**Método de la ingeniería**

**Paso 1. Identificación del Problema**

Se necesita un software con las siguientes características:

* Ser un videojuego
* Que funcione mediante una implementación propia de grafos
* El grafo debe estar implementado de dos formas: a modo de lista enlazada o a modo de matriz de adyacencia
* El software debe usar al menos dos algoritmos de grafos, pero tener 6 funcionales (DFS, BFS, Dijkstra, Kruskal, Floyd Warshall y Prim).

**Paso 2. Recopilación de Información**

**Grafo:**

Un grafo es una estructura de datos que consiste en un grupo de nodos o vértices y aristas, los primeros representando las unidades de los elementos que se buscan representar y las aristas las conexiones de estos elementos. Un ejemplo de un grafo sería una red social, las personas serían los vértices y las relaciones de amistad son aristas que juntan los vértices.

Hay varios tipos de grafos pero los importantes más importantes son los grafos dirigidos y los no dirigidos: en el primero las relaciones no son binarias, es decir, que el vértice “A” tenga una relación “B” con el vértice “C” no es lo mismo que el vértice “C” tenga una relación “B” con el vértice “A”; por el contrario, en el grafo no dirigido, si hay una relación “B” de “A” a “C”, siempre es verdad que hay una relación “B” de “C” a “A”.

Otro tipo de grafo son los grafos con pesos, estos indican cierta “resistencia” de llegar de un vértice a otro a través de una arista. Un ejemplo de grafos con pesos serían las carreteras: para llegar de una ciudad a otra hay un coste, un peso, ya sea de tiempo o de consumo de gasolina.

Para representar un grafo se tienen dos formas principales: la lista de adyacencia y la matriz de adyacencia. En el primer tipo, para saber a que nodos se puede acceder el nodo numero “i” se tiene que llamar a los valores que hay en esa posición. En el segundo tipo se tiene una matriz que contiene los pesos (si es un grafo sin pesos se pone 1) para llegar del nodo de la fila al de las columnas, un ejemplo sería que al ver la posición (i, j) se obtiene un número (el peso) que dice si se puede llegar del nodo i al nodo j.

Hay varias operaciones con grafos como el añadir, quitar o modificar vértices y aristas. Adicionalmente hay varios algoritmos para recorrer los grafos como DFS y BFS para recorridos en grafos sin peso, Floyd warshall y Dijkstra para caminos mínimos en grafos con peso y Prim y Kruskal para hallar el subgrafo con menores pesos posibles.

GeeksforGeeks. (2023). Graph Data Structure and Algorithms. https://www.geeksforgeeks.org/graph-data-structure-and-algorithms/

**Paso 3. Búsqueda de Soluciones Creativas**

Se tienen dos principales ideas, la primera es hacer una versión propia del popular juego del Buscaminas y la otra es hacer un juego original con el siguiente enunciado:

"Venganza Romana" es un emocionante juego de estrategia basado en un grafo no dirigido con más de 50 nodos y aristas que representa una parte de la Europa medieval. El jugador comienza en la histórica ciudad de Sicilia con el objetivo final de conquistar la base vikinga. Para lograrlo, deberá conquistar diferentes reinos, ciudades y pueblos, enfrentándose a desafíos estratégicos en cada territorio.

El mapa está organizado en niveles, con Sicilia como punto de partida. El jugador avanza a través de pueblos interconectados, cada uno con rutas que conducen a niveles superiores Cada territorio tiene un peso asociado, indicando la dificultad de conquista. El jugador debe tomar decisiones estratégicas para maximizar recursos y minimizar pérdidas de soldados.

**Paso 4. Transición de las Ideas a los Diseños Preliminares**

La idea general es usar al menos 2 algoritmos de grafos, por ejemplo, el usar BFS en el buscaminas para hallar “zonas libres de minas” de forma automática, o usar Dijkstra u Floyd Warshall para hallar el camino mínimo hacia los vikingos en “Venganza Romana”. También se podría usar Prim en venganza romana para hallar las ciudades más fáciles de conquistar y así ganar más puntos.

Cabe recalcar que hay que usar una interfaz gráfica por lo que se pensó en JavaFX, Java Swing, SWT o AWT.

**Paso 5. Evaluación y Selección de la Mejor Solución**

**Paso 6. Preparación de Informes y Especificaciones**

**Paso 7. Implementación del Diseño**