Reporte Proyecto parte 3 Dalgo

0.identificación

Andrés Fernando Galvis

Josué Ladino Rivera-201914138

Luimarco Daniel Santiago Carrascal Díaz-201620630

1. Algoritmo de solución:

Lo primero que se realizo fue un algoritmo de recursivo que encontrara y probara todos los caminos posibles en la recta de rango 0 a *m,* para luego comparar con casos más grandes en la solución eficiente para estar completamente seguros del funcionamiento de esta.

Posteriormente se pensó una solución tipo iterativa que partía de una lista inicial en la que se calculaban todos los múltiplos de k menores o iguales a m y partir de esa lista ese calculaba todos los posibles valores sucesores de estos, de manera que solo los que fuera menores a m, se añadían a una lista temporal, y los que eran iguales a m indicaban un camino que se le añadía al contador. Posteriormente se copiaba el contenido de la lista temporal en la lista auxiliar inicial y la lista temporal se limpiaba y el proceso se repite hasta que la lista inicial se encuentre vacía, es decir que no haya más caminos. Sin embargo, aunque la solución fuera correcta, se expandía de manera horizontal de manera exponencial y generaba errores de memoria debido a que tenía en cuenta valores repetidos de caminos posibles distintos.

Luego se intentó adaptar la solución de manera que solo se iterara por el k dentro de una lista de adyacencias para para llevar control de los valores repetidos y generar un árbol de caminos que convergía en un nodo m, pero el problema fue que la navegación y generación aumentaban la complejidad de manera exponencial al intentar llamar a los caminos de manera recursivas.

Nuestro algoritmo final es de tipo iterativo, al igual que el primer algoritmo lo primero que hace es inicializar una lista donde se van a ir guardando los números de movimiento posibles, pero la diferencia es que adicionalmente se crea un diccionario para realizar un control de los números que se van a ir repitiendo en diferentes caminos para realizar un manejo de la memoria más eficiente y solo tener que realizar operaciones posteriores una vez por cada iteración que varias veces por iteración con creaciones de caminos muertos.

Luego de la inicialización de la lista y del diccionario de repeticiones iniciales, se va a empezar a iterar por el número de movimientos realizados, dentro de esta iteración se van a evaluar las posiciones dentro de la lista y se van a empezar a generar los movimientos posibles que van a ir siendo añadidos si no son repetidos en una lista temporal y se van a añadir a un diccionario de repeticiones temporal de la itera y cuando se repiten se van a realizar la operación de modificar el valor de la llave del elemento repetido dentro del diccionario de manera que se tiene en cuenta de cuantas formas se va a poder llegar a ese punto desde el elemento actual que se está evaluando teniendo en cuenta sus repeticiones a partir del diccionario anterior. Luego de realizar la evaluación de los elementos en la lista inicial de la iteración se va a realizar una suma de las repeticiones de m en diccionario inicial y el temporal, se van a cambiar los contenidos de la lista y diccionario inicial con los contenidos de los temporales generados en

el ciclo. La iteración a partir de los movimientos realizados terminara cuando ya no se encuentren más continuaciones posibles. Y al final en las repeticiones de m en el diccionario inicial, indican cuantas maneras se pueden llegar a m desde el punto inicial de la ejecución.

Adicionalmente la razón por la que se usan diccionarios y no listas para llevar el control de las repeticiones de valores es porque para la búsqueda de un elemento es O (1) si existe dentro del diccionario, si se realizara en una lista se debería recorrer la lista múltiples veces para realizar una sola evaluación dentro de una iteración. Y en la suma de repeticiones de m en los diccionarios se emplea el uso de la operación módulo de 998244353 debido a que, para los casos de los valores m más grandes con divisores k propiamente pequeños a veces se generan números extremadamente de caminos, por lo que se realiza modulo en las sumas para que no se desborde y no genere error de memoria.

2.Análisis de complejidades espacial y temporal:

El algoritmo de solución se compone de 2 fases, la inicialización y la transformación.

La inicialización es un ciclo dentro de un rango de 1 a m+1, donde se generan todos los múltiplos de k que son menores o iguales a m, por lo que su complejidad siempre será O(m/k).

La transformación consiste en un ciclo que itera sobre el número de movimientos y si la lista está vacía, dentro de este ciclo se van a realizar evaluaciones de los múltiplos de k actual a partir de los elementos de la lista inicial, luego de evaluar se copian los valores de la lista y diccionario temporal en las estructuras de datos iniciales y se limpian los temporales, de manera que se pasa a realizar las evaluaciones con el siguiente k.

Luego de ver el comportamiento se puede establecer que la iteración sobre k nunca será mayor a la mitad del número m, por lo que se puede tomar a m/2 como cota superior

Las evaluaciones se dan de acuerdo a la longitud de la lista inicial, es decir se ejecutará m/k veces, cada evaluación, no agregará más de m/k elementos al arreglo temporal, que luego reemplazará a la inicial.

Y al finalizar la evaluación sobre la lista, se va a copiar y luego limpiar el arreglo y el diccionario temporal, para poder realizar la siguiente iteración de acuerdo a k actual. Las operaciones de copiar y borrar de una lista o un diccionario tienen un costo igual a la longitud de lo que quieran copiar o borrar.

Entonces de acuerdo a esto se puede armar la siguiente formula de complejidad:

$$SolIterDp(m,k) = inicilizacion + iterk(eval\_lista(eval\_valores) + 4copia + 4vaciar)$$

$$SolIterDP(m,k) = \frac{m}{k} + \frac{m}{2}\left(\frac{m}{k}\left(\frac{m}{k}\right) + 4\frac{m}{k} + 4\frac{m}{k}\right)$$

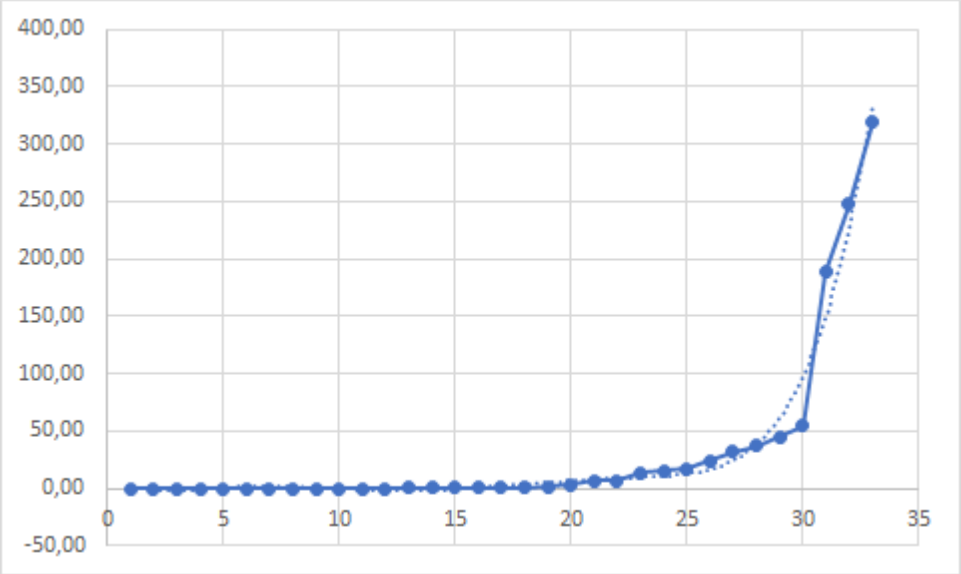
$$SolIterDP(m,k) = \frac{m}{k} + \frac{m}{2}\left(\frac{m^2}{k^2} + 8\frac{m}{k}\right)$$

$$SolIterDP(m,k) = \frac{m}{k} + \frac{m^3}{2k^2} + 4\frac{m^2}{k}$$

$$SolIterDP(m,k) = \frac{m^3}{2k^2} + 4\frac{m^2}{k} + \frac{m}{k}$$

Entonces se puede decir que la complejidad es $O(m^3/k^2)$

Se realizaron pruebas no solo con casos dados por el profesor, sino que también se realizaron 100 casos aleatorios, donde se fue incrementando el m, pero el k se mantuvo constante y con los resultados de pudo observar la siguiente tendencia:



3. Respuestas a los escenarios de comprensión de problemas algorítmicos:

i) que nuevos retos presupone este nuevo escenario -si aplica-?

(ii)que cambios -si aplica- le tendría que realizar a su solución para que se adapte a este nuevo escenario?

ESCENARIO 1: Suponga que tiene un número limitado de movimientos t para llegar a un valor determinado m.

(i)El reto que establece este escenario es establecer el número limitado de pasos t como una restricción en la ejecución de las trasformaciones sin que se generen errores

(ii) En nuestra solución, se realizaría un cambio en el ciclo de transformaciones a partir de la inicialización del arreglo, para que no ejecute hasta que pueda si no que se ejecute t-1 veces, ya que la inicialización cuenta como 1 movimiento.

ESCENARIO 2: El robot no parte de cero si no de un número p>0 primo.

i) El reto dado sería verificar que el primo dado sea primo y de ahí empezar el ciclo de inicialización del ciclo de manera que todos los múltiplos de *k estén entre* el rango del *p* dado y el numero *m*

(ii) Se realizarían 2 cambios, antes de iniciar todo el resto del proceso se va a verificar que el número dado *p* es primo y en caso de que se cumpla, el segundo cambio se realiza en el rango del ciclo que inicia la lista, de manera que genere los múltiplos de *k* tal que sean menores o iguales a *m* de manera que también sean mayores o iguales a *p*. El resto del funcionamiento sería el mismo.