**Proyecto Final**

Julián Andrés Bermúdez Valderrama, Juan Camilo Lara Navarro

{ja.bermudez10, jc.lara10}@uniandes.edu.co

{201519648, 201424726}

Universidad de los Andes, Bogotá, Colombia

**Problema A**

1. **Algoritmo de solución.**

Caso 1:

El problema se abordó mediante el reconocimiento y almacenamiento de las posiciones cuyo valor en el conjunto (arreglo) era igual a cero (0). Esto se hizo con el fin de facilitar la búsqueda del sub-arreglo de mayor tamaño.

Es así como, por ejemplo, para el caso del arreglo , las posiciones de los ceros apartadas en otro arreglo serían: .

De este modo, se define el rango de búsqueda, el cual es dado por la entrada . Sí se sabe que en el sub-arreglo a encontrar debe haber a lo sumo ceros, se puede llegar a un acercamiento a lo que sería el tamaño del sub-arreglo en , mediante el establecimiento de una cota mínima dada por y una cota máxima dada por en donde se cumple qué, . Inicialmente , por ende, el primer índice en será . Por otra parte, dado que es un arreglo basado en cero (zero-based, es decir que su primer elemento está en la posición ), se concluye que el índice del cero en Z, estará en .

A continuación, se muestra una gráfica para aclarar el concepto.

Con la entrada: , se tiene entonces:

, la dimensión del arreglo.

, el máximo número de ceros en un sub-arreglo.



Arreglo de entrada (Z)



Arreglo de índices de cero en Z

Ahora, se establece la cota mínima y la cota máxima:

Se recuperan los índices de en , así:



Índices para establecer búsqueda en



Rango de búsqueda básico

Si bien en el rango de búsqueda establecido se cumple con la condición del máximo número de ceros posibles , aun no se cumple con el tamaño máximo a nivel local en este punto de la ejecución.

Se necesita entonces incluir aquellos elementos distintos de cero , ubicados hacia los extremos de las cotas. Tomando la cota mínima como referencia (), se aumentará el rango del sub-arreglo moviendo a la izquierda siempre y cuando se cumpla qué ; De igual manera, tomando la cota máxima como referencia ), se aumentará el rango del sub-arreglo moviendo a la derecha siempre y cuando se cumpla qué . Así:



Máximo sub-arreglo local

Tamaño sub-arreglo: 8

El cálculo del tamaño del arreglo es dado por el último valor de menos el último valor tomado por , más uno . Teniendo entonces al mayor tamaño del sub-arreglo local como, .

Finalmente, se procede a actualizar las cotas, tanto la mínima como la máxima, avanzando su índice al siguiente cero perteneciente en (siempre y cuando sea posible).

Se repite el proceso de:

* Establecer rango y/o rango de búsqueda con base en cotas.
* Expandir rango y/o rango hacia los extremos de las cotas, para encontrar el sub-arreglo.
* Actualizar el máximo tamaño encontrado hasta el momento con un tamaño mayor.

El siguiente avance completo es:



Actualización de índices para establecer búsqueda en



Rango de búsqueda básico (Iteración 2)



Máximo sub-arreglo local (Iteración 2)

Tamaño sub-arreglo: 11

El máximo tamaño del sub-arreglo, es dado después de

* El Avance de las cotas.
* La expansión del rango.
* La actualización del tamaño máximo (hasta que no se pueda actualizar más).

Caso 2:

Cuando el número de ceros en el arreglo de entrada es menor o igual a , se asume que el tamaño del sub-arreglo más largo es . Dado que es una cota superior, el problema así lo describe: *“****Problema*** *Encontrar la longitud del sub-arreglo más largo de a que tiene, a lo sumo, c ceros.”*

Con la entrada: , se tiene entonces:

, la dimensión del arreglo.

, el máximo número de ceros en un sub-arreglo.



Caso 2

Tamaño sub-arreglo: 15

Caso 3:

Cuando el número de ceros en el arreglo de entrada es igual a , se asume que el tamaño del sub-arreglo más largo es . Puesto que si todos los elementos en , son cero , lo anterior se cumple.

Con la entrada: , se tiene entonces:

, la dimensión del arreglo.

, el máximo número de ceros en un sub-arreglo.



Caso 3

Tamaño sub-arreglo: 3

* Métodos:
  + Principales:

1. ConstruirArreglo



1. ObtenerIndicesCeros



1. CalcularMayorSubarreglo



* + Auxiliares:

1. ContarCeros

Este método cuenta la cantidad de ceros presentes en el arreglo de entrada.



1. **Análisis de complejidades espacial y temporal.**

* Complejidad temporal
  + CalcularMayorSubarreglo

En el peor caso, se puede realizar permutaciones.

|  |  |
| --- | --- |
| **Operación** | **Veces** |
| Asignación (:=) |  |
| Comparación (>=, <=) |  |

Complejidad:

* Complejidad espacial

La complejidad espacial a diferencia de la temporal es , puesto que se trabaja sobre el arreglo solamente.

1. **Comentarios finales.**

Utilizando los casos de prueba suministrados en el documento y otros casos planteados por nosotros, nuestra solución demora tiempos aceptables en resolver el problema.

Una de las decisiones importantes fue definir los posibles casos de reconocimiento del sub-arreglo más largo (los cuales se pueden ver en el literal 1. Algoritmo de solución.).

Reconocer los 3 casos allí mencionados, permitió facilitar el cálculo de la respuesta, en busca de optimizar la respuesta del programa.