Diseño y Análisis de Algoritmos: Proyecto - Parte 2

Camilo Morillo Cervantes (202015224)
Juan Sebastián Alegría (202011282)

Departamento de Ingeniería de Sistemas y Computación, Universidad de los Andes

10 de mayo de 2022

I. Algoritmo de solución.

En primer lugar, se crearán arreglos de enteros de acuerdo al número de casos de entrada, representando cada posición un componente. Al llamarse a la función solución con cada arreglo, se inicializará una variable con el número de componentes conectados y otra que marcará el mayor índice encontrado en el grafo.

Luego de esto, se usará un arreglo para almacenar arreglos de subgrafos indicando en su primera posición el componente inicial y en su segunda el componente con menor valor conectado a ese subgrafo, si no hay ningún componente conectado (un subgrafo de un solo nodo) será infinito.

Se añadirá en primer lugar el componente inicial y luego se iterará todos los demás componentes del grafo original para determinar los subcomponentes conectados de acuerdo a las siguientes reglas:
- Si el componente actual es mayor al componente con el mayor valor anterior, se creará un nuevo subgrafo, se incrementará el número de componentes conectados, y el componente actual pasará a ser el mayor índice hasta el momento.
- De lo contrario, si el componente actual es menor al conectado en el último subgrafo encontrado, se reemplazará el valor de la segunda posición del último subgrafo con el componente actual, indicando que es el componente con menor valor en ese grupo.

Para terminar, la función solución itera nuevamente pero esta vez de forma inversa sobre los subgrafos obtenidos de la primera iteración.
Esta iteración hace una comparacion entre el nodo con menor valor del subgrafo actual y el nodo mayor del componente siguiente (el cual tiene un índice menor dado que estamos iterando de forma inversa), en caso de que dicho nodo mayor del índice menor sea menor al nodo menor del índice mayor entonces dichos subgrafos se conectan y por lo tanto le restamos 1 a la variable que lleva el total de componentes conectados. Cabe notar que el nodo menor estará dentro de una variable general que llevara la cuenta de cual es el nodo menor llevado hasta el momento de la iteración por lo tanto siempre y cuando un índice menor tenga un nodo con mayor valor a un índice mayor siempre se conectaran esos dos subgrafos.
Esta última iteración se realiza con el fin de conectar subgrafos con conexiones compatibles entre sí que no eran adyacentes en un principio.

II. Análisis de complejidades.

La función solución tiene la siguiente complejidad, siendo c_1 asignaciones, c_2 comparaciones, y c_3 operaciones:

```
def grafosConectados(data):
    num, indexG = 1, 0  # c1
    subGraphs = []   s  # c1
    subGraphs.append([data[indexG], math.inf])  # c3

    for i in range(1, len(data)):  # n
```

```
        if data[i] > data[indexG]:  # c2
            subGraphs.append([data[i], math.inf])  # c3
            num = num+1  # c1 + c3
            indexG = i  # c1
        else:
            if data[i] < subGraphs[len(subGraphs)-1][1]:  # c2
                subGraphs[len(subGraphs)-1][1] = data[i]  # c1

    men = math.inf  # c1
    for i in range(len(subGraphs)-1, 0, -1):  # n
        if subGraphs[i][1] < men:  # c2
            men = subGraphs[i][1]  # c1
        if subGraphs[i-1][0] > men:  # c2
            num = num-1  # c1 + c3

    return num
```

Sumatoria de complejidades:

$T(n) = c_1 + c_1 + c_3 + n(c_2 + c_3 + c_1 + c_3 + c_1 + c_1) + n(c_2 + c_1 + c_2 + c_1 + c_3)$
$T(n) = 2c_1 + c_3 + 3nc_2 + 3nc_3 + 5nc_1$

Por lo tanto, este algoritmo tiene una complejidad O(n) donde n es el número de componentes en el grafo de entrada.

Como está función de solución se ejecuta el número de casos de entrada, entonces la complejidad total del programa es $O(n_{casos} * n_{maxComponentes})$

III. Otros escenarios.

ESCENARIO 1: Se le pide devolver como salida el número mínimo de aristas que se deben añadir para que exista un único componente conectado:

Si se necesita devolver como salida el número mínimo de aristas que se deben añadir para que solamente exista un único componente conectado, se debe restar 1 al resultado de la función planteada originalmente.

Esto dado que sin modificaciones se esta retornando el número de componentes encontrados, por lo que para conectar los componentes separados necesitaremos 1 arista por cada pareja de subgrafos.

Ejemplo:

Si se tiene como arreglo de entrada [1, 2] tendremos como resultado el componente [1] y el componente [2], por lo que la función solución inicial retornaría 2 como número de componentes, y la modificada en este escenario retornaría $2-1 = 1$ indicando que se necesita una arista puente para conectar estos componentes.

Como dificultades se tendría que necesariamente se debe incumplir la regla para crear aristas "v_i, v_j $(i < j)$ si y solo si $p_i > p_j$", porque de lo contrario no se podrían conectar casos como el ejemplo anterior.

ESCENARIO 2: Se mantiene el problema de obtener los componentes conectados, pero se cambia la instrucción de construcción de aristas así:
Una arista es creada para los vértices v_i, v_j $(i < j)$ si y solo si $p_i < p_j$:

Para este necesario, se deberá invertir la condición de creación de aristas. Por lo que al inicio, en vez de tener una variable con el mayor índice encontrado en el grafo, se tendrá una variable con el menor índice encontrado hasta el momento.

El arreglo de componentes ahora almacenará en su primera posición el mismo nodo inicial, pero en la segunda tendrá no el menor valor conectado a ese componente sino el mayor valor que se haya conectado en ese subgrafo. Si no hay ningún componente conectado, este segundo elemento tendrá el valor de -1.

Se añadirá en primer lugar el componente inicial y luego se iterará todos los demás componentes del grafo original para determinar los subcomponentes conectados de acuerdo a las siguientes reglas:
- Si el componente actual es menor al componente con el menor valor anterior, se creará un nuevo subgrafo, se incrementará el número de componentes conectados, y el componente actual pasará a ser el menor índice hasta el momento.
- De lo contrario, si el componente actual es mayor al conectado en el último subgrafo encontrado, se reemplazará el valor de la segunda posición del último subgrafo con el componente actual, indicando que es el componente con mayor valor en ese grupo.

Para terminar, la función solución itera nuevamente pero esta vez de forma inversa sobre los subgrafos obtenidos de la primera iteración.
Esta iteración hace una comparacion entre el nodo con mayor valor del subgrafo actual y el nodo menor del componente siguiente (el cual tiene un índice menor dado que estamos iterando de forma inversa), en caso de que dicho nodo menor del índice menor sea mayor al nodo mayor del índice mayor entonces dichos subgrafos se conectan y por lo tanto le restamos 1 a la variable que lleva el total de componentes conectados. Cabe notar que el nodo mayor estará dentro de una variable general que llevara la cuenta de cual es el nodo mayor llevado hasta el momento de la iteración por lo tanto siempre y cuando un índice menor tenga un nodo con menor valor a un índice mayor siempre se conectaran esos dos subgrafos.
Esta última iteración se realiza con el fin de conectar subgrafos con conexiones compatibles entre sí que no eran adyacentes en un principio.