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

1 Algoritmo de solución

La solución propuesta está dividida en tres métodos, el primero se encarga netamente en leer las líneas de entrada, dividiendo la entrada a en el carácter '='. Cuando se termina de recibir una línea este ejecuta el método alf() en donde los parámetros son las dos partes de la línea de entrada.

En el método alf() como contexto tenemos dos arreglos que contienen las palabras y las operaciones validas de cada lado de la ecuación. En este método se busca terminar con un arreglo de restricciones para cada una de las letras del alfabeto ingles donde la posición corresponde al código ASCI-65 (si son primera en palabra, si no están de primera, o no están presentes en el acertijo), adicionalmente tener un arreglo con los caracteres del acertijo. Una vez agregadas las letras si el número de letras agregadas resulta mayor a 10 se termina el programa con el mensaje de no solución. En caso contrario se hace un llamado al método bucarRespuesta().

Por último, el método buscarRespuesta() el cual es invocado una vez por alf(), como contexto se tienen de parámetros, un arreglo de restricciones para las letras (la posición corresponde al código ASCI-65) , los caracteres presentes en el acertijo y los arreglos de las cadenas de entrada de palabras de ambos lados de la igualdad y se define un límite de iteraciones. Seguido se realizan ciclos donde cada vez se instancia una posible combinación de números para cada letra presente en el acertijo considerando la restricción de no poder asignar 0 si la letra inicia palabra, como invariante se tiene que cada letra tendrá un numero distinto en cada combinación y las primeras letras de las palabras no son 0. Ya con una combinación valida se procede a evaluar la suma y la equivalencia en cada lado, como pos condición, si se encuentra solución se termina el ciclo con respuesta. De otra manera, se termina sin solución cuando ya no hay más intentos.

2 Análisis de complejidades espacial y temporal

2.1 Complejidad espacial

La complejidad espacial de la solución propuesta en función de la cantidad de letras resulta O(n) , esto ya que solo se está guardando la combinación de números que se van a probar, si no sirve sobre escribe otra posible solución en el mismo arreglo.

2.2 Complejidad temporal

Ya que se requiere recorrer todo el espacio de búsqueda para determinar si hay o no solución la complejidad temporal es todo el conjunto de posibles soluciones de n caracteres presentes en las palabras. Lo que significaría que el tamaño de busqueda en función del número de caracteres de entrada. En general, el número de posibles soluciones está dado por el número de combinaciones que se puedan hacer con los 10 caracteres. En caso de que entre un menor número de letras, sus espacios serán reemplazados por '*'.

Como se vuelve necesario generar las posibles combinaciones, para después si poder buscar la solución al problema se puede definir que la complejidad temporal está dada por generar los vectores que podrían ser solución. De esta manera, se deduce que la complejidad está dada por O(n!).

3 Comentarios finales

Las restricciones y los números de las letras se guardaron en arreglos donde su código ASCI menos 65, de esta manera se evita hacer recorridos al momento de buscar el número asignado a cada letra, cuando se va a evaluar la equivalencia se recorren las palabras la letra hace de "llave" para obtener el número asignado. Adicional a esto se consideró tener un arreglo con las potencias de 10 que fueran a usarse para el cálculo de la suma (solo se soportan palabras de hasta 10 caracteres), estas decisiones se tomaron pensando en tener un mejor desempeño al hacer uso de los arreglos.

Este algoritmo propuesto es una aproximación de fuerza bruta que solo evita evaluar las soluciones donde la primera letra es cero, una mejor implementación resultaría al implementar una búsqueda con algún predicado dominó que

descarte soluciones antes de evaluarlas. Sin embargo, no encontramos un predicado que permitiera realizarlo. También el método propuesto no garantiza que se busquen todas las posibles soluciones ya que no se implementó que el espacio de búsqueda fuera secuencial evitando ciclos sino por el contrario se toman valores aleatorios. En consecuencia, cuando no hay solución el programa se para estimando que el número de intentos aleatorios cubre todo el espacio de búsqueda. De igual forma, en los casos donde existe más de una solución al problema, el algoritmo devolverá la primera que encuentre. Por esta razón, no se puede garantizar que se obtenga la misma solución esperada.