Diseño y Análisis de Algoritmos: Solución problema C del proyecto

Jaime Carvajal M 201632567, Nicole Bahamon M 201629594

16 de diciembre de 2020

Algoritmo de solución

El problema C es una versión más compleja de un problema al que uno de nuestros integrantes se enfrentó en uno de los exámenes de esta materia, por lo cual sabíamos que se debía llevar a cabo de manera dinámica.
Nuestro algoritmo se puede dividir en dos partes: La inicialización de las variables y la llenada de los arreglos, con una "tercera"parte siendo la entrega de la respuesta. A continuación se muestran los valores que se utilizaron organizados en una tabla.

E/S/C	Nombre	Tipo	Descripción
E	arr	Int[]	El arreglo que simboliza el camino
E	n	Int	La longitud del arreglo
E	p	Int	La cantidad de huecos que se pueden pasar para llegar a la meta
S	resp	Int	El valor de la distancia más corta para llegar desde el inicio hasta el final.
C	pasos	Int[]	La cantidad de pasos desde cada posición hasta la meta.
C	copy	Int[]	Nuevo tamaño de arreglos

La primera parte anteriormente mencionada requiere antes que se entiendan las precondiciones del problema, por lo que estas se explican a continuación.
Precondición:
p <n $\wedge n \leq 1000 \wedge arr[0]! = -1 \wedge arr[n-1] = 0$
Esta precondición nos índica que la tabla para pasar precipicios no debe ser de tamaño mayor a la longitud del arreglo, que el arreglo no puede ser mayor a 1000, que el primer valor del arreglo no puede ser un precipicio y que el último valor del arreglo debe ser la meta.
Para la primera parte, se debe llevar a cabo la inicialización de todos los valores que se denominan C en la tabla (Por creados). Esto quiere decir que se debe inicializar pasos y copy. Pasos se inicializa como un arreglo de tamaño n, donde cada casilla tiene el valor infinito guardado, con la excepción de la útlima, que guarda 0. Copy es una copia dirigida a otro punto de memoria de pasos, para poder compararlas y en un futuro usar esta comparación para revisar la completitud del código.
La postcondición de esta parte del algoritmo se explica a continuación:

copy = pasos $\wedge len(pasos) = n \wedge pasos[n-1] = 0 \wedge (\forall i | 0 \leq i < n-1 : pasos[i] = \infty$
Esta también funciona como la precondición para la segunda parte del método, en la que se llena la tabla de resps utilizando la optimización de Knuth. Se va llenando los valores de pasos con los resultados encontrados de ver qué posiciones pueden llegar directamente al cero por medio de caminar, saltar o usar el tablón, y luego qué posiciones pueden llegar a estas. A cada posición se le asigna el valor de la posición con menor valor a la que puede llegar sumado a uno. Luego de asegurar que se haya llegado al final (Si el arreglo de pasos no cambia después de que se hayan revisado todos los valores), se devuelve el valor encontrado en pasos en 0, que es la distancia mínima para llegar desde el inicio hasta el final.

Análisis de complejidades espacial y temporal

La complejidad espacial se mide teniendo en cuenta que los ints valen 4 bytes cada uno, por lo que se puede decir que, teniendo en cuenta que se crean dos arreglos de ints, ambos de dimensión n, se puede decir que la complejidad espacial tiene una ecuación de 4(n + n), lo que indica que crece de forma lineal y se puede denominar complejidad espacial lineal.
La complejidad temporal en este caso se mide teniendo en cuenta el while por fuera del for, que utiliza una función con un for dentro de ella, por lo que se puede decir que la complejidad temporal es de O(n^3)$dondeneslacantidaddeespaciosenelarreglo, yaseanabismosotrampolines.$

Comentarios finales

Este fue el algoritmo más difícil de sacar para nosotros, debido especialmente al tablón, que fue difícil de cuadrar para que funcionara sin dejar que el jugador saltara cualquier abismo en su camino. En lo que se refiere a velocidad de ejecución, el algoritmo es especialmente rápido para peticiones pequeñas, y si se piden respuestas grandes, dura un máximo de alrededor de seis segundos, lo que demuestra que no es perfecto, pero sigue teniendo bastante buen tiempo.