Step 1. Problem Identification (Provide Context, State Needs and Symptoms, Define the Problem)

Desarrollo de un Juego con Grafos

Debemos diseñar un juego que pueda ser modelado utilizando grafos. El juego debe tener al menos 50 vértices y 50 aristas en su representación de grafo, y debe permitir la aplicación de al menos dos de los siguientes algoritmos de grafos que se estudiarán durante el curso: Recorridos sobre Grafos (BFS, DFS), Caminos de Peso Mínimo (Dijkstra, Floyd-Warshall), y Árbol de Recubrimiento Mínimo (Prim, Kruskal).

**Requisitos mínimos:**

**Implementación de Dos Versiones de Grafo:** Debe desarrollarse no una, sino dos versiones de la representación del grafo en el juego. Esto implica que el juego debe funcionar sin problemas con ambas implementaciones, lo que significa que el programa debe admitir el cambio de la implementación utilizada en cualquier momento y seguir funcionando sin problemas, independientemente de la implementación que se esté utilizando. Esto brinda flexibilidad y robustez al juego.

**Método de Ingeniería para la Solución:** El proyecto debe seguir todas las fases del método de ingeniería para la solución del problema. Esto incluye un análisis detallado, diseño del TAD (Tipo Abstracto de Datos), diagramas de clase y objetos, y diseño de casos de prueba para las pruebas unitarias automáticas. Cada fase debe estar bien documentada y respaldada.

**Interfaz Gráfica de Usuario (GUI):** Su juego debe contar con una interfaz gráfica de usuario que permita a los jugadores interactuar con las funcionalidades que se alinean con los requisitos del juego. La GUI debe ser amigable y atractiva para los usuarios.

**Contexto:**

En el juego "Aventura del Tesoro Perdido," los jugadores se embarcan en una emocionante búsqueda de un antiguo tesoro oculto en un mundo lleno de enigmas y peligros. Este mundo se representa como un grafo, en el cual los vértices simbolizan ubicaciones, y las aristas representan los caminos que conectan estas ubicaciones. Los jugadores deben emplear algoritmos de recorrido para descubrir nuevas ubicaciones y recolectar pistas que los conduzcan al ansiado tesoro.

**Necesidades y Síntomas Identificados:**

* Los jugadores enfrentan el desafío de administrar eficazmente su exploración y gestión de pistas en medio de un mundo complejo de ubicaciones interconectadas.
* Distinguir entre rutas disponibles y rutas bloqueadas puede resultar confuso y dificultar la planificación de la búsqueda.
* La búsqueda de pistas y objetos importantes puede ser una tarea desafiante, y algunas pistas pueden estar ocultas, lo que requiere una búsqueda minuciosa.
* Los obstáculos y desafíos en ciertas ubicaciones pueden dificultar el progreso del jugador y requerir estrategia para superarlos.
* El objetivo principal del juego, encontrar el tesoro, depende de la planificación de recorridos y la resolución de desafíos.

**Definición del Problema:**

El problema central en "Aventura del Tesoro Perdido" es la gestión eficiente de la exploración de un mundo complejo representado como un grafo. Los jugadores necesitan navegar por un entorno interconectado, diferenciar entre rutas disponibles y bloqueadas, descubrir pistas y objetos ocultos, superar obstáculos y planificar recorridos para alcanzar el objetivo final: encontrar el tesoro. Para garantizar una experiencia de juego fluida y satisfactoria, es esencial desarrollar una solución que aborde estos desafíos y facilite la exploración y la búsqueda del tesoro en este mundo lleno de misterios.

**Requerimientos Funcionales:**

**Modelado del Juego:** El sistema debe permitir el modelado de un juego que pueda ser representado utilizando grafos con un mínimo de 50 vértices y 50 aristas.

**Algoritmos de Grafos:** La solución debe ser capaz de aplicar al menos dos (2) de los siguientes algoritmos de grafos: Recorridos sobre Grafos (BFS, DFS), Caminos de Peso Mínimo (Dijkstra, Floyd-Warshall), Árbol de Recubrimiento Mínimo (MST) (Prim, Kruskal).

**Desarrollo de 2 Versiones de Grafo:** El sistema debe permitir el desarrollo de dos versiones de grafo, y ambas versiones deben funcionar correctamente. Debe ser posible cambiar la implementación del grafo en cualquier momento sin que afecte el funcionamiento del programa.

**Especificación de Requerimientos:** Debe generarse un documento de especificación de requerimientos que describa las necesidades y funcionalidades del juego.

**Diseño del TAD:** El sistema debe incluir el diseño del Tipo Abstracto de Datos (TAD) que permita representar eficazmente el juego utilizando grafos.

**Diagramas de Clase y Objetos:** Deben crearse diagramas de clase y objetos que representen las estructuras y relaciones del juego y sus componentes.

**Casos de Pruebas de Pruebas Unitarias Automáticas:** Deben ser diseñados casos de pruebas de pruebas unitarias automáticas para verificar la funcionalidad de cada componente del juego.

**Interfaz Gráfica de Usuario:** El programa debe contar con una interfaz gráfica de usuario que permita a los jugadores utilizar las funcionalidades necesarias para jugar el juego.

**Requerimientos No Funcionales:**

**Calidad del Código:** El código desarrollado debe seguir buenas prácticas de programación y ser de alta calidad.

**Documentación Clara y Concisa**: Toda la documentación, incluyendo especificaciones, diseños y pruebas, debe ser clara y fácil de entender.

**Eficiencia y Rendimiento:** El sistema debe ser eficiente en términos de consumo de recursos y debe funcionar con buen rendimiento.

**Flexibilidad y Mantenibilidad:** El código y la arquitectura del sistema deben ser flexibles y mantenibles para facilitar futuras actualizaciones o modificaciones.

**Usabilidad de la Interfaz Gráfica:** La interfaz gráfica de usuario debe ser intuitiva y fácil de usar para los jugadores.

**Compatibilidad:** El sistema debe ser compatible con las tecnologías y plataformas especificadas para la interfaz gráfica de usuario.

**Seguridad:** Se deben implementar medidas de seguridad para proteger los datos y la integridad del juego.

**Robustez y Tolerancia a Errores**: El sistema debe ser robusto y capaz de manejar errores de manera adecuada para garantizar la estabilidad del juego.

**Diseño Estético:** La interfaz gráfica de usuario debe tener un diseño estético atractivo para los jugadores.

**Rendimiento de Grafos:** Los algoritmos de grafos implementados deben ser eficientes y manejar grafo de gran tamaño de manera efectiva.

**Step 2.** Information Gathering (Recopilación de Información sobre las Estructuras de Datos a Utilizar)

**1. Grafo del Mundo:**

**Descripción:** El grafo del mundo es la estructura central del juego que representa el entorno en el que los jugadores operan. Cada ubicación importante se representa como un vértice, y las rutas entre ubicaciones se representan como aristas.

**Tipo de Estructura de Datos:** Para representar eficazmente el grafo del mundo, se utilizará una estructura de datos de grafo. Puede ser una matriz de adyacencia o una lista de adyacencia, dependiendo de los requisitos de rendimiento y memoria.

**2. Objetos y Pistas:**

**Descripción:** Los jugadores pueden encontrar objetos y pistas en cada ubicación, lo que afectará su progreso en la búsqueda del tesoro.

**Tipo de Estructura de Datos:** Se utilizarán estructuras de datos como listas o diccionarios para almacenar información sobre los objetos y pistas encontradas en cada ubicación. Esto permitirá un acceso eficiente a estos elementos.

**3. Algoritmos de Grafos:**

**Descripción:** Se deben aplicar al menos dos (2) algoritmos de grafos en el juego para permitir a los jugadores explorar el mundo y encontrar pistas.

**Tipo de Estructura de Datos:** Los algoritmos de grafos mencionados (BFS, DFS, Dijkstra, Floyd-Warshall, Prim, Kruskal) requerirán estructuras de datos específicas para su implementación. Por ejemplo, BFS y DFS pueden utilizar una cola o una pila, mientras que Dijkstra y Floyd-Warshall requerirán matrices o listas para representar los grafos ponderados.

**1. BFS (Breadth-First Search):**

**Cola (Queue):** BFS utiliza una estructura de datos tipo cola para llevar a cabo el recorrido del grafo. La cola permite un orden FIFO (primero en entrar, primero en salir) para explorar los vértices de manera sistemática.

**2. DFS (Depth-First Search):**

* **Pila (Stack):** DFS utiliza una estructura de datos tipo pila para llevar a cabo el recorrido del grafo. La pila sigue un enfoque de LIFO (último en entrar, primero en salir) para explorar los vértices de manera sistemática.

**3. Dijkstra (Algoritmo de Caminos Más Cortos):**

* **Matriz de Adyacencia o Lista de Adyacencia:** Dijkstra se utiliza para encontrar los caminos más cortos desde un vértice fuente a todos los demás vértices en un grafo ponderado. Para su implementación, se requiere una representación eficiente del grafo, que puede ser una matriz de adyacencia o una lista de adyacencia.

**4. Floyd-Warshall (Algoritmo de Todos los Pares más Cortos):**

* **Matriz de Adyacencia:** El algoritmo de Floyd-Warshall se utiliza para encontrar los caminos más cortos entre todos los pares de vértices en un grafo ponderado. Requiere una matriz de adyacencia para almacenar la información de las distancias entre los vértices.

**5. Prim (Algoritmo de Árbol de Recubrimiento Mínimo):**

* **Conjunto de Vértices y Conjunto de Aristas:** Prim se utiliza para encontrar un árbol de recubrimiento mínimo en un grafo ponderado. Requiere el mantenimiento de dos conjuntos: uno que contiene los vértices incluidos en el árbol y otro que contiene las aristas seleccionadas para el árbol.

**6. Kruskal (Algoritmo de Árbol de Recubrimiento Mínimo):**

* **Conjunto de Vértices y Conjunto de Aristas:** Al igual que Prim, Kruskal se utiliza para encontrar un árbol de recubrimiento mínimo. Requiere el mantenimiento de dos conjuntos: uno que contiene los vértices incluidos en el árbol y otro que contiene las aristas seleccionadas para el árbol.

**4. Rutas Bloqueadas y Desafíos:**

**Descripción:** Algunas rutas entre ubicaciones pueden estar bloqueadas hasta que se cumplan ciertos requisitos o se superen desafíos. Es importante realizar un seguimiento de estas condiciones.

**Tipo de Estructura de Datos:** Estructuras de datos como listas de control de acceso o estados de desafío se utilizarán para gestionar las rutas bloqueadas y los desafíos superados.

**5. Interfaz Gráfica de Usuario:**

**Descripción:** La interfaz gráfica de usuario del juego debe representar visualmente el grafo del mundo, las ubicaciones, pistas y objetos. Además, debe ser interactiva y permitir la exploración del mundo del juego.

**Tipo de Estructura de Datos:** Se utilizarán elementos gráficos, como imágenes, botones y elementos de navegación, para crear la interfaz gráfica del juego. Las coordenadas y referencias a ubicaciones del grafo se utilizarán para posicionar elementos en la interfaz.