**Laboratorio S04. AYED-02**

Jefer Alexis González Romero

**DOCUMENTO TÉCNICO PROBLEMA A**

**Requisitos**

**Especificación**

**Entrada:**

The input will consist of a series of pairs of integers i and j, one pair of integers per line. All integers will be less than 10,000 and greater than 0. You should process all pairs of integers and for each pair determine the maximum cycle length over all integers between and including i and j. You can assume that no operation overflows a 32-bit integer.

**Salida:**

For each pair of input integers i and j you should output i, j, and the maximum cycle length for integers between and including i and j. These three numbers should be separated by at least one space with all three numbers on one line and with one line of output for each line of input. The integers i and j must appear in the output in the same order in which they appeared in the input and should be followed by the maximum cycle length (on the same line).

**Diseño**

**Estrategia**

Se hace una función que encuentre la duración del ciclo dado un número entero. En otra función se llamará a la descrita anteriormente y si la duración del ciclo obtenida es mayor a la que estaba guardada como mayor entonces se reemplaza, esto se hace hasta que se evalúen todos los números ente i y j. Al final se imprime el número que corresponde a la máxima duración del ciclo que se obtuvo entre el rango de número dados.

Adjunto el programa con nombre “A”.

**Invariante #1 (ver código)**

Por cada valor que está en el ciclo del número entero se suma uno a la variable length.

* Iniciación: Numero al cual se le quiere hallar la duración de su ciclo, length es igual a 1.
* Estabilidad: El número es mayor a uno.
* Terminación: El número es menor o igual a 1.

**Invariante #2 (ver código)**

Por cada número en el rango dado se le encuentra su duración de ciclo y se evalúa con la mayor obtenida hasta ese momento.

* Iniciación: la variable mayor es igual a menos infinito
* Estabilidad: Por cada número dentro del rango se encuentra su duración y se evalúa.
* Terminación: El número es igual al entero j.

**Casos prueba**

|  |  |  |
| --- | --- | --- |
| Entrada | Justificación | Salida |
| 14 14 | i = j | 14 14 18 |
| 23 19 | i > j | 23 19 21 |
| 134 561 | Números grandes | 134 561 144 |

**Análisis**

**Temporal**

En el mejor de los casos T(n) = Ω(1)

En el peor de los casos T(n) = O(n2)

**Código**

**Documentación**

Dentro del código

**DOCUMENTO TÉCNICO PROBLEMA B**

**Requisitos**

**Especificación**

**Entrada:**

The first line contains two space separated integers, ***n*** and ***k***.

**Salida:**

Return the super digit of ***p***, where ***p*** is created as described above.

**Diseño**

**Estrategia**

Se concatena el número n consigo mismo k veces, al número obtenido se le halla el super digito (el cual se consigue sumando todos los dígitos que tiene el número) y si este tiene más de un digito se le vuelve hallar el super dígito, se hace recursivamente hasta que el resultado solo tenga un dígito.

Adjunto el programa con nombre “B”.

**Invariante**

Cada digito en el número se suma a un contador.

* Iniciación: Primer dígito del número, contador es igual a 0.
* Estabilidad: Por cada dígito dentro del número se suma el dígito al contador.
* Terminación: Se sumaron todos los dígitos dentro del número.

**Casos prueba**

|  |  |  |
| --- | --- | --- |
| Entrada | Justificación | Salida |
| 237 0 | k = 0  concatenar el número cero veces | 0 |
| 5 1 | Concatenar un número con solo un dígito una vez. | 5 |
| 0 15 | n = 0 | 0 |
| 476 973 | Número grandes | 8 |

**Análisis**

**Temporal**

En el mejor de los casos T(n) = Ω(1)

En el peor de los casos T(n) = O(n)

**Código**

**Documentación**

Dentro del código

**DOCUMENTO TÉCNICO PROBLEMA C**

**Requisitos**

**Especificación**

**Entrada:**

The first line contains an integer ***X***.

The second line contains an integer ***N***.

**Salida:**

Output a single integer, the number of possible combinations calculated.

**Diseño**

**Estrategia**

En una función recursiva que recibe los enteros X y N, se toman las diferentes configuraciones de sumas de números (que empiezan desde 1) con exponente N, cada configuración se detiene hasta que el resultado de la suma sea igual o mayor a X, si se da el primer caso se retorna uno, este significa que hay una nueva posibilidad, si se da el segundo se retorna 0 debido a que esa configuración no da el número dado X. Añl final se devuelve el total de posibilidades.

Adjunto el programa con nombre “C”.

**Casos prueba**

|  |  |  |
| --- | --- | --- |
| Entrada | Justificación | Salida |
| 1  5 | X = 1  Mínimo número para X | 1 |
| 46  10 | N = 10  Máximo número para N dado por el problema | 0 |
| 1000  2 | X = 1000  Máximo número para X dado por el problema | 1269 |

**Análisis**

**Temporal**

En el mejor de los casos T(n) = Ω(1)

**Código**

**Documentación**

Dentro del código

**DOCUMENTO TÉCNICO PROBLEMA D**

**Requisitos**

**Especificación**

**Entrada:**

The input will consist of a set of numbers between 0 and 30000 inclusive, one per line in the input file. The input must be read from standard input.

**Salida:**

The output will consist of the appropriate statement from the selection below on a single line in the output file for each input value. The number m is the number your program computes, n is the input value. There are m ways to produce n cents change. There is only 1 way to produce n cents change. The output must be written to standard output.

**Diseño**

**Estrategia**

En una función recursiva que recibe un entero X que representa el cambio que se debe dar, se toman las diferentes configuraciones de suma de las monedas de EE.UU, cada configuración se detiene hasta que el resultado de la suma sea igual o mayor a X, si se da el primer caso se retorna uno, este significa que hay una nueva combinación, si se da el segundo se retorna 0 debido a que esa configuración no da el número nado X. Al final se devuelve el total de caminos que hay para dar de cambio X.

Se usa la estructura de datos diccionario para evitar el recalculo, en esta se va guardando la llave que es el cambio junto con un índice de la lista de monedas y el valor es las posibles combinaciones.

Adjunto el programa con nombre “D”.

**Casos prueba**

|  |  |  |
| --- | --- | --- |
| Entrada | Justificación | Salida |
| 0 | No se debe dar nada de cambio | 1 |
| 30000 | Máximo cambio que da el problema | 543427145501 |
| 1 | El cambio es igual a la moneda de menor valor de EE.UU | 1 |

**Análisis**

**Temporal**

En el mejor de los casos T(n) = Ω(1)

**Código**

**Documentación**

Dentro del código.