Universidad Simón Bolívar
Departamento de Computación y Tecnología de la Información
CI2692 - Laboratorio de Algoritmos y Estructuras II
Trimestre Enero-Marzo 2024

Examen 2

1. Problema 1

1.1. Descripción del problema

Dado una secuencia A de números enteros, que puede contener elementos repetidos, se quiere obtener una secuencia B en el cual todos los elementos repetidos estén juntos. El orden en el que los elementos aparecen en B corresponde al orden de la primera aparición del elemento en A. Dado una secuencia de n elementos, su solución debe ser $O(n)$.

1.2. Detalles de la implementación

La solución de este problema debe ser implementada en archivo llamado `Problema1.kt`. Además de este archivo, la solución puede incluir otros que considere conveniente. La entrada de los datos es por la entrada estándar. Debe crear un shell script, llamado `runProblema1.sh`, que ejecuta el programa `Problema1.kt`. Además debe proporcionar un archivo Makefile que compile el código fuente de este problema.

1.3. Entrada y salida del problema

Por la entrada estándar se recibe el arreglo a procesar. Suponga que se ejecuta la siguiente línea de comando, la cual corresponde al ejemplo dado:

```
>./runProblema1.sh 1 4 3 3 4 8 0 1 8 8 1 4 1
```

La salida correcta es la siguiente:

```
1 1 1 1 4 4 4 3 3 8 8 8 0
```

2. Problema 2

2.1. Descripción del problema

Dada una secuencia de n números enteros, se desea saber si esta posee o no un elemento mayoritario. En caso de poseer un elemento mayoritario, se debe indicarse cual es. Un elemento mayoritario es aquel que se encuentra más de $n/2$ veces en una secuencia. La salida del programa es por la salida estándar, y en caso de haber un elemento mayoritario se debe imprimir el mismo. En caso contrario, se debe imprimir la frase *No hay elemento mayoritario*. La solución debe ser $O(n)$.

2.2. Detalles de la implementación

La solución de este problema debe ser implementada en archivo llamado `Problema2.kt`. Además de este archivo, la solución puede incluir otros que considere conveniente. La entrada de los datos es por la entrada estándar. Debe crear un shell script, llamado `runProblema2.sh`, que ejecuta el programa `Problema2.kt`. Además debe proporcionar un archivo Makefile que compile el código fuente de este problema.

2.3. Entrada y salida del problema

Por la entrada estándar se recibe la secuencia a procesar. Suponga que se ejecuta la siguiente línea de comando:

```
>./runProblema2.sh 0 4 5 0 0 5 0 2 1 0 0
```

En este caso como hay un elemento mayoritario que es cero. La salida correcta es la siguiente:

```
0
```

3. Requerimientos de las implementaciones

No puede usar librerías de Kotlin que correspondan a contenedores o estructuras de datos contenedoras. Las estructuras de datos de Kotlin que puede usar se restringen a arreglos, tuplas, y pares. Para la solución de estos problemas deben hacer uso de alguna de las estructuras de datos usadas en clase, tales como listas, pilas, colas, tablas de hash y árboles, entre otras. Dependiendo de su solución debe escoger las estructuras de datos más adecuadas. Deben implementar todas las estructuras de datos que vayan a usar en solución. Las estructuras de datos a crear, deben estar contenidas cada en archivos aparte. Su programa debe funcionar en la plataforma Linux. Si su entrega no se compila o no se ejecuta en Linux la nota del examen será *cero*.

4. Condiciones de entrega

Los códigos fuentes de su programa, y la declaración de autenticidad firmada, deben estar contenidas en un archivo comprimido, con formato *tar.xz*, llamado *examen2_X.tar.xz*, donde X es el número de carné del estudiante. La entrega del archivo *examen2_X.tar.xz*, debe hacerse por la plataforma Classroom, antes de las 12:00 PM del día viernes 05 de abril de 2024.