Proyecto 3
Diseño de Algoritmos I

Fabio Castro 10-10132, Leopoldo Pimentel 06-40095

13/03/2015

1 Problema ACMAKER

1.1 Descripción del diseño del algoritmo implementado para la solución óptima

Para esta solucion, se guardo la posicion de la palabra que esta analizando y la posicion de la abrevicion. Tambien se tuvo en cuenta si se habia usado ya esa palabra en la abrevicion o si se habia usado ya la letra en las palabras.

Tambien, antes de correr el algoritmo, se limpio la oracion de las palabras innecesarias para el acronimo. De esta forma se pudo reducir el nivel de complejidad de los casos del algoritmo.

1.2 Descripción de la función de optimización empleada

Para este problema se uso una funcion de optimizacion Top-Down, en programacion dinamica, donde se hizo de forma recursiva todo el planteamiento del algoritmo.

1.3 Estrategia de programación dinámica seguida

Como estrategia de nuestra solucion, implementamos una matriz de tamaño M x N x 2. En dicha matriz fuimos almacenando en la posicion de la palabra y en el posicion de la oracion, guardamos cuantas formas tengo de lograr el acronimo en ese momento.

1.4 Análisis de complejidad en tiempo y espacio

Como es una matriz de tamaño M x N x 2. Donde M es el tamaño de la palabra por N el tamaño de la abreviacion. Como en el peor caso el algoritmo tiene que recorrer toda la matriz entonces, que nuestra solucion es de 0(MN2), ahora M y N son dos variables independientes no se pueden unir. Como el 2 es constante, entonces podemos asegurar que el algoritmo es de orden O(MN)

2 Problema BABY

2.1 Descripción del diseño del algoritmo implementado para la solución óptima

Nuestro algoritmo, toma la cantidad de reinas. Una vez tomado el numero de reinas, se comenzo con la solucion de colocar N reinas en un tablero. Para esto, se trabajo con la ayuda de una mascara de bits que iba trabajando con un OR logico. Con esto, sabiamos si ya se habia colocado una reina en esa columna o fila

2.2 Descripción de la función de optimización empleada

Para este problema se uso una funcion de optimizacion Top-Down, en programacion dinamica, donde se hizo de forma recursiva todo el planteamiento del algoritmo.

2.3 Estrategia de programación dinámica seguida

Para la estrategia de nuestra solucion, en una matriz de tamaño 1.En dicha matriz se almaceno una mascara de bits. Con la finalidad de poder hace un OR logico. Asi vemos donde estan las reinas ya colocadas en el tablero. Con esto, tenemos una solucion bastante optima para el problema.

2.4 Análisis de complejidad en tiempo y espacio

Para este algoritmo se uso el O($Nx2^n$). Donde N es la cantidad de reinas y 2^n es la cantidad de mascaras que tiene que calcular para saber cuantas reinas existen.

3 Problema BORW

3.1 Descripción del diseño del algoritmo implementado para la solución óptima

Para la solucion optima de este algoritmo, se uso una matriz de 3 dimensiones, donde se fue guardando las solciones parciales. Esto es debido a que plantenado el problema de una forma recursiva podemos ver que muchas de las soluciones se solapan. Gracias a este comportamiento, es que se pudo idear un algoritmo que iba almacenando las solciones parciales y se tenian a la mano en caso de ser necesarias.

3.2 Descripción de la función de optimización empleada

Para este problema se uso una funcion de optimizacion Top-Down, en programacion dinamica, donde se hizo de forma recursiva todo el planteamiento del algoritmo.

3.3 Estrategia de programación dinámica seguida

Para esta estrategia, se usa una matriz de tamaño N x N x N, donde fuimos almacenando por indice las secuencias que ibamos trabajando y se fue guardando el numero de la secuencia que se iba alcanzado.

3.4 Análisis de complejidad en tiempo y espacio

La complejidad de este algotmimo es de orden O(n^3) N es el tamaño de la entrada. Es decir la cantidad de elementos que entran. Igualmente en cuanto a espacio, se habla de una matriz de tamaño de 3 dimensiones de N

4 Problema MAXWOODS

4.1 Descripción del diseño del algoritmo implementado para la solución óptima

Para este problema, se uso un pensamiento bottom-up. En el cual, se hizo de forma lineal la solucion del problema. Para poder hacer esto, se construyo una matriz donde se fue almacenando la informacion de cuantos arboles se iban cortando hasta un punto. Esto es debido a que para la solucion parcial solo es necesario saber el nivel en el que estas y el nivel superior. Con esta informacion, tienes suficiente para correr el algoritmo.

4.2 Descripción de la función de optimización empleada

Para este algoritmo, se uso una funcion bottom-up. Esta funcion plantea el uso de un algoritmo lineal. Asi pues la solucion de nuestro algoritmo es de forma lineal.

4.3 Estrategia de programación dinámica seguida

Para la estrategia de este algoritmo, se uso una matriz de tamaño N x 2, que representa el total de columnas que tiene la matriz de entrada y otra dimension adicional para guardar el piso de arriba. En el arreglo auxiliar, fuimos guardando la maxima cantidad posibles de arboles cortados. De esta forma, siempre tuvimos bajo nuesta mira; el nivel donde estabamos y el nivel de arriba.

4.4 Análisis de complejidad en tiempo y espacio

La complejidad de este algoritmo es de orden O(n^2), esto es debido a que aunque nuestra solicion del dp es casi lineal, nuestra implementacion siempre tiene que recorrer toda la matriz dada para poder obtener la respuesta.