INTEGRANTES:

Gabriel Restrepo Giraldo

Juan Camilo Gonzalez Torres

Método de la ingeniería:

Contexto del problema:

Se plantea el desarrollo de un software que simula el recorrido que lleva a cabo una empresa cualquiera de envíos a domicilio. El programa permite encontrar el recorrido más corto desde el punto a donde el domiciliario llega (O bien, desde donde este se encuentra), hasta su siguiente destino, modelando el recorrido que debería hacer el domiciliario mediante conexiones dirigidas entre las casas, y tomando como el peso de las aristas (caminos entre domicilios) la distancia que existe entre una casa y otra. Los recorridos que debe hacer el programa se establecen de acuerdo con las entradas que propicia el usuario. Hay dos tipos de entrada de instrucciones: la primera, permite al usuario definir un único domicilio al cual debe llegar; la segunda, permite al usuario definir un conjunto de domicilios cuyo tiempo de trayecto en conjunto no exceda las 24 horas. En ambos casos el sistema muestra al usuario el tiempo de recorrido y los domicilios por los cuales pasó el domiciliario.

Definición del problema:

Se debe determinar el camino más corto entre dos puntos mediante los algoritmos de recorrido dijkstra y warshall haciendo uso de matrices y listas de adyacenci**a.**

Etapa 1: Requerimientos funcionales y no funcionales

Requerimientos funcionales:

R1: El programa permite generar automáticamente un número de casas determinado por el usuario (Admite más de 50 casas), con las conexiones dirigidas (aleatorias) entre ellas, incluyendo el peso de las mismas. El punto desde donde el domiciliario parte también se establece aleatoriamente (Este punto es denominado Sede de la empresa de envíos).

R2: El programa debe permitir al usuario visualizar el mapa de los domicilios (Mostrando los nodos representativos de cada domicilio y una representación clara de la arista). Cada domicilio cuenta con un código único asignado que funge como referencia, el cual es mostrado al usuario junto con el mapa..

R3: El programa permite al usuario hacer un envío único. El cual empieza desde un origen determinado por el usuario, y finaliza en el domicilio indicado por el usuario. Para seleccionar el destino y el origen, el usuario escoge ambos en una lista de selección múltiple.

R4: El programa le muestra al usuario la distancia total recorrida (se debe interpretar como metros). El programa debe completar el recorrido con el camino más corto, y al final de la simulación se mostrará una pantalla emergente con la información del destino.

R5: El programa permite al usuario simular la entrega hacia todos los domicilios generados. Al final de la simulación el usuario verá la distancia total del grafo al recorrerlo por completo. Este recorrido del grafo corresponde al camino más eficiente para entregar a todos los domicilios.

Etapa 2: Investigación de estructuras para modelar el problema.

- **Marco teórico:**

- **Recorrido BFS:** También conocido como recorrido en amplitud, es una forma sistemática de visitar los vértices. Este enfoque se denomina en amplitud porque desde cada vértice v que se visita se busca en forma tan amplia como sea posible, visitando todos los vértices adyacentes a v. Es una generalización del recorrido por niveles de un árbol. Este método
- **Recorrido DFS:** La estrategia de este metodo consiste en partir de un vértice determinado Y y a partir de allí, cuando se visita un nuevo vértice, explorar cada camino que salga de él. Hasta que no se haya finalizado de explorar uno de los caminos no se comienza con el siguiente. Un camino deja de explorarse cuando se llega a un vértice ya visitado.
- **Dijkstra:** También llamado algoritmo de caminos mínimos, es un algoritmo para la determinación del camino más corto dado un vértice origen al resto de vértices en un grafo con pesos en cada arista. Su nombre se refiere a Edsger Dijkstra, quien lo describió por primera vez en 1959.
- **Floyd Warshall:** En informática, el algoritmo de Floyd-Warshall es un algoritmo de análisis sobre grafos para encontrar el camino mínimo en grafos dirigidos ponderados. El algoritmo encuentra el camino entre todos los pares de vértices en una única ejecución. Este algoritmo puede ser más eficiente que el algoritmo de Dijkstra en algunos casos.

Fuentes:

- *BFS - Recorrido en amplitud*. (s. f.). Recorridos sobre grafos. Recuperado 22 de mayo de 2022, de http://163.10.22.82/OAS/recorrido_grafos/bfs__recorrido_en_amplitud.html
- *DFS - Recorrido en profundidad*. (s. f.). Recorridos sobre grafos. Recuperado 22 de mayo de 2022, de http://163.10.22.82/OAS/recorrido_grafos/dfs__recorrido_en_profundidad.html#:~:te

xt=Un%20Recorrido%20en%20profundidad%20(en,recorrido%20preorden%20de%20un%20%C3%A1rbol

- *Algoritmo de Dijkstra*. (s. f.). EcuRed. Recuperado 22 de mayo de 2022, de https://www.ecured.cu/Algoritmo_de_Dijkstra
- *Algoritmo de Floyd-Warshall*. (s. f.). Wikipedia. Recuperado 22 de mayo de 2022, de https://es.wikipedia.org/wiki/Algoritmo_de_Floyd-Warshall

Etapa 3:

Idea 1: Se pretende usar el algoritmo de DFS para saber si el grafo es conexo, de manera que se pueda crear un grafo conexo siempre.

Idea 2: Podemos usar el algoritmo de Floyd Warshall para encontrar el camino con la distancia mínima.

Idea 3: Se pretende usar el algoritmo de Dijkstra para encontrar el camino más corto entre dos nodos.

Idea 4: Se pretende usar el algoritmo BFS para saber si el grafo es conexo o no lo es. Esta información la podemos usar para crear un grafo conexo.

Idea 5: Se pretende usar el algoritmo de Kruskal para encontrar el árbol generador mínimo. Esto para simular el recorrido de todos los nodos.

Idea 6: Se pretende usar el algoritmo de Prim para hallar el árbol de expansión mínima MST. Esto será usado para simular el recorrido de todos los nodos del grafo.

Etapa 4:

Ideas descartadas:

Idea 1: Descartamos el uso de DFS dado que encontramos más sencillo implementar el BFS, además, el DFS es más útil cuando el grafo no es conexo. En nuestro caso el grafo es conexo siempre..

Idea 2: Descartamos el uso del algoritmo de Floyd Warshall porque este es más útil cuando lo que se quiere es conocer la distancia total del recorrido más corto.

Idea 5: Descartamos el uso del algoritmo de Kruskal para encontrar el árbol de expansión mínimo. Usaremos en su lugar el algoritmo de Prim dada la simplicidad y sinergia que tiene con la implementación de grafo utilizada.

Ideas utilizadas:

Idea 3: El algoritmo de Dijkstra, también llamado algoritmo de caminos mínimos, es un algoritmo para la determinación del camino más corto, dado un vértice origen, hacia el resto de los vértices en un grafo que tiene pesos en cada arista (Wikipedia, s.f). La implementación de este algoritmo tiene una complejidad manejable. Para efectos de uso en el proyecto manejamos el recorrido con lista de adyacencia y matriz de adyacencia.

Idea 4: Una búsqueda en profundidad (DFS) es un algoritmo de búsqueda para lo cual recorre los nodos de un grafo. Su funcionamiento consiste en ir expandiendo cada uno de los nodos que va localizando, de forma recurrente (desde el nodo padre hacia el nodo hijo) (Encora, s.f). Este algoritmo es especialmente útil al momento de definir si un grafo es conexo o no lo es. Durante la implementación en el proyecto se hará uso del algoritmo con este mismo fin.

Idea 5: Algoritmo de Prim es un algoritmo perteneciente a la teoría de los grafos para encontrar un árbol recubridor mínimo en un grafo conexo, no dirigido y cuyas aristas están etiquetadas (Ecured, s.f). Este algoritmo va incrementando el tamaño del árbol de expansión mínimo MST. Durante la implementación en el programa se usará el árbol de expansión mínimo generado por el algoritmo de Prim para mostrar el recorrido mínimo de todo el grafo. Esto va de la mano con el requerimiento R5.

Etapa 5:

Algoritmo	Precisión de la solución		Eficiencia		Completitud		Facilidad de implementación		TOTAL
	25%		25%		25%		25%		
Prim	Este algoritmo resuelve de forma satisfactoria el objetivo de encontar el arbol de expansión mínimo	5	Este algoritmo tiene una complejidad que permite encontrar el MST de forma rápida y efectiva	5	Este algoritmo cumple con los requerimientos del programa	5	Su implementación es sencilla, partiendo de la estructura de grafo con la que se cuenta	5	5
DFS	Este algoritmo resuelve de forma satisfactoria el objetivo de recorrer todo el arbol y determinar si el grafo es conexo o no	5	Este algoritmo se ejecuta de forma eficiente y el resultado es efectivo en todos los casos	5	Este algoritmo cumple con los requerimientos del programa	5	La implementacion de este algoritmo es sencilla	5	5
Dijkstra	Este algoritmo encuentra de forma satisfactoria el camino más corto entre 2 nodos	5	Este algoritmo tiene una complejidad que permite encontrar rápidamente la ruta de nodos más corta, lo que hace eficiente al programa	5	Este algoritmo cumple con los requerimientos del programa	5	La implementacion de este algoritmo no es tan sencilla	4.8	4.95

Dado el resultado efectivo de los 3 algoritmos, consideramos el uso de los tres para la implementación de la solución