. si n = 1 entonces STOP

4. si n es impar entonces n ←- 3n + 1

5. si no n ←- n/2

6. GOTO 2

1. Definimos el superDigit de un número entero ***X*** utilizando las siguientes reglas: Dado un número entero, tenemos que encontrar el superDigit del número entero.

Si ***X*** sólo tiene 1 dígito, entonces su superDigit es. En caso contrario, el superDigit de ***X*** es igual al superDigit de la suma de los dígitos de ***X***.

1. **Requisitos**
   1. Especificación
      1. Entrada
2. 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.
3. Esta función recibe como parámetros de entrada una serie de pares de enteros i y j. Todos los enteros serán menores que 10.000 y mayores que 0. Se puede asumir que ninguna operación desborda un entero de 32 bits.
4. Esta función recibe como parámetros de entrada una serie de pares de enteros ***n*** y ***k***. ***n*** debe ser mayor igual que 1 y menor igual que y k debe ser mayor igual que 1 y menor igual que .
   * 1. Salida
5. Esta función retorna un único número entero, el cual es el número de combinaciones posibles calculadas.
6. Esta función retorna i, j y la longitud máxima del ciclo, estos tres números están separados por al menos un espacio. Los enteros i y j aparecen en la salida en el mismo orden en que aparecieron en la entrada.
7. Esta función retorna el superDigit de ***p***, donde ***p*** se crea como se ha descrito anteriormente (ver **1. Contexto. C.**).
8. **Diseño**
   1. Estrategia
      1. Descripción general
9. 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 , 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***.
10. Para resolver este problema utilizamos estrategia incremental, planteamos el caso base cuando n es igual a 1, para los casos inductivos plantamos que cuando n mod 2 es igual a 0, entonces retorne la suma entre 1 y el llamado de la función con un nuevo parámetro que es n//2, sino retorne la suma entre 1 y el llamado de la función con un nuevo parámetro que es 3\*n+1.
11. Para resolver este problema utilizamos estrategia incremental, planteamos a una variable ***suma*** la cual es la suma de los dígitos de una cadena de números multiplicada por ***k***, dado que la cadena de dígitos se repite ***k*** veces. Para el caso base es cuando ***suma*** tiene solo un digito y retornamos esa ***suma***. Para el caso inductivo retornamos el llamado de la función con los parámetros ***suma*** casteada a *string* y 1.
    1. Casos de Prueba

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

Como casos de prueba se tomaron los siguientes casos:

* 1 10
* 100 200
* 201 210
* 900 1000

Los resultados de estos casos fueron los siguientes:

* 1 10 20
* 100 200 125
* 201 210 89
* 900 1000 174

Como casos de prueba se tomaron los siguientes casos:

* 148 3
* 9875 4
* 123 3

Los resultados de estos casos fueron los siguientes:

* 3
* 8
* 9

1. **Análisis**
   1. Temporal
2. Esta función tiene un costo de O(): exponencial, dado que se evaluarán los números dos veces cuando se tomen y cuando no.
3. Esta función tiene un costo de O(n): lineal, dado que se evaluarán número en un intervalo de i hasta j, es decir, n números de i hasta j.
4. Esta función tiene un costo de O(n): lineal, dado que la llamada recursiva no va a tener varios llamados recursivos.