MÉTODO DE LA INGENIERÍA

**FASE 1**

**Contexto del problema:**

Diseño de un juego interactivo con conceptos de la teoría de grafos

**Identificación del problema:**

En el ámbito del desarrollo de juegos, tenemos la tarea de crear un juego interactivo, "El Viajero Astuto", que combina la emoción de un viaje con la aplicación práctica de los conceptos de la teoría de grafos. El reto consiste en crear una experiencia de juego inmersiva en la que los jugadores aprendan a gestionar eficazmente recursos limitados, como el combustible, el tiempo y el hambre, mientras navegan hacia su destino.

**Síntomas y Necesidades:**

Gestión de Recursos en el Juego: Los jugadores necesitan aprender a administrar de manera efectiva los recursos limitados presentes en el juego. Esto implica tomar decisiones estratégicas sobre la cantidad de gasolina a llevar, cuándo hacer paradas para comer y cómo gestionar sabiamente el tiempo durante el viaje. La necesidad de administrar recursos de manera efectiva es fundamental para superar los desafíos y tener éxito en el juego.

Toma de Decisiones Estratégicas en el Viaje: "El Viajero Astuto" debe permitir a los jugadores tomar decisiones estratégicas a lo largo de su viaje. Esto implica seleccionar rutas entre ciudades y lugares, considerando los recursos disponibles y los obstáculos en el camino. La toma de decisiones estratégicas es un elemento esencial que influye en el resultado del juego.

Comprender los Algoritmos Clave: Los jugadores necesitan comprender y aplicar los algoritmos de Dijkstra y BFS (Búsqueda en Anchura) en el contexto del juego. Estos algoritmos les permiten encontrar las rutas más eficientes y cortas en términos de tiempo y recursos. Los algoritmos son herramientas esenciales para alcanzar el objetivo del juego.

**Recolección de Información:**

Recorridos sobre Grafos:

Búsqueda en Amplitud (BFS): Algoritmo de búsqueda que explora todos los vértices de un grafo de manera sistemática, nivel por nivel, partiendo desde un nodo raíz. Se utiliza en grafos no ponderados para encontrar el camino más corto en términos de número de pasos.

Búsqueda en Profundidad (DFS): La Búsqueda en Profundidad explora una rama del grafo lo máximo posible antes de retroceder. Se utiliza para clasificar los vértices por orden de visita y detectar ciclos en el grafo.

Caminos de Peso Mínimo:

Algoritmo de Dijkstra: Este algoritmo es esencial para encontrar los caminos más cortos desde un vértice origen a todos los demás vértices en un grafo ponderado. Se utiliza ampliamente en redes de transporte y logística.

Algoritmo de Floyd-Warshall: El algoritmo de Floyd-Warshall es capaz de encontrar los caminos más cortos entre todos los pares de vértices en un grafo ponderado. Aunque es más lento que Dijkstra, puede manejar grafos con ciclos negativos.

Árbol de Recubrimiento Mínimo (MST):

Algoritmo de Prim: El Algoritmo de Prim se utiliza para encontrar un árbol de recubrimiento mínimo en un grafo no dirigido ponderado. Es ampliamente empleado en redes de comunicación y diseño de circuitos.

Algoritmo de Kruskal: Similar al algoritmo de Prim, Kruskal encuentra un árbol de recubrimiento mínimo mediante la unión paso a paso de aristas de menor peso. Es útil en aplicaciones como diseño de redes y planificación de cableado.

Algoritmos de Ordenamiento:

Quicksort: Quicksort es un algoritmo de ordenamiento eficiente que utiliza la estrategia "divide y vencerás". Es ampliamente utilizado en bibliotecas de lenguajes de programación.

Mergesort: Mergesort es un algoritmo de ordenamiento que divide la lista en dos mitades, las ordena por separado y luego las fusiona en una lista ordenada.

Heapsort: Heapsort es un algoritmo de ordenamiento basado en una estructura de datos llamada "montículo" o "heap". Se utiliza en aplicaciones en las que se necesita un ordenamiento en su lugar.

Estructuras de Datos:

Pilas (Stacks): Las pilas son estructuras de datos lineales que siguen el principio LIFO (Last-In, First-Out). Se utilizan para administrar datos de manera eficiente en aplicaciones como navegadores web y editores de texto.

Colas (Queues): Las colas son estructuras lineales que siguen el principio FIFO (First-In, First-Out). Son fundamentales en la administración de tareas en sistemas operativos y en la planificación de recursos.

Árboles Binarios: Los árboles binarios son estructuras jerárquicas que constan de nodos, donde cada nodo tiene, a lo sumo, dos nodos secundarios. Son ampliamente utilizados en la búsqueda y organización de datos.

Complejidad Computacional (Big O): La notación Big O se utiliza para medir la eficiencia o complejidad de un algoritmo en función del tamaño de la entrada. Ejemplos de notaciones Big O incluyen O(1), O(log n), O(n), O(n log n), O(n^2), entre otras. Comprender la complejidad computacional es fundamental para la optimización de algoritmos.

Pruebas Unitarias: Las pruebas unitarias son un método para verificar que componentes individuales del software funcionen correctamente. Se emplean para asegurarse de que cada parte del programa cumple con sus especificaciones.

Grafos Ponderados vs. No Ponderados:

Grafos Ponderados: En los grafos ponderados, cada arista tiene un peso o costo asociado. Se utilizan en situaciones donde las conexiones entre nodos tienen valores significativos, como redes de transporte o rutas óptimas.

Grafos No Ponderados: En los grafos no ponderados, todas las aristas tienen el mismo valor y no se les asigna ningún peso. Se utilizan para representar conexiones binarias donde la existencia de una conexión es más relevante que su valor numérico.

**Fase 3: Posible soluciones**

Implementación de un Grafo Ponderado: Para modelar las conexiones entre ciudades y puntos de interés en el juego. Esto permitirá calcular rutas óptimas y evaluar el costo del viaje.

Algoritmo de Búsqueda en Amplitud (BFS): Para encontrar rutas cortas y brindar a los jugadores la experiencia de comprender conceptos clave del BFS mientras juegan.

Algoritmo de Dijkstra: Para calcular la distancia más corta entre ciudades y ayudar a los jugadores a tomar decisiones basadas en tiempo y gasolina.

Integración de JavaFX y SceneBuilder: Para desarrollar una interfaz de usuario atractiva y amigable con gráficos de mapas y elementos interactivos.

Matrices de Adyacencia y Listas de Adyacencia: Para representar las relaciones entre lugares y conexiones en el juego. Matrices para grafos no dirigidos y listas para dirigidos.

Pila (Stack) para Tareas Pendientes y Opciones de Mapa: Se utilizará una pila para administrar tareas pendientes y opciones de mapa en el juego. Los jugadores agregarán y eliminarán elementos de la pila para planificar su viaje y tomar decisiones sobre las opciones de mapa.

Cola (Queue) para Acciones a Deshacer: Se empleará una cola para gestionar acciones que los jugadores puedan deshacer en el juego, como retroceder en su ruta o reconsiderar decisiones. Las acciones se agregarán a la cola y se recuperarán en el orden en que se agregaron (FIFO - First-In, First-Out).

Implementación de Heapsort y Quicksort: Para ordenar elementos, como las ciudades a visitar, en función de prioridades y distancias.

Algoritmo de Árbol de Recubrimiento Mínimo: Para sugerir rutas más eficientes en el juego, ayudando a los jugadores a optimizar su viaje.

Pruebas Unitarias y Validación del Juego: Para garantizar la calidad y la integridad del juego.

Grafos No Ponderados: Para representar situaciones en el juego donde las conexiones no tienen un costo asociado, como eventos inesperados o sorpresas en el camino.

**Fase 4: Transición de Ideas a Diseños Preliminares**

Modo de Juego 1: Modo de Búsqueda en Amplitud (BFS)

Descripción: En este modo de juego, los jugadores disfrutan de emocionantes viajes sin la necesidad de comprender en detalle el algoritmo de Búsqueda en Amplitud (BFS). Aquí, la premisa es tomar decisiones desafiantes en un tiempo limitado, sin la gestión de recursos, y al final de la partida, la aplicación evaluará automáticamente la eficacia de las elecciones realizadas por los jugadores durante el juego.

Características Técnicas Relevantes:

Acceso Rápido a Gráficos Dispersos: Este modo utiliza representación de listas de adyacencia para facilitar el acceso a las conexiones entre ubicaciones en un grafo disperso y no necesariamente conexo.

Grafo No Ponderado: En este modo, los grafos no están ponderados, lo que permite a los jugadores tomar decisiones sin preocuparse por valores numéricos asociados a las conexiones.

Modo de Juego 2: Modo de Dijkstra

Descripción: En este modo de juego, los jugadores toman decisiones estratégicas para optimizar sus viajes en función del tiempo y los recursos, sin requerir un profundo conocimiento del algoritmo de Dijkstra. La aplicación de Dijkstra se ejecuta automáticamente al final de la partida para determinar la eficiencia de las decisiones tomadas por los jugadores.

Características Técnicas Relevantes:

Optimización de Rutas Ponderadas: En este modo, se aplican grafos ponderados para reflejar la optimización de rutas en términos de tiempo y recursos, lo que requiere que los jugadores tomen decisiones estratégicas basadas en valores numéricos asociados a las conexiones.

Evaluación Automática: La evaluación del rendimiento se basa en la ejecución automática del algoritmo de Dijkstra al final de la partida, lo que garantiza una medición objetiva de las elecciones realizadas por los jugadores.

Ideas Generales:

Uso de Matrices de Adyacencia: Las matrices de adyacencia se utilizan para representar las conexiones en "El Viajero Astuto" en situaciones donde la mayoría de los lugares están directamente conectados, lo que proporciona acceso rápido a las conexiones y es útil para grafos densos y conexos.

Uso de Listas de Adyacencia: Se prefieren las listas de adyacencia en el juego debido a la naturaleza dispersa del grafo, donde no todos los lugares están directamente conectados. Esto permite una representación eficiente de las conexiones entre lugares y puntos de interés, lo que es esencial para que los jugadores tomen decisiones estratégicas basadas en la optimización de recursos y tiempo.

Grafos Ponderados: Los grafos ponderados se aplican al modo de juego de Dijkstra para reflejar la optimización del tiempo y los recursos. Al igualar los pesos a 0, los grafos ponderados pueden considerarse como grafos no ponderados para el modo de juego de BFS.

Grafo No Conexo: Se emplea un grafo no conexo para reflejar la diversidad de rutas y ubicaciones en el juego. Esta elección contribuye a la autenticidad de la experiencia, ya que no todas las ubicaciones están directamente conectadas. Los jugadores deben tomar decisiones estratégicas para planificar su ruta a través de ubicaciones intermedias, lo que añade un nivel de complejidad y desafío al juego, fomentando la inmersión y decisiones significativas.

Gestión de Recursos con Colas y Pilas:

Pila (Stack): Se utiliza una pila para administrar tareas pendientes y opciones de mapa. Los jugadores pueden agregar y eliminar elementos de la pila para planificar su viaje y tomar decisiones.

Cola (Queue): Una cola se emplea para manejar acciones a deshacer de manera automática, permitiendo que los jugadores deshagan sus acciones en el orden en que se realizaron, siguiendo el principio FIFO (First-In, First-Out).

Interfaz de Usuario:

La elección de JavaFX y SceneBuilder asegura una interfaz de usuario atractiva y amigable para mejorar la experiencia del jugador.

Pruebas Unitarias: Se incluyen pruebas unitarias para garantizar la calidad del juego y su correcto funcionamiento.

**Fase 5: Criterios de Evaluación para las Ideas**

Para evaluar las propuestas de implementación del juego "El Viajero Astuto", hemos establecido criterios específicos. A continuación, se definen los criterios y las escalas para evaluar cada alternativa:

Eficiencia del Algoritmo:

Descripción: Evalúa la eficiencia de los algoritmos implementados (Dijkstra y BFS) en términos de tiempo de ejecución.

Escala:

1: Pobre eficiencia

2: Moderada eficiencia

3: Buena eficiencia

Evaluación:

Dijkstra: 3

BFS: 2

Experiencia del Usuario:

Descripción: Evalúa la interfaz de usuario y la experiencia general del jugador.

Escala:

1: Experiencia deficiente

2: Experiencia aceptable

3: Experiencia sobresaliente

Evaluación:

Interfaz de usuario: 3

Adaptabilidad a Modos de Juego:

Descripción: Evalúa qué tan bien se adaptan las implementaciones a los dos modos de juego distintos.

Escala:

1: Baja adaptabilidad

2: Adaptabilidad moderada

3: Alta adaptabilidad

Evaluación:

Modo 1 (Dijkstra): 3

Modo 2 (BFS): 2

Manejo de Recursos:

Descripción: Evalúa la eficacia del manejo de recursos mediante el uso de colas y pilas.

Escala:

1: Ineficiente manejo de recursos

2: Manejo de recursos aceptable

3: Eficiente manejo de recursos

Evaluación:

Colas y Pilas: 3

Complejidad y Desafío del Juego:

Descripción: Evalúa la complejidad y el desafío proporcionados por la estructura del grafo no conexo.

Scales:

1: Baja complejidad y desafío

2: Moderada complejidad y desafío

3: Alta complejidad y desafío

Ideas Evaluadas:

Idea 1 (Implementación actual):

Eficiencia del Algoritmo: 3 (Dijkstra y BFS implementados eficientemente).

Experiencia del Usuario: 3 (Interfaz de usuario atractiva y amigable).

Adaptabilidad a Modos de Juego: 3 (Algoritmos y estructuras de datos adaptados para ambos modos).

Manejo de Recursos: 3 (Optimizado con el uso de colas y pilas).

Complejidad y Desafío del Juego: 3 (Estructura de grafo no conexo agrega desafío).

Idea 2 (Otra propuesta):

Eficiencia del Algoritmo: 2 (BFS es rápido, pero no tan eficiente como Dijkstra en términos de costos).

Experiencia del Usuario: 3 (Interfaz de usuario atractiva y amigable).

Adaptabilidad a Modos de Juego: 2 (Adaptabilidad moderada a los modos).

Manejo de Recursos: 2 (Manejo de recursos aceptable).

Complejidad y Desafío del Juego: 2 (Complejidad moderada y desafío).

**TEST**

**GrafoListaAdyacenciaGenerico:**

* Método a probar: agregar Nodo

| **Clase** | **Método a probar** | **Escenario** | **Entradas** | **Salida Esperada** |
| --- | --- | --- | --- | --- |
| **GrafoListaAdyacenciaGenerico** | **agregarNodo** | **Estándar: Nodo no existente en el grafo** | **Valor del nodo: "A"** | **Nodo "A" agregado al grafo** |
| **GrafoListaAdyacenciaGenerico** | **agregarNodo** | **Límite: Nodo ya existente en el grafo** | **Valor del nodo: "A"** | **No se agrega un nuevo nodo al grafo** |
| **GrafoListaAdyacenciaGenerico** | **agregarNodo** | **Interesante: Agregar varios nodos diferentes** | **Valores de nodos: "B", "C", "D"** | **Nodos "B", "C", "D" agregados al grafo** |

* Método a probar: agregarArista

| **Clase** | **Método a probar** | **Escenario** | **Entradas** | **Salida Esperada** |
| --- | --- | --- | --- | --- |
| **GrafoListaAdyacenciaGenerico** | **agregarArista** | **Estándar: Arista entre nodos existentes** | **Valor del nodo 1: "A", Valor del nodo 2: "B"** | **Arista agregada entre nodos "A" y "B"** |
| **GrafoListaAdyacenciaGenerico** | **agregarArista** | **Límite: Nodo 1 no existente en el grafo** | **Valor del nodo 1: "X", Valor del nodo 2: "Y"** | **No se agrega arista, ya que el nodo "X" no existe** |
| **GrafoListaAdyacenciaGenerico** | **agregarArista** | **Interesante: Crear un ciclo en el grafo** | **Valores de nodos: "A", "B", "C"** | **Aristas agregadas para formar un ciclo "A-B-C-A"** |

* Método a probar: eliminarNodo

| **Clase** | **Método a probar** | **Escenario** | **Entradas** | **Salida Esperada** |
| --- | --- | --- | --- | --- |
| **GrafoListaAdyacenciaGenerico** | **eliminarNodo** | **Estándar: Nodo existente con aristas** | **Valor del nodo a eliminar: "A"** | **Nodo "A" y sus aristas eliminados del grafo** |
| **GrafoListaAdyacenciaGenerico** | **eliminarNodo** | **Límite: Nodo no existente en el grafo** | **Valor del nodo a eliminar: "X"** | **No se realiza ninguna acción, ya que el nodo "X" no existe** |
| **GrafoListaAdyacenciaGenerico** | **eliminarNodo** | **Interesante: Eliminar nodo con aristas y verificar conexiones restantes** | **Valor del nodo a eliminar: "B"** | **Nodo "B" y sus aristas eliminados del grafo, conexiones restantes ajustadas** |

**GrafoMatrizAdyacenciaGenerico:**

* Método a probar: agregarNodo

| **Clase** | **Método a probar** | **Escenario** | **Entradas** | **Salida Esperada** |
| --- | --- | --- | --- | --- |
| **GrafoMatrizAdyacenciaGenerico** | **agregarNodo** | **Estándar: Nodo no existente en el grafo** | **Valor del nodo: "A"** | **Nodo "A" agregado al grafo** |
| **GrafoMatrizAdyacenciaGenerico** | **agregarNodo** | **Límite: Nodo ya existente en el grafo** | **Valor del nodo: "A"** | **No se agrega un nuevo nodo al grafo** |
| **GrafoMatrizAdyacenciaGenerico** | **agregarNodo** | **Interesante: Agregar varios nodos diferentes** | **Valores de nodos: "B", "C", "D"** | **Nodos "B", "C", "D" agregados al grafo** |

* Método a probar: agregarArista

| **Clase** | **Método a probar** | **Escenario** | **Entradas** | **Salida Esperada** |
| --- | --- | --- | --- | --- |
| **GrafoMatrizAdyacenciaGenerico** | **agregarArista** | **Estándar: Arista entre nodos existentes** | **Valor del nodo 1: "A", Valor del nodo 2: "B"** | **Arista agregada entre nodos "A" y "B"** |
| **GrafoMatrizAdyacenciaGenerico** | **agregarArista** | **Límite: Nodo 1 no existente en el grafo** | **Valor del nodo 1: "X", Valor del nodo 2: "Y"** | **No se agrega arista, ya que el nodo "X" no existe** |
| **GrafoMatrizAdyacenciaGenerico** | **agregarArista** | **Interesante: Crear un ciclo en el grafo** | **Valores de nodos: "A", "B", "C"** | **Aristas agregadas para formar un ciclo "A-B-C-A"** |

* Método a probar: eliminarNodo

| **Clase** | **Método a probar** | **Escenario** | **Entradas** | **Salida Esperada** |
| --- | --- | --- | --- | --- |
| **GrafoMatrizAdyacenciaGenerico** | **eliminarNodo** | **Estándar: Nodo existente con aristas** | **Valor del nodo a eliminar: "A"** | **Nodo "A" y sus aristas eliminados del grafo** |
| **GrafoMatrizAdyacenciaGenerico** | **eliminarNodo** | **Límite: Nodo no existente en el grafo** | **Valor del nodo a eliminar: "X"** | **No se realiza ninguna acción, ya que el nodo "X" no existe** |
| **GrafoMatrizAdyacenciaGenerico** | **eliminarNodo** | **Interesante: Eliminar nodo con aristas y verificar conexiones restantes** | **Valor del nodo a eliminar: "B"** | **Nodo "B" y sus aristas eliminados del grafo, conexiones restantes ajustadas** |