INF-343 SISTEMAS DISTRIBUIDOS

TAREA Nº3: "Aplicación e implementación del algoritmo de Suzuki-Kasami para asegurar exclusión mutua distribuida en un sistema de reserva de pasajes"

11 de noviembre de 2022

ANTECEDENTES GENERALES

Objetivos:

- Aplicar un algoritmo de exclusión mutua distribuida para coordinar la actualización de datos compartidos por parte de procesos concurrentes.

Fechas importantes:

Ayudantía por Zoom: Miércoles, 16 de noviembre de 2022
Fecha de entrega de la tarea: Domingo, 4 de diciembre de 2022

ESPECIFICACIÓNDE LA TAREA

Descripción

Una empresa de buses requiere el desarrollo de un sistema que permita a los pasajeros seleccionar sus asientos al momento de realizar una reserva. Los requerimientos fundamentales para que el sistema funcione correctamente es que solo un pasajero se encuentre relacionado con un asiento y que ningún otro pasajero pueda tomar un asiento ya ocupado. Además, la empresa también quiere conocer la ganancia obtenida por la ocupación del bus.

Para un mapa de asientos, consideraremos el siguiente archivo:

`mapa.txt`

```
04 08 12 16 20 24 28 32 36 40 44 48
03 07 11 15 19 23 27 31 35 39 43 47

02 06 10 14 18 22 26 30 34 38 42 46
01 05 09 13 17 21 25 29 33 37 41 45
```

Las tarifas para cada asiento se rigen por los siguientes tramos:

- Tramo 1, asientos 01 – 16: $8.000 CLP.
- Tramo 2, asientos 17 – 32: $6.000 CLP.
- Tramo 3, asientos 33 – 48: $4.000 CLP.

En cuanto a los pasajeros, éstos se identificarán en el siguiente archivo:

pasajeros.txt

```
ROJAS 33
SOTO 15
DIAZ 04
CONTRERAS 24
SILVA 23
TORRES 41
MUNOZ 48
PEREZ 18
```

El formato para cada línea es APELLIDO ASIENTO, por lo que el archivo anterior es solo un ejemplo y puede contener una cantidad mayor de pasajeros.

Teniendo en cuenta ambos archivos mencionados anteriormente, uno de los objetivos de la empresa es obtener el archivo `mapa.txt` modificado, donde los asientos ocupados sean reemplazados por el *string* XX. Observe el siguiente ejemplo, donde los asientos ocupados corresponden a los listados en `pasajeros.txt`:

mapa.txt

```
XX 08 12 16 20 XX 28 32 36 40 44 XX
03 07 11 XX 19 XX 27 31 35 39 43 47

02 06 10 14 XX 22 26 30 34 38 42 46
01 05 09 13 17 21 25 29 XX 37 XX 45
```

Cabe mencionar que cada vez que se procese una reserva, el archivo `pasajeros.txt` disminuirá su cantidad de líneas a medida que el archivo `mapa.txt` se actualiza. Una vez que todos los pasajeros han sido procesados, el archivo `pasajeros.txt` quedará vacío.

Por otro lado, se requiere la modificación del siguiente archivo:

ganancias.txt

```
0 0 0
```

Donde el primer valor corresponde a la ganancia por los asientos del tramo 1, el segundo al tramo 2 y el tercero al tramo 3.

Si tenemos en cuenta los mismos pasajeros de `pasajeros.txt`, el archivo `ganacias.txt` contendrá los siguientes valores:

ganancias.txt

```
16000 18000 12000
```

Desarrollo:

La estrategia para resolver la presente tarea será el uso del **algoritmo de Suzuki–Kasami**, para asegurar la exclusión mutua en la manipulación de los archivos descritos anteriormente por parte de varios procesos concurrentes.

Independiente del número de archivos que cree, el algoritmo debe inicializarse en un archivo `main.go`. Además, el número de procesos concurrentes será establecido como argumento en la ejecución. Un ejemplo de ejecución se muestra a continuación, donde el algoritmo se iniciará con 3 procesos.

```
go build main.go
./main 3
```

Cada proceso procesará secuencial e iterativamente una entrada por vez de `pasajeros.txt`, hasta que el archivo quede vacío. Para simular la realización de otras tareas de un proceso entre el procesamiento de dos líneas de larchivo, se esperará 1 [s].

Los procesos generarán un *log* con las entradas procesadas por cada uno de ellos, según el siguiente formato de acuerdo al ejemplo:

`procesados.txt`

```
P1: ROJAS 33
P2: SOTO 15
P3: DIAZ 04
P2: CONTRERAS 24
P1: SILVA 23
P3: TORRES 41
P2: MUNOZ 48
P3: PEREZ 18
```

Cada línea es agregada por el propio proceso identificado, antes de entregar el *token*.

Consideraciones:

- No es necesario realizar ninguna validación para los datos contenidos en los `archivos.txt` entregados.
- La versión de Go es la 1.19.X.
- Para consultas, escriba a hugo.leyton@sansano.usm.cl.

Reglas de la tarea:

- La tarea se entrega en grupos de 3 personas.
- La fecha de entrega es el día **4 de diciembre a las 23:55**.
- La tarea debe ser subida al mismo repositorio creado en la `tarea 1`.
- Todo el contenido de la tarea debe estar en una nueva rama llamada “`tarea_3`”.
- En el `README` del repositorio, incluya el nombre y apellido de cada integrante.