El HACKER viajero

Computación y estructuras discretas I

Integrantes:

Omhes Samuel Leon Diaz- A00406645

Luis Fernando Soto Bedoya- A00406591

Miguel Perez Ojeda-A00407054

Cliente	Sistema Virtual de Seguridad Distribuido
Usuario	Jugadores del juego "El Hacker Viajero" y diseñadores del sistema de navegación en redes digitales.
Requerimientos funcionales	RF1 - Visualizar grafo de la red digital (nodos y enlaces) RF2 - Seleccionar nodo inicial y nodo destino para recorrido RF3 - Recorrer la red con algoritmos BFS y DFS para exploración RF4 - Calcular rutas óptimas con Dijkstra y Floyd-Warshall RF5 - Cambiar entre implementaciones de grafo: lista de adyacencia y matriz de adyacencia RF6 - Mostrar rutas recorridas, claves encontradas y costos de movimiento RF7 - Validar que el grafo tenga al menos 50 vértices y 50 aristas RF8 - Ejecutar pruebas unitarias automáticas para operaciones del grafo y algoritmos implementados
Contexto del problema	En un mundo digital altamente vigilado, el hacker debe infiltrarse en una red de servidores representada como un grafo con nodos y enlaces con latencias variables. Para lograrlo, el jugador debe optimizar sus rutas utilizando algoritmos clásicos de grafos, explorando nodos y evitando sistemas de seguridad para recuperar claves secretas. El sistema debe soportar dos implementaciones de grafo para mayor flexibilidad y debe ofrecer una interfaz gráfica que permita la interacción directa con la red y la visualización de resultados.
Requerimientos no funcionales	RNF1 – Implementar dos estructuras de datos para grafo: lista y matriz de adyacencia RNF2 - Interfaz gráfica amigable e interactiva RNF3 - Código modular y documentado que facilite mantenimiento y pruebas RNF4 - Pruebas unitarias automáticas para garantizar la fiabilidad de algoritmos y operaciones del grafo RNF5 - Rendimiento óptimo para grafos con al menos 50 vértices y 50 aristas

Identificador y nombre	*RF1 – Visualizar Red del Juego*		
Resumen	*Este requerimiento permite mostrar gráficamente el grafo que representa la red de nodos del juego. El usuario podrá observar los vértices y sus conexiones (aristas) con los pesos asociados.*		
Entradas	**Nombre entrada**	**Tipo de dato**	**Condición valores válidos**
	grafo	Objeto	*Debe contener al menos 50 vértices y 50 aristas*
Resultado o Postcondición	El grafo se visualiza correctamente en la interfaz gráfica del sistema, si no existe un grafo válido se muestra un mensaje de error.		
Salidas	Mensaje exitoso	String	"La red del juego ha sido visualizada correcta
	Mensaje error	String	"No hay un grafo válido para visualizar."

Identificador y nombre	*RF2 – Seleccionar Nodo Inicial y Final*		
Resumen	*Este requerimiento permite al jugador seleccionar los nodos de inicio y destino para aplicar algoritmos sobre la red.*		
Entradas	**Nombre entrada**	**Tipo de dato**	**Condición valores válidos**
	nodoInicio	String / int	*Debe existir en el grafo*
	nodoDestino	String / int	Debe existir en el grafo
Resultado o Postcondición	*Los nodos seleccionados quedan guardados para su uso en algoritmos de recorrido o búsqueda. Si alguno no existe, se muestra un mensaje de error.*		
Salidas	Mensaje exitoso	String	"Los nodos han sido seleccionados correctamente." "Los nodos han sido seleccionados correctamente."
	Mensaje error	String	"Nodo seleccionado no existe en el grafo."

Identificador y nombre	***RF3 - Recorrer la red con algoritmos BFS y DFS para exploración***		
Resumen	*Ejecuta un recorrido por el grafo desde el nodo origen, utilizando BFS o DFS para explorar la red.*		
Entradas	**Nombre entrada**	**Tipo de dato**	**Condición valores válidos**
	Costo total	double	Número positivo (suma de pesos del camino)

Identificador y nombre	*RF5 - Cambiar entre implementaciones de grafo: lista de adyacencia y matriz de adyacencia*
Resumen	*Permite cambiar la implementación del grafo, sin alterar su funcionamiento.*

Entradas	**Nombre entrada**	**Tipo de dato**	**Condición valores válidos**
	tipoGrafo	String	"Lista" o "Matriz"
Resultado o Postcondición	*La implementación del grafo cambia con éxito, conservando los datos cargados.*		
Salidas	Mensaje exitoso	String	"Implementación cambiada exitosamente"
	Mensaje error	String	"Tipo de grafo no reconocido"

Identificador y nombre	*RF6 - Mostrar rutas recorridas, claves encontradas y costos de movimiento*
Resumen	*Despliega las rutas que el jugador ha recorrido, claves digitales encontradas y el costo acumulado.*

Entradas	**Nombre entrada**	**Tipo de dato**	**Condición valores válidos**
Resultado o Postcondición	Se muestran los datos de progreso del jugador en pantalla.		
Salidas	Rutas recorridas	Lista<String>	Lista de nodos visitados
	Claves encontradas	Lista<String>	[Clave1, Clave2, ...]

Identificador y nombre	*RF7 - Validar que el grafo tenga al menos 50 vértices y 50 aristas*		
Resumen	*Valida si el grafo cumple con los requisitos mínimos para que el juego funcione correctamente.*		
Entradas	**Nombre entrada**	**Tipo de dato**	**Condición valores válidos**
Resultado o Postcondición	*Se determina si el grafo cargado es válido para ejecutar el juego.*		
Salidas	Validación grafo	String	"Grafo válido" o "Debe tener al menos 50 vértices y 50 aristas"

Identificador y nombre	*RF8 - Ejecutar pruebas unitarias automáticas para operaciones del grafo y algoritmos implementados*		
Resumen	*Permite ejecutar las pruebas automáticas del sistema para validar la funcionalidad de los métodos y algoritmos.*		
Entradas	**Nombre entrada**	**Tipo de dato**	**Condición valores válidos**
Resultado o Postcondición	*El sistema muestra el resultado de cada prueba unitaria, indicando éxito o fallo.*		
Salidas	Mensaje general	String	"Pruebas completadas"