Luisa Fernanda Castaño Pino A00380290

Kevin Vincent Loachamin Almeida A00382106

Manuel Alejandro Herrera Uribe A00381987

**Tarea Integradora 2**

**Enunciado**

Después del excelente rendimiento académico de usted y sus compañeros, se les eligió por un directivo de la universidad, para realizar la primera versión de un sistema que permita solucionar un problema del que se les informa a continuación, en la universidad se presenta un nuevo lugar que dispondrá de alimentos a domicilios para los estudiantes, directivos, etc, llamado LOTTO, pero las personas encargadas de dichos domicilios requieren saber de la ruta más óptima y rápida para sus entregas de domicilios desde distintos puntos dentro de la universidad.

Dado que este programa será ejecutado por personas encargadas de los domicilios, la institución entiende que al ser un nuevo sistema, seguramente las personas encargadas cometerán errores, por lo que será necesario encargarse de ajustarlo lo suficiente para evitar errores de aquellas personas y que sea usado de la mejor manera posible sin presentar dificultades a los usuarios del programa, por ello dicho programa solicitará una serie de datos para generar la ruta mas rapida.

**Método de la ingeniería:**

**Fase 1: Identificación del problema**

Este cuenta con el problema dado la dificultad de encontrar una ruta o camino en la universidad que sea eficiente para el restaurante LOTTO, con respecto al punto de partida para poder realizar de manera correcta la entrega de los domicilios dentro de la universidad en un menor recorrido, pues tal y como lo menciona el enunciado se espera que saber la ruta más óptima y rápida para las entregas.

Además de esto, se le puede agregar que para la solución del problema hay que tener en cuenta posibles errores humanos de ejecución de la solución mientras sea muy nuevo y mientras las personas encargadas se acostumbran y se familiarizan con el sistema que dará solución a este problema.

**Especificación del problema:**

| Cliente | Restaurante LOTTO |
| --- | --- |
| Usuario | Domiciliarios |
| Requerimientos funcionales | R1. El sistema debe ser capaz de leer cada una de las etiquetas y pesos de las aristas.  R2. El sistema debe poder generar una respuesta de la ruta óptima.  R3. El sistema debe poder informarle al usuario que cometió un error ingresando información. |
| Contexto del problema | Un restaurante dentro de la universidad ICESI, que comenzó a vender a domicilio en dicha universidad, y requiere de un sistema que les facilite y agilice el proceso de domicilios. |
| Requerimientos no funcionales | Adecuada operatividad del sistema, mostrando mensajes dado que se comentan errores. |

**Fase 2: Recopilación de la información necesaria**

De acuerdo con el problema propuesto por el restaurante LOTTO y que nos presentan con los requerimientos establecidos previamente, se puede establecer cierta información que nos puede ser útil para la solución del problema propuesto:

**R1**. El sistema debe ser capaz de leer el punto de partida y de llegada: el restaurante presentará los puntos de llegada y de partida para poder ejecutar la solución de la ruta más óptima, por lo que la solución a este problema debe poder leer esta información ingresada.

**R2**. El sistema debe poder generar una respuesta de la ruta óptima: dado los parámetros ingresados por el usuario como el origen y el destino, el sistema debe poder de acuerdo a esto establecer la opción más rápida que les facilite el proceso de domicilios al restaurante LOTTO.

**R3.** El sistema debe poder informarle al usuario que cometió un error ingresando información: pues el problema registrará y recibirá mucha información que si no se maneja de la mejor manera, se pueden llegar a dar problemas en el uso del sistema solución, por lo que al menos cuando haya alguna información ingresada de la manera incorrecta se le deberá informar la usuario para que lo corrija.

Además de esto, se emplea un mapa de la universidad pues es el lugar en el que se realizarán los domicilios, dado esto, es de gran importancia pues nos ayuda a establecer posibles distancias entre los lugares y también sus conexiones.



**Fase 3: Búsqueda de soluciones creativas:**

La lluvia de ideas es una de las técnicas más populares para resolver el tipo de problemas que deben ser resueltos en grupo, además de estimular activamente la generación de ideas aunque no sean posibles, ya que esto minimiza la posibilidad de un estado emocional negativo entre los participantes y así se contribuye como equipo a encontrar la solución al problema, pues se plantea además una participación activa.

A través de la lluvia de ideas, se generaron las siguientes:

1. Usar el algoritmo de Dijkstra para la determinación del camino más corto usando las aristas como caminos y los pesos cómo Km.
2. Hacer un hashMap con los nombres y las distancias de cada uno y sumar sus distancias.
3. Usar la búsqueda heurística para el grafo.
4. Usar el algoritmo de Bellman-Ford para hallar el camino más corto de un punto a otro en el grafo.
5. Implementar el algoritmo Floyd-Warshall para calcular la mínima distancia entre los pares de vértices del grafo.
6. Preguntarle directamente al usuario por medio del sistema, las opciones por las que puede pasar, incluyendo la distancia entre los diferentes puntos.

**Fase 4: Desarrollo de diseños preliminares:**

Dada la lluvia de ideas, procederemos a elegir una de ellas y a definir la razón por la que las otras opciones no son válidas o no resuelven el problema de forma correcta y eficaz.

**Idea**:

1. Usar el algoritmo de Dijkstra para la determinación del camino más corto usando las aristas como caminos y los pesos cómo Km.

Esta idea es de las opciones más eficientes para poder encontrar la distancia de menor tamaño entre dos vértices dado la existencia de un grafo que cuente con pesos no negativos.

**Idea:**

1. Hacer un hashMap con los nombres y las distancias de cada uno y sumar sus distancias.

En esta opción de idea no se denota un alto nivel de eficiencia, pues a pesar que se le puede asignar determinados valores y nombres a cada una de las opciones item que se cuenta, la búsqueda sería de forma lineal, por lo que esta opción se descarta.

**Idea:**

1. Usar la búsqueda heurística para el grafo.

La idea de búsqueda heurística se descarta ya que solo funciona si el grafo G es de ramificación finita (o bien, la relación .→. es localmente finita), i.e., SUC(x) es finito, para cualquier x∈BUSQ. o si Naturalmente, SOL≠∅.

**Idea:**

1. Usar el algoritmo de Bellman-Ford para hallar el camino más corto de un punto a otro en el grafo.

Esta opción detona pocos niveles de eficiencia, pues en este caso se tendrán que admitir pesos negativos, pero se puede hallar el camino más corto de una manera más rápida pues con otros algoritmos se halla este en un menor tiempo.

**Idea:**

1. Implementar el algoritmo Floyd-Warshall para calcular la mínima distancia entre los pares de vértices del grafo.

Esta opción se descarta de entre los propuestos, pues no se ve reflejado un mínimo de efectividad al hallar el resultado calculando la distancia mínima entre todos los pares de vértices de un grafo ponderado.

**Idea:**

1. Preguntarle directamente al usuario por medio del sistema, las opciones por las que puede pasar, incluyendo la distancia entre los diferentes puntos, sin hacer uso de algoritmo de búsqueda de la distancia mínima.

Esta opción es la menos óptima de las propuestas, pues en primer lugar la solución en este caso depende de que el usuario esté en pleno conocimiento de todos los puntos por los que puede pasar desde su lugar de origen hasta su destino, además de esto debe estar en pleno conocimiento de la distancia entre cada uno de ellos y por último debe contar con disponibilidad de tiempo necesario para digitar cada uno de estos datos, lo cual directamente hace que sea la opción menos óptima porque no haría uso de un algoritmo como tal que ayude a la facilidad de elección y por lo tanto sea descartada inmediatamente.

**Fase 5: Evaluación**

En este punto es necesario establecer un criterio para poder seleccionar la opción o idea correcta que nos ayude a solucionar el problema de la manera más óptima posible, estos criterios se puede establecer de acuerdo a la precisión de solución, eficiencia, que tan completa es como solución y la facilidad de uso de su implementación.

**Criterio A. Precisión de la solución.** ­

[2] Exacta

­[1] Aproximada ­

**Criterio B. Eficiencia.**

[3] Exacta

­[2] Aproximada ­

[1] Nula­

**Criterio C. Completitud.**

[3] Exacta

­[2] Aproximada ­

[1] Nula­

**Criterio D. Facilidad en implementación algorítmica: ­**

[2] Compatible con las operaciones aritméticas básicas de un equipo de cómputo moderno ­

[1] No compatible completamente con las operaciones aritméticas básicas de un equipo de cómputo moderno

|  | Criterio A | Criterio B | Criterio C | Criterio D | Total |
| --- | --- | --- | --- | --- | --- |
| Idea 1 | [2] | [3] | [3] | [2] | 10 |
| Idea 4 | [1] | [2] | [2] | [1] | 6 |
| Idea 5 | [1] | [1] | [1] | [1] | 4 |

De acuerdo con lo anterior, se logra establecer que la solución ideal para el problema propuesto inicialmente, es la idea 1 en el que se hace uso del algoritmo de Dijkstra, pues este es el que nos brinda la manera más eficiente para encontrar la distancia mínima entre dos vértices de un grafo con pesos que no son negativos, como se logra ver de acuerdo a los criterios de evaluación.

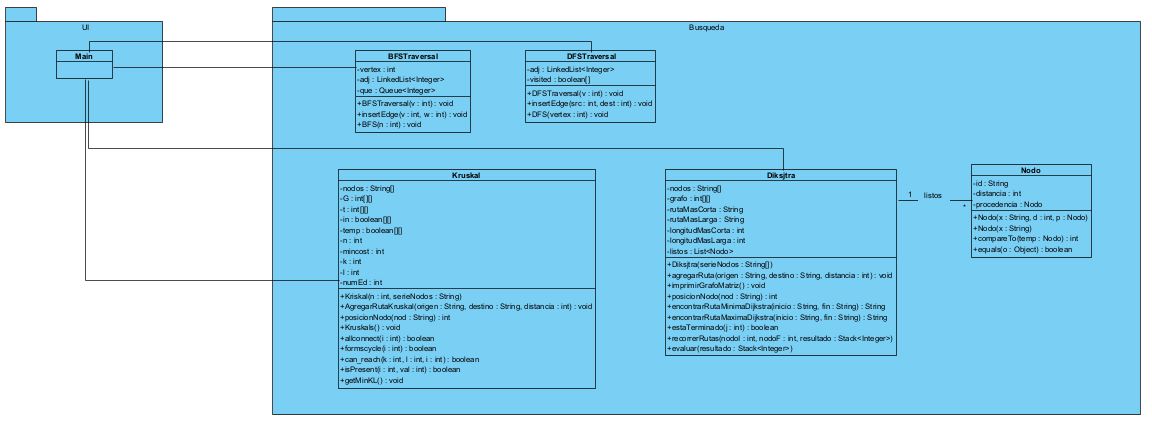
La idea principal del algoritmo es muy sencilla: Iremos construyendo un grafo F, inicialmente formado únicamente por el nodo origen. En cada paso miraremos todos los nodos a los que podamos llegar directamente desde F, es decir aquellos que compartan arista con un nodo de F, y añadiremos el nodo cuya distancia al origen sea mínima. Pararemos el algoritmo en el momento en que el vértice a incluir a F sea el nodo destino.

**Especificación de requerimientos:**

| NOMBRE O IDENTIFICADOR | R1. El sistema debe ser capaz de leer cada una de las etiquetas y pesos de las aristas. | | |
| --- | --- | --- | --- |
| RESUMEN | El sistema debe permitir leer la información registrada | | |
| ENTRADAS | **Nombre de entrada** | **Tipo de dato** | Condición de selección o repetición |
| lugarInicial | String | **--------** |
| lugarFinal | String | **--------** |
|  |  |  |
| ACTIVIDADES GENERALES NECESARIAS PARA OBTENER LOS RESULTADOS | Dada la información registrada, se generará por medio de un método especializado en buscar la ruta óptima para esta situación. | | |
| RESULTADO O POSTCONDICIÓN |  | | |
| SALIDAS | Nombre de salida | Tipo de dato | Condición de selección o repetición |
| mensaje | String |  |

| NOMBRE O IDENTIFICADOR | R2. El sistema debe poder generar una respuesta de la ruta óptima. | | |
| --- | --- | --- | --- |
| RESUMEN | El sistema debe permitir leer la información registrada y de acuerdo a esto generar la ruta óptima para entregar el domicilio. | | |
| ENTRADAS | **Nombre de entrada** | **Tipo de dato** | Condición de selección o repetición |
| lugarInicial | String | **--------** |
| lugarFinal | String | **--------** |
|  |  |  |
| ACTIVIDADES GENERALES NECESARIAS PARA OBTENER LOS RESULTADOS | Dada la información registrada, se generará por medio de un método especializado en buscar la ruta óptima para esta situación. | | |
| RESULTADO O POSTCONDICIÓN | Información al usuario de la ruta más óptima para entregas según los puntos dados | | |
| SALIDAS | Nombre de salida | Tipo de dato | Condición de selección o repetición |
| mensaje | String |  |

| NOMBRE O IDENTIFICADOR | R3. El sistema debe poder informarle al usuario que cometió un error ingresando información. | | |
| --- | --- | --- | --- |
| RESUMEN | Dada la información registrada | | |
| ENTRADAS | **Nombre de entrada** | **Tipo de dato** | Condición de selección o repetición |
| dato | String | **--------** |
| dato | Int | **--------** |
|  |  |  |
| ACTIVIDADES GENERALES NECESARIAS PARA OBTENER LOS RESULTADOS | Dada la información registrada, se generará por medio de un mensaje una alerta de error dado algún error cometido en los datos ingresados. | | |
| RESULTADO O POSTCONDICIÓN | Mensaje de error si lo cometió. | | |
| SALIDAS | Nombre de salida | Tipo de dato | Condición de selección o repetición |
| mensaje | String |  |

****

**Para mejor visualización entrar el archivo vpp que contiene este diagrama.**

| TAD Grafo | | |
| --- | --- | --- |
| Abstract object: | | |
| Invariant: | | |
| Operations: | | |
| Operation | Input | Output |
| Graph |  | String[] |
| diksjtra | String[] | diksjtra |
| kruskal | size, String[] | kruskal |

**TAD estructuras utilizadas**

| TAD Diksjtra | | |
| --- | --- | --- |
| Abstract object: | | |
| Invariant: | | |
| Operations: | | |
| Operation | Input | Output |
| Diksjtra | String[] | Diksjtra |
| agregarRuta | String,String,int |  |
| imprimirGrafoMatriz |  | String |
| posicionNodo | String | int |
| encontrarRutaMinimaDiksjtra | String, String | String |
| encontrarRutaMaximaDiksjtra | String, String | String |
| estaTerminado | int | boolean |
| recorrerRutas | int, int, Stack<Integer> |  |
| evaluar | Stack<Integer> |  |

| TAD Kruskal | | |
| --- | --- | --- |
| Abstract object: | | |
| Invariant: | | |
| Operations: | | |
| Operation | Input | Output |
| Kruskal | String[] | Kruskal |
| agregarRutaKruskal | String,String,int |  |
| posicionNodo | String | int |
| Kruskals |  |  |
| allconnect | int | boolean |
| formscycle | int | boolean |
| canReach | int, int, int | boolean |
| isPresent | int, int | boolean |
| getMinKL |  |  |