Diseño y Análisis de Algoritmos

Parte II – Proyecto

Integrantes:

Santiago Mora

Sergio Guillen

David Cruz

Juan Felipe Castaño

201913351

201912757

201912150

201820865

Tabla de Contenido

Algoritmo de Solución

El problema por solucionar es la búsqueda de los componentes conectados (CC) dadas unas condiciones sobre un arreglo de vértices, una permutación de n elementos, que se unen por aristas. La condición principal para la creación de una arista es $vi, v\,j\ (i < j) si\ y\ solo\ si\ pi > pj$.

Para la construcción de la solución se determinaron dos casos base principales:

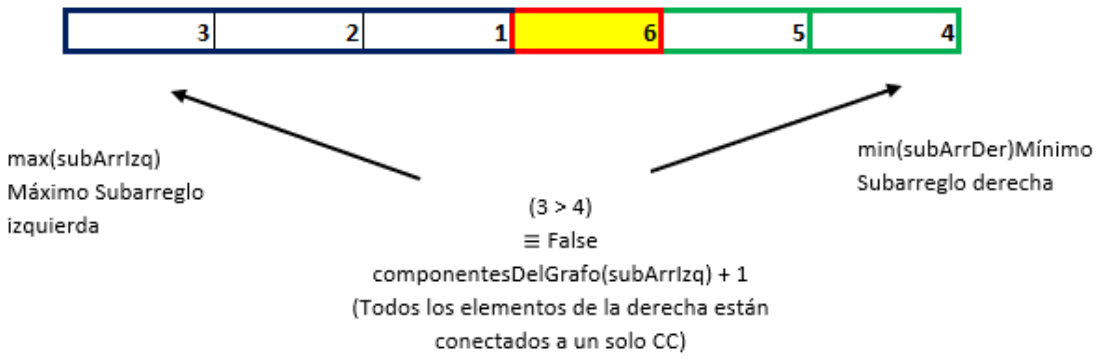
1. Sí el primer elemento del arreglo es el **máximo**, entonces todos los elementos están conectados al mismo CC. Se termina la ejecución.
2. Si el primer elemento del arreglo es el **mínimo,** entonces este elemento se encuentra separado y en un solo CC. Entonces se procede a operar otra vez sobre un subarreglo que ya no incluya este elemento porque ya pertenece a un CC.

Cuando no se cumplen estos dos casos base, se encuentra el **máximo** elemento del arreglo sobre el cual se va a dividir este arreglo en dos. Esto porque si se encuentra el elemento máximo y se divide en dos arreglos donde uno empieza por el máximo, se cumple un caso base y es un CC.

Después de hacer la división del arreglo en otros dos subArreglos, se encuentra el **máximo** del subarreglo que se encontraba antes del elemento máximo, y se encuentra el elemento **mínimo** del arreglo posterior al elemento máximo. Esta verificación es para determinar si ambos subArreglos (CC) se encuentran conectados por al menos una arista.

Si el máximo del subArreglo izquierda es mayor que el mínimo del subarreglo izquierda, todos los elementos de la izquierda están conectados al subArreglo CC de la derecha del máximo. Por el contrario, si el máximo del subarreglo izquierda es menor que el mínimo de la derecha se vuelve a ejecutar a hacer la operación en busca de un caso base solo sobre el subarreglo de la izquierda y se suma uno al total de CC, pues todos los elementos después del máximo están conectados a un solo CC.

Finalmente se retorna el número de CC encontrados cuando no se cumpla ninguna condición de las anteriormente mencionadas. Otra acercamiento que se tuvo para resolución del problema era agregar todos los elementos del arreglo en un grafo, agregar las aristas de acuerdo a la condición original del problema y por ultimo hacer DFS o BFS sobre cada nodo. Al final no se implementó ya que el acercamiento a partir de divide y vencerás era más eficiente y requería menos líneas de código



Total: 2 Componentes conectados.

El componente después del máximo (1) + El componente de la izquierda donde hay un caso base en el que el primer elemento es el máximo del arreglo (1).

Especificación

A continuación, se presentarán las entradas y salidas del algoritmo desarrollado componentesDelGrafo()

	Nombre	**Tipo**	**Descripción**
E	Arr	Array of Int	Lista de Entrada
S	cc	Int	Hace referencia a la cantidad de componentes conectadas.

Precondición

$$P\colon (\forall k \mid Arr \neq \phi \wedge k \in Arr : k \in nat)$$

Para la precondición tenemos primeramente que verificar que el arreglo sea distinto del conjunto vacío. Por otro lado, tenemos que **k** representa un elemento del arreglo ***Arr***, y este debe ser un numero natural incluyendo el 0.

Postcondición

$$R\colon (\forall k \mid k \in Arr : CC \geq 1 \wedge componentesDelGrafo(Arr) = CC)$$

Para la postcondición se tiene que se debe cumplir que para cada elemento k del arreglo de entrada Arr, el número de componentes conectado CC es mayor o igual a 1, es decir, que si existe al menos un elemento en el arreglo; el número de componentes conectados es mayor a 0 y se cumple que el número de componentes conectados devuelto sea el número de componentes conectados CC real del grafo.

Análisis de complejidad Espacial y Temporal

Concluimos que el algoritmo solución tiene una complejidad temporal de O(n), debido a que las búsquedas de los mínimos y máximos sobre el arreglo de permutaciones son llamados sobre arreglos de tamaño fijo, por lo tanto estos se pueden tomar como n*O(1) = O(1) y tenemos dos casos distintos al base uno es en el que el primer elemento del arreglo es el mínimo del arreglo por lo cual en un peor caso de un arreglo ordenado de manera ascendente tendríamos que quitar n-1 elementos para lograr que se cumpla uno de los casos base. Para el otro caso recursivo miramos los elementos a partir del pivote en el que las componentes del grafo no están conectadas al elemento máximo, por lo cual tenemos que dividir el arreglo (desde 0 hasta el pivote seleccionado) y verificar los componentes del nuevo subarreglo, esto tiene un costo de m*O(n/2) donde m es la profundidad

de la recursión, siendo m= (n-1)/2 y al hacer la suma en conjunto obtenemos algo lineal como resultado.

Para el análisis espacial se debe tener en cuenta la cantidad de elementos que utiliza el algoritmo respecto a la entrada y el espacio auxiliar que requerimos, el espacio que ocupa nuestro algoritmo es de $\frac{n}{2}$ para cada arreglo auxiliar creado para hacer la recursión unido con la recursión de n-1 elementos y por último tenemos que en el peor de los casos alguna de las recursiones o se va a repetir $\left(\frac{n}{2}\right) * m \vee n * (n-1)$ donde n es el tamaño del arreglo y m es la cantidad de veces que se repite la recursión llegando a un resultado de $n^2 * \log_m \left(\frac{n}{2}\right)$ para la complejidad espacial.

Escenarios de comprensión de problemas algorítmicos

Escenario 1

Sabemos que el algoritmo solución encuentra los componentes conectados de un grafo por lo tanto y teniendo en cuenta que nuestra implementación divide y vencerás hace la división del arreglo respecto a un pivote deberíamos agregar una arista para cada separación del arreglo, esta arista iría desde el pivote al valor siguiente del mismo ej: pivote = 6 entonces arista = (6,7) y así una vez identificamos todas las componentes conectadas del grafo las contamos y añadimos las aristas mínimas para poder dejar un solo componente conectado, por ultimo para retornar el número mínimo de aristas necesarias deberíamos retornar la cantidad de componentes conectados que existen y restarle uno, ejemplo: componente conectado A: [1,2,3], componente conectado B: [4,5,6] el número de componentes conectados es dos y siguiendo la lógica explicada arriba necesitaríamos una arista adicional de 3 a 4 para que quede un solo componente conectado

Escenario 2

Para este escenario se puede plantear la misma solución que se propuso para el problema original con algunos ligeros cambios. El primero y más importante, es el caso base donde el elemento más pequeño o mínimo del arreglo está conectado con todos los elementos siguientes. El segundo es que, si el elemento más grande o máximo del arreglo está en la posición al inicio del arreglo, este es un CC único. El tercer cambio debería ser que, para poder dividir el arreglo en izquierda y derecha, se busca el mínimo del arreglo y posteriormente el mínimo del subarreglo de la izquierda y el máximo del arreglo de la derecha. De esta forma se puede identificar si son un solo CC y la verificación de todos los Componentes y elementos del arreglo estaría completa.