**INFORME DE INGENIERÍA**

**PROYECTO FINAL**

**MARIA CAMILA LENIS RESTREPO**

**JUAN SEBASTIAN PALMA GARCÍA**

**JAVIER ANDRÉS TORRES REYES**

**ALGORITMOS Y ESTRUCTURAS DE DATOS**

**2018-2**

**INFORME DE INGENIERÍA**

**Paso 1: Identificación del problema**

**Definición del problema**

Implementar una guía, como un plano, para los visitantes de la mansión Winchester.

**Justificación**

La mansión Winchester ha existido desde hace mucho tiempo y al convertirse en un museo se ha decidido implementar una renovación al tipo de planificación utilizando la mansión. Primero, se ha buscado implementar un nuevo mapa que innove y facilite la movilidad de los visitantes de una sala a otra y que les sirva como guía para llegar a una sala que ellos quieran. Además, han decidido modificar el sistema de inventario de la mansión ya que se han descubierto las misteriosas desapariciones de objetos valiosos de algunas habitaciones y así evitar la perdida de estos valiosos objetos. Inclusive, se ha tomado en cuenta que la mansión al tener una gran antigüedad se requiere la demolición y reconstrucción o adición de nuevas habitaciones para que el museo logre mantenerse abierto al publico y no imponer algún tipo de riesgo a cualquier visitante. Por último, entre estos nuevos planes se ha hecho una inversión mayor en los sistemas de comunicación dentro de la mansión para permitir una mayor facilidad de comunicación entre el museo y los visitantes en caso del cierre del museo y así evitar que algunas personas terminan encerradas dentro de la casa embrujada como ha ocurrido antes.

**Requerimientos funcionales**

1. Dada una habitación de la mansión se debe encontrar el camino más rápido, en minutos, desde esa habitación hasta la salida. Si la habitación no tiene salida, de debe mostrar un mensaje de advertencia.
2. El sistema debe encontrar el camino que pase por menos habitaciones desde un punto a otro de la mansión. El usuario debe ingresar el punto de partida y el de llegada, y recibe una secuencia de habitaciones incluyendo el punto de partida y el de llegada.
3. El sistema debe transmitir el mensaje de cierre a todos los rincones de la casa, de manera que este llegue de la forma más rápida posible teniendo en cuenta lo que se demora cruzar de una habitación a otra. Es importante aclarar que el mensaje, por su misma naturaleza, puede ignorar la dirección de los corredores, de manera que es suficiente que un corredor vaya de B a A para que el mensaje pueda ir de A a B.
4. Añadir una habitación a la mansión. La nueva habitación debe contener el indicador, las habitaciones a las cuales se puede llegar a través de ella, y las habitaciones de las cuales se puede llegar a ella.
5. Dado el indicador de la habitación se debe eliminar la habitación del mapa. Si la habitación contenía tesoros, estos deben quedar en el registrados en el museo.
6. Dado el indicador de dos habitaciones se deben añadir un pasillo entre ella de la siguiente manera: primera habitación indica desde dónde, segunda habitación indica hasta donde y el tiempo de minutos que toma cruzar el pasillo entre ambas habitaciones. Si ya existe un pasillo entre la primera y la segunda, muestra un mensaje de error.
7. Dado el indicador de dos habitaciones se debe eliminar el pasillo que exista entre ellas. Si no existe ningún pasillo se muestra un mensaje de alerta.
8. Dado el indicador de la habitación se deben registrar tesoros encontrados. Se debe añadir el nombre y el valor del tesoro y la habitación a la cual pertenece.
9. Visualizar los tesoros encontrados, ya sea que aún pertenezcan a la habitación o que pertenezcan al museo. Se debe mostrar su nombre, valor, habitación a la que pertenece o, en su defecto, que pertenece al museo.

**Paso 2: Recopilación de la información**

**Sobre la mansión Winchester**

Se sabe que la mansión Winchester fue empezada a construir alrededor de los años 1881 en California, luego de la muerte de William Winchester, y no cesó hasta el fallecimiento de Sarah Winchester, quien ponía reglas muy claras a los obreros: nada de planos. Por ende, nadie saber ciertamente como está compuesta, pero de lo que se ha logrado apreciar tiene:

* Cuatro pisos, aunque hubo un tiempo donde alcanzó a tener siete pisos
* Dos hectáreas de longitud.
* 160 cuartos
* 40 recámaras
* Dos salones de baile
* Habitaciones secretas
* Infinitud de ventanas, puertas y muchas de ellas conducen a una pared o al vacío.
* Decoraciones con el número 13 (candelabros con 13 velas, ventanas con 13 vidrios, mosaicos con 13 particiones).
* 476 entradas.

De lo anterior se puede abstraer que para nuestra solución se debe tomar en cuenta que:

* Existen múltiples entradas, y por ende salidas, de la mansión a la hora de hallar el camino más corto hacia una salida.
* Los tesoros encontrados estarían relacionados con tener alguna característica con el número 13.
* No siempre se va a tener un camino hasta la salida

**Aproximación de la mansión Winchester**

De manera muy general, la mansión Winchester se aproxima a lo que es un laberinto, y puede ser visto de la siguiente manera:

Imagen que contiene texto, mapa

Descripción generada con confianza muy alta

Donde se puede concluir que los pasos de un lado a otro se conectan y hay puntos de salida y puntos de llegada, lo que puede representarse en un grafo donde a) es una aproximación de una matriz de adyacencia, y en b) como la gráfica del grafo como tal.

**Grafo**

Un grafo es una estructura de datos no linear que se compone de Vértices y Aristas. Los vértices o nodos son los componentes del grafo que contienen un objeto y las aristas son caminos que conectan un vértice a otra. Un grafo puede ser direccionado o no direccionado, en el cual si este es direccionado solo puede ir de un vértice inicial a un final sin poder devolverse a la inicial a menos de que exista una arista que conecte este nodo con el inicial. El grafo no direccionado tiene aristas que permiten el viaje entre vértices mediante el uso de una sola arista ya que no tiene ningún limitante. Un grafo se considera conexo cuando se encuentra completamente conectado y no existe mas de un grafo, el no conexo esta separado en dos subgrafos.

Tomado de <https://www.geeksforgeeks.org/graph-and-its-representations/> & <https://www.geeksforgeeks.org/graph-data-structure-and-algorithms/>

Como representar un Grafo:

* Lista de Adyacencia: Es una Lista de Listas que contiene todos los vértices y marca dentro de cada una en base a un vértice si tiene aristas que los conectan entre si.
* Matriz de Adyacencia: Es una matriz de vértices que compara cada vértice con otro y si alguno tiene un vértice que lo conecta con otro se marca dentro de la intersección de la matriz la existencia de la arista, con su peso o un uno.

**DFS**

DFS o Depth First Search, es un algoritmo que crea un recorrido desde un nodo inicial A, hacia el resto sin repetir los nodos que visita. Este algoritmo recorre la profundidad de un nodo hasta que no existan más nodos por los cuales se puede recorrer, antes de pasar al siguiente. Este método utiliza un arreglo booleano que mantiene un chequeo de los nodos o vértices ya visitados por la búsqueda. Al final se retorna un árbol con los órdenes de los vértices mediante profundidad.

Tomado de <https://www.geeksforgeeks.org/depth-first-search-or-dfs-for-a-graph/>

**BFS**

BFS o Breath First Search, es un algoritmo que recorre desde un nodo inicial A, a todos los demás, sin repetir visitas. Este algoritmo recorre todos los nodos adyacentes antes de seguir bajando niveles dentro de cada nodo. Este método utiliza un arreglo booleano que mantiene un chequeo de los nodos o vértices ya visitados por la búsqueda. Al final se retorna un árbol con los órdenes de los vértices mediante su nivel de adyacencia.

Tomado de <https://www.geeksforgeeks.org/breadth-first-search-or-bfs-for-a-graph/>

**Recorrido de Camino Mínimo**

Existen tres algoritmos de búsqueda de camino mínimo:

* Disjtrak: Este es un algoritmo que genera un árbol de camino más corto en el cual empieza desde un Vértice Inicial y se asignan todos los valores a cada vértice como infinito o un numero extremadamente grande, y se empieza a recorrer cada vértice sumando sus pesos y asignándoselos al mismo, y evalúa si otro nodo por otro lado llega a ese mismo vértice y evalúa cual es menor. Al final arma un árbol con los caminos más bajos desde el nodo elegido.

Tomado de <https://www.geeksforgeeks.org/dijkstras-shortest-path-algorithm-greedy-algo-7/>

* Bellman-Ford: Este es un algoritmo que logra hacer lo mismo que el algoritmo de Disjtrak, solo que este es capaz de manejar los vértices que tienen pesos negativos sin llegar a un bucle infinito. Este método consiste en crear una lista de todos los vértices que contendrá la distancias de estos a la inicial. Todas las distancias se colocan como infinitas excepto la de la inicial y luego se van llenando las distancias con un recorrido, y al mismo tiempo se evalúa si por otro vértice existe una conexión y este camino tiene menor peso al ya establecido
  + Este método maneja los pesos negativos a través de una comparación en la cual la distancia de V es mayor a la distancia de U mas la arista entre UV, existe un valor negativo.

Tomado de <https://www.geeksforgeeks.org/bellman-ford-algorithm-dp-23/>

* Floyd-Warshall: este método genera un bosque de arboles de caminos más cortos, y lo aplica para cada uno de los vértices. Este algoritmo mediante una matriz de vértices para buscar el camino más corto en cada árbol y así crear el bosque de árboles. Utiliza un método muy parecido al de Disjtrak para evaluar las distancias dentro de los árboles.

Tomado de <https://www.geeksforgeeks.org/floyd-warshall-algorithm-dp-16/>

**Arboles de Recubrimiento mínimo**

Los arboles de recubrimiento mínimo son subgrafos provenientes de un grafo conectado y no dirigido que tienen la función de conectar todos los vértices. Estos arboles buscan crear el camino de menor peso entre un vértice y todos los demás mediante la suma mínima de los pesos de las aristas del nodo. Los dos métodos que se utilizan son:

* Prim: Este método toma los vértices adyacentes y compara los pesos. Este método contiene una lista que mantiene la cuenta de vértices que ya han sido des cubiertos por el algoritmo y les asigna a todos los vértices un valor infinito y al inicial de 0, luego revisa los adyacentes y agrega el menor peso, este evalúa después si existe una ruta de menor peso, si la hay reemplaza el actual por el nuevo y repite el proceso con los nuevos nodos descubiertos.
* Kruskal: Este método lista todos los pesos de las aristas de manera ascendente, se agrega la arista más baja y se agrega, luego se toma la siguiente y se verifica que no haya un ciclo, cuando esta condición se cumple lo agrega al árbol. Esto se hace hasta que se recorren todas las aristas y los vértices se encuentran conectados sin la existencia de un ciclo completo.

Tomado de: <https://www.geeksforgeeks.org/kruskals-minimum-spanning-tree-algorithm-greedy-algo-2/> y <https://www.geeksforgeeks.org/prims-minimum-spanning-tree-mst-greedy-algo-5/>

**Paso 3: Búsqueda de soluciones creativas**

**Para la guía**

Se realizó una lista de atributos para determinar la estructura de datos a usar para implementar la guía en la cual se resumen todas las características que debe cumplir y sus restricciones:

* Un punto donde guardar el objeto habitación.
* Conexión entre habitaciones, las cuales deben permitir ir de una a otra, pero no necesariamente el regreso.
* Habitaciones que no conduzcan a ninguna parte.
* Caminos para llegar a la salida.
* Permitir agregar y borrar conexiones y habitaciones.

Debido a los atributos que la estructura requiere se llegó a la conclusión de que las siguientes alternativas corresponden a una solución para el problema:

1. Grafo simple
2. Multígrafo
3. Pseudografo
4. Grafo dirigido
5. Multígrafo dirigido
6. Árbol n-ario
7. Hash Table

**Para la recolección de tesoros**

Se realizó una lluvia de ideas para manejar la recolección de tesoros dentro de la guía de la mansión. Cabe resaltar que la funcionalidad de la recolección de tesoros sirve para llevar un registro de los tesoros encontrados empíricamente, no dentro de la aplicación, es decir, la persona ingresa los tesoros que encuentra en la mansión, no los encuentra en la aplicación. Por lo tanto, se tienes las siguientes alternativas para guardar los tesoros:

1. Los tesoros pertenecen a la mansión, por lo tanto, quedan guardados en una lista enlazada que le pertenece a la mansión.
2. Los tesoros serían guardados en un HashMap.
3. Los tesoros serían guardados en una lista enlazada perteneciente a cada habitación.
4. Para facilitar la búsqueda de los tesoros, son guardados en un árbol binario de búsqueda.
5. Crear un grafo para mostrar la distribución de los tesoros.
6. Utilizar una pila para ir guardando los tesoros encontrados
7. Cada habitación tendría un ArrayList donde guardaría los tesoros encontrados.

**Paso 4: Transición de ideas a los diseños preliminares**

**Para la guía**

Teniendo en cuenta que las conexiones entre las habitaciones podrían acabar siendo cíclicas, y de esta forma tendría conexiones entre los hijos de una raíz y otra; esto no debe ocurrir en un árbol, de esta manera no cumpliría los requisitos para ser un árbol. Por otro lado, un Hash Table pretende ser un diccionario, el cual puede servir para guardar el índice o clave de la habitación para acceder a ella directamente en memoria, pero no modelaría las conexiones entre habitaciones. Por lo tanto, ambas ideas quedan descartadas.

Las ideas restantes corresponden a tipos de grafos, cuyas características están descritas por la siguiente tabla:

|  |  |  |  |
| --- | --- | --- | --- |
|  | Tipos de aristas | ¿Admite aristas múltiples? | ¿Admite bucles? |
| Grafo simple | No dirigidas | No | No |
| Multígrafo | No dirigidas | Si | No |
| Pseudografo | No dirigidas | Si | Si |
| Grafo dirigido | Dirigidas | No | Si |
| Multígrafo dirigido | Dirigidas | Si | Si |

**Para la recolección de tesoros**

**Paso 5: Evaluación o selección de la mejor solución (Criterios y selección)**

**Para la guía**

Criterio A: Conexión entre habitaciones. La conexión entre habitaciones debe permitir llegar de una a otra, pero no necesariamente de permitir el regreso.

* [3] Permite llegar de una habitación a otra, pero no necesariamente el regreso
* [1] Permite llegar de una habitación a otra y viceversa.

Criterio B: Conexión hacia sí mismo. Permite modelar la situación en que una habitación no tiene salida, por lo tanto, su única arista adyacente es hacia el mismo vértice.

* [3] Admite bucles
* [1] No admite bucles

Criterio C: Aristas múltiples. Para llegar de una habitación a otra, directamente, solo existe un pasillo, por ende, no tiene aristas múltiples.

* [3] No admite aristas múltiples
* [1] Admite aristas múltiples

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
|  | Criterio A | Criterio B | Criterio C | Total |
| Grafo simple | 1 | 1 | 3 | 5 |
| Multígrafo | 1 | 1 | 3 | 5 |
| Pseudografo | 1 | 3 | 1 | 5 |
| Grafo dirigido | 3 | 3 | 3 | 9 |
| Multígrafo dirigido | 3 | 3 | 1 | 7 |

El tipo de grafo escogido para modelar la guía de la mansión es un **grafo dirigido**, porque cumple con la capacidad de conexión restringida entre habitaciones, modelar una habitación sin salida, y no tiene múltiples conexiones entre dos habitaciones.

**Para la recolección de tesoros**

**Paso 6: Preparación de informes**

**Diagrama de clases de la solución**

**Diagrama de objetos**

**Diseño de casos de pruebas unitarias**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Prueba 1: | | | | |
| Clase | Método | Escenario | Entrada | Resultado |
|  |  |  |  |  |
|  |  |  |  |  |
|  |  |  |  |  |
|  |  |  |  |  |
|  |  |  |  |  |

**Bibliografía**

Anónimo. (Desconocido). “Algoritmo de Prim”. Tomado de: <https://sites.google.com/site/complejidadalgoritmicaes/prim>

Anonimo. (Desconocido). “Bellma-Ford Algorithm | DP-23”. Tomado de: <https://www.geeksforgeeks.org/bellman-ford-algorithm-dp-23/>

Anonimo. (Desconocido). “Breadth First Search or BFS for a Graph”. Tomado de: <https://www.geeksforgeeks.org/breadth-first-search-or-bfs-for-a-graph/>

Anonimo. (Desconocido). “Depth First Search or DFS for a Graph”. Tomado de: <https://www.geeksforgeeks.org/depth-first-search-or-dfs-for-a-graph/>

Anonimo. (Desconocido). “Dijkstra´s shortest path algorithm | Greed Algo-7”. Tomado de: <https://www.geeksforgeeks.org/dijkstras-shortest-path-algorithm-greedy-algo-7/>

Anonimo. (Desconocido). “Floyd Warshall Algorithm | DP-16”. Tomado de: <https://www.geeksforgeeks.org/floyd-warshall-algorithm-dp-16/>

Anonimo. (Desconocido). “Graph and its representations”. Tomado de: <https://www.geeksforgeeks.org/graph-and-its-representations/>

Anonimo. (Desconocido). “Graph Data Structure and Algorithms”. Tomado de: <https://www.geeksforgeeks.org/graph-data-structure-and-algorithms/>

Anonimo. (Desconocido). “Kruska´ls Minimum Spanning Tree Algorithm | Greed Algo-2”. Tomado de: <https://www.geeksforgeeks.org/kruskals-minimum-spanning-tree-algorithm-greedy-algo-2/>

Anónimo. (Mayo 09 del 2018). “La tenebrosa historia de la casa Winchester (+Video)” Tomado de: <https://culturizando.com/la-tenebrosa-historia-de-la-casa/>

Anonimo.(Desconocido). “Prim´s Algorithm”. Tomado de: <https://www.programiz.com/dsa/prim-algorithm>

Anonimo.(Desconocido). “Prim´s Minimum Spanning Tree(MST) | Greedy Algo-5”. Tomado de: <https://www.geeksforgeeks.org/prims-minimum-spanning-tree-mst-greedy-algo-5/>

Anónimo. (Desconocido). “Sarah´s Story”. Tomado de:<https://winchestermysteryhouse.com/sarahs-story/>

Anónimo. (Desconocido). “Teoría de Grafos”. Tomado de:<http://www.ma.uva.es/~antonio/Industriales/Apuntes_05-06/LabM/B_T-Grafos.pdf>

Miller, B. (2014, julio 2) “El tipo abstracto de datos grafo”. Tomado de: <http://interactivepython.org/runestone/static/pythoned/Graphs/ElTipoAbstractoDeDatosGrafo.html>

Pérez, L. (Desconocido). “¿Visitarías una mansión embrujada de 2 hectáreas? Conoce la historia de la Mansión Winchester”. Tomado de: <https://www.vix.com/es/mundo/190470/visitarias-una-mansion-embrujada-de-2-hectareas-conoce-la-historia-de-la-mansion-winchester>