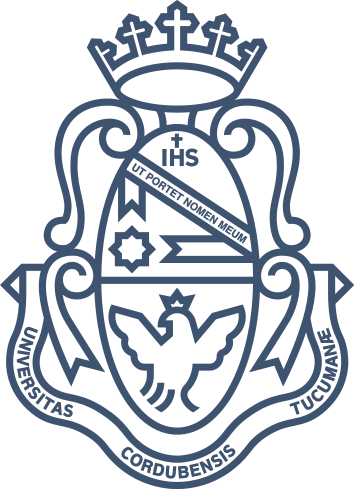
UNIVERSIDAD NACIONAL DE CÓRDOBA

FACULTAD DE CIENCIAS EXACTAS, FÍSICAS Y NATURALES



**TRABAJO PRÁCTICO INTEGRADOR**

**PROGRAMACIÓN CONCURRENTE**

ALUMNOS:

Castagno, Gustavo Daniel

Siñanez, María Edith

TEMA:

Circuito Ferroviario

DOCENTE:

Dr. Orlando Micolini

DOCENTE ADJUNTO:

Ing. Luis Orlando Ventre

2018

**ÍNDICE**

1. Objetivos 03
2. Enunciado 04
3. Consignas 05
4. Introducción 06
5. Consideraciones Generales para la RdP y el modelo 07
6. Consideraciones para la implementación del codigo 08
7. Red de Petri 09
   1. Estación Simplificada 10
   2. Estación Simplificada Máquina/Vagón 09
   3. Cuatro Estaciones con Máquina 09
   4. Análisis de T Invariantes 10
   5. Análisis de P Invariantes 11
   6. Matriz de Incidencia 12
   7. Sifones 13
   8. Trampas 13
   9. Matriz de Brazos Inhibidores 15
   10. Grafo de Alcanzabilidad/Cobertura 16

6.11 Simulación 17

1. Tabla de Eventos 18
2. Tabla de Estados o Actividades 20
3. Hilos, Justificación y Tareas realizadas 21
4. Diagrama de Clases 23
5. Diagrama de Secuencias 24
6. Código 25
   1. Main 25
   2. Monitor 31
   3. Paso Nivel 55
   4. Tren 56
7. Simulación 57
8. Testing 58
9. Conclusión 67
10. Bibliografía 68

**OBJETIVO**

El principal objetivo de este trabajo integrador es aplicar los conocimientos adquiridos durante el cursado de la materia para resolver problemas de sincronización e interbloqueo y lograr el paralelismo y la concurrencia entre programas y procesos.

## **Objetivos secundarios**

* Aprender el uso y la importancia de las redes de Petri y los monitores en los sistemas concurrentes.
* Evitar la inanición de los procesos.
* Lograr la implementación de un circuito ferroviario a través del uso de las redes de Petri y los monitores en Java.

**ENUNCIADO**

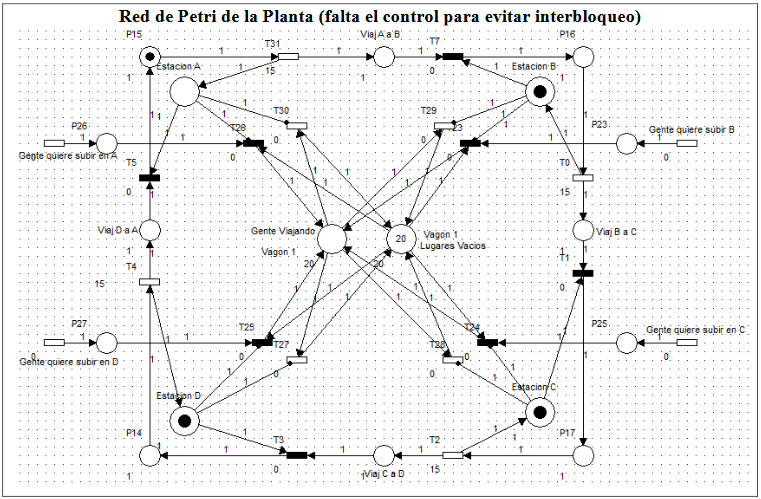
En este práctico se debe resolver el problema de control de un circuito ferroviario. Como dato se propone la red de Petri que modela una planta con 4 estaciones, un vagón, sin barreras. La red debe ser modificada con el fin de modelar la planta requerida y evitar interbloqueos. Luego simular la solución en un proyecto desarrollado con la herramienta adecuada (explique por qué eligió la herramienta usada).

La planta requerida está formada por 4 estaciones (Estación A, Estación B, Estación C y Estación D), una máquina y un vagón. La capacidad de la maquina es de 30 pasajeros, mientras que la capacidad del vagón es de 20 pasajeros. En cada estación los pasajeros pueden subir o bajar; no pudiendo descender en cada estación los pasajeros que han ascendido en esa (no es necesario identificar los pasajeros, solo número).

Los tramos de unión entre las estaciones A y B y las estaciones C y D tienen un paso a nivel. En este paso a nivel se debe controlar la barrera para el paso de los vehículos y el tren. La barrera debe bajar 30 metros antes que llegue el tren a paso nivel y subir después de 20 metros que el tren ha atravesado el paso a nivel.

El tren debe detenerse en cada estación no menos de 10 segundos y debe arrancar una vez que hayan subido todos los pasajeros o no haya lugar en máquina ni vagón.

El sistema controlador debe estar conformado por distintos hilos, los cuales deben ser asignados a cada conjunto de responsabilidades afines en particular. Por ej. Manejar el tren, manejar las barreras, etc.



**CONSIGNAS**

El modelo ha sido editado con la herramienta HPSim; está disponible en el LEV2.

Realizar:

* Colocar las restricciones a la RdP para evitar el interbloqueo, mostrarlo con la herramienta elegida y justificarlo.
* Colocar los tiempos en las estaciones (en las transiciones correspondientes).
* Hacer la tabla de eventos.
* Hacer la tabla de estados o actividades.
* Determinar la cantidad de hilos necesarios (justificarlo).
* Implementar dos casos de Políticas para producir:
  + Prioridad a los pasajeros que bajan.
  + Prioridad a los pasajeros que suben.
* Hacer el diagrama de clases.
* Hacer los diagramas de secuencias.
* Hacer el código.
* Hacer el testing.

**INTRODUCCIÓN**

En el desarrollo de este trabajo aplicamos conocimientos adquiridos en clases respecto a las Redes de Petri, utilizamos Java como lenguaje de programación ya que está orientado a objetos e integramos el concepto de monitores.

De esta manera pudimos comprender el alcance de las herramientas de trabajo que nos facilitaron durante el cursado de la materia, hablando más específicamente de las redes de Petri y el uso de monitores. Asimismo, empleamos aprendizajes previos a la asignatura para integrar todo lo recibido hasta el momento y lograr un mejor acercamiento a la resolución del problema en cuestión.

Además, a partir del modelo propuesto fuimos desmenuzándolo hasta lograr la mayor simplificación posible del mismo haciendo uso de la propiedad de simetría propia de la red y salvando que dicha reducción no afectara al enunciado original ni su complejidad.

Por otra parte, mediante el uso de los papers brindados por los docentes logramos resolver ciertas dudas respecto a la obtención de la matriz de incidencia y despejar otras acerca de los arcos inhibidores.

Por último, debemos mencionar el uso de internet que fue fundamental y el compañerismo mutuo de gran ayuda para encarar la problemática propuesta y encontrar la solución más óptima o al menos la que hallamos más conveniente.

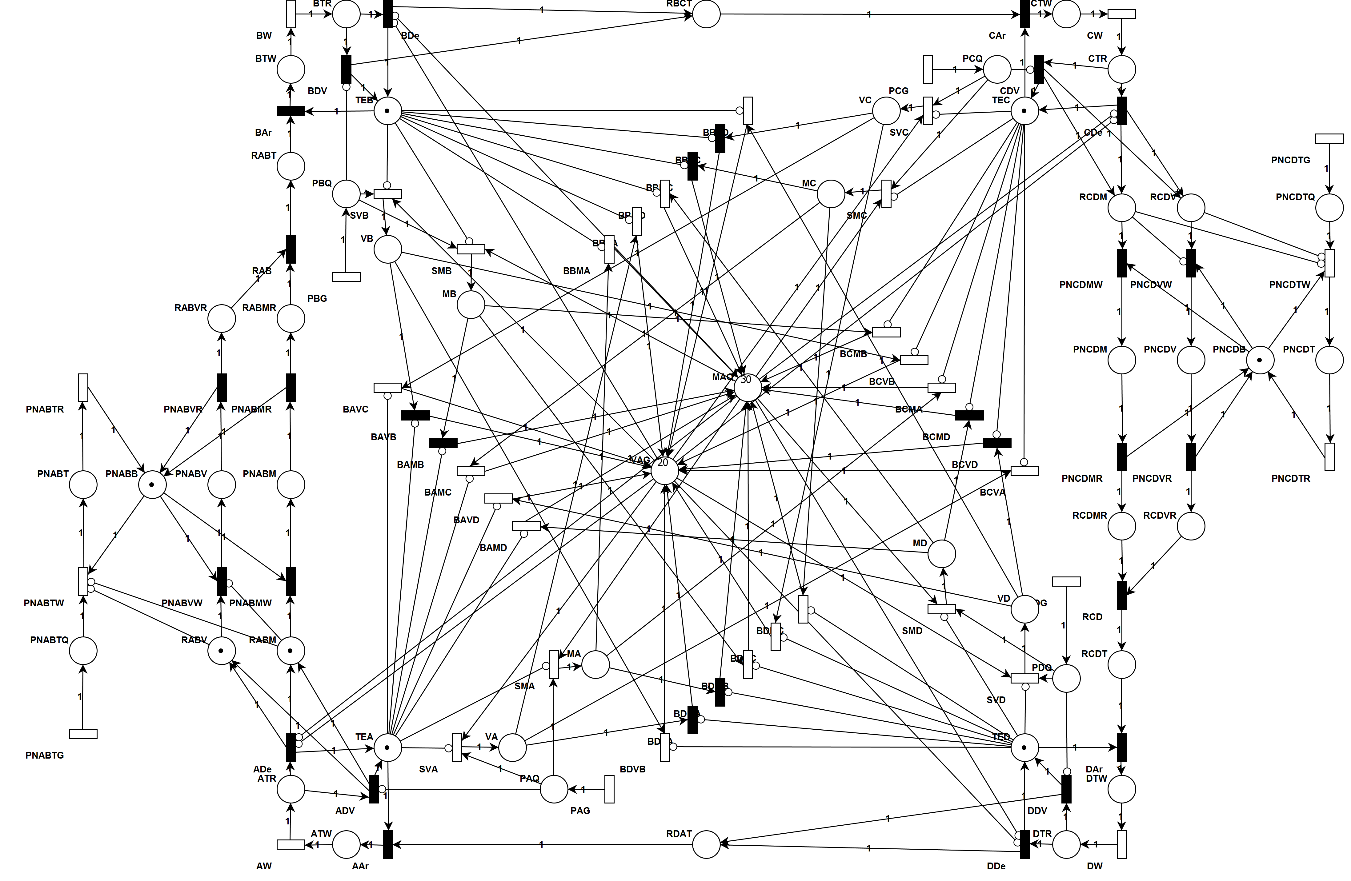
**CONSIDERACIONES GENERALES PARA LA RdP Y EL MODELO**

* En primera instancia tomamos la decisión de utilizar el enunciado general, sin tomar en cuenta la simplificación propuesta por el docente Micolini.
* El pasajero debe descender antes de llegar a la estación en la que subió.
* Los pasajeros pueden subirse a la máquina o al vagón de forma indistinta.
* Como herramienta de trabajo utilizamos solamente el PIPE para elaborar y simular la RdP.
* Debido a que el PIPE no permite especificar el tiempo de disparo, sino el ratio de disparo, utilizamos ratios para los tiempos de espera en las estaciones.
* Además, empleamos ratios para simular bajadas aleatorias de los pasajeros de la estación previa y la opuesta, mientras que para forzar el descenso de todos los pasajeros de la estación siguiente se utilizan transiciones inmediatas.
* Para poder ver la simulación adecuadamente, debimos utilizar para la subida transiciones con tiempo y no inmediatas, de lo contrario la estación se vaciaría instantáneamente o el tren se llenaría instantáneamente una vez finalizado el tiempo de espera en cada estación, de esta manera al tardar algo de tiempo para que puedan subir al tren los pasajeros, es más fácil visualizar cual de las dos condiciones se cumple primero.
* Debido a que la RdP posee simetría, se recurrió a un modelo simplificado para los análisis posteriores, asumiendo que todas las propiedades del sistema simplificado se cumplen en la red completa.
* Respecto a los pasos de nivel, se trató a la máquina y el vagón por separado a pedido de los docentes como un requisito adicional, tomando como prioridades primero a la máquina, luego el vagón y por último el tránsito.
* Para conservar invariante los lugares en la máquina y el vagón, se evitó utilizar la capacidad máxima de las plazas a pedido de los profesores.

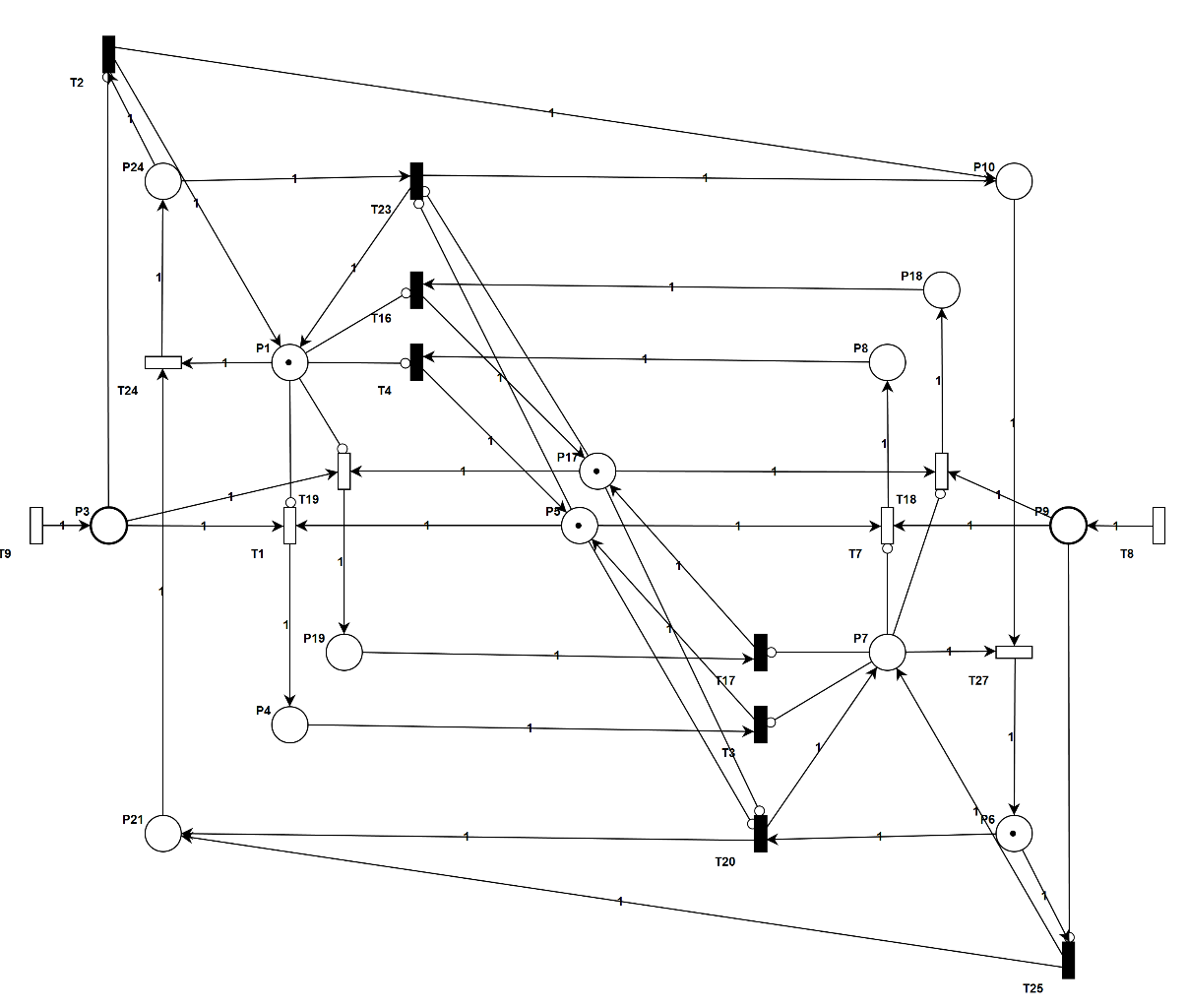
**CONSIDERACIONES PARA LA IMPLEMENTACION DEL CODIGO**

* Usamos las matrices de incidencia I+ e I- para simplificar la parte de la programación y el cálculo en vez de la matriz I combinada.
* El monitor no tiene threads.
* No consideramos al main como un hilo que llama o accede al monitor. Las clases Tren, Generador, PasoNivel, SubirPasajeros y BajarPasajeros implementan la interfaz Thread y llaman al monitor simulando eventos externos.
* En la clase Generador hay dos tipos de generadores…
* Se comenzó con una secuencia de disparos preestablecida

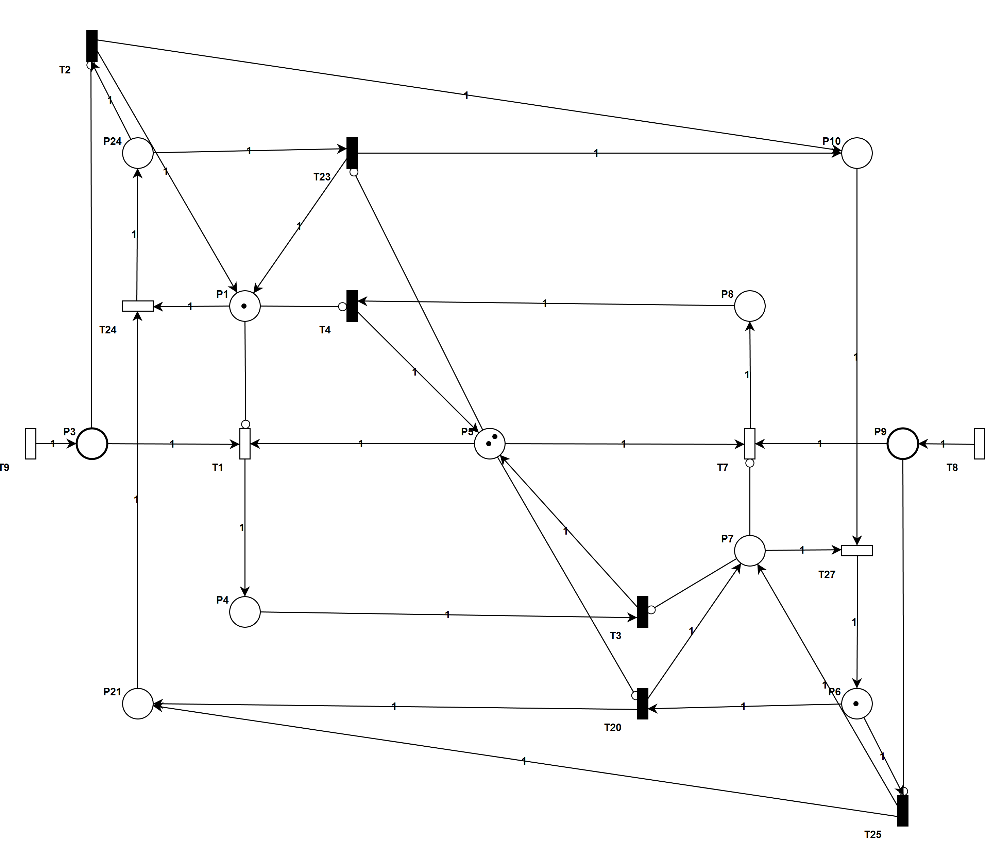
**RED DE PETRI**

****

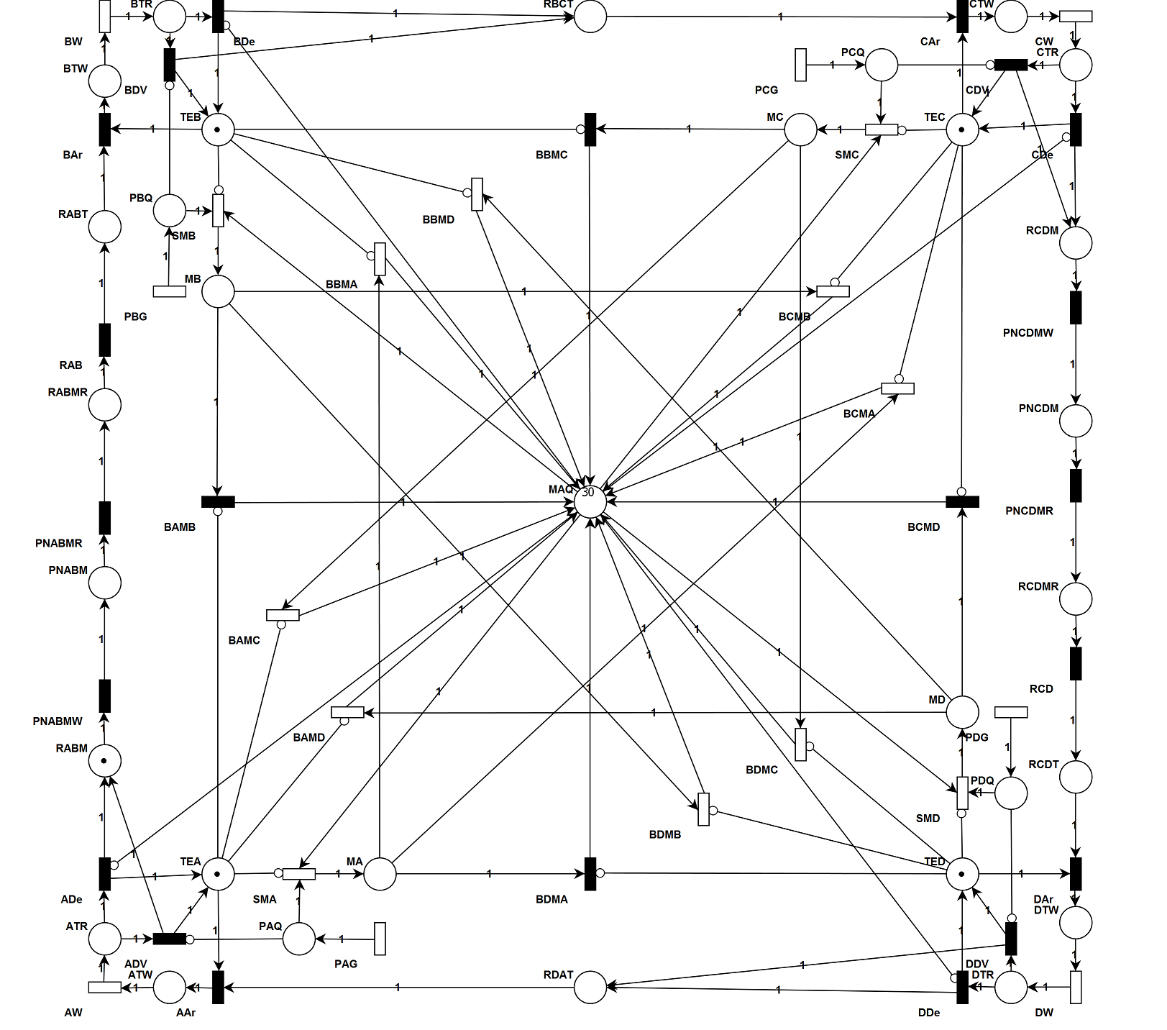
**Estación Simplificada:**

****

**Estación Simplificada Máquina/Vagón:**

****

**Cuatro estaciones con Máquina:**

****

**Análisis de T Invariantes:**

|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| AAr | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| ADe | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| AW | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| BAMB | 1 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| BAMC | 0 | 1 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| BAMD | 0 | 0 | 1 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| BAr | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| BAVB | 0 | 0 | 0 | 1 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| BAVC | 0 | 0 | 0 | 0 | 1 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| SVA | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 1 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 1 | 0 | 0 | 0 | 1 | 0 |
| BBMC | 0 | 0 | 0 | 0 | 0 | 1 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| BBVC | 0 | 0 | 0 | 0 | 0 | 0 | 1 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| BCMD | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 1 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| BCVD | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 1 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| BDe | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| BDMA | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 1 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| BDVA | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 1 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| BW | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| CAr | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| CDe | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| CW | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| DAr | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| DDe | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| DW | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| PNABMR | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| PNABMW | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| PNABTR | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 1 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| PNABTW | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 1 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| PNABVR | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| PNABVW | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| PNCDM | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 1 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| PNCDMR | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| PNCDTR | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 1 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| PNCDTW | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 1 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| PNCDVR | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| PNCDVW | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| RAB | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| RCD | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| SMA | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 1 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 1 | 0 | 0 | 0 | 0 | 0 | 0 |
| BAVD | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 1 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| BBMD | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 1 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| CDV | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| BCMB | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 1 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| PCG | 0 | 1 | 0 | 0 | 1 | 1 | 1 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 1 | 0 | 0 | 0 | 1 | 0 | 0 | 0 |
| BDV | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| DDV | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| PBG | 1 | 0 | 0 | 1 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 1 | 1 | 0 | 0 | 0 | 0 | 0 | 1 | 0 | 1 |
| PNCDTG | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 1 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| ADV | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| PAG | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 1 | 1 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 1 | 1 | 0 | 0 | 0 | 1 | 0 |
| PDG | 0 | 0 | 1 | 0 | 0 | 0 | 0 | 1 | 1 | 0 | 0 | 0 | 0 | 0 | 1 | 1 | 0 | 0 | 0 | 0 | 0 | 1 | 0 | 0 | 0 | 0 |
| PNABTG | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 1 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| BDMB | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 1 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| BDMC | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 1 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| BCMA | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 1 | 0 | 0 | 0 | 0 | 0 | 0 |
| BBVA | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 1 | 0 | 0 | 0 | 0 | 0 |
| SMD | 0 | 0 | 1 | 0 | 0 | 0 | 0 | 1 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 1 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| BBVD | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 1 | 0 | 0 | 0 | 0 |
| SVB | 0 | 0 | 0 | 1 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 1 | 0 | 1 |
| SVD | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 1 | 0 | 0 | 0 | 0 | 0 | 1 | 0 | 0 | 0 | 0 | 0 | 0 | 1 | 0 | 0 | 0 | 0 |
| BDVC | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 1 | 0 | 0 | 0 |
| BDVB | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 1 | 0 | 0 |
| BCVA | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 1 | 0 |
| BCVB | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 1 |
| SVC | 0 | 0 | 0 | 0 | 1 | 0 | 1 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 1 | 0 | 0 | 0 |
| BBMA | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| SMB | 1 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 1 | 1 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| SMC | 0 | 1 | 0 | 0 | 0 | 1 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 1 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

**Análisis de P Invariantes:**

|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| ATR | ATW | BTR | BTW | CTR | CTW | DTR | DTW | MA | MAQ | MB | MC | MD | PBQ | PCQ | PNCDTQ | PAQ | PDQ | PNABTQ | PNABB | PNABM | PNABT | PNABV | PNCDB | PNCDM | PNCDT | PNCDV | RABM | RABMR | RABT | RABV | RABVR | RBCT | RCDM | RCDMR | RCDT | RCDV | RCDVR | RDAT | TEA | TEB | TEC | TED | VA | VAG | VB | VC | VD |
| 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 1 | 1 | 1 | 1 | 1 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 1 | 1 | 1 | 1 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 1 | 1 | 1 | 1 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| 1 | 1 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 1 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| 0 | 0 | 1 | 1 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 1 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| 0 | 0 | 0 | 0 | 1 | 1 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 1 | 0 | 0 | 0 | 0 | 0 | 0 |
| 0 | 0 | 0 | 0 | 0 | 0 | 1 | 1 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 1 | 0 | 0 | 0 | 0 | 0 |
| 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 1 | 1 | 1 | 1 | 1 |

*Ecuaciones de P-Invariantes:*

M(MA) + M(MAQ) + M(MB) + M(MC) + M(MD) = 6  
M(PNABB) + M(PNABM) + M(PNABT) + M(PNABV) = 1  
M(PNCDB) + M(PNCDM) + M(PNCDT) + M(PNCDV) = 1  
M(ATR) + M(ATW) + M(TEA) = 1  
M(BTR) + M(BTW) + M(TEB) = 1  
M(CTR) + M(CTW) + M(TEC) = 1  
M(DTR) + M(DTW) + M(TED) = 1  
M(VA) + M(VAG) + M(VB) + M(VC) + M(VD) = 4

**Matriz de Incidencia:**

|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
|  | ATR | ATW | BTR | BTW | CTR | CTW | DTR | DTW | MA | MAQ | MB | MC | MD | PAQ | PBQ | PCQ | PDQ | PNABB | | PNABM | PNABT | PNABTQ | PNABV | PNCDB | PNCDM | PNCDT | PNCDTQ | PNCDV | RABM | RABMR | RABT | RABV | RABVR | RBCT | RCDM | RCDMR | RCDT | RCDV | RCDVR | RDAT | TEA | TEB | TEC | TED | VA | VAG | VB | VC | VD | |
| AAr | 0 | 1 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | -1 | -1 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| ADe | -1 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 1 | 0 | 0 | 1 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 1 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| ADV | -1 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 1 | 0 | 0 | 1 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 1 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| AW | 1 | -1 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| BAMB | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 1 | -1 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| BAMC | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 1 | 0 | -1 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| BAMD | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 1 | 0 | 0 | -1 | 0 | 0 | 0 | 0 | 0 | 0 | | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| BAr | 0 | 0 | 0 | 1 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | -1 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | -1 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| BAVB | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 1 | -1 | 0 | 0 |
| BAVC | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 1 | 0 | -1 | 0 |
| BAVD | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 1 | 0 | 0 | -1 |
| BBMA | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | -1 | 1 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| BBMC | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 1 | 0 | -1 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| BBMD | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 1 | 0 | 0 | -1 | 0 | 0 | 0 | 0 | 0 | 0 | | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| BBVA | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | -1 | 1 | 0 | 0 | 0 |
| BBVC | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 1 | 0 | -1 | 0 |
| BBVD | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 1 | 0 | 0 | -1 |
| BCMA | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | -1 | 1 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| BCMB | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 1 | -1 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| BCMD | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 1 | 0 | 0 | -1 | 0 | 0 | 0 | 0 | 0 | 0 | | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| BCVA | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | -1 | 1 | 0 | 0 | 0 |
| BCVB | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 1 | -1 | 0 | 0 |
| BCVD | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 1 | 0 | 0 | -1 |
| BDe | 0 | 0 | -1 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 1 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 1 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| BDMA | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | -1 | 1 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| BDMB | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 1 | -1 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| BDMC | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 1 | 0 | -1 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| BDV | 0 | 0 | -1 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 1 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 1 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| BDVA | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | -1 | 1 | 0 | 0 | 0 |
| BDVB | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 1 | -1 | 0 | 0 |
| BDVC | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 1 | 0 | -1 | 0 |
| BW | 0 | 0 | 1 | -1 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| CAr | 0 | 0 | 0 | 0 | 0 | 1 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | -1 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | -1 | 0 | 0 | 0 | 0 | 0 | 0 |
| CDe | 0 | 0 | 0 | 0 | -1 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 1 | 0 | 0 | 1 | 0 | 0 | 0 | 0 | 1 | 0 | 0 | 0 | 0 | 0 | 0 |
| CDV | 0 | 0 | 0 | 0 | -1 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 1 | 0 | 0 | 1 | 0 | 0 | 0 | 0 | 1 | 0 | 0 | 0 | 0 | 0 | 0 |
| CW | 0 | 0 | 0 | 0 | 1 | -1 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| DAr | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 1 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | -1 | 0 | 0 | 0 | 0 | 0 | 0 | -1 | 0 | 0 | 0 | 0 | 0 |
| DDe | 0 | 0 | 0 | 0 | 0 | 0 | -1 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 1 | 0 | 0 | 0 | 1 | 0 | 0 | 0 | 0 | 0 |
| DDV | 0 | 0 | 0 | 0 | 0 | 0 | -1 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 1 | 0 | 0 | 0 | 1 | 0 | 0 | 0 | 0 | 0 |
| DW | 0 | 0 | 0 | 0 | 0 | 0 | 1 | -1 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| PAG | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 1 | 0 | 0 | 0 | 0 | 0 | | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| PBG | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 1 | 0 | 0 | 0 | 0 | | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| PCG | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 1 | 0 | 0 | 0 | | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| PDG | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 1 | 0 | 0 | | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| PNABMR | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 1 | -1 | | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 1 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| PNABMW | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | -1 | 1 | | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | -1 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| PNABTG | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | | 0 | 1 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| PNABTR | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 1 | 0 | | -1 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| PNABTW | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | -1 | 0 | | 1 | -1 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| PNABVR | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 1 | 0 | | 0 | 0 | -1 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 1 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| PNABVW | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | -1 | 0 | | 0 | 0 | 1 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | -1 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| PNCDMR | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | | 0 | 0 | 0 | 1 | -1 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 1 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| PNCDMW | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | | 0 | 0 | 0 | -1 | 1 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | -1 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| PNCDTG | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | | 0 | 0 | 0 | 0 | 0 | 0 | 1 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| PNCDTR | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | | 0 | 0 | 0 | 1 | 0 | -1 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| PNCDTW | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | | 0 | 0 | 0 | -1 | 0 | 1 | -1 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| PNCDVR | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | | 0 | 0 | 0 | 1 | 0 | 0 | 0 | -1 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 1 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| PNCDVW | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | | 0 | 0 | 0 | -1 | 0 | 0 | 0 | 1 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | -1 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| RAB | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | -1 | 1 | 0 | -1 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| RCD | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | -1 | 1 | 0 | -1 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| SMA | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 1 | -1 | 0 | 0 | 0 | -1 | 0 | 0 | 0 | 0 | 0 | | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| SMB | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | -1 | 1 | 0 | 0 | 0 | -1 | 0 | 0 | 0 | 0 | | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| SMC | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | -1 | 0 | 1 | 0 | 0 | 0 | -1 | 0 | 0 | 0 | | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| SMD | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | -1 | 0 | 0 | 1 | 0 | 0 | 0 | -1 | 0 | 0 | | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| SVA | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | -1 | 0 | 0 | 0 | 0 | 0 | | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 1 | -1 | 0 | 0 | 0 |
| SVB | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | -1 | 0 | 0 | 0 | 0 | | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | -1 | 1 | 0 | 0 |
| SVC | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | -1 | 0 | 0 | 0 | | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | -1 | 0 | 1 | 0 |
| SVD | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | -1 | 0 | 0 | | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | -1 | 0 | 0 | 1 |

**Sifones:**

{VA, VAG, VB, VC, VD}

{PNCDB, PNCDM, PNCDT, PNCDV}

{PNABB, PNABM, PNABT, PNABV}

{MA, MAQ, MB, MC, MD}

{DTR, DTW, TED}

{CTR, CTW, TEC}  
{BTR, BTW, TEB}  
{ATR, ATW, TEA}  
{ATR, ATW, BTR, BTW, CTR, CTW, DTR, DTW, PNABM, PNCDM, RABM, RABMR, RABT, RBCT, RCDM, RCDMR, RCDT, RDAT}  
{ATR, ATW, BTR, BTW, CTR, CTW, DTR, DTW, PNABV, PNCDM, RABT, RABV, RABVR, RBCT, RCDM, RCDMR, RCDT, RDAT}  
{ATR, ATW, BTR, BTW, CTR, CTW, DTR, DTW, PNABM, PNCDV, RABM, RABMR, RABT, RBCT, RCDT, RCDV, RCDVR, RDAT}  
{ATR, ATW, BTR, BTW, CTR, CTW, DTR, DTW, PNABV, PNCDV, RABT, RABV, RABVR, RBCT, RCDT, RCDV, RCDVR, RDAT}

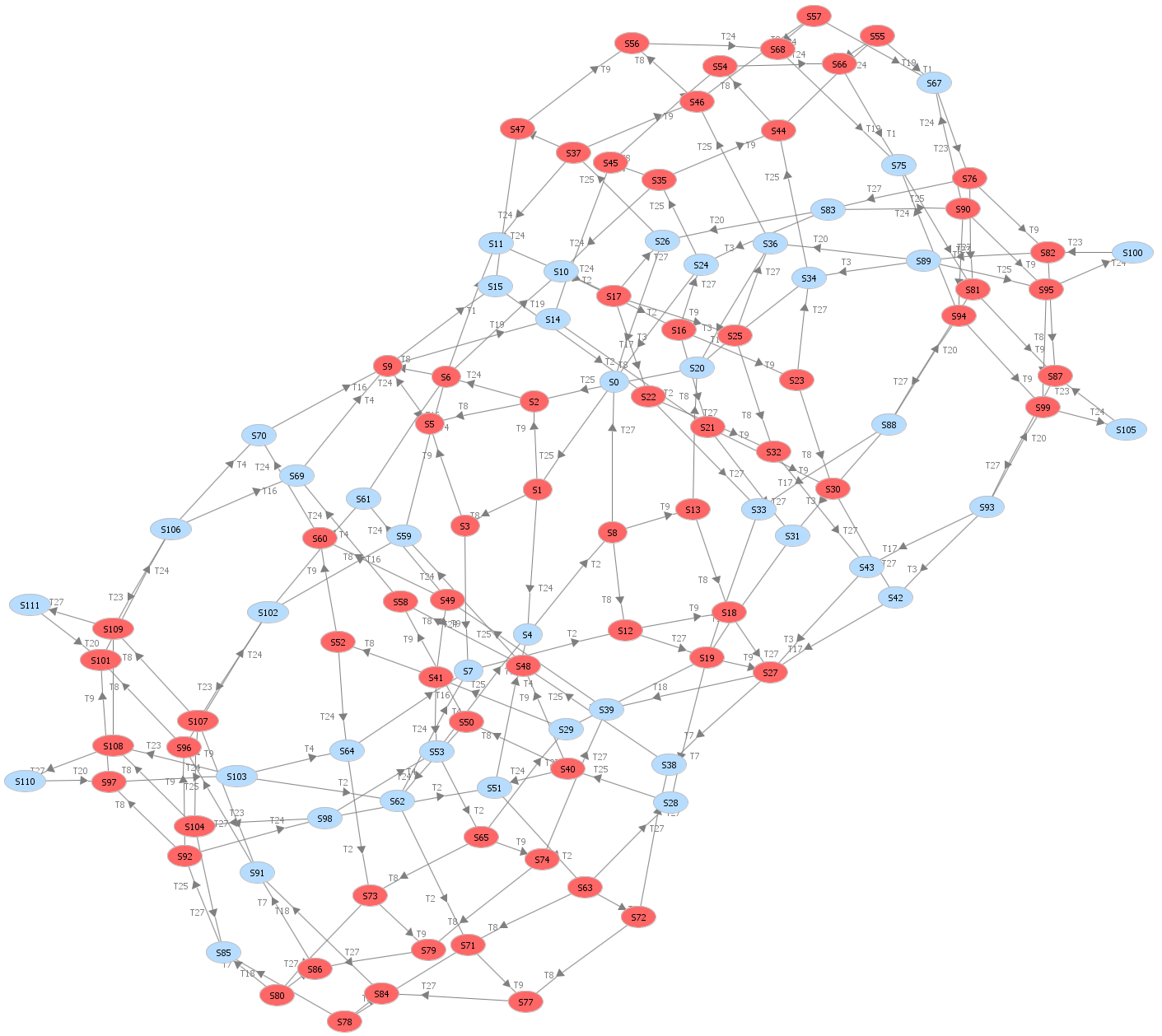
**Trampas:**

{VA, VAG, VB, VC, VD}  
{PNCDB, PNCDM, PNCDT, PNCDV}  
{PNABB, PNABM, PNABT, PNABV}  
{MA, MAQ, MB, MC, MD}  
{DTR, DTW, TED}  
{CTR, CTW, TEC}  
{BTR, BTW, TEB}  
{ATR, ATW, TEA}  
{ATR, ATW, DTR, RDAT, TEA}  
{ATR, ATW, BTR, BTW, DTR, PNABV, RABT, RABV, RABVR, RDAT, TEB}  
{ATR, ATW, BTR, BTW, DTR, PNABM, RABM, RABMR, RABT, RDAT, TEB}  
{ATR, ATW, BTR, BTW, DTR, PNABM, PNABV, RABM, RABMR, RABT, RABVR, RDAT, TEB}  
{ATR, ATW, DTR, DTW, RDAT, TEA}  
{ATR, ATW, BTR, BTW, DTR, DTW, PNABV, RABT, RABV, RABVR, RDAT, TEB}  
{ATR, ATW, BTR, BTW, DTR, DTW, PNABM, RABM, RABMR, RABT, RDAT, TEB}  
{ATR, ATW, BTR, BTW, CTR, CTW, DTR, DTW, PNABM, PNCDV, RABM, RABMR, RABT, RBCT, RCDT, RCDV, RCDVR, RDAT}  
{ATR, ATW, BTR, BTW, CTR, CTW, DTR, DTW, PNABM, PNCDM, RABM, RABMR, RABT, RBCT, RCDM, RCDMR, RCDT, RDAT}  
{ATR, ATW, BTR, BTW, DTR, DTW, PNABM, PNABV, RABM, RABMR, RABT, RABVR, RDAT, TEB}  
{ATR, ATW, BTR, BTW, CTR, CTW, DTR, DTW, PNABV, PNCDV, RABT, RABV, RABVR, RBCT, RCDT, RCDV, RCDVR, RDAT}  
{ATR, ATW, BTR, BTW, CTR, CTW, DTR, DTW, PNABV, PNCDM, RABT, RABV, RABVR, RBCT, RCDM, RCDMR, RCDT, RDAT}

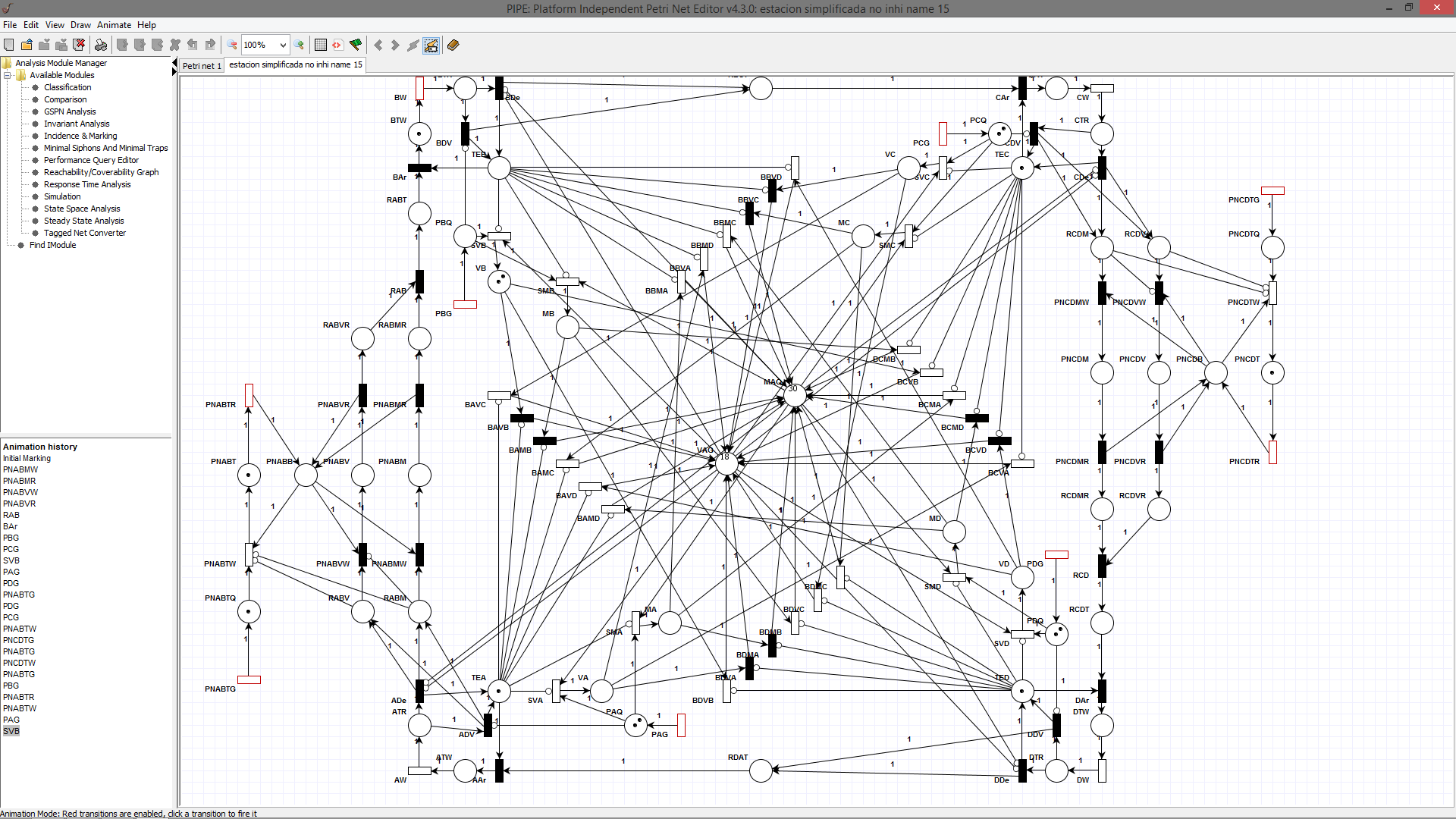
**Matriz de Brazos inhibidores:**

|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
|  | ATR | ATW | BTR | BTW | CTR | CTW | DTR | DTW | MA | MAQ | MB | MC | MD | PAQ | PBQ | PCQ | PDQ | PNABB | PNABM | PNABT | PNABTQ | PNABV | PNCDB | PNCDM | PNCDT | PNCDTQ | PNCDV | RABM | RABMR | RABT | RABV | RABVR | RBCT | RCDM | RCDMR | RCDT | RCDV | RCDVR | RDAT | TEA | TEB | TEC | TED | VA | VAG | VB | VC | VD |
| AAr | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| ADe | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 1 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 1 | 0 | 0 | 0 |
| ADV | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 1 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| AW | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| BAMB | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 1 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| BAMC | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 1 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| BAMD | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 1 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| BAr | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| BAVB | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 1 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| BAVC | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 1 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| BAVD | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 1 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| BBMA | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 1 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| BBMC | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 1 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| BBMD | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 1 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| BBVA | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 1 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| BBVC | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 1 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| BBVD | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 1 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| BCMA | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 1 | 0 | 0 | 0 | 0 | 0 | 0 |
| BCMB | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 1 | 0 | 0 | 0 | 0 | 0 | 0 |
| BCMD | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 1 | 0 | 0 | 0 | 0 | 0 | 0 |
| BCVA | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 1 | 0 | 0 | 0 | 0 | 0 | 0 |
| BCVB | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 1 | 0 | 0 | 0 | 0 | 0 | 0 |
| BCVD | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 1 | 0 | 0 | 0 | 0 | 0 | 0 |
| BDe | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 1 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 1 | 0 | 0 | 0 |
| BDMA | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 1 | 0 | 0 | 0 | 0 | 0 |
| BDMB | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 1 | 0 | 0 | 0 | 0 | 0 |
| BDMC | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 1 | 0 | 0 | 0 | 0 | 0 |
| BDV | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 1 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| BDVA | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 1 | 0 | 0 | 0 | 0 | 0 |
| BDVB | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 1 | 0 | 0 | 0 | 0 | 0 |
| BDVC | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 1 | 0 | 0 | 0 | 0 | 0 |
| BW | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| CAr | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| CDe | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 1 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 1 | 0 | 0 | 0 |
| CDV | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 1 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| CW | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| DAr | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| DDe | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 1 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 1 | 0 | 0 | 0 |
| DDV | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 1 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| DW | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| PAG | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| PBG | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| PCG | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| PDG | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| PNABMR | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| PNABMW | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| PNABTG | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| PNABTR | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| PNABTW | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 1 | 0 | 0 | 1 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| PNABVR | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| PNABVW | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 1 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| PNCDMR | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| PNCDMW | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| PNCDTG | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| PNCDTR | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| PNCDTW | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 1 | 0 | 0 | 1 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| PNCDVR | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| PNCDVW | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 1 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| RAB | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| RCD | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| SMA | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 1 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| SMB | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 1 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| SMC | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 1 | 0 | 0 | 0 | 0 | 0 | 0 |
| SMD | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 1 | 0 | 0 | 0 | 0 | 0 |
| SVA | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 1 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| SVB | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 1 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| SVC | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 1 | 0 | 0 | 0 | 0 | 0 | 0 |

**Grafo de Alcanzabilidad/Cobertura:**

****

**Simulación:**

****

**TABLA DE EVENTOS**

|  |  |
| --- | --- |
| Transición | Descripción |
| AAr  AW  ADe  PNABMW  PNABVW  PNABMR  PNABVR  RAB  BAr  BW  BDe  CAr  CW  CDe  PNCDMW  PNCDVW  PNCDMR  PNCDVR  RCD  DAr  DW  DDe | Arribo de pasajeros a la estación A.  Espera mínima del tren en la estación A.  Espera que el tren se llene en la estación A para partir.  Espera de la Máquina por el Paso Nivel de A a B (recurso barrera).  Espera del Vagón por el Paso Nivel de A a B (recurso barrera).  Máquina libera el Paso Nivel de A a B (recurso barrera).  Vagón libera el Paso Nivel de A a B (recurso barrera).  Recorrido de la estación A a la estación B.  Arribo de pasajeros a la estación B.  Espera mínima del tren en la estación B.  Espera que el tren se llene en la estación B para partir.  Arribo de pasajeros a la estación C.  Espera mínima del tren en la estación C.  Espera que el tren se llene en la estación C para partir.  Espera de la Máquina por el Paso Nivel de C a D (recurso barrera).  Espera del Vagón por el Paso Nivel de C a D (recurso barrera).  Máquina libera el Paso Nivel de C a D (recurso barrera).  Vagón libera el Paso Nivel de C a D (recurso barrera).  Recorrido de la estación C a la estación D.  Arribo de pasajeros a la estación D.  Espera mínima del tren en la estación D.  Espera que el tren se llene en la estación D para partir. |
| PNABTG  PNABTW  PNABTR  PNCDTG  PNCDTW  PNCDTR | Generador de vehículos para el Paso de Nivel entre A y B.  Espera del Tránsito por el Paso Nivel de la estación A a la estación B.  El Tránsito Libera el Paso Nivel de A a B.  Generador de vehículos para el Paso de Nivel entre C y D.  Espera del Tránsito por el Paso Nivel de la estación C a la estación D.  El Transito Libera el Paso Nivel de C a D. |
| BAVC  BAVB  BAMB  BAMC  BAVD  BAMD  PAG  SMA  SVA  ADV | Bajada en la estación A de los pasajeros del Vagón subidos en la estación C.  Bajada en la estación A de los pasajeros del Vagón subidos en la estación B.  Bajada en la estación A de los pasajeros de la Máquina subidos en la estación B.  Bajada en la estación A de los pasajeros de la Maquina subidos en la estación C.  Bajada obligatoria en la estación A de pasajeros del Vagón subidos en la estación D.  Bajada obligatoria en la estación A de pasajeros de la Maquina subidos en la estación D.  Generador de Pasajeros de la estación A.  Subida de los pasajeros a la Máquina en la estación A.  Subida de los pasajeros al Vagón en la estación A.  Se sensibiliza cuando la estación A está vacía (partida del tren). |
| BBVD  BBVC  BBMC  BBMD  BBVA  BBMA  PBG  SMB  SVB  BDV | Bajada en la estación B de los pasajeros del Vagón subidos en la estación D.  Bajada obligatoria en la estación B de pasajeros del Vagón subidos en la estación C.  Bajada obligatoria en la estación B de pasajeros de la Máquina subidos en la estación C.  Bajada en la estación B de los pasajeros de la Máquina subidos en la estación D.  Bajada en la estación B de los pasajeros del Vagón subidos en la estación A.  Bajada en la estación B de los pasajeros de la Máquina subidos en la estación A.  Generador de Pasajeros de la estación B.  Subida de los pasajeros a la Máquina en la estación B.  Subida de los pasajeros al Vagón en la estación B.  Se sensibiliza cuando la estación B está vacía (partida del tren). |
| BCVA  BCVD  BCMD  BCMA  BCVB  BCMB  PCG  SMC  SVC  CDV | Bajada en la estación C de los pasajeros del Vagón subidos en la estación A.  Bajada obligatoria en la estación C de los pasajeros del Vagón subidos en la estación D.  Bajada obligatoria en la estación C de pasajeros de la Maquina subidos en la estación D.  Bajada en la estación C de los pasajeros de la Máquina subidos en la estación A.  Bajada en la estación C de los pasajeros del Vagón subidos en la estación B.  Bajada en la estación C de los pasajeros de la Máquina subidos en la estación B.  Generador de Pasajeros de la estación C.  Subida de los pasajeros a la Maquina en la estación C.  Subida de los pasajeros al Vagón en la estación C.  Se sensibiliza cuando la estación C está vacía (partida del tren). |
| BDVB  BDVA  BDMA  BDMB  BDVC  BDMC  PDG  SMD  SVD  DDV | Bajada en la estación D de los pasajeros del Vagón subidos en la estación B.  Bajada obligatoria en la estación D de pasajeros del Vagón subidos en la estación A.  Bajada obligatoria en la estación D de pasajeros de la Máquina subidos en la estación A.  Bajada en la estación D de los pasajeros de la Máquina subidos en la estación B.  Bajada en la estación D de los pasajeros del Vagón subidos en la estación C.  Bajada en la estación D de los pasajeros de la Máquina subidos en la estación C.  Generador de Pasajeros de la estación D.  Subida de los pasajeros a la Máquina en la estación D.  Subida de los pasajeros al Vagón en la estación D.  Se sensibiliza cuando la estación D está vacía (partida del tren). |

**TABLA DE ESTADOS O ACTIVIDADES**

|  |  |
| --- | --- |
| Plazas | Descripción |
| MAQ  VAG | Lugares disponibles en la Máquina.  Lugares disponibles en el Vagón. |
| RDAT  ATW  ATR  RABM  RABV  PNABM  PNABV  PNABB  PNABTQ  PNABT  RABMR  RABVR  RABT  BTW  BTR  RBCT  CTW  CTR  RCDM  RCDV  PNCDM  PNCDV  PNCDB  PNCDTQ  PNCDT  RCDMR  RCDVR  RCDT  DTW  DTR | Recorrido del Tren de la estación D a A.  Espera del Tren en la estación A.  Tren esperando subida de pasajeros en la estación A.  Máquina espera el cruce por el Paso Nivel entre A y B.  Vagón espera el cruce por el Paso Nivel entre A y B.  Máquina cruzando el Paso Nivel de A a B.  Vagón cruzando el Paso Nivel de A a B.  Barrera del Paso Nivel entre A y B.  Tránsito en cola esperando por el Paso Nivel entre A y B.  Transito cruzando el Paso Nivel de A a B.  Máquina esperando que el Vagón cruce el Paso de Nivel de A a B.  Vagón Listo para continuar el Recorrido junto a la Maquina de A a B.  Recorrido del Tren de la estación A a B.  Espera del Tren en la estación B.  Tren esperando subida de pasajeros en la estación B.  Recorrido del tren de la estación B a C.  Espera del Tren en la estación C.  Tren esperando subida de pasajeros en la estación C.  Máquina espera el cruce por el Paso Nivel entre C y D.  Vagón espera el cruce por el Paso Nivel entre C y D.  Máquina cruzando el Paso Nivel de C a D.  Vagón cruzando el Paso Nivel de C a D.  Barrera del Paso Nivel entre C y D.  Tránsito en cola esperando por el Paso Nivel entre C y D.  Transito cruzando el Paso Nivel de C a D.  Máquina esperando que el Vagón cruce el Paso de Nivel de C a D.  Vagón Listo para continuar el Recorrido junto a la Máquina de C a D.  Recorrido del Tren de la estación C a D.  Espera del Tren en la estación D.  Tren esperando subida de pasajeros en la estación D. |
| PAQ  TEA  VA  MA | Pasajeros esperando en Cola en la estación A.  Tren en la Estación A.  Pasajeros subidos en el Vagón en la estación A.  Pasajeros subidos en la Máquina en la estación A. |
| PBQ  TEB  VB  MB | Pasajeros esperando en Cola en la estación B.  Tren en la Estación B.  Pasajeros subidos en el Vagón en la estación B.  Pasajeros subidos en la Máquina en la estación B. |
| PCQ  TEC  VC  MC | Pasajeros esperando en Cola en la estación C.  Tren en la Estación C.  Pasajeros subidos en el Vagón en la estación C.  Pasajeros subidos en la Máquina en la estación C. |
| PDQ  TED  VD  MD | Pasajeros esperando en Cola en la estación D.  Tren en la Estación D.  Pasajeros subidos en el Vagón en la estación D.  Pasajeros subidos en la Máquina en la estación D. |

**HILOS (justificación)**

Utilizamos veintinueve hilos en total, tres hilos por cada estación (es decir doce). Además, un hilo correspondiente al tren, dos hilos por cada paso de nivel que hay entre dos estaciones contiguas y un último hilo para el main pero éste muere al instante.

**Tareas Realizadas por cada hilo:**

* Hilos de cada estación: Dos hilos se encargan de subir pasajeros, uno principal y otro auxiliar, dos hilos se encargan de bajar pasajeros, uno principal y otro auxiliar y un quinto hilo se encarga de generar gente para viajar.
* Hilo del tren: Espera en cada estación. Se agregó un hilo auxiliar de arribo para la llegada del tren y otro para la partida, chequea las condiciones de que el tren este lleno o la estación vacía.
* Hilos de cada paso nivel: Dos hilos generadores de los automóviles que desean cruzar y dos hilos auxiliares que hacen cruzar el tránsito por cada paso de nivel y liberan la barrera del tránsito, del vagón y de la máquina. Además, un hilo auxiliar para hacer cruzar a la máquina y otro auxiliar para hacer cruzar al vagón.
* Hilo del main: Inicializa las matrices, el monitor y el resto de los hilos, luego de ello muere por eso no lo contamos como parte del total de hilos utilizados en el monitor.

|  |  |  |
| --- | --- | --- |
| Nombre del hilo | Hilos que cumplen dicha función | Total de hilos |
| Arribo Aux | AAr, BAr, CAr, DAr | 1 |
| Partida Aux | ADe, BDe, CDe, DDe | 2 |
| Partida Aux | ADV, BDV, CDV, DDV | 2 |
| Partida Aux | RAB, RCD | 2 |
| Tren | AW, BW, CW, DW | 3 |
| Máquina | PNABMW, PNCDMW | 4 |
| Vagón | PNABVW, PNCDVW | 5 |
| Paso Nivel Tránsito Auxiliar AB | PNABTR, PNABVR, PNABMR, PNABTW | 6 |
| Paso Nivel Tránsito Auxiliar CD | PNCDTR, PNCDVR, PNCDMR, PNCDTW | 7 |
| Hilo Generador Tránsito AB | PNABTG | 8 |
| Hilo Generador Tránsito CD | PNCDTG | 9 |
| Hilo Generador Pasajeros A | PAG | 10 |
| Hilo Generador Pasajeros B | PBG | 11 |
| Hilo Generador Pasajeros C | PCG | 12 |
| Hilo Generador Pasajeros D | PDG | 13 |
| Subida A, Subida A Aux | SMA, SVA | 14, 15 |
| Subida B, Subida B Aux | SMB, SVB | 16, 17 |
| Subida C, Subida C Aux | SMC, SVC | 18, 19 |
| Subida D, Subida D Aux | SMD, SVD | 20, 21 |
| Bajada A, Bajada A Aux | BAVC, BAVB, BAMB, BAMC, BAVD, BAMD | 22, 23 |
| Bajada B, Bajada B Aux | BBVD, BBVC, BBMC, BBMD, BBVA, BBMA | 24, 25 |
| Bajada C, Bajada C Aux | BCVA, BCVD, BCMD, BCMA, BCVB, BCMB | 26, 27 |
| Bajada D, Bajada D Aux | BDVB, BDVA, BDMA, BDMB, BDVC, BDMC | 28, 29 |

**DIAGRAMA DE CLASES**

****

**DIAGRAMA DE SECUENCIAS**

****

**CÓDIGO**

* *Main:*

|  |
| --- |
| package main; |
|  |
|  |  |
|  | import java.io.File; |
|  | import java.io.FileNotFoundException; |
|  | import java.io.FileOutputStream; |
|  | import java.io.FileReader; |
|  | import java.io.PrintWriter; |
|  | import java.util.ArrayList; |
|  | import java.util.LinkedHashMap; |
|  | import java.util.Scanner; |
|  |  |
|  | public class Main extends ConstantesComunes { |
|  | private static final String marcadoInicial = "./src/main/MarcadoInicial.html"; |
|  | private static final String matrizInhibicion = "./src/main/MatrizInhibicion.html"; |
|  | private static final String matrizIMas = "./src/main/MatrizIMas.html"; |
|  | private static final String matrizIMenos = "./src/main/MatrizIMenos.html"; |
|  | private static final String testOutput = "./src/test/testOutput.txt"; |
|  |  |
|  | public static void main(String[] args) { |
|  |  |
|  | PrintWriter printWriter = null; |
|  |  |
|  | try { |
|  | File file = new File(testOutput); |
|  | if(file.exists()) { |
|  | file.delete(); |
|  | } |
|  | file.createNewFile(); |
|  | FileOutputStream fileOutputStream = new FileOutputStream(file); |
|  | printWriter = new PrintWriter(fileOutputStream); |
|  | } catch (Exception e) { |
|  | e.printStackTrace(); |
|  | } |
|  |  |
|  | System.out.println("\nMatriz de Incidencia Positiva"); |
|  | Integer[][] matrizMas = getMatrix(matrizIMas); |
|  | System.out.println("\nMatriz de Incidencia Negativa"); |
|  | Integer[][] matrizMenos = getMatrix(matrizIMenos); |
|  | System.out.println("\nMatriz de Arcos Inhibidores"); |
|  | Integer[][] matrizInhibidora = getMatrix(matrizInhibicion); |
|  |  |
|  | ArrayList<String> transiciones = getTransiciones(matrizIMas); |
|  |  |
|  | LinkedHashMap<String, Integer> marcadoInicial = marcadoInicial(Main.marcadoInicial); |
|  |  |
|  |  |
|  | Monitor monitor = new Monitor(matrizMas, matrizMenos, matrizInhibidora, marcadoInicial, transiciones, printWriter); |
|  |  |
|  | SubirPasajeros subirPasajerosA = new SubirPasajeros(monitor, estacionA, precedenciaPrincipal); |
|  | subirPasajerosA.start(); |
|  | SubirPasajeros subirPasajerosB = new SubirPasajeros(monitor, estacionB, precedenciaPrincipal); |
|  | subirPasajerosB.start(); |
|  | SubirPasajeros subirPasajerosC = new SubirPasajeros(monitor, estacionC, precedenciaPrincipal); |
|  | subirPasajerosC.start(); |
|  | SubirPasajeros subirPasajerosD = new SubirPasajeros(monitor, estacionD, precedenciaPrincipal); |
|  | subirPasajerosD.start(); |
|  |  |
|  | SubirPasajeros subirPasajerosAuxA = new SubirPasajeros(monitor, estacionA, precedenciaAuxiliar); |
|  | subirPasajerosAuxA.start(); |
|  | SubirPasajeros subirPasajerosAuxB = new SubirPasajeros(monitor, estacionB, precedenciaAuxiliar); |
|  | subirPasajerosAuxB.start(); |
|  | SubirPasajeros subirPasajerosAuxC = new SubirPasajeros(monitor, estacionC, precedenciaAuxiliar); |
|  | subirPasajerosAuxC.start(); |
|  | SubirPasajeros subirPasajerosAuxD = new SubirPasajeros(monitor, estacionD, precedenciaAuxiliar); |
|  | subirPasajerosAuxD.start(); |
|  |  |
|  | BajarPasajeros bajarPasajerosA = new BajarPasajeros(monitor, estacionA, precedenciaPrincipal); |
|  | bajarPasajerosA.start(); |
|  | BajarPasajeros bajarPasajerosB = new BajarPasajeros(monitor, estacionB, precedenciaPrincipal); |
|  | bajarPasajerosB.start(); |
|  | BajarPasajeros bajarPasajerosC = new BajarPasajeros(monitor, estacionC, precedenciaPrincipal); |
|  | bajarPasajerosC.start(); |
|  | BajarPasajeros bajarPasajerosD = new BajarPasajeros(monitor, estacionD, precedenciaPrincipal); |
|  | bajarPasajerosD.start(); |
|  |  |
|  | BajarPasajeros bajarPasajerosAuxA = new BajarPasajeros(monitor, estacionA, precedenciaAuxiliar); |
|  | bajarPasajerosAuxA.start(); |
|  | BajarPasajeros bajarPasajerosAuxB = new BajarPasajeros(monitor, estacionB, precedenciaAuxiliar); |
|  | bajarPasajerosAuxB.start(); |
|  | BajarPasajeros bajarPasajerosAuxC = new BajarPasajeros(monitor, estacionC, precedenciaAuxiliar); |
|  | bajarPasajerosAuxC.start(); |
|  | BajarPasajeros bajarPasajerosAuxD = new BajarPasajeros(monitor, estacionD, precedenciaAuxiliar); |
|  | bajarPasajerosAuxD.start(); |
|  |  |
|  | Tren tren = new Tren(monitor, precedenciaPrincipal); |
|  | tren.start(); |
|  | Tren trenAuxiliarArrivo = new Tren(monitor, precedenciaAuxiliarArrivo); |
|  | trenAuxiliarArrivo.start(); |
|  | Tren trenAuxiliarPartida = new Tren(monitor, precedenciaAuxiliarPartida); |
|  | trenAuxiliarPartida.start(); |
|  |  |
|  | Generador generadorPasajerosEstacionA = new Generador(monitor, generadorPasajeros, estacionA); |
|  | generadorPasajerosEstacionA.start(); |
|  | Generador generadorPasajerosEstacionB = new Generador(monitor, generadorPasajeros, estacionB); |
|  | generadorPasajerosEstacionB.start(); |
|  | Generador generadorPasajerosEstacionC = new Generador(monitor, generadorPasajeros, estacionC); |
|  | generadorPasajerosEstacionC.start(); |
|  | Generador generadorPasajerosEstacionD = new Generador(monitor, generadorPasajeros, estacionD); |
|  | generadorPasajerosEstacionD.start(); |
|  |  |
|  | Generador generadorTransitoEstacionA = new Generador(monitor, generadorTransito, recorridoAB); |
|  | generadorTransitoEstacionA.start(); |
|  | Generador generadorTransitoEstacionB = new Generador(monitor, generadorTransito, recorridoCD); |
|  | generadorTransitoEstacionB.start(); |
|  |  |
|  | PasoNivel pasoDeNivelMaquina = new PasoNivel(monitor, maquinaTren, precedenciaPrincipal); |
|  | pasoDeNivelMaquina.start(); |
|  | PasoNivel pasoDeNivelVagon = new PasoNivel(monitor, vagonTren, precedenciaPrincipal); |
|  | pasoDeNivelVagon.start(); |
|  | PasoNivel pasoDeNivelTransitoAuxiliarAB = new PasoNivel(monitor, pasoNivelTransitoAB, precedenciaAuxiliar); |
|  | pasoDeNivelTransitoAuxiliarAB.start(); |
|  | PasoNivel pasoDeNivelTransitoAuxiliarCD = new PasoNivel(monitor, pasoNivelTransitoCD, precedenciaAuxiliar); |
|  | pasoDeNivelTransitoAuxiliarCD.start(); |
|  |  |
|  | } |
|  |  |
|  |  |
|  | static private Integer[][] getMatrix(String pathName) { |
|  | Integer[][] matriz = new Integer[100][100]; |
|  | Integer[][] matrizIncidencia = null; |
|  |  |
|  | FileReader matrizFile; |
|  | try { |
|  | matrizFile = new FileReader(pathName); |
|  | Scanner scanFile = new Scanner(matrizFile); |
|  | // System.out.println(scanFile.hasNext()); |
|  |  |
|  | String tempString = scanFile.nextLine(); |
|  | int i = 0; |
|  | int j = 0; |
|  | for (i = 0; !tempString.contains("</table"); i++) { |
|  | while(!tempString.contains("<tr")) { |
|  | tempString = scanFile.nextLine(); |
|  | } |
|  | for (j = 0; !tempString.contains("</tr"); j++) { |
|  | while(!tempString.contains("<td")) { |
|  | tempString = scanFile.nextLine(); |
|  | } |
|  |  |
|  | while(!tempString.contains("</td")) { |
|  | if(tempString.contains("<td class=\"cell\">")) { |
|  | tempString = scanFile.nextLine(); |
|  | matriz[i][j] = Integer.valueOf(tempString.trim()); |
|  | } else { |
|  | tempString = scanFile.nextLine(); |
|  | } |
|  | } |
|  | tempString = scanFile.nextLine(); |
|  | } |
|  | tempString = scanFile.nextLine(); |
|  | } |
|  |  |
|  | System.out.println("i: "+(i-1)+ " - j:"+(j-1)); |
|  | matrizIncidencia = new Integer[(i-1)][(j-1)]; |
|  |  |
|  | for(int n = 0; n < (i-1); n++) { |
|  | for(int m = 0; m < (j-1); m++) { |
|  | matrizIncidencia[n][m] = matriz[n+1][m+1]; |
|  | System.out.print(" "+matrizIncidencia[n][m]); |
|  | } |
|  | System.out.println(" "); |
|  | } |
|  |  |
|  | scanFile.close(); |
|  | } catch (FileNotFoundException e) { |
|  | e.printStackTrace(); |
|  | } |
|  |  |
|  | return matrizIncidencia; |
|  | } |
|  |  |
|  | static private LinkedHashMap<String, Integer> marcadoInicial(String pathName) { |
|  | LinkedHashMap<String, Integer> marcadoInicial = new LinkedHashMap<>(); |
|  |  |
|  | FileReader vectorMarcado; |
|  | try { |
|  | vectorMarcado = new FileReader(pathName); |
|  | Scanner scanFile = new Scanner(vectorMarcado); |
|  |  |
|  | if(!scanFile.nextLine().contains("<table")) { |
|  | scanFile.close(); |
|  | return null; |
|  | } |
|  | String tempString = scanFile.nextLine(); |
|  | while(!tempString.contains("</table")) { |
|  | if(!tempString.contains("<tr")) { |
|  | tempString = scanFile.nextLine(); |
|  | continue; |
|  | } |
|  | tempString = scanFile.nextLine(); |
|  | int index = 0; |
|  | while(!tempString.contains("</tr")) { |
|  | if(!tempString.contains("<td")) { |
|  | tempString = scanFile.nextLine(); |
|  | continue; |
|  | } |
|  | while(!tempString.contains("</td")) { |
|  | if(tempString.contains("<td class=\"colhead\">")) { |
|  | tempString = scanFile.nextLine(); |
|  | // System.out.print(" "+tempString.trim()); |
|  | marcadoInicial.put(tempString.trim(), null); |
|  | index = index + 1; |
|  | } else if(tempString.contains("<td class=\"cell\">")) { |
|  | tempString = scanFile.nextLine(); |
|  | // System.out.print(" "+tempString.trim()); |
|  | marcadoInicial.put((String)marcadoInicial.keySet().toArray()[index], Integer.valueOf(tempString.trim())); |
|  | index = index + 1; |
|  | } else { |
|  | tempString = scanFile.nextLine(); |
|  | } |
|  | } |
|  | } |
|  | // System.out.println(" "); |
|  | } |
|  |  |
|  | System.out.println(""); |
|  | scanFile.close(); |
|  | } catch (FileNotFoundException e) { |
|  | e.printStackTrace(); |
|  | } |
|  |  |
|  | return marcadoInicial; |
|  | } |
|  |  |
|  | static private ArrayList<String> getTransiciones(String pathName) { |
|  | ArrayList<String> transiciones = new ArrayList<>(); |
|  |  |
|  | FileReader matrizIPlus; |
|  | try { |
|  | matrizIPlus = new FileReader(pathName); |
|  | Scanner scanFile = new Scanner(matrizIPlus); |
|  |  |
|  | if(!scanFile.nextLine().contains("<table")) { |
|  | scanFile.close(); |
|  | return null; |
|  | } |
|  | String tempString = scanFile.nextLine(); |
|  | while(!tempString.contains("</table")) { |
|  | if(!tempString.contains("<tr")) { |
|  | tempString = scanFile.nextLine(); |
|  | continue; |
|  | } |
|  | tempString = scanFile.nextLine(); |
|  | while(!tempString.contains("</tr")) { |
|  | if(!tempString.contains("<td")) { |
|  | tempString = scanFile.nextLine(); |
|  | continue; |
|  | } |
|  | while(!tempString.contains("</td")) { |
|  | if(tempString.contains("<td class=\"colhead\">")) { |
|  | tempString = scanFile.nextLine(); |
|  | // System.out.print(" "+tempString.trim()); |
|  | transiciones.add(tempString.trim()); |
|  | } else { |
|  | tempString = scanFile.nextLine(); |
|  | } |
|  | } |
|  | } |
|  | break; |
|  | } |
|  |  |
|  | System.out.println(""); |
|  | scanFile.close(); |
|  | } catch (FileNotFoundException e) { |
|  | e.printStackTrace(); |
|  | } |
|  |  |
|  | return transiciones; |
|  | } |
|  |  |
|  | } |

* *Monitor:*

|  |  |
| --- | --- |
| Package main; |  |
|  | import java.io.PrintWriter; |
|  | import java.util.ArrayList; |
|  | import java.util.Arrays; |
|  | import java.util.Collections; |
|  | import java.util.Date; |
|  | import java.util.LinkedHashMap; |
|  | import java.util.concurrent.locks.Condition; |
|  | import java.util.concurrent.locks.ReentrantLock; |
|  |  |
|  | public class Monitor extends ConstantesComunes { |
|  |  |
|  | private Integer[][] matrizMas; |
|  | private Integer[][] matrizMenos; |
|  | private Integer[][] matrizInhibicion; |
|  |  |
|  | private Integer[] marcado; |
|  | private LinkedHashMap<String, Integer> marcadoInicial; |
|  | private ArrayList<String> plazas; |
|  | private LinkedHashMap<String, Integer> printOrder; |
|  |  |
|  | private ArrayList<String> transiciones; |
|  | private LinkedHashMap<String, Condition> colaCondicion; |
|  | private ArrayList<String> transicionesGeneradoras; |
|  |  |
|  | private LinkedHashMap<String, String> abordarTren; |
|  | private LinkedHashMap<String, String> descenderTren; |
|  |  |
|  | private ArrayList<String> estaciones; |
|  | private LinkedHashMap<String, Date> ultimaSubidaEstacion; |
|  | private Date ultimoArrivoEstacion; |
|  | private final ReentrantLock lock = new ReentrantLock(false); |
|  |  |
|  | private PrintWriter printWriter; |
|  |  |
|  | private final Condition fullTrenOrEmptyEstacion = lock.newCondition(); |
|  | private final Condition tiempoDeEspera = lock.newCondition(); |
|  | private final Condition trenArriboEstacion = lock.newCondition(); |
|  |  |
|  | private final Condition subidaEstacionA = lock.newCondition(); |
|  | private final Condition bajadaEstacionA = lock.newCondition(); |
|  | private final Condition subidaEstacionB = lock.newCondition(); |
|  | private final Condition bajadaEstacionB = lock.newCondition(); |
|  | private final Condition subidaEstacionC = lock.newCondition(); |
|  | private final Condition bajadaEstacionC = lock.newCondition(); |
|  | private final Condition subidaEstacionD = lock.newCondition(); |
|  | private final Condition bajadaEstacionD = lock.newCondition(); |
|  |  |
|  | private final Condition pasajeroGeneradorEstacionA = lock.newCondition(); |
|  | private final Condition pasajeroGeneradorEstacionB = lock.newCondition(); |
|  | private final Condition pasajeroGeneradorEstacionC = lock.newCondition(); |
|  | private final Condition pasajeroGeneradorEstacionD = lock.newCondition(); |
|  |  |
|  | private final Condition pasoDeNivelTransitoGeneradorAB = lock.newCondition(); |
|  | private final Condition pasoDeNivelTransitoGeneradorCD = lock.newCondition(); |
|  |  |
|  | private final Condition pasoDeNivelMaquina = lock.newCondition(); |
|  |  |
|  | private final Condition pasoDeNivelVagon = lock.newCondition(); |
|  |  |
|  | private final Condition liberarBarreraPasoNivelAB = lock.newCondition(); |
|  | private final Condition liberarBarreraPasoNivelCD = lock.newCondition(); |
|  |  |
|  | /\* |
|  | \* 1) Disparo red con el disparo correspondiente a la transicion representada por la condicion a la que se notifico. |
|  | \* 2) Si el disparo es completado pido Vs y Vc intersecto y uso politicas para decidir que condicion notificar (despertar hilo). |
|  | \* 3) Si el disparo no es completado libera el lock y sale del monitor. |
|  | \* |
|  | \* Nota: Las transiciones sensibilizadas que no tienen una cola de condicion asociada se disparan inmediatamente. |
|  | \* |
|  | \*/ |
|  |  |
|  | public Monitor(Integer[][] matrizMas, Integer[][] matrizMenos, Integer[][] matrizInhibicion, LinkedHashMap<String, Integer> marcado, ArrayList<String> transiciones, PrintWriter printWriter) { |
|  | this.printWriter = printWriter; |
|  | this.marcadoInicial = marcado; |
|  | this.marcado = marcado.values().toArray(new Integer[marcado.values().size()]); |
|  | this.plazas = new ArrayList<>(this.marcadoInicial.keySet()); |
|  |  |
|  | this.transiciones = transiciones; |
|  | this.matrizMas = matrizMas; |
|  | this.matrizMenos = matrizMenos; |
|  | this.matrizInhibicion = matrizInhibicion; |
|  |  |
|  | this.colaCondicion = new LinkedHashMap<>(); |
|  |  |
|  | this.colaCondicion.put(tranBajadaMaquinaAEstacionD, bajadaEstacionD); |
|  | this.colaCondicion.put(tranBajadaMaquinaBEstacionA, bajadaEstacionA); |
|  | this.colaCondicion.put(tranBajadaMaquinaCEstacionB, bajadaEstacionB); |
|  | this.colaCondicion.put(tranBajadaMaquinaDEstacionC, bajadaEstacionC); |
|  | this.colaCondicion.put(tranBajadaVagonAEstacionD, bajadaEstacionD); |
|  | this.colaCondicion.put(tranBajadaVagonBEstacionA, bajadaEstacionA); |
|  | this.colaCondicion.put(tranBajadaVagonCEstacionB, bajadaEstacionB); |
|  | this.colaCondicion.put(tranBajadaVagonDEstacionC, bajadaEstacionC); |
|  |  |
|  | this.colaCondicion.put(tranBajadaMaquinaAEstacionB, bajadaEstacionB); |
|  | this.colaCondicion.put(tranBajadaMaquinaAEstacionC, bajadaEstacionC); |
|  | this.colaCondicion.put(tranBajadaMaquinaBEstacionC, bajadaEstacionC); |
|  | this.colaCondicion.put(tranBajadaMaquinaBEstacionD, bajadaEstacionD); |
|  | this.colaCondicion.put(tranBajadaMaquinaCEstacionA, bajadaEstacionA); |
|  | this.colaCondicion.put(tranBajadaMaquinaCEstacionD, bajadaEstacionD); |
|  | this.colaCondicion.put(tranBajadaMaquinaDEstacionA, bajadaEstacionA); |
|  | this.colaCondicion.put(tranBajadaMaquinaDEstacionB, bajadaEstacionB); |
|  | this.colaCondicion.put(tranBajadaVagonAEstacionB, bajadaEstacionB); |
|  | this.colaCondicion.put(tranBajadaVagonAEstacionC, bajadaEstacionC); |
|  | this.colaCondicion.put(tranBajadaVagonBEstacionC, bajadaEstacionC); |
|  | this.colaCondicion.put(tranBajadaVagonBEstacionD, bajadaEstacionD); |
|  | this.colaCondicion.put(tranBajadaVagonCEstacionA, bajadaEstacionA); |
|  | this.colaCondicion.put(tranBajadaVagonCEstacionD, bajadaEstacionD); |
|  | this.colaCondicion.put(tranBajadaVagonDEstacionA, bajadaEstacionA); |
|  | this.colaCondicion.put(tranBajadaVagonDEstacionB, bajadaEstacionB); |
|  |  |
|  | this.colaCondicion.put(tranSubidaMaquinaEstacionA, subidaEstacionA); |
|  | this.colaCondicion.put(tranSubidaMaquinaEstacionB, subidaEstacionB); |
|  | this.colaCondicion.put(tranSubidaMaquinaEstacionC, subidaEstacionC); |
|  | this.colaCondicion.put(tranSubidaMaquinaEstacionD, subidaEstacionD); |
|  | this.colaCondicion.put(tranSubidaVagonEstacionA, subidaEstacionA); |
|  | this.colaCondicion.put(tranSubidaVagonEstacionB, subidaEstacionB); |
|  | this.colaCondicion.put(tranSubidaVagonEstacionC, subidaEstacionC); |
|  | this.colaCondicion.put(tranSubidaVagonEstacionD, subidaEstacionD); |
|  |  |
|  | this.colaCondicion.put(tranTrenLlenoA, fullTrenOrEmptyEstacion); |
|  | this.colaCondicion.put(tranTrenLlenoB, fullTrenOrEmptyEstacion); |
|  | this.colaCondicion.put(tranTrenLlenoC, fullTrenOrEmptyEstacion); |
|  | this.colaCondicion.put(tranTrenLlenoD, fullTrenOrEmptyEstacion); |
|  |  |
|  | this.colaCondicion.put(tranEstacionVaciaA, fullTrenOrEmptyEstacion); |
|  | this.colaCondicion.put(tranEstacionVaciaB, fullTrenOrEmptyEstacion); |
|  | this.colaCondicion.put(tranEstacionVaciaC, fullTrenOrEmptyEstacion); |
|  | this.colaCondicion.put(tranEstacionVaciaD, fullTrenOrEmptyEstacion); |
|  |  |
|  | this.colaCondicion.put(tranTrenEsperandoA, tiempoDeEspera); |
|  | this.colaCondicion.put(tranTrenEsperandoB, tiempoDeEspera); |
|  | this.colaCondicion.put(tranTrenEsperandoC, tiempoDeEspera); |
|  | this.colaCondicion.put(tranTrenEsperandoD, tiempoDeEspera); |
|  |  |
|  | this.colaCondicion.put(tranTrenArriboA, trenArriboEstacion); |
|  | this.colaCondicion.put(tranTrenArriboB, trenArriboEstacion); |
|  | this.colaCondicion.put(tranTrenArriboC, trenArriboEstacion); |
|  | this.colaCondicion.put(tranTrenArriboD, trenArriboEstacion); |
|  |  |
|  | this.colaCondicion.put(tranRecorridoTrenAB, fullTrenOrEmptyEstacion); |
|  | this.colaCondicion.put(tranRecorridoTrenCD, fullTrenOrEmptyEstacion); |
|  |  |
|  | this.colaCondicion.put(tranPasajerosAGenerador, pasajeroGeneradorEstacionA); |
|  | this.colaCondicion.put(tranPasajerosBGenerador, pasajeroGeneradorEstacionB); |
|  | this.colaCondicion.put(tranPasajerosCGenerador, pasajeroGeneradorEstacionC); |
|  | this.colaCondicion.put(tranPasajerosDGenerador, pasajeroGeneradorEstacionD); |
|  |  |
|  |  |
|  |  |
|  | this.colaCondicion.put(tranPasoNivelABMaquinaReady, liberarBarreraPasoNivelAB); |
|  | this.colaCondicion.put(tranPasoNivelCDMaquinaReady, liberarBarreraPasoNivelCD); |
|  | this.colaCondicion.put(tranPasoNivelABMaquinaWait, pasoDeNivelMaquina); |
|  | this.colaCondicion.put(tranPasoNivelCDMaquinaWait, pasoDeNivelMaquina); |
|  |  |
|  | this.colaCondicion.put(tranPasoNivelABVagonReady, liberarBarreraPasoNivelAB); |
|  | this.colaCondicion.put(tranPasoNivelCDVagonReady, liberarBarreraPasoNivelCD); |
|  | this.colaCondicion.put(tranPasoNivelABVagonWait, pasoDeNivelVagon); |
|  | this.colaCondicion.put(tranPasoNivelCDVagonWait, pasoDeNivelVagon); |
|  |  |
|  | this.colaCondicion.put(tranPasoNivelABTransitoReady, liberarBarreraPasoNivelAB); |
|  | this.colaCondicion.put(tranPasoNivelCDTransitoReady, liberarBarreraPasoNivelCD); |
|  | this.colaCondicion.put(tranPasoNivelABTransitoWait, liberarBarreraPasoNivelAB); |
|  | this.colaCondicion.put(tranPasoNivelCDTransitoWait, liberarBarreraPasoNivelCD); |
|  |  |
|  | this.colaCondicion.put(tranPasoNivelABTransitoGenerador, pasoDeNivelTransitoGeneradorAB); |
|  | this.colaCondicion.put(tranPasoNivelCDTransitoGenerador, pasoDeNivelTransitoGeneradorCD); |
|  |  |
|  |  |
|  | this.abordarTren = new LinkedHashMap<>(); |
|  | this.abordarTren.put(tranSubidaMaquinaEstacionA, trenEstacionA); |
|  | this.abordarTren.put(tranSubidaVagonEstacionA, trenEstacionA); |
|  | this.abordarTren.put(tranSubidaMaquinaEstacionB, trenEstacionB); |
|  | this.abordarTren.put(tranSubidaVagonEstacionB, trenEstacionB); |
|  | this.abordarTren.put(tranSubidaMaquinaEstacionC, trenEstacionC); |
|  | this.abordarTren.put(tranSubidaVagonEstacionC, trenEstacionC); |
|  | this.abordarTren.put(tranSubidaMaquinaEstacionD, trenEstacionD); |
|  | this.abordarTren.put(tranSubidaVagonEstacionD, trenEstacionD); |
|  |  |
|  |  |
|  | this.descenderTren = new LinkedHashMap<>(); |
|  |  |
|  | this.descenderTren.put(tranBajadaMaquinaBEstacionA, trenEstacionA); |
|  | this.descenderTren.put(tranBajadaVagonBEstacionA, trenEstacionA); |
|  |  |
|  | this.descenderTren.put(tranBajadaMaquinaCEstacionB, trenEstacionB); |
|  | this.descenderTren.put(tranBajadaVagonCEstacionB, trenEstacionB); |
|  |  |
|  | this.descenderTren.put(tranBajadaMaquinaDEstacionC, trenEstacionC); |
|  | this.descenderTren.put(tranBajadaVagonDEstacionC, trenEstacionC); |
|  |  |
|  | this.descenderTren.put(tranBajadaMaquinaAEstacionD, trenEstacionD); |
|  | this.descenderTren.put(tranBajadaVagonAEstacionD, trenEstacionD); |
|  |  |
|  |  |
|  | this.descenderTren.put(tranBajadaMaquinaCEstacionA, trenEstacionA); |
|  | this.descenderTren.put(tranBajadaMaquinaDEstacionA, trenEstacionA); |
|  | this.descenderTren.put(tranBajadaVagonCEstacionA, trenEstacionA); |
|  | this.descenderTren.put(tranBajadaVagonDEstacionA, trenEstacionA); |
|  |  |
|  | this.descenderTren.put(tranBajadaMaquinaAEstacionB, trenEstacionB); |
|  | this.descenderTren.put(tranBajadaMaquinaDEstacionB, trenEstacionB); |
|  | this.descenderTren.put(tranBajadaVagonAEstacionB, trenEstacionB); |
|  | this.descenderTren.put(tranBajadaVagonDEstacionB, trenEstacionB); |
|  |  |
|  | this.descenderTren.put(tranBajadaMaquinaAEstacionC, trenEstacionC); |
|  | this.descenderTren.put(tranBajadaMaquinaBEstacionC, trenEstacionC); |
|  | this.descenderTren.put(tranBajadaVagonAEstacionC, trenEstacionC); |
|  | this.descenderTren.put(tranBajadaVagonBEstacionC, trenEstacionC); |
|  |  |
|  | this.descenderTren.put(tranBajadaMaquinaBEstacionD, trenEstacionD); |
|  | this.descenderTren.put(tranBajadaMaquinaCEstacionD, trenEstacionD); |
|  | this.descenderTren.put(tranBajadaVagonBEstacionD, trenEstacionD); |
|  | this.descenderTren.put(tranBajadaVagonCEstacionD, trenEstacionD); |
|  |  |
|  |  |
|  | estaciones = new ArrayList<>(Arrays.asList(estacion)); |
|  | this.ultimaSubidaEstacion = new LinkedHashMap<>(); |
|  | Date fechaActual = new Date(); |
|  | ultimaSubidaEstacion.put(trenEstacionA, fechaActual); |
|  | ultimaSubidaEstacion.put(trenEstacionB, fechaActual); |
|  | ultimaSubidaEstacion.put(trenEstacionC, fechaActual); |
|  | ultimaSubidaEstacion.put(trenEstacionD, fechaActual); |
|  |  |
|  | ultimoArrivoEstacion = fechaActual; |
|  |  |
|  |  |
|  | printOrder = new LinkedHashMap<>(); |
|  | printOrder.put(maq, plazas.indexOf(maq)); |
|  | printOrder.put(vag, plazas.indexOf(vag)); |
|  |  |
|  | printOrder.put(trenEstacionA, plazas.indexOf(trenEstacionA)); |
|  | printOrder.put(maqA, plazas.indexOf(maqA)); |
|  | printOrder.put(vagA, plazas.indexOf(vagA)); |
|  | printOrder.put(pasajerosEsperandoSubidaA, plazas.indexOf(pasajerosEsperandoSubidaA)); |
|  |  |
|  | printOrder.put(trenEstacionB, plazas.indexOf(trenEstacionB)); |
|  | printOrder.put(maqB, plazas.indexOf(maqB)); |
|  | printOrder.put(vagB, plazas.indexOf(vagB)); |
|  | printOrder.put(pasajerosEsperandoSubidaB, plazas.indexOf(pasajerosEsperandoSubidaB)); |
|  |  |
|  | printOrder.put(trenEstacionC, plazas.indexOf(trenEstacionC)); |
|  | printOrder.put(maqC, plazas.indexOf(maqC)); |
|  | printOrder.put(vagC, plazas.indexOf(vagC)); |
|  | printOrder.put(pasajerosEsperandoSubidaC, plazas.indexOf(pasajerosEsperandoSubidaC)); |
|  |  |
|  | printOrder.put(trenEstacionD, plazas.indexOf(trenEstacionD)); |
|  | printOrder.put(maqD, plazas.indexOf(maqD)); |
|  | printOrder.put(vagD, plazas.indexOf(vagD)); |
|  | printOrder.put(pasajerosEsperandoSubidaD, plazas.indexOf(pasajerosEsperandoSubidaD)); |
|  |  |
|  | printOrder.put(trenEstacionAArribo, plazas.indexOf(trenEstacionAArribo)); |
|  | printOrder.put(trenEstacionAEspera, plazas.indexOf(trenEstacionAEspera)); |
|  | printOrder.put(trenEstacionAPartida, plazas.indexOf(trenEstacionAPartida)); |
|  | printOrder.put(pasoNivelABMaquinaEsperando, plazas.indexOf(pasoNivelABMaquinaEsperando)); |
|  | printOrder.put(pasoNivelABMaquina, plazas.indexOf(pasoNivelABMaquina)); |
|  | printOrder.put(pasoNivelABMaquinaUnion, plazas.indexOf(pasoNivelABMaquinaUnion)); |
|  | printOrder.put(pasoNivelABVagonEsperando, plazas.indexOf(pasoNivelABVagonEsperando)); |
|  | printOrder.put(pasoNivelABVagon, plazas.indexOf(pasoNivelABVagon)); |
|  | printOrder.put(pasoNivelABVagonUnion, plazas.indexOf(pasoNivelABVagonUnion)); |
|  |  |
|  | printOrder.put(trenEstacionBEspera, plazas.indexOf(trenEstacionBEspera)); |
|  | printOrder.put(trenEstacionBPartida, plazas.indexOf(trenEstacionBPartida)); |
|  | printOrder.put(trenEstacionBArribo, plazas.indexOf(trenEstacionBArribo)); |
|  |  |
|  | printOrder.put(trenEstacionCArribo, plazas.indexOf(trenEstacionCArribo)); |
|  | printOrder.put(trenEstacionCEspera, plazas.indexOf(trenEstacionCEspera)); |
|  | printOrder.put(trenEstacionCPartida, plazas.indexOf(trenEstacionCPartida)); |
|  | printOrder.put(pasoNivelCDMaquinaEsperando, plazas.indexOf(pasoNivelCDMaquinaEsperando)); |
|  | printOrder.put(pasoNivelCDMaquina, plazas.indexOf(pasoNivelCDMaquina)); |
|  | printOrder.put(pasoNivelCDMaquinaUnion, plazas.indexOf(pasoNivelCDMaquinaUnion)); |
|  | printOrder.put(pasoNivelCDVagonEsperando, plazas.indexOf(pasoNivelCDVagonEsperando)); |
|  | printOrder.put(pasoNivelCDVagon, plazas.indexOf(pasoNivelCDVagon)); |
|  | printOrder.put(pasoNivelCDVagonUnion, plazas.indexOf(pasoNivelCDVagonUnion)); |
|  |  |
|  | printOrder.put(trenEstacionDEspera, plazas.indexOf(trenEstacionDEspera)); |
|  | printOrder.put(trenEstacionDPartida, plazas.indexOf(trenEstacionDPartida)); |
|  | printOrder.put(trenEstacionDArribo, plazas.indexOf(trenEstacionDArribo)); |
|  |  |
|  | printOrder.put(pasoNivelABBarrera, plazas.indexOf(pasoNivelABBarrera)); |
|  | printOrder.put(pasoNivelABTransitoEsperando, plazas.indexOf(pasoNivelABTransitoEsperando)); |
|  | printOrder.put(pasoNivelABTransito, plazas.indexOf(pasoNivelABTransito)); |
|  |  |
|  | printOrder.put(pasoNivelCDBarrera, plazas.indexOf(pasoNivelCDBarrera)); |
|  | printOrder.put(pasoNivelCDTransitoEsperando, plazas.indexOf(pasoNivelCDTransitoEsperando)); |
|  | printOrder.put(pasoNivelCDTransito, plazas.indexOf(pasoNivelCDTransito)); |
|  |  |
|  |  |
|  | transicionesGeneradoras = new ArrayList<>(); |
|  | transicionesGeneradoras.add(tranPasajerosAGenerador); |
|  | transicionesGeneradoras.add(tranPasajerosBGenerador); |
|  | transicionesGeneradoras.add(tranPasajerosCGenerador); |
|  | transicionesGeneradoras.add(tranPasajerosDGenerador); |
|  | transicionesGeneradoras.add(tranPasoNivelABTransitoGenerador); |
|  | transicionesGeneradoras.add(tranPasoNivelCDTransitoGenerador); |
|  | } |
|  |  |
|  | public Long continuarRecorridoTren() throws InterruptedException { |
|  | lock.lock(); |
|  |  |
|  | try { |
|  | Date fechaActual = new Date(); |
|  | while( ( |
|  | marcado[plazas.indexOf(trenEstacionAEspera)] == 0 && marcado[plazas.indexOf(trenEstacionBEspera)] == 0 && |
|  | marcado[plazas.indexOf(trenEstacionCEspera)] == 0 && marcado[plazas.indexOf(trenEstacionDEspera)] == 0 |
|  | ) || ( |
|  | ( |
|  | marcado[plazas.indexOf(trenEstacionAEspera)] == 1 || marcado[plazas.indexOf(trenEstacionBEspera)] == 1 || |
|  | marcado[plazas.indexOf(trenEstacionCEspera)] == 1 || marcado[plazas.indexOf(trenEstacionDEspera)] == 1 |
|  | ) && |
|  | (fechaActual.getTime() - ultimoArrivoEstacion.getTime()) < 10000 |
|  | ) |
|  | ) { |
|  | // tiempoDeEspera.awaitNanos((10000 - (new Date().getTime() - ultimoArrivoEstacion.getTime())) \* 1000); |
|  | if((fechaActual.getTime() - ultimoArrivoEstacion.getTime()) < 10000) { |
|  | return (10000L - ( (new Date()).getTime() - ultimoArrivoEstacion.getTime() ) ); |
|  | } |
|  | tiempoDeEspera.await(); |
|  | fechaActual = new Date(); |
|  | } |
|  |  |
|  | for(String estacion: estaciones) { |
|  | if(marcado[plazas.indexOf(estacion + trenEstacionAEspera.substring(1, trenEstacionAEspera.length()))] == 1) { |
|  | dispararRed(estacion + tranTrenEsperandoA.substring(1, tranTrenEsperandoA.length())); |
|  | break; |
|  | } |
|  | } |
|  |  |
|  | String transicion = interseccionPrioritarias(); |
|  | if(transicion != null) { |
|  | colaCondicion.get(transicion).signal(); |
|  | } |
|  | } finally { |
|  | lock.unlock(); |
|  | } |
|  | return 0L; |
|  | } |
|  |  |
|  | public void partidaTren() throws InterruptedException { |
|  | lock.lock(); |
|  |  |
|  | try { |
|  | while( ( marcado[plazas.indexOf(trenEstacionAPartida)] == 0 && marcado[plazas.indexOf(trenEstacionBPartida)] == 0 && |
|  | marcado[plazas.indexOf(trenEstacionCPartida)] == 0 && marcado[plazas.indexOf(trenEstacionDPartida)] == 0 && |
|  | (marcado[plazas.indexOf(pasoNivelABMaquinaUnion)] == 0 || marcado[plazas.indexOf(pasoNivelABVagonUnion)] == 0) && |
|  | (marcado[plazas.indexOf(pasoNivelCDMaquinaUnion)] == 0 || marcado[plazas.indexOf(pasoNivelCDVagonUnion)] == 0) |
|  | ) || |
|  |  |
|  | ( (marcado[plazas.indexOf(trenEstacionAPartida)] == 0 && marcado[plazas.indexOf(trenEstacionBPartida)] == 0 && |
|  | marcado[plazas.indexOf(trenEstacionCPartida)] == 0 && marcado[plazas.indexOf(trenEstacionDPartida)] == 0) && |
|  | !(marcado[plazas.indexOf(pasoNivelABMaquinaUnion)] == 1 && marcado[plazas.indexOf(pasoNivelABVagonUnion)] == 1 || |
|  | marcado[plazas.indexOf(pasoNivelCDMaquinaUnion)] == 1 && marcado[plazas.indexOf(pasoNivelCDVagonUnion)] == 1) |
|  | ) || |
|  | ( |
|  | (marcado[plazas.indexOf(trenEstacionAPartida)] == 1 || marcado[plazas.indexOf(trenEstacionBPartida)] == 1 || |
|  | marcado[plazas.indexOf(trenEstacionCPartida)] == 1 || marcado[plazas.indexOf(trenEstacionDPartida)] == 1) && ( |
|  | (marcado[plazas.indexOf(trenEstacionAPartida)] == 1 && marcado[plazas.indexOf(pasajerosEsperandoSubidaA)] != 0 || |
|  | marcado[plazas.indexOf(trenEstacionBPartida)] == 1 && marcado[plazas.indexOf(pasajerosEsperandoSubidaB)] != 0 || |
|  | marcado[plazas.indexOf(trenEstacionCPartida)] == 1 && marcado[plazas.indexOf(pasajerosEsperandoSubidaC)] != 0 || |
|  | marcado[plazas.indexOf(trenEstacionDPartida)] == 1 && marcado[plazas.indexOf(pasajerosEsperandoSubidaD)] != 0) && |
|  | (marcado[plazas.indexOf(vag)] != 0 || marcado[plazas.indexOf(maq)] != 0) |
|  | ) ) ) { |
|  | fullTrenOrEmptyEstacion.await(); |
|  | } |
|  |  |
|  |  |
|  | if(marcado[plazas.indexOf(pasoNivelABMaquinaUnion)] == 1 && marcado[plazas.indexOf(pasoNivelABVagonUnion)] == 1) { |
|  | dispararRed(tranRecorridoTrenAB); |
|  | } else if(marcado[plazas.indexOf(pasoNivelCDMaquinaUnion)] == 1 && marcado[plazas.indexOf(pasoNivelCDVagonUnion)] == 1) { |
|  | dispararRed(tranRecorridoTrenCD); |
|  | } else { |
|  | for(String estacion: estaciones) { |
|  | if(marcado[plazas.indexOf(estacion + trenEstacionAPartida.substring(1))] == 1) { |
|  | if(marcado[plazas.indexOf(vag)] == 0 && marcado[plazas.indexOf(maq)] == 0) { |
|  | dispararRed(estacion + tranTrenLlenoA.substring(1)); |
|  | break; |
|  | } |
|  | if(marcado[plazas.indexOf(pasajerosEsperandoSubidaA.substring(0, 1) + estacion + pasajerosEsperandoSubidaA.substring(pasajerosEsperandoSubidaA.length()-1) )] == 0) { |
|  | dispararRed(estacion + tranEstacionVaciaA.substring(1)); |
|  | break; |
|  | } |
|  |  |
|  | } |
|  | } |
|  | } |
|  |  |
|  | String transicion = interseccionPrioritarias(); |
|  | if(transicion != null) { |
|  | colaCondicion.get(transicion).signal(); |
|  | } |
|  | } finally { |
|  | lock.unlock(); |
|  | } |
|  | } |
|  |  |
|  | public void arrivoTrenEstacion() throws InterruptedException { |
|  | lock.lock(); |
|  |  |
|  | try { |
|  | while(( marcado[plazas.indexOf(trenEstacionAArribo)] == 0 && |
|  | marcado[plazas.indexOf(trenEstacionBArribo)] == 0 && |
|  | marcado[plazas.indexOf(trenEstacionCArribo)] == 0 && |
|  | marcado[plazas.indexOf(trenEstacionDArribo)] == 0 |
|  | ) ) { |
|  | trenArriboEstacion.await(); |
|  | } |
|  |  |
|  | for(String estacionActual: estaciones) { |
|  | if(marcado[plazas.indexOf(trenEstacionAArribo.substring(0 ,1) + estacion[(estaciones.indexOf(estacionActual)+3)%4] + estacionActual + trenEstacionAArribo.substring(trenEstacionAArribo.length() - 1))] == 1) { |
|  | if(dispararRed(estacionActual + tranTrenArribo)) { |
|  | ultimoArrivoEstacion = new Date(); |
|  | break; |
|  | } |
|  | } |
|  | } |
|  |  |
|  | String transicion = interseccionPrioritarias(); |
|  | if(transicion != null) { |
|  | colaCondicion.get(transicion).signal(); |
|  | } |
|  | } finally { |
|  | lock.unlock(); |
|  | } |
|  | } |
|  |  |
|  | public void abordarTren() throws InterruptedException { |
|  | lock.lock(); |
|  |  |
|  | String threadName = Thread.currentThread().getName(); |
|  |  |
|  | try { |
|  | while( trenEstacionA.endsWith(threadName.substring(threadName.length() - 1)) && (marcado[plazas.indexOf(trenEstacionA)] != 0 || |
|  | marcado[plazas.indexOf(maq)] == 0 && marcado[plazas.indexOf(vag)] == 0 || marcado[plazas.indexOf(pasajerosEsperandoSubidaA)] == 0) ) { |
|  | subidaEstacionA.await(); |
|  | } |
|  | while( trenEstacionB.endsWith(threadName.substring(threadName.length() - 1)) && (marcado[plazas.indexOf(trenEstacionB)] != 0 || |
|  | marcado[plazas.indexOf(maq)] == 0 && marcado[plazas.indexOf(vag)] == 0 || marcado[plazas.indexOf(pasajerosEsperandoSubidaB)] == 0) ) { |
|  | subidaEstacionB.await(); |
|  | } |
|  | while( trenEstacionC.endsWith(threadName.substring(threadName.length() - 1)) && (marcado[plazas.indexOf(trenEstacionC)] != 0 || |
|  | marcado[plazas.indexOf(maq)] == 0 && marcado[plazas.indexOf(vag)] == 0 || marcado[plazas.indexOf(pasajerosEsperandoSubidaC)] == 0) ) { |
|  | subidaEstacionC.await(); |
|  | } |
|  | while( trenEstacionD.endsWith(threadName.substring(threadName.length() - 1)) && (marcado[plazas.indexOf(trenEstacionD)] != 0 || |
|  | marcado[plazas.indexOf(maq)] == 0 && marcado[plazas.indexOf(vag)] == 0 || marcado[plazas.indexOf(pasajerosEsperandoSubidaD)] == 0) ) { |
|  | subidaEstacionD.await(); |
|  | } |
|  |  |
|  | boolean disparoExitoso = false; |
|  | ArrayList<String> listaSubidas = new ArrayList<>(Arrays.asList(abordarTren.keySet().toArray(new String[abordarTren.keySet().size()]))); |
|  | for(String subida: listaSubidas) { |
|  | if( marcado[plazas.indexOf(subida.startsWith("SM")? maq : vag)] != 0 && |
|  | marcado[plazas.indexOf(trenEstacion + threadName.substring(threadName.length() - 1))] == 0 && |
|  | abordarTren.get(subida).endsWith(threadName.substring(threadName.length() - 1))) { |
|  | disparoExitoso = dispararRed(subida); |
|  | if(disparoExitoso) { |
|  | ultimaSubidaEstacion.put(trenEstacion + threadName.substring(threadName.length() - 1), new Date()); |
|  | break; |
|  | } |
|  | } |
|  | } |
|  |  |
|  |  |
|  | String transicion = interseccionPrioritarias(); |
|  | if(transicion != null) { |
|  | colaCondicion.get(transicion).signal(); |
|  | } |
|  | } finally { |
|  | lock.unlock(); |
|  | } |
|  | } |
|  |  |
|  | public Long descenderTren() throws InterruptedException { |
|  | lock.lock(); |
|  |  |
|  | String threadName = Thread.currentThread().getName(); |
|  |  |
|  | try { |
|  | String estacionAnteriorTren = estacion[(estaciones.indexOf(threadName.substring(threadName.length() - 1)) + 3)%4]; |
|  | String estacionOpuestaTren = estacion[(estaciones.indexOf(threadName.substring(threadName.length() - 1)) + 2)%4]; |
|  |  |
|  | Date actual = new Date(); |
|  |  |
|  | while( trenEstacionA.endsWith(threadName.substring(threadName.length() - 1)) && (marcado[plazas.indexOf(trenEstacionA)] != 0 || |
|  | (marcado[plazas.indexOf(maqB)] == 0 && marcado[plazas.indexOf(vagB)] == 0 && |
|  | (marcado[plazas.indexOf(maqC)] == 0 && marcado[plazas.indexOf(vagC)] == 0 || ultimaSubidaEstacion.get(trenEstacion + estacionOpuestaTren).getTime() > actual.getTime()) && |
|  | (marcado[plazas.indexOf(maqD)] == 0 && marcado[plazas.indexOf(vagD)] == 0 || ultimaSubidaEstacion.get(trenEstacion + estacionAnteriorTren).getTime() > actual.getTime()) ) ) ) { |
|  |  |
|  | if(ultimaSubidaEstacion.get(trenEstacion + estacionOpuestaTren).getTime() > actual.getTime() && ultimaSubidaEstacion.get(trenEstacion + estacionAnteriorTren).getTime() > actual.getTime()) { |
|  | if(ultimaSubidaEstacion.get(trenEstacion + estacionOpuestaTren).getTime() > ultimaSubidaEstacion.get(trenEstacion + estacionAnteriorTren).getTime()) { |
|  | return ultimaSubidaEstacion.get(trenEstacion + estacionAnteriorTren).getTime() - actual.getTime(); |
|  | } else { |
|  | return ultimaSubidaEstacion.get(trenEstacion + estacionOpuestaTren).getTime() - actual.getTime(); |
|  | } |
|  | } else if(ultimaSubidaEstacion.get(trenEstacion + estacionOpuestaTren).getTime() > actual.getTime()) { |
|  | return ultimaSubidaEstacion.get(trenEstacion + estacionOpuestaTren).getTime() - actual.getTime(); |
|  | } else if(ultimaSubidaEstacion.get(trenEstacion + estacionAnteriorTren).getTime() > actual.getTime()) { |
|  | return ultimaSubidaEstacion.get(trenEstacion + estacionAnteriorTren).getTime() - actual.getTime(); |
|  | } |
|  | bajadaEstacionA.await(); |
|  | actual = new Date(); |
|  | } |
|  | while( trenEstacionB.endsWith(threadName.substring(threadName.length() - 1)) && (marcado[plazas.indexOf(trenEstacionB)] != 0 || |
|  | (marcado[plazas.indexOf(maqC)] == 0 && marcado[plazas.indexOf(vagC)] == 0 && |
|  | (marcado[plazas.indexOf(maqD)] == 0 && marcado[plazas.indexOf(vagD)] == 0 || ultimaSubidaEstacion.get(trenEstacion + estacionOpuestaTren).getTime() > actual.getTime()) && |
|  | (marcado[plazas.indexOf(maqA)] == 0 && marcado[plazas.indexOf(vagA)] == 0 || ultimaSubidaEstacion.get(trenEstacion + estacionAnteriorTren).getTime() > actual.getTime()) ) ) ) { |
|  |  |
|  | if(ultimaSubidaEstacion.get(trenEstacion + estacionOpuestaTren).getTime() > actual.getTime() && ultimaSubidaEstacion.get(trenEstacion + estacionAnteriorTren).getTime() > actual.getTime()) { |
|  | if(ultimaSubidaEstacion.get(trenEstacion + estacionOpuestaTren).getTime() > ultimaSubidaEstacion.get(trenEstacion + estacionAnteriorTren).getTime()) { |
|  | return ultimaSubidaEstacion.get(trenEstacion + estacionAnteriorTren).getTime() - actual.getTime(); |
|  | } else { |
|  | return ultimaSubidaEstacion.get(trenEstacion + estacionOpuestaTren).getTime() - actual.getTime(); |
|  | } |
|  | } else if(ultimaSubidaEstacion.get(trenEstacion + estacionOpuestaTren).getTime() > actual.getTime()) { |
|  | return ultimaSubidaEstacion.get(trenEstacion + estacionOpuestaTren).getTime() - actual.getTime(); |
|  | } else if(ultimaSubidaEstacion.get(trenEstacion + estacionAnteriorTren).getTime() > actual.getTime()) { |
|  | return ultimaSubidaEstacion.get(trenEstacion + estacionAnteriorTren).getTime() - actual.getTime(); |
|  | } |
|  |  |
|  | bajadaEstacionB.await(); |
|  | actual = new Date(); |
|  | } |
|  | while( trenEstacionC.endsWith(threadName.substring(threadName.length() - 1)) && (marcado[plazas.indexOf(trenEstacionC)] != 0 || |
|  | (marcado[plazas.indexOf(maqD)] == 0 && marcado[plazas.indexOf(vagD)] == 0 && |
|  | (marcado[plazas.indexOf(maqA)] == 0 && marcado[plazas.indexOf(vagA)] == 0 || ultimaSubidaEstacion.get(trenEstacion + estacionOpuestaTren).getTime() > actual.getTime()) && |
|  | (marcado[plazas.indexOf(maqB)] == 0 && marcado[plazas.indexOf(vagB)] == 0 || ultimaSubidaEstacion.get(trenEstacion + estacionAnteriorTren).getTime() > actual.getTime()) ) ) ) { |
|  |  |
|  | if(ultimaSubidaEstacion.get(trenEstacion + estacionOpuestaTren).getTime() > actual.getTime() && ultimaSubidaEstacion.get(trenEstacion + estacionAnteriorTren).getTime() > actual.getTime()) { |
|  | if(ultimaSubidaEstacion.get(trenEstacion + estacionOpuestaTren).getTime() > ultimaSubidaEstacion.get(trenEstacion + estacionAnteriorTren).getTime()) { |
|  | return ultimaSubidaEstacion.get(trenEstacion + estacionAnteriorTren).getTime() - actual.getTime(); |
|  | } else { |
|  | return ultimaSubidaEstacion.get(trenEstacion + estacionOpuestaTren).getTime() - actual.getTime(); |
|  | } |
|  | } else if(ultimaSubidaEstacion.get(trenEstacion + estacionOpuestaTren).getTime() > actual.getTime()) { |
|  | return ultimaSubidaEstacion.get(trenEstacion + estacionOpuestaTren).getTime() - actual.getTime(); |
|  | } else if(ultimaSubidaEstacion.get(trenEstacion + estacionAnteriorTren).getTime() > actual.getTime()) { |
|  | return ultimaSubidaEstacion.get(trenEstacion + estacionAnteriorTren).getTime() - actual.getTime(); |
|  | } |
|  |  |
|  | bajadaEstacionC.await(); |
|  | actual = new Date(); |
|  | } |
|  | while( trenEstacionD.endsWith(threadName.substring(threadName.length() - 1)) && (marcado[plazas.indexOf(trenEstacionD)] != 0 || |
|  | (marcado[plazas.indexOf(maqA)] == 0 && marcado[plazas.indexOf(vagA)] == 0 && |
|  | (marcado[plazas.indexOf(maqB)] == 0 && marcado[plazas.indexOf(vagB)] == 0 || ultimaSubidaEstacion.get(trenEstacion + estacionOpuestaTren).getTime() > actual.getTime()) && |
|  | (marcado[plazas.indexOf(maqC)] == 0 && marcado[plazas.indexOf(vagC)] == 0 || ultimaSubidaEstacion.get(trenEstacion + estacionAnteriorTren).getTime() > actual.getTime()) ) ) ) { |
|  |  |
|  | if(ultimaSubidaEstacion.get(trenEstacion + estacionOpuestaTren).getTime() > actual.getTime() && ultimaSubidaEstacion.get(trenEstacion + estacionAnteriorTren).getTime() > actual.getTime()) { |
|  | if(ultimaSubidaEstacion.get(trenEstacion + estacionOpuestaTren).getTime() > ultimaSubidaEstacion.get(trenEstacion + estacionAnteriorTren).getTime()) { |
|  | return ultimaSubidaEstacion.get(trenEstacion + estacionAnteriorTren).getTime() - actual.getTime(); |
|  | } else { |
|  | return ultimaSubidaEstacion.get(trenEstacion + estacionOpuestaTren).getTime() - actual.getTime(); |
|  | } |
|  | } else if(ultimaSubidaEstacion.get(trenEstacion + estacionOpuestaTren).getTime() > actual.getTime()) { |
|  | return ultimaSubidaEstacion.get(trenEstacion + estacionOpuestaTren).getTime() - actual.getTime(); |
|  | } else if(ultimaSubidaEstacion.get(trenEstacion + estacionAnteriorTren).getTime() > actual.getTime()) { |
|  | return ultimaSubidaEstacion.get(trenEstacion + estacionAnteriorTren).getTime() - actual.getTime(); |
|  | } |
|  |  |
|  | bajadaEstacionD.await(); |
|  | actual = new Date(); |
|  | } |
|  |  |
|  |  |
|  | // pasajerosAnterior calcula bajadas estocasticas para los pasajeros subidos en la estacion anterior utilizando la hora de la ultima subida en esa estacion |
|  | // la estacion anterior se obtiene buscando en la lista de estaciones la estacion correspondiente al indice anterior al de la estacion actual. En el caso en que el indice |
|  | // sea 0 la estacion anterior se encontraria en el indice -1 si ultilizamos un offset negativo por lo que se suma el tamanio del array y se calcula el modulo para que el indice |
|  | // siempre este dentro del array. (-1 + 4) = 3 para la estacion anterior y (-2 + 4) = 2 para la estacion opuesta de esta forma usamos al array como un anillo (campo finito cerrado o campo de Galois) |
|  |  |
|  | ArrayList<String> listaBajadas = new ArrayList<>(Arrays.asList(descenderTren.keySet().toArray(new String[descenderTren.keySet().size()]))); |
|  | for(String bajada: listaBajadas) { |
|  | if( bajada.startsWith("B"+ threadName.substring(threadName.length() - 1)) && // Si el thread baja pasajeros en la estacion de la tansicion |
|  | marcado[plazas.indexOf(trenEstacion + threadName.substring(threadName.length() - 1))] == 0 && // Si el tren se encuentra en la estacion del thread |
|  | marcado[plazas.indexOf(bajada.substring(bajada.length() - 2, bajada.length()))] != 0 && ( // Si hay pasajeros viajando desde la estacion de la transicion |
|  | bajada.endsWith(estacionAnteriorTren) && ultimaSubidaEstacion.get(trenEstacion + estacionAnteriorTren).getTime() <= actual.getTime() || // Si la transicion baja pasajeros de la estacion anterior |
|  | bajada.endsWith(estacionOpuestaTren) && ultimaSubidaEstacion.get(trenEstacion + estacionOpuestaTren).getTime() <= actual.getTime() || // Si la transicion baja pasajeros de la estacion opuesta |
|  | bajada.endsWith(estacion[(estaciones.indexOf(threadName.substring(threadName.length() - 1)) + 1)%4]) // Si la transicion baja pasajeros de la estacion siguiente |
|  | ) ) { |
|  | if(dispararRed(bajada)) { |
|  | if(bajada.endsWith(estacionAnteriorTren)) { |
|  | int tiempoEsperadoAnterior = TiempoDeEspera.getInstance(5000, 97L).getNextRandom(); |
|  | ultimaSubidaEstacion.put(trenEstacion + estacionAnteriorTren, new Date(ultimaSubidaEstacion.get(trenEstacion + estacionAnteriorTren).getTime() + tiempoEsperadoAnterior)); |
|  | } |
|  | if(bajada.endsWith(estacionOpuestaTren)) { |
|  | int tiempoEsperadoOpuesta = TiempoDeEspera.getInstance(5000, 97L).getNextRandom(); |
|  | ultimaSubidaEstacion.put(trenEstacion + estacionOpuestaTren, new Date(ultimaSubidaEstacion.get(trenEstacion + estacionOpuestaTren).getTime() + tiempoEsperadoOpuesta)); |
|  | } |
|  |  |
|  | break; |
|  | } |
|  | } |
|  | } |
|  |  |
|  | String transicion = interseccionPrioritarias(); |
|  | if(transicion != null) { |
|  | colaCondicion.get(transicion).signal(); |
|  | } |
|  | } finally { |
|  | lock.unlock(); |
|  | } |
|  | return 0L; |
|  | } |
|  |  |
|  |  |
|  | public void cruzarPasoNivel() throws InterruptedException { |
|  | lock.lock(); |
|  |  |
|  | String threadName = Thread.currentThread().getName(); |
|  |  |
|  | try { |
|  |  |
|  | while( threadName.endsWith(pasoNivelVagon) && ( |
|  | marcado[plazas.indexOf(pasoNivelABVagonEsperando)] == 0 && marcado[plazas.indexOf(pasoNivelCDVagonEsperando)] == 0 || |
|  | marcado[plazas.indexOf(pasoNivelABVagonEsperando)] != 0 && ( |
|  | marcado[plazas.indexOf(pasoNivelABBarrera)] == 0 || |
|  | marcado[plazas.indexOf(pasoNivelABMaquinaEsperando)] != 0 |
|  | ) || |
|  | marcado[plazas.indexOf(pasoNivelCDVagonEsperando)] != 0 && ( |
|  | marcado[plazas.indexOf(pasoNivelCDBarrera)] == 0 || |
|  | marcado[plazas.indexOf(pasoNivelCDMaquinaEsperando)] != 0 |
|  | ) |
|  | ) |
|  | ) { |
|  | pasoDeNivelVagon.await(); |
|  | } |
|  |  |
|  | while( threadName.endsWith(pasoNivelMaquina) && ( |
|  | marcado[plazas.indexOf(pasoNivelABMaquinaEsperando)] == 0 && marcado[plazas.indexOf(pasoNivelCDMaquinaEsperando)] == 0 || |
|  | marcado[plazas.indexOf(pasoNivelABMaquinaEsperando)] != 0 && marcado[plazas.indexOf(pasoNivelABBarrera)] == 0 || |
|  | marcado[plazas.indexOf(pasoNivelCDMaquinaEsperando)] != 0 && marcado[plazas.indexOf(pasoNivelCDBarrera)] == 0 |
|  | ) |
|  | ) { |
|  | pasoDeNivelMaquina.await(); |
|  | } |
|  |  |
|  | if(threadName.endsWith(pasoNivelMaquina)) { |
|  | if(marcado[plazas.indexOf(pasoNivelABMaquinaEsperando)] != 0) { |
|  | dispararRed(tranPasoNivelABMaquinaWait); |
|  | } |
|  | if(marcado[plazas.indexOf(pasoNivelCDMaquinaEsperando)] != 0) { |
|  | dispararRed(tranPasoNivelCDMaquinaWait); |
|  | } |
|  | } |
|  | if(threadName.endsWith(pasoNivelVagon)) { |
|  | if(marcado[plazas.indexOf(pasoNivelABVagonEsperando)] != 0) { |
|  | dispararRed(tranPasoNivelABVagonWait); |
|  | } |
|  | if(marcado[plazas.indexOf(pasoNivelCDVagonEsperando)] != 0) { |
|  | dispararRed(tranPasoNivelCDVagonWait); |
|  | } |
|  | } |
|  |  |
|  | String transicion = interseccionPrioritarias(); |
|  | if(transicion != null) { |
|  | colaCondicion.get(transicion).signal(); |
|  | } |
|  | } finally { |
|  | lock.unlock(); |
|  | } |
|  | } |
|  |  |
|  | public void liberarBarreraPasoNivel() throws InterruptedException { |
|  | lock.lock(); |
|  |  |
|  | String threadName = Thread.currentThread().getName(); |
|  |  |
|  | try { |
|  | while( threadName.endsWith(pasoNivelTransitoAB) && |
|  | ( marcado[plazas.indexOf(pasoNivelABBarrera)] == 1 && |
|  | ( marcado[plazas.indexOf(pasoNivelABTransitoEsperando)] == 0 || |
|  | marcado[plazas.indexOf(pasoNivelABMaquinaEsperando)] != 0 || |
|  | marcado[plazas.indexOf(pasoNivelABVagonEsperando)] != 0 |
|  | ) |
|  | ) |
|  | ) { |
|  | liberarBarreraPasoNivelAB.await(); |
|  | } |
|  |  |
|  | while( threadName.endsWith(pasoNivelTransitoCD) && |
|  | ( marcado[plazas.indexOf(pasoNivelCDBarrera)] == 1 && |
|  | ( marcado[plazas.indexOf(pasoNivelCDTransitoEsperando)] == 0 || |
|  | marcado[plazas.indexOf(pasoNivelCDMaquinaEsperando)] != 0 || |
|  | marcado[plazas.indexOf(pasoNivelCDVagonEsperando)] != 0 |
|  | ) |
|  | ) |
|  | ) { |
|  | liberarBarreraPasoNivelCD.await(); |
|  | } |
|  |  |
|  | boolean disparoExitoso = false; |
|  | if(threadName.endsWith(pasoNivelTransitoAB)) { |
|  | if(marcado[plazas.indexOf(pasoNivelABMaquina)] != 0 && !disparoExitoso) { |
|  | disparoExitoso = dispararRed(tranPasoNivelABMaquinaReady); |
|  | } |
|  | if(marcado[plazas.indexOf(pasoNivelABVagon)] != 0 && !disparoExitoso) { |
|  | disparoExitoso = dispararRed(tranPasoNivelABVagonReady); |
|  | } |
|  | if(marcado[plazas.indexOf(pasoNivelABTransito)] != 0 && !disparoExitoso) { |
|  | disparoExitoso = dispararRed(tranPasoNivelABTransitoReady); |
|  | } |
|  | if(marcado[plazas.indexOf(pasoNivelABTransitoEsperando)] != 0 && !disparoExitoso) { |
|  | disparoExitoso = dispararRed(tranPasoNivelABTransitoWait); |
|  | } |
|  | } |
|  | if(threadName.endsWith(pasoNivelTransitoCD)) { |
|  | if(marcado[plazas.indexOf(pasoNivelCDMaquina)] != 0 && !disparoExitoso) { |
|  | disparoExitoso = dispararRed(tranPasoNivelCDMaquinaReady); |
|  | } |
|  | if(marcado[plazas.indexOf(pasoNivelCDVagon)] != 0 && !disparoExitoso) { |
|  | disparoExitoso = dispararRed(tranPasoNivelCDVagonReady); |
|  | } |
|  | if(marcado[plazas.indexOf(pasoNivelCDTransito)] != 0 && !disparoExitoso) { |
|  | disparoExitoso = dispararRed(tranPasoNivelCDTransitoReady); |
|  | } |
|  | if(marcado[plazas.indexOf(pasoNivelCDTransitoEsperando)] != 0 && !disparoExitoso) { |
|  | disparoExitoso = dispararRed(tranPasoNivelCDTransitoWait); |
|  | } |
|  | } |
|  |  |
|  | String transicion = interseccionPrioritarias(); |
|  | if(transicion != null) { |
|  | colaCondicion.get(transicion).signal(); |
|  | } |
|  | } finally { |
|  | lock.unlock(); |
|  | } |
|  | } |
|  |  |
|  |  |
|  | public void generarPasajeros() throws InterruptedException { |
|  | lock.lock(); |
|  |  |
|  | String threadName = Thread.currentThread().getName(); |
|  |  |
|  | try { |
|  |  |
|  | for(String estacion: estaciones) { |
|  | if(threadName.endsWith(estacion)) { |
|  | dispararRed(tranPasajerosAGenerador.substring(0, 1) + estacion + tranPasajerosAGenerador.substring(tranPasajerosAGenerador.length() - 1)); |
|  | break; |
|  | } |
|  | } |
|  |  |
|  | String transicion = interseccionPrioritarias(); |
|  | if(transicion != null) { |
|  | colaCondicion.get(transicion).signal(); |
|  | } |
|  | } finally { |
|  | lock.unlock(); |
|  | } |
|  | } |
|  |  |
|  |  |
|  | public void generarTransito() throws InterruptedException { |
|  | lock.lock(); |
|  |  |
|  | String threadName = Thread.currentThread().getName(); |
|  |  |
|  | try { |
|  |  |
|  | if(threadName.endsWith(recorridoAB)) { |
|  | dispararRed(tranPasoNivelABTransitoGenerador); |
|  | } |
|  | if(threadName.endsWith(recorridoCD)) { |
|  | dispararRed(tranPasoNivelCDTransitoGenerador); |
|  | } |
|  |  |
|  | String transicion = interseccionPrioritarias(); |
|  | if(transicion != null) { |
|  | colaCondicion.get(transicion).signal(); |
|  | } |
|  | } finally { |
|  | lock.unlock(); |
|  | } |
|  | } |
|  |  |
|  | private void imprimirMarcado() { |
|  | ArrayList<String> printOrderKeys = new ArrayList<>(Arrays.asList(printOrder.keySet().toArray(new String[printOrder.keySet().size()]))); |
|  | ArrayList<Integer> colWidthPrint = new ArrayList<>(); |
|  | for(String plaza: printOrderKeys) { |
|  | System.out.print(" "+plaza); |
|  | colWidthPrint.add(plaza.length()); |
|  | } |
|  | System.out.print("\n"); |
|  | for(String plaza: printOrderKeys) { |
|  | for(int i = 0; i < (colWidthPrint.get(printOrderKeys.indexOf(plaza)) - String.valueOf(marcado[printOrder.get(plaza)]).length()); i++) { |
|  | System.out.print(" "); |
|  | } |
|  | System.out.print(" "+marcado[printOrder.get(plaza)]); |
|  | colWidthPrint.add(plaza.length()); |
|  | } |
|  | System.out.print("\n"); |
|  | } |
|  |  |
|  | private String interseccionPrioritarias() { |
|  | ArrayList<String> prioritarias = new ArrayList<>(Arrays.asList(colaCondicion.keySet().toArray(new String[colaCondicion.keySet().size()]))); |
|  | LinkedHashMap<String, Boolean> vectorInterseccion = getInterseccionCondicion(getInterseccionInhibicion(getSensibilizadas(), getInhibidas())); |
|  | for(String transicion: prioritarias) { |
|  | if(vectorInterseccion.get(transicion)) { |
|  | return transicion; |
|  | } |
|  | } |
|  | return null; |
|  | } |
|  |  |
|  | private LinkedHashMap<String, Boolean> getInterseccionCondicion(LinkedHashMap<String, Boolean> vectorSensibilizadas) { |
|  | LinkedHashMap<String, Boolean> interseccion = new LinkedHashMap<>(); |
|  |  |
|  | System.out.println(" "); |
|  | for(String transicion: this.transiciones) { |
|  | if(vectorSensibilizadas.get(transicion) && !transicionesGeneradoras.contains(transicion)) { |
|  | System.out.print(" "+transicion+" ("+(colaCondicion.containsKey(transicion)? lock.getWaitQueueLength(colaCondicion.get(transicion)) != 0 : true)+")"); |
|  | } |
|  | interseccion.put(transicion, vectorSensibilizadas.get(transicion) && (colaCondicion.containsKey(transicion)? lock.getWaitQueueLength(colaCondicion.get(transicion)) != 0 : true)); |
|  | } |
|  | System.out.print(" - "+Thread.currentThread().getName()); |
|  | System.out.println(" "); |
|  |  |
|  | return interseccion; |
|  | } |
|  |  |
|  | private LinkedHashMap<String, Boolean> getInterseccionInhibicion(LinkedHashMap<String, Boolean> vectorSensibilizadas, LinkedHashMap<String, Boolean> vectorInhibidas) { |
|  | LinkedHashMap<String, Boolean> interseccion = new LinkedHashMap<>(); |
|  |  |
|  | for(String transicion: this.transiciones) { |
|  | interseccion.put(transicion, vectorSensibilizadas.get(transicion) && (vectorInhibidas.get(transicion)!=null? !vectorInhibidas.get(transicion): true)); |
|  | } |
|  |  |
|  | return interseccion; |
|  | } |
|  |  |
|  | private LinkedHashMap<String, Boolean> getSensibilizadas(){ |
|  | LinkedHashMap<String, Boolean> sensibilizadas = new LinkedHashMap<>(); |
|  |  |
|  | for (String transicion : transiciones) { |
|  | Integer sign = 0; |
|  | for(int i = 0; i < matrizMenos.length; i++) { |
|  | sign = new Integer(marcado[i]) - matrizMenos[i][transiciones.indexOf(transicion)]; |
|  | sensibilizadas.put(transicion, (sensibilizadas.get(transicion)!=null? sensibilizadas.get(transicion) : true) && !(sign < 0)); |
|  | } |
|  | } |
|  |  |
|  | return sensibilizadas; |
|  | } |
|  |  |
|  | private LinkedHashMap<String, Boolean> getInhibidas(){ |
|  | LinkedHashMap<String, Boolean> inhibidas = new LinkedHashMap<>(); |
|  |  |
|  | for (String transicion : transiciones) { |
|  | for(int i = 0; i < matrizInhibicion.length; i++) { |
|  | if(matrizInhibicion[i][transiciones.indexOf(transicion)] != 0) { |
|  | Integer sign = 0; |
|  | sign = new Integer(marcado[i]) - matrizInhibicion[i][transiciones.indexOf(transicion)]; |
|  | inhibidas.put(transicion, (inhibidas.get(transicion)!=null? inhibidas.get(transicion) : false) || sign >= 0); |
|  | } |
|  | } |
|  | } |
|  |  |
|  | return inhibidas; |
|  | } |
|  |  |
|  | private boolean dispararRed(String transicion) { |
|  | System.out.println(" "+transicion); |
|  | Integer[] vectorDisparo = Collections.nCopies(transiciones.size(), 0).toArray(new Integer[0]); |
|  | vectorDisparo[transiciones.indexOf(transicion)] = 1; |
|  |  |
|  | return dispararRed(vectorDisparo); |
|  | } |
|  |  |
|  | private boolean dispararRed(Integer[] vectorDisparo) { |
|  | String transicionDisparada = ""; |
|  | Integer sumatoriaDisparo = 0; |
|  | for (int i = 0; i < vectorDisparo.length; i++) { |
|  | if(vectorDisparo[i] != 0) { |
|  | transicionDisparada = transiciones.get(i); |
|  | } |
|  | sumatoriaDisparo += vectorDisparo[i]; |
|  | } |
|  | if(!sumatoriaDisparo.equals(1)) { |
|  | return false; |
|  | } |
|  |  |
|  | Integer[] postDisparo = new Integer[marcado.length]; |
|  | for(int i = 0; i < matrizMenos.length; i++) { |
|  | postDisparo[i] = new Integer(marcado[i]); |
|  | for (int j = 0; j < matrizMenos[i].length; j++) { |
|  | postDisparo[i] = postDisparo[i] - matrizMenos[i][j] \* vectorDisparo[j]; |
|  | } |
|  | if(postDisparo[i] < 0) { |
|  | return false; |
|  | } |
|  | } |
|  |  |
|  | for(int i = 0; i < matrizMas.length; i++) { |
|  | for (int j = 0; j < matrizMas[i].length; j++) { |
|  | postDisparo[i] = postDisparo[i] + matrizMas[i][j] \* vectorDisparo[j]; |
|  | } |
|  | if(postDisparo[i] < 0) { |
|  | return false; |
|  | } |
|  | } |
|  |  |
|  |  |
|  | this.marcado = postDisparo; |
|  | this.printWriter.print(transicionDisparada+" "); |
|  | this.printWriter.flush(); |
|  | imprimirMarcado(); |
|  | return true; |
|  | } |
|  | } |

* *Paso Nivel:*

|  |
| --- |
| package main; |
|  |  |
|  | public class PasoNivel extends Thread { |
|  |  |
|  | private Monitor monitorTren; |
|  |  |
|  | public PasoNivel(Monitor monitor, String maquinaVagonRecorrido, String precedencia) { |
|  | monitorTren = monitor; |
|  | setName(precedencia + ConstantesComunes.pasoNivel + maquinaVagonRecorrido); |
|  | } |
|  |  |
|  | @Override |
|  | public void run() { |
|  | try { |
|  | while(true) { |
|  | if(Thread.currentThread().getName().endsWith(ConstantesComunes.maquinaTren) || Thread.currentThread().getName().endsWith(ConstantesComunes.vagonTren)) { |
|  | monitorTren.cruzarPasoNivel(); |
|  | } else { |
|  | monitorTren.liberarBarreraPasoNivel(); |
|  | } |
|  | } |
|  | } catch (InterruptedException e) { |
|  | e.printStackTrace(); |
|  | } |
|  | } |
|  | } |

* *Tren:*

|  |
| --- |
| package main; |
|  |  |
|  | public class Tren extends Thread { |
|  |  |
|  | private Monitor monitorTren; |
|  |  |
|  | public Tren(Monitor monitor, String precedencia) { |
|  | monitorTren = monitor; |
|  | setName(ConstantesComunes.tren + precedencia); |
|  | } |
|  |  |
|  | @Override |
|  | public void run() { |
|  | try { |
|  | while(true) { |
|  | if(Thread.currentThread().getName().endsWith(ConstantesComunes.precedenciaAuxiliarArrivo)) { |
|  | monitorTren.arrivoTrenEstacion(); |
|  | } |
|  | if(Thread.currentThread().getName().endsWith(ConstantesComunes.precedenciaAuxiliarPartida)) { |
|  | monitorTren.partidaTren(); |
|  | } |
|  | if(Thread.currentThread().getName().endsWith(ConstantesComunes.precedenciaPrincipal)) { |
|  | Long tiempoEsperaContinuarRecorrido = monitorTren.continuarRecorridoTren(); |
|  | sleep(tiempoEsperaContinuarRecorrido); |
|  | } |
|  | } |
|  | } catch (InterruptedException e) { |
|  | e.printStackTrace(); |
|  | } |
|  | } |
|  | } |

**SIMULACIÓN**

**TESTING**

|  |  |
| --- | --- |
| Caso de test: T Invariantes |  |
| Secuencia a testear | Invariantes de transición para el recorrido cuando las estaciones A, B, C y D se vacian. |
| Expresión regular |  |
| Descripción | El marcado que conforma el recorrido vuelve a su estado original. |
| Resultado esperado | Mediante la siguiente secuencia de disparos (insertar secuencia), se espera que la red vuelva al estado del que partió. |
| Resultado obtenido | Pass |

|  |  |
| --- | --- |
| Caso de test: T Invariantes |  |
| Secuencia a testear | Invariantes de transición para el recorrido cuando A, B y C se vacian y el tren se llena en D. |
| Expresión regular |  |
| Descripción | El marcado que conforma el recorrido vuelve a su estado original. |
| Resultado esperado | Mediante la siguiente secuencia de disparos (insertar secuencia), se espera que la red vuelva al estado del que partió. |
| Resultado obtenido | Pass |

|  |  |
| --- | --- |
| Caso de test: T Invariantes |  |
| Secuencia a testear | Invariantes de transición para el recorrido cuando A, B y D se vacian y el tren se llena en C. |
| Expresión regular |  |
| Descripción | El marcado que conforma el recorrido vuelve a su estado original. |
| Resultado esperado | Mediante la siguiente secuencia de disparos (insertar secuencia), se espera que la red vuelva al estado del que partió. |
| Resultado obtenido | Pass |

|  |  |
| --- | --- |
| Caso de test: T Invariantes |  |
| Secuencia a testear | Invariantes de transición para el recorrido cuando A y B se vacian y el tren se llena en C y D. |
| Expresión regular |  |
| Descripción | El marcado que conforma el recorrido vuelve a su estado original. |
| Resultado esperado | Mediante la siguiente secuencia de disparos (insertar secuencia), se espera que la red vuelva al estado del que partió. |
| Resultado obtenido | Pass |

|  |  |
| --- | --- |
| Caso de test: T Invariantes |  |
| Secuencia a testear | Invariantes de transición para el recorrido cuando A, C y D se vacian y el tren se llena en B. |
| Expresión regular |  |
| Descripción | El marcado que conforma el recorrido vuelve a su estado original. |
| Resultado esperado | Mediante la siguiente secuencia de disparos (insertar secuencia), se espera que la red vuelva al estado del que partió. |
| Resultado obtenido | Pass |

|  |  |
| --- | --- |
| Caso de test: T Invariantes |  |
| Secuencia a testear | Invariantes de transición para el recorrido cuando A y C se vacian y el tren se llena en B y D. |
| Expresión regular |  |
| Descripción | El marcado que conforma el recorrido vuelve a su estado original. |
| Resultado esperado | Mediante la siguiente secuencia de disparos (insertar secuencia), se espera que la red vuelva al estado del que partió. |
| Resultado obtenido | Pass |

|  |  |
| --- | --- |
| Caso de test: T Invariantes |  |
| Secuencia a testear | Invariantes de transición para el recorrido cuando A y D se vacian y el tren se llena en B y C. |
| Expresión regular |  |
| Descripción | El marcado que conforma el recorrido vuelve a su estado original. |
| Resultado esperado |  |
| Resultado obtenido | Pass |

|  |  |
| --- | --- |
| Caso de test: T Invariantes |  |
| Secuencia a testear | Invariantes de transición para el recorrido cuando A está vacia y el tren se llena en B, C y D. |
| Expresión regular |  |
| Descripción | El marcado que conforma el recorrido vuelve a su estado original. |
| Resultado esperado | Mediante la siguiente secuencia de disparos (insertar secuencia), se espera que la red vuelva al estado del que partió. |
| Resultado obtenido | Pass |

|  |  |
| --- | --- |
| Caso de test: T Invariantes |  |
| Secuencia a testear | Invariantes de transición para el recorrido cuando B, C y D se vacian y el tren se llena en A. |
| Expresión regular |  |
| Descripción | El marcado que conforma el recorrido vuelve a su estado original. |
| Resultado esperado | Mediante la siguiente secuencia de disparos (insertar secuencia), se espera que la red vuelva al estado del que partió. |
| Resultado obtenido | Pass |

|  |  |
| --- | --- |
| Caso de test: T Invariantes |  |
| Secuencia a testear | Invariantes de transición para el recorrido cuando B y C se vacian y el tren se llena en A y D. |
| Expresión regular |  |
| Descripción | El marcado que conforma el recorrido vuelve a su estado original. |
| Resultado esperado | Mediante la siguiente secuencia de disparos (insertar secuencia), se espera que la red vuelva al estado del que partió. |
| Resultado obtenido | Pass |

|  |  |
| --- | --- |
| Caso de test: T Invariantes |  |
| Secuencia a testear | Invariantes de transición para el recorrido cuando B y D se vacian y el tren se llena en A y C. |
| Expresión regular |  |
| Descripción | El marcado que conforma el recorrido vuelve a su estado original. |
| Resultado esperado | Mediante la siguiente secuencia de disparos (insertar secuencia), se espera que la red vuelva al estado del que partió. |
| Resultado obtenido | Pass |

|  |  |
| --- | --- |
| Caso de test: T Invariantes |  |
| Secuencia a testear | Invariantes de transición para el recorrido cuando B se vacia y el tren se llena en A, C y D. |
| Expresión regular |  |
| Descripción | El marcado que conforma el recorrido vuelve a su estado original. |
| Resultado esperado | Mediante la siguiente secuencia de disparos (insertar secuencia), se espera que la red vuelva al estado del que partió. |
| Resultado obtenido | Pass |

|  |  |
| --- | --- |
| Caso de test: T Invariantes |  |
| Secuencia a testear | Invariantes de transición para el recorrido cuando C y D se vacian y el tren se llena en A y B. |
| Expresión regular |  |
| Descripción | El marcado que conforma el recorrido vuelve a su estado original. |
| Resultado esperado | Mediante la siguiente secuencia de disparos (insertar secuencia), se espera que la red vuelva al estado del que partió. |
| Resultado obtenido | Pass |

|  |  |
| --- | --- |
| Caso de test: T Invariantes |  |
| Secuencia a testear | Invariantes de transición para el recorrido cuando C se vacia y el tren se llena en A, B y D. |
| Expresión regular |  |
| Descripción | El marcado que conforma el recorrido vuelve a su estado original. |
| Resultado esperado | Mediante la siguiente secuencia de disparos (insertar secuencia), se espera que la red vuelva al estado del que partió. |
| Resultado obtenido | Pass |

|  |  |
| --- | --- |
| Caso de test: T Invariantes |  |
| Secuencia a testear | Invariantes de transición para el recorrido cuando D se vacia y el tren se llena en A, B y C. |
| Expresión regular |  |
| Descripción | El marcado que conforma el recorrido vuelve a su estado original. |
| Resultado esperado | Mediante la siguiente secuencia de disparos (insertar secuencia), se espera que la red vuelva al estado del que partió. |
| Resultado obtenido | Pass |

|  |  |
| --- | --- |
| Caso de test: T Invariantes |  |
| Secuencia a testear | Invariantes de transición para el recorrido cuando el tren se llena en A, B, C y D. |
| Expresión regular |  |
| Descripción | El marcado que conforma el recorrido vuelve a su estado original. |
| Resultado esperado | Mediante la siguiente secuencia de disparos (insertar secuencia), se espera que la red vuelva al estado del que partió. |
| Resultado obtenido | Pass |

|  |  |
| --- | --- |
| Caso de test: T Invariantes |  |
| Secuencia a testear | Invariantes de transición para el viaje que realiza un pasajero desde la estación A. |
| Expresión regular |  |
| Descripción | El pasajero subido en A puede bajarse en B, C o D. |
| Resultado esperado | Mediante la siguiente secuencia de disparos (insertar secuencia), se espera que la red vuelva al estado del que partió. |
| Resultado obtenido | Pass |

|  |  |
| --- | --- |
| Caso de test: T Invariantes |  |
| Secuencia a testear | Invariantes de transición para el viaje que realiza un pasajero desde la estación B. |
| Expresión regular |  |
| Descripción | El pasajero subido en B puede bajarse en A, C o D. |
| Resultado esperado | Mediante la siguiente secuencia de disparos (insertar secuencia), se espera que la red vuelva al estado del que partió. |
| Resultado obtenido | Pass |

|  |  |
| --- | --- |
| Caso de test: T Invariantes |  |
| Secuencia a testear | Invariantes de transición para el viaje que realiza un pasajero desde la estación C. |
| Expresión regular |  |
| Descripción | El pasajero subido en C puede bajarse en A, B o D. |
| Resultado esperado | Mediante la siguiente secuencia de disparos (insertar secuencia), se espera que la red vuelva al estado del que partió. |
| Resultado obtenido | Pass |

|  |  |
| --- | --- |
| Caso de test: T Invariantes |  |
| Secuencia a testear | Invariantes de transición para el viaje que realiza un pasajero desde la estación D. |
| Expresión regular |  |
| Descripción | El pasajero subido en D puede bajarse en A, B o C. |
| Resultado esperado | Mediante la siguiente secuencia de disparos (insertar secuencia), se espera que la red vuelva al estado del que partió. |
| Resultado obtenido | Pass |

|  |  |
| --- | --- |
| Caso de test: T Invariantes |  |
| Secuencia a testear | Invariantes de transición para el cruce que realiza un vehiculo en el paso de nivel entre las estaciones A y B. |
| Expresión regular |  |
| Descripción | El marcado que conforma el recorrido vuelve a su estado original. |
| Resultado esperado | Mediante la siguiente secuencia de disparos (insertar secuencia), se espera que la red vuelva al estado del que partió. |
| Resultado obtenido | Pass |

|  |  |
| --- | --- |
| Caso de test: T Invariantes |  |
| Secuencia a testear | Invariantes de transición para el cruce que realiza un vehiculo en el paso de nivel entre las estaciones C y D. |
| Expresión regular |  |
| Descripción | El marcado que conforma el recorrido vuelve a su estado original. |
| Resultado esperado | Mediante la siguiente secuencia de disparos (insertar secuencia), se espera que la red vuelva al estado del que partió. |
| Resultado obtenido | Pass |

|  |  |
| --- | --- |
| Caso de test: P Invariantes |  |
| Restricción a testear | El marcado para los pasajeros subidos en las estaciones A, B, C y D que viajan en la maquina junto a los lugares libres de la misma permanece constante en 30. |
| Expresión regular |  |
| Resultado esperado | La suma de los tokens en todas las plazas mencionadas se mantenga constante. |
| Resultado obtenido | Pass |

|  |  |
| --- | --- |
| Caso de test: P Invariantes |  |
| Restricción a testear | El marcado para los pasajeros subidos en las estaciones A, B, C y D que viajan en el vagon junto a los lugares libres del mismo permanece constante en 20. |
| Expresión regular |  |
| Resultado esperado | La suma de los tokens en todas las plazas mencionadas se mantenga constante. |
| Resultado obtenido | Pass |

|  |  |
| --- | --- |
| Caso de test: P Invariantes |  |
| Restricción a testear | El marcado conformado por el paso nivel entre A y B se mantiente constante en 1. |
| Expresión regular |  |
| Resultado esperado | Se espera que el token se encuentre en una y solo una de las plazas que satisfacen el paso nivel entre A y B (PNABB, PNABM, PNABT o PNABV). |
| Resultado obtenido | Pass |

|  |  |
| --- | --- |
| Caso de test: P Invariantes |  |
| Restricción a testear | El marcado conformado por el paso nivel entre C y D se mantiente constante en 1. |
| Expresión regular |  |
| Resultado esperado | Se espera que el token se encuentre en una y solo una de las plazas que satisfacen el paso nivel entre C y D (PNCDB, PNCDM, PNCDT o PNCDV). |
| Resultado obtenido | Pass |

|  |  |
| --- | --- |
| Caso de test: P Invariantes |  |
| Restricción a testear | Invariante de plaza generado artificialmente por un “flag” usado para inhibir las bajadas y subidas. |
| Expresión regular |  |
| Resultado esperado | Se espera que el token se encuentre en alguna de las siguientes plazas: ATR, ATW o TEA. |
| Resultado obtenido | Pass |

|  |  |
| --- | --- |
| Caso de test: P Invariantes |  |
| Restricción a testear | Invariante de plaza generado artificialmente por un “flag” usado para inhibir las bajadas y subidas. |
| Expresión regular |  |
| Resultado esperado | Se espera que el token se encuentre en alguna de las siguientes plazas: BTR, BTW o TEB. |
| Resultado obtenido | Pass |

|  |  |
| --- | --- |
| Caso de test: P Invariantes |  |
| Restricción a testear | Invariante de plaza generado artificialmente por un “flag” usado para inhibir las bajadas y subidas. |
| Expresión regular |  |
| Resultado esperado | Se espera que el token se encuentre en alguna de las siguientes plazas: CTR, CTW o TEC. |
| Resultado obtenido | Pass |

|  |  |
| --- | --- |
| Caso de test: P Invariantes |  |
| Restricción a testear | Invariante de plaza generado artificialmente por un “flag” usado para inhibir las bajadas y subidas. |
| Expresión regular |  |
| Resultado esperado | Se espera que el token se encuentre en alguna de las siguientes plazas: DTR, DTW o TED. |
| Resultado obtenido | Pass |

**CONCLUSIÓN**

El monitor cumplió con el objetivo principal, pues se pudo ver el funcionamiento de un circuito ferroviario tal y como lo sugería la propuesta de trabajo.

Un propósito importante plasmado a medida que fuimos avanzando en la resolución del problema fue la adquisición de conocimientos necesarios para ser capaces de implementar las redes de Petri en los sistemas concurrentes y de esta manera ampliar nuestro campo de estudio brindando una herramienta más para nuestra labor como futuros ingenieros.

El mayor inconveniente se presentó a la hora de seleccionar qué hilo despertar ya que en nuestro caso muchas de las transiciones no estaban asociadas a una cola de condición, por lo que se debió consultar a los docentes y opiniones de diversos compañeros para poder solucionar esta dificultad.

Por último, se debe salvar el hecho de la colaboración mutua y el compañerismo que fue de gran ayuda para alcanzar todo lo expuesto y a lo cual se recurrió cada vez que hubo dudas en el entendimiento y/o implementación de la lógica del monitor.

**BIBLIOGRAFÍA**

https://en.wikipedia.org/wiki/Petri\_net

https://www.techfak.uni-bielefeld.de/~mchen/BioPNML/Intro/pnfaq.html

http://www.scholarpedia.org/article/Petri\_net

http://www.isr.umd.edu/Labs/CIM/miscs/wmsor97.pdf

http://www.stevens-tech.edu/wireless/research/petrinet/reading-petri-net-tutorial-zurawski-zhou.pdf

http://bluehawk.monmouth.edu/~jwang/Petri%20Nets%20--%20Introduction.pdf

https://www.javaworld.com/article/2077769/core-java/better-monitors-for-java.html

https://es.wikipedia.org/wiki/Monitor\_(concurrencia)

https://programaressencillo.wordpress.com/2014/11/25/java-monitores-ejemplo-productor-consumidor/

<http://www.tecdis-eu.es/web/sites/default/files/u42/Teoria/Tema_5_Monitores.pdf>

<https://docs.oracle.com/javase/tutorial/essential/concurrency/interrupt.html>

http://www.chuidiang.org/java/hilos/wait\_y\_notify.php

Manuales USERS.code JAVA LA GUÍA TOTAL DEL PROGRAMADOR - Sergio Dos Santos.

Redes de Petri en sistemas concurrentes - Micolini, Ventre, Cebollada y Eschoyez.

Ecuación de estado generalizada para redes de Petri no autónomas con distintos tipos de arcos y semánticas temporales - Micolini, Ventre, Cebollada, Eschoyez y Schild.

Programación Concurrente - Palma.