Problema C

Ricardo Enrique González

Camilo Zambrano

201425733

201515438

1. El algoritmo cuenta con tres ciclos. El primero, es encargado de repetirse para tomar la siguiente imagen de entrada con sus tamaños y los otros dos se encargan de tomar los tamaños y la imagen para calcular la respuesta de esa imagen en cuestión. En general el primer ciclo no es tan importante, solo se ejecuta una vez para cada imagen de la entrada, en cambio, los otros dos ciclos son los que más peso tienen en la ejecución del programa y los que al final permiten calcular si la mayor cantidad de pixeles blancos seguidos se encontraban en una fila (“H” en la salida) o en una columna (“V” en la salida). A continuación, se provee el código en GCL de los dos ciclos utilizados en el cálculo de la respuesta en donde los píxeles de entrada son vistos como una matriz de m filas y n columnas.

```
[| {CTX: B[0..m,0..n]:{1∧0} , m,n:nat}

actual,maxC,maxV,i,j:= 0,0,0,0,0;

V[0..n,0..1]:=0; {vector para guardar la cantidad de 1’s en cada columna.}

{Inv I1:0<i<m ∧ 0<j<=n, b[i,j] se está procesando y es necesario procesar lo que
queda de b[0..m,0..n]}

{Cota: m-i}

do i<m ∧ j<=n ⟶ if j<n⟶ if b[i,j]=1 ⟶actual,v[j,1],j:=actual+1,v[j,1]+1,j+1;

                          [] b[i,j]=0⟶if v[j,0]<v[j,1]⟶v[j,0]:=v[j,1] fi v[j,0]:=0;

                             if maxC<actual⟶maxC:=actual fi actual,j:=0,0;

                          fi

                 [] j=n⟶if maxC<actual⟶maxC:=actual fi actual,j,i:=0,0,i+1;

                 fi

od

{R1: maxC ∧ v[0...n,0..1]:nat}

i:=0;

{Inv I2: 0<i<n, maxV:=(maxV,max(v[i,0],v[i,1]))}

{Cota: n-i}

do i<m ⟶ if maxV<v[i,0] ∧ v[i,1]<v[i,0] ⟶maxV:=v[j,0];

             [] maxV<v[i,1] ∧ v[i,0]<v[i,1] ⟶maxV:=v[j,1];
```

```
            if
od
{R2: maxV ≠0}
if maxC>=maxV ⟶respuesta:='H'+maxC;
[] maxV>=maxC ⟶respuesta:='V'+maxV;
fi
{R: respuesta = 'H'+maxC V 'V'+maxV} |]
```

Como se puede ver el primer ciclo del algoritmo recorre las filas de la imagen pixel por pixel y va guardando la en la mayor cantidad de 1s seguidos de todas las filas en la variable `maxC`. Para las columnas se crea una matriz de n columnas y dos filas que permite guardar mientras se recorren las filas los valores máximos de cada columna y el valor que se está procesando por ellas. Cuando termina el primer ciclo `maxC` contiene la fila máxima de 1s seguidos en la matriz, pero `maxV` aún sigue siendo cero, no posee el valor máximo de cada columna, es por esto que se utiliza un segundo ciclo que recorra la matriz creada como si fuera un vector buscando el valor más grande de cada columna. Antes de explicar el segundo ciclo es necesario notar que para la última fila puede ocurrir que el valor que va siendo procesado en cada una de las columnas no quede guardado dentro de la primera fila de la matriz de columnas creadas(`V[0..n,0..1]`), por esta razón el siguiente ciclo necesita revisar para cada `V[i,0..1]` cuál es el mayor entre `V[i,0]` y `V[i,1]` a la vez que revisa si los valores de estos son mayores que `maxV` , de esta forma cuando este ciclo termine `maxV` va a contener el valor de la columna con más 1s seguidos. Al final se responde con un string basado en el mayor número entre `maxC` y `maxV`.

2. El orden espacial de este algoritmo es igual a la nueva matriz creada para guardar el valor de las columnas al procesar la imagen. Por lo tanto: S(m,n) = O(n+n)=O(2n).
El orden temporal se basa directamente en los dos ciclos creados para procesar la imagen. La suma de la complejidad temporal de ambos ciclos sería igual a la complejidad temporal del algoritmo. Por lo tanto, si el primer ciclo recorre toda la imagen una vez y se puede decir que el segundo ciclo recorre la matriz como si fuera un vector (`x[0..n]`) entonces la complejidad temporal seria: T(m,n)=O(mn+n).
En resumen, las complejidades serían del ejercicio serían:

$$S(m,n) = O(2n)$$
$$T(m,n) = O(mn+n)$$

3. La solución es eficiente en cuanto a que recorre una sola vez cada una de las imágenes a procesar, sin embargo, el uso de memoria extra también presenta un peso extra en el tiempo de ejecución, si se consigue que el segundo ciclo se ejecute sin tener en cuenta la segunda fila de la matriz, es decir, si el valor máximo de cada columna siempre se encuentra en la primera fila de la matriz, el recorrido que se debe de realizar en el segundo ciclo solo tendría una condición a revizar en lugar de dos y sería un ciclo que en cada iteración ocupe menos tiempo de procesamiento.
Nota: para que aparezca en consola la respuesta a la última salida es necesario agregar un "enter" o un " " al final de la entrada.