Diseño y Análisis de Algoritmos
Proyecto 2017-10
Problema A
Juan Sebastian Barragan – Stephannie Jimenez
201212774 201423727

1 Algoritmo de solución Para la resolución de este problema se implementó el método `r(int p, int q, int k, int[] act)` el cual realiza una k-rotación sobre el arreglo *act* dado por los parámetros asignados previamente. Como precondición se tiene que todos los parámetros no son nulos, que $0 \leq p \leq q < n$ donde n es el número de elementos en *act* y k es un número entero. Como pos-condición se obtiene un arreglo de la misma longitud que *act* después de realizar la rotación dada por p,q y k.

Esta solución se divide en tres partes principales. La primera, consiste en definir cuanto se va a rotar el arreglo. Para ello, se utilizó la fórmula dada en el enunciado, `mov= k%(q-p)`. Una vez realizado este cálculo, se prosigue a realizar una copia temporal del arreglo ingresado por parámetro, de manera que se pueda utilizar como referencia para el que se requiere calcular. Seguidamente, se realiza la rotación solamente sobre los elementos que se encuentren entre los índices p y q.

Es importante recalcar, que la solución dada es sensible a los casos donde el movimiento es hacia la derecha, con *mov* positivo o hacia la izquierda, con *mov* negativo. Este paso era necesario, puesto que el comportamiento del índice cuando excede el tamaño del arreglo a modificar varia y obtiene valores al inicio o fin. Como sólo se están teniendo en cuenta las posiciones que van a cambiar, no es necesario recorrer todo el arreglo cuando no se especifique por los parámetros p y q.

En cuanto a la lectura de los datos, las k-rotaciones se realizan a medida que se hace la lectura de las líneas. Es decir, sólo se almacena el vector intermedio entre las n k-rotaciones que se desean calcular. Una vez completadas las rotaciones, se muestra un mensaje al usuario por consola donde se encuentra el vector resultante después de todos los cálculos.

2 Análisis de complejidades espacial y temporal

2.1 Complejidad espacial

En cuanto a la complejidad espacial, se encuentra que al inicio del algoritmo es necesario guardar en un vector la secuencia de los n números. Al igual, se mantienen en tres variables temporales los valores *actuales* de p,q y k. Adicional a esto, al hacer una k-rotación se encontró necesario guardar una copia del vector con n posiciones. De esta manera, se encuentra que la complejidad espacial de la solución es de $O(n)$.

2.2 Complejidad temporal

La complejidad temporal del algoritmo realizado, está dado por el número de iteraciones que tiene que realizar para poder realizar una k-rotación. En este caso, está dado como la diferencia entre los parámetros p y q. Es decir, para poder realizar una k-rotación, es necesario recorrer todos los elementos que se encuentran dentro de los índices p y q-1. En el peor caso, es necesario realizar el cálculo para todo el vector de n elementos, $O(n)$.

Como el problema se basa en realizar un numero de m rotaciones sobre el arreglo, se encuentra que la complejidad temporal va a ser el tiempo que se demore calculando m rotaciones. De esta manera, se puede encontrar que el tiempo esperado para realizar todas las operaciones sobre el arreglo de n elementos con m rotaciones va a ser $O(m*n)$.

3 Comentarios finales

Al realizar los casos de prueba, se encuentra que el algoritmo es capaz de encontrar una solución en un tiempo menor a un segundo. Adicionalmente, se puede afirmar que el algoritmo solución no tiene una estabilidad en cuanto a su complejidad temporal. Es decir, aunque se ponga a prueba el mismo caso, los resultados del tiempo de ejecución fluctúan y no tienden a un mismo valor. Aun así, siempre se encuentran menores a un segundo.