1.) Identificación del problema

El problema planteado en el contexto del proyecto "Fuga del Edificio D: Misión Profesor Icesi" se centra en el desarrollo de un juego que involucra la representación y el uso de la teoría de grafos para crear una experiencia de juego desafiante y entretenida. El objetivo principal del juego es guiar al protagonista, un profesor hastiado de su rutina académica, a través de un edificio universitario compuesto por cinco pisos, cada uno con sus respectivos salones y pasillos. El desafío radica en evitar a los estudiantes que pueblan los pasillos mientras se navega por el edificio. Este juego plantea diversas necesidades clave que deben abordarse para lograr su implementación exitosa.

Las necesidades planteadas en este contexto son reales y fundamentales para el desarrollo del juego "Fuga del Edificio D: Misión Profesor Icesi", incluyen la creación de una representación precisa y eficiente del edificio universitario en forma de grafo, la implementación de al menos dos algoritmos de grafos (por ejemplo, BFS y Dijkstra) para encontrar rutas óptimas y evadir a los estudiantes, y la construcción de una interfaz gráfica de usuario que permita a los jugadores interactuar con el juego de manera efectiva.

En resumen, el problema identificado se relaciona con la creación de un juego basado en la teoría de grafos, que requiere la implementación de al menos dos algoritmos de grafos, una representación eficiente del edificio universitario y una interfaz gráfica de usuario. Estas necesidades se han definido a grandes rasgos y se han diferenciado de las soluciones específicas, evitando prejuicios en la definición del problema. El enfoque se centra en la creación de una experiencia de juego desafiante y entretenida para los jugadores.

2.) Recopilación de información

Con fin de suplir todas las necesidades actuales y las posibles necesidades emergentes del problema en cuestión, la búsqueda empezará por los siguientes conceptos: Grafo, Vértices y Aristas, Grafo Dirigido y No Dirigido, Grafo Ponderado y No Ponderado, Ciclo y Camino, Grado de un Vértice, Conectividad, Grafo Acíclico, Árbol, Árbol de Expansión Mínima (MST), Algoritmos de Búsqueda, Algoritmos de Caminos Más Cortos, Algoritmos de Flujo Máximo, Teorema de Euler y Hamilton, Aplicaciones en la Vida Real.

Para esto tomaremos varías páginas referencia con fin de clarificar la teoría y comprender su implementación real dentro del entorno que estamos tratando:

<https://ccc.inaoep.mx/~villasen/CursoMatDiscretas/Grafos.pdf>

<https://elpixelamplificado.wordpress.com/2020/04/07/las-matematicas-de-los-videojuegos-los-grafos/>

<https://www.freecodecamp.org/espanol/news/algoritmo-de-la-ruta-mas-corta-de-dijkstra-introduccion-grafica/#:~:text=El%20algoritmo%20de%20Dijkstra%20encuentra,los%20dem%C3%A1s%20nodos%20del%20grafo>.

<https://e-archivo.uc3m.es/bitstream/handle/10016/23954/TFG_Alvaro_Parra_De_Miguel_2015.pdf?sequence=1&%3BisAllowed=y>(Sección algoritmos de Búsqueda).

<https://core.ac.uk/download/pdf/421934731.pdf>(Sección Teoría de grafos (2.2))

**Conceptos y definiciones recopiladas:**

1. ***Grafo***: Un grafo es una estructura matemática que consta de un conjunto de nodos (vértices) y un conjunto de aristas (bordes) que conectan estos nodos. Los grafos se utilizan para representar relaciones entre objetos.

2***. Vértices y Aristas***: Los vértices (nodos) son los puntos en un grafo, y las aristas (bordes) son las conexiones entre los vértices. Las aristas pueden ser dirigidas (de un vértice a otro) o no dirigidas.

3. ***Grafo Dirigido y No Dirigido***: En un grafo dirigido, las aristas tienen una dirección, es decir, van desde un vértice de origen a un vértice de destino. En un grafo no dirigido, las aristas no tienen una dirección específica.

4. ***Grafo Ponderado y No Ponderado***: Un grafo ponderado asigna un peso o valor a cada arista, lo que puede representar distancias, costos o cualquier otro atributo relevante. En un grafo no ponderado, todas las aristas tienen el mismo valor.

5. ***Ciclo y Camino***: Un ciclo es una secuencia de vértices y aristas que comienza y termina en el mismo vértice. Un camino es una secuencia de vértices y aristas que conecta dos vértices diferentes.

6***. Grado de un Vértice***: El grado de un vértice es el número de aristas que se conectan a ese vértice. En un grafo dirigido, se distingue entre el grado de entrada (aristas que llegan al vértice) y el grado de salida (aristas que salen del vértice).

7. ***Conectividad***: Un grafo se considera conectado si hay un camino entre cualquier par de vértices. Un grafo no conectado se divide en componentes conectados.

8. ***Grafo Acíclico***: Un grafo acíclico es aquel que no contiene ciclos, es decir, no se puede seguir un camino cerrado.

9. ***Árbo***l: Un árbol es un grafo no dirigido, conectado y acíclico. Un árbol es una estructura jerárquica que se utiliza en muchas aplicaciones, incluidas las estructuras de datos.

10. ***Árbol de Expansión Mínima*** (Minimum Spanning Tree - MST): Un MST de un grafo no dirigido y ponderado es un subgrafo que contiene todos los vértices y la menor cantidad de aristas posible para conectar todos los vértices sin formar ciclos.

11. ***Algoritmos de Búsqueda***: Los algoritmos de búsqueda en grafos, como BFS (Breadth-First Search) y DFS (Depth-First Search), se utilizan para explorar y buscar información en un grafo.

12. ***Algoritmos de Caminos Más Cortos***: Algoritmos como Dijkstra y Bellman-Ford se utilizan para encontrar el camino más corto entre dos vértices en un grafo ponderado.

13. ***Algoritmos de Flujo Máximo***: Estos algoritmos se utilizan en problemas de redes para encontrar el flujo máximo a través de una red de nodos y aristas.

14. ***Teorema de Euler y Hamilton***: Estos teoremas se relacionan con circuitos y caminos en grafos y proporcionan propiedades interesantes.

3.) ***Búsqueda de Soluciones Creativas***

Representación grafica :

Piso 1:

***S1 -- S2***

***| |***

***S3 -- S4***

Piso 2:

**S5 -- S6**

**| |**

**S7 -- S8**

Piso 3:

**S9 -- S10**

**| |**

**S11 -- S12**

En esta representación, los | representan las aristas verticales que conectan los salones en cada piso, y los -- representan las aristas horizontales que conectan los salones en el mismo piso. Cada sala está etiquetada como S1, S2, S3, y así sucesivamente. Lo anterior simplemente fue una representación guardando proporciones mas pequeñas, en el proyecto real cada grafo tendrá 10 vértices.

***Representación del Edificio como un Grafo:***

El edificio se representa como un grafo G = (V, E), donde V es el conjunto de vértices (S1 a S12) y E es el conjunto de aristas (pasillos). Esto proporciona una estructura matemática para el edificio.

***Búsqueda de Rutas Óptimas y Evitación de Estudiantes:***

Supongamos que aplicamos el algoritmo BFS (Breadth-First Search) desde la ubicación actual del profesor (por ejemplo, S1) al ascensor (por ejemplo, S9) en el tercer piso. Este algoritmo calcula la ruta más corta considerando el peso de las aristas (representando la dificultad de pasar en presencia de estudiantes).

Usando Dijkstra, podríamos encontrar la ruta más segura para evitar a los estudiantes. Dijkstra calcula la ruta más corta teniendo en cuenta el peso de las aristas. El profesor podría utilizar esta información para eludir a los estudiantes en los pasillos.

***Cálculo de Rutas para Ascensores:***

Supongamos que el profesor desea llegar al ascensor en el tercer piso (S9). Aplicamos Dijkstra desde su ubicación actual (por ejemplo, S4) para encontrar la ruta más corta hasta el ascensor. Esto se hace considerando el peso de las aristas (representando la dificultad de los pasillos).

***Dinámica del Juego y Aumento de la Dificultad:***

A medida que el profesor avanza de un piso a otro, se pueden agregar más estudiantes en los pasillos, aumentando así el peso de las aristas correspondientes a esos pasillos. Esto se puede representar matemáticamente al actualizar los valores de peso en las aristas a medida que el profesor avanza.

***Estrategias de Huida y Caminos Alternativos:***

El juego puede permitir al jugador calcular rutas alternativas utilizando algoritmos como BFS o Dijkstra para evitar a los estudiantes y llegar al ascensor de manera segura.

***Control de Flujo del Juego:***

Podemos usar algoritmos de grafos para modelar cuándo se produce un evento de "descubrimiento" por parte de los estudiantes, como calcular cuándo el profesor y los estudiantes están en el mismo pasillo.

***Otras alternativas un poco mas avanzadas*** que involucran inteligencia artificial y algoritmos mas avanzados y que darían solución al juego son:

Propuesta tentativa:

***Algoritmo A:\****

En lugar de usar Dijkstra para encontrar la ruta más corta, podrías implementar el algoritmo A\*. A\* es una variante de búsqueda informada que utiliza una heurística para estimar la distancia restante hasta el objetivo. Puede proporcionar rutas más eficientes en términos de tiempo de ejecución que Dijkstra.

***Algoritmo de Evitación de Obstáculos:***

Además de considerar la presencia de estudiantes como obstáculos, podríamos implementar un algoritmo de evitación de obstáculos que permita al profesor planificar rutas evitando obstáculos dinámicos en tiempo real. Esto agregaría un elemento de estrategia adicional.

***Algoritmo Genético:***

Podríamos explorar la implementación de algoritmos genéticos para que el profesor "aprenda" a evadir a los estudiantes de manera más efectiva a lo largo del juego. Esto introduciría la inteligencia artificial en el juego.

***Algoritmo de Recorrido del Vendedor Viajero (TSP):***

En lugar de buscar una ruta simple hacia el ascensor, podríamos formular el juego como un problema del TSP, donde el profesor debe visitar todos los salones antes de llegar al ascensor. Esto haría que el juego sea más complejo y desafiante.

***Algoritmos de Redes Neuronales Artificiales:***

Podríamos implementar redes neuronales artificiales para modelar el comportamiento de los estudiantes y ajustar su "inteligencia" durante el juego. Esto permitiría que los estudiantes se vuelvan más astutos a medida que el juego avanza.

***Algoritmos de Difusión:***

En lugar de considerar a los estudiantes como obstáculos fijos, podrías usar algoritmos de difusión o propagación en el grafo para simular el movimiento de los estudiantes a lo largo del tiempo. Esto haría que la dinámica de los estudiantes sea más dinámica y realista.

***Algoritmo de Evolución de Comportamiento:***

Implementaríamos un algoritmo de evolución de comportamiento que permita a los estudiantes desarrollar estrategias de búsqueda del profesor de manera adaptativa. Los estudiantes pueden aprender y mejorar su estrategia con el tiempo.

***Variación en las Reglas del Juego:***

Modificaríamos las reglas del juego a medida que el profesor avanza, como agregar elementos de juego, power-ups o obstáculos que cambian dinámicamente. Esto puede agregar variedad y emoción al juego.