Tarea Integradora 3

Santiago Trochez Velasco - A00369326
Gianni Stiven Benavides - A00362358
Luis Miguel Ossa Arias - A00369982
Johnatan Garzón Montesdeoca

Algoritmos y Estructura de Datos, Ingeniería de Sistemas
Santiago de Cali, Colombia
2021

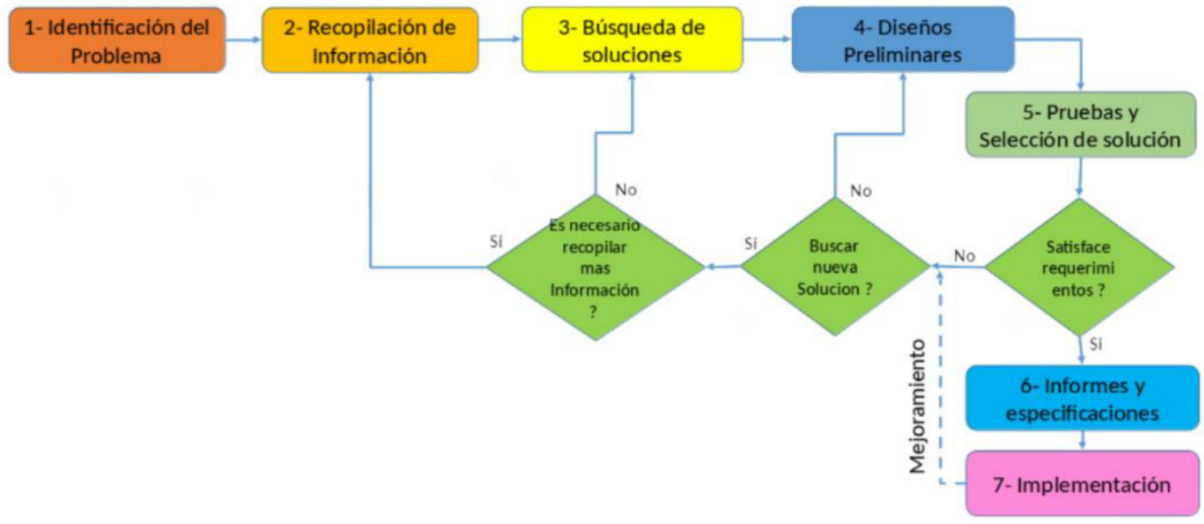
Etapa 1. Contexto Problemático.

Un grupo de personas emprendedoras ha decidido crear una aerolínea llamada Butterfly, la cual prestará servicios de vuelo por todo el mundo. Para empezar, se han instalado en diez países, los cuales son, España, Japón, Estados Unidos, Rusia, Colombia, Nigeria, Dubai, Portugal, Australia y Madagascar, para que se realicen vuelos entre estos, los cuales pueden ser directos o en escalas. Para que esta aerolínea pueda destacar entre las demás respecto a precios, se requiere un software que permita identificar los menores costos posibles entre cada viaje para los diez países en donde prestan su servicio. Se busca que el software esté en la capacidad de establecer y modificar los costos de cada viaje, como también definir los posibles recorridos que se pueden presentar en los diez países. Estos datos serán guardados en la memoria secundaria del sistema.

Definición del Problema:
La aerolínea Butterfly requiere de un software que le permita identificar el menor costo posible en un viaje de un país a otro, teniendo en cuenta que el viaje puede ser directo o con escalas.

Desarrollo de la solución
Para sobrellevar el problema anteriormente mencionado se seleccionó el Método de la Ingeniería para desarrollar una solución, mediante un enfoque sistemático y acorde a la situación problemática planteada.

Para esto, se implementó la descripción del Método de la Ingeniería del libro “Introduction to Engineering” de Paul Wright, se define el siguiente diagrama de flujo, para seguir los pasos en el desarrollo de la solución.



Especificación de Requerimientos.
El software debe estar en la capacidad de:

Req. 1. Gestionar los viajes que puede realizar la aerolínea.
Req. 1.1. Agregar un viaje que contiene un costo, un país de origen y uno de destino.
Req. 1.2. Eliminar un viaje a partir de su país de origen y de destino.
Req. 1.3 Modificar un viaje, en donde se puede cambiar su país de destino y su costo; no su país de origen.

Req 2. Representar los viajes que puede realizar la aerolínea a partir de líneas con colores, los cuales representan si un viaje puede ser realizado en un solo sentido (solo ida) o en doble sentido (ida y vuelta).

Req 3. Mostrar los 10 países con su información correspondiente (nombre, año de fundación y población) a los que la aerolínea Butterfly puede realizar viajes.

Req 4. Buscar los países a los cuales un país determinado puede viajar.

Req 5. Identificar el costo mínimo de un vuelo entre un país y otro, teniendo en cuenta que el vuelo puede ser directo o con escalas.

Req 6. Guardar la información de cada uno de los vuelos en memoria secundaria, de forma persistente.

Etapa 2. Marco teórico.
Con el objetivo de afrontar el problema con la mayor claridad posible, se buscarán las definiciones relacionadas con el problema previamente definido.

Definiciones:

Fuentes:
https://www.espanol.skyscanner.com/
https://es.wikipedia.org/wiki/Wikipedia
https://latam.kaspersky.com/blog/
https://concepto.de/
https://techlib.net/index.html

Aerolínea: es una empresa que se dedica al transporte de pasajeros o carga.

Vuelo directo: un vuelo directo es aquel que te llevará desde el aeropuerto más cercano hasta el aeropuerto de tu destino final, sin parar en ningún otro aeropuerto en el trayecto.

Vuelo a escala: un vuelo con escalas te llevará desde el aeropuerto más cercano a un punto intermedio o de conexión donde tendrás que descender del primer avión, pasar por los filtros de seguridad o migración necesarios y abordar otro avión que te llevará ya sea a tu destino final o a otro(s) aeropuerto(s) antes de hacerlo.

Grafo: es un tipo abstracto de datos (TAD), que consiste en un conjunto de nodos (también llamados vértices) y un conjunto de arcos (aristas) que establecen relaciones entre los nodos. El concepto de grafo TAD desciende directamente del concepto matemático de grafo.

Recorrido: una aeronave no vuela en línea recta; se mueve de un lugar a otro. En distancias más grandes, dicha ruta en forma poligonal casi se ajusta a la línea directa. La razón es simple y lógica: cuanto más corta es la distancia, menor es la cantidad de combustible consumido.

Costo: es el desembolso económico que se realiza para la producción de algún bien o la oferta de algún servicio.

Memoria secundaria: memoria secundaria se refiere a dispositivos de almacenamiento, tales como unidades de disco duro y a discos de estado sólido. También puede referirse a medios de almacenamiento extraíbles, como USB, unidades flash, CDs y DVDs.

Etapa 3. Búsqueda de soluciones creativas.

Se realiza la búsqueda de diferentes soluciones que puedan resolver el problema de la aerolínea Butterfly, el cual es identificar el menor costo posible en un viaje de un país a otro, teniendo en cuenta que el viaje puede ser directo o con escalas. Para plantear las ideas que puedan solucionar el problema anteriormente mencionado, se optó por realizar una **lluvia de ideas**, para así poder generar una cantidad considerable de opciones, y en donde se plantearon las siguientes:

Alternativa 1. *Búsqueda del costo mínimo de un determinado viaje por medio del algoritmo Floyd Warshall usando grafos.*
Este algoritmo es uno de los mejores para realizar la búsqueda de valores mínimos, ya que para este se implementa una matriz en donde la intersección de cada uno de los vértices contiene el costo mínimo de un viaje entre dos países. Este algoritmo tiene una complejidad

temporal de $\Theta(V^3)$ donde V es el número de vértices que contiene el grafo. Además, al utilizar grafos, se pueden representar los viajes posibles entre cada par de países con la implementación de grafos dirigidos, donde se identifica a el país de origen y de destino del viaje.

Alternativa 2. *Búsqueda del costo mínimo de un determinado viaje por medio del algoritmo Dijkstra usando grafos.*

Este algoritmo es muy similar al de Floyd Warshall, ya que también busca e identifica el costo o camino mínimo entre dos vértices, con la diferencia de que el algoritmo Dijsktra no usa matrices para encontrar el camino mínimo. Este algoritmo lo hace por medio de colas ordenadas, de menor a mayor costo de aristas y un arreglo de distancias, donde cada vértice tiene su propio valor de distancia mínima. Este algoritmo tiene una complejidad temporal de $O(V^2)$ o O(E log V), donde E es el número de aristas y V el número de vértices. Además, por medio de este método se usan grafos dirigidos, en donde cada relación representa el país de origen y el país de destino de cada viaje.

Alternativa 3. *Búsqueda del costo mínimo de un determinado viaje por medio de un recorrido en Inorden usando árboles binarios.*

Este algoritmo permitirá crear un árbol binario por cada viaje, cada uno contará con el costo que ese viaje tendrá. para que de esta forma se pueda evaluar el menor costo posible de un viaje, realizando una búsqueda por cada uno de los nodos izquierdos de este árbol. Este algoritmo tiene una complejidad temporal de O(n).
Los costos que tendrá el árbol, serán de costos de vuelos directos y vuelos con escalas.

Alternativa 4. *Búsqueda del costo mínimo de un determinado viaje por medio de un arraylist de costos.*

Esta solución es muy sencilla de aplicar, ya que se implementará un arraylist que contenga los costos de todos los viajes. Cada elemento del arraylist tendrá atributos como país de origen, país de destino y costo del viaje. Cuando se tengan todos los costos se empezará a buscar por medio del arraylist un elemento que contenga los mismos países, tanto de origen como de destino, sin importar si estos viajes serán directos o con escala, para luego obtener el costo de dicho elemento y así compararlo con los costos de los demás elementos que contengan el mismo atributo de país de destino y de origen. Al finalizar el arraylist se tendrá un elemento con un país de origen y de destino esperado, así como con el menor costo respecto a los demás elementos del arraylist con el mismo país de origen y de destino.

Alternativa 5. *Búsqueda del costo mínimo de un determinado viaje por medio de una Queue de costos.*

Esta solución busca aplicar una cola de costos para cada uno de los viajes posibles que tiene un país, en donde no importa si los viajes son con escala o no. Después de tener la cola de

cada viaje implementada, se implementará una variable global que almacenará el menor costo. Para empezar, se va a asignar el primer elemento de la cola a la variable, para que de esta forma luego se pueda comparar dicha variable con los siguientes costos almacenados en la cola. Al final, la variable tendrá el costo mínimo de un viaje, que va desde un país de origen hasta un país de destino y la cola estará vacía.

Alternativa 6. *Búsqueda del costo mínimo de un determinado viaje por medio de una Stack de costos.*

En esta solución se propone el uso de la estructura de datos llamada Stack. Cada país tendrá una stack por cada viaje o país de destino en donde se almacenarán todos los costos. Al momento de comparar los costos, se hará el uso de una variable global que tendrá el costo mínimo del viaje, pero que al iniciar, tomará el primer valor proveniente de la stack. Al obtener el primer valor, la stack empezará a sacar cada uno de sus elementos para que se compare con la variable global y se van a asignar a esta solo si su costo es menor. Al finalizar la búsqueda del mínimo costo del viaje, la stack estará vacía y tendrá el costo mínimo del viaje.

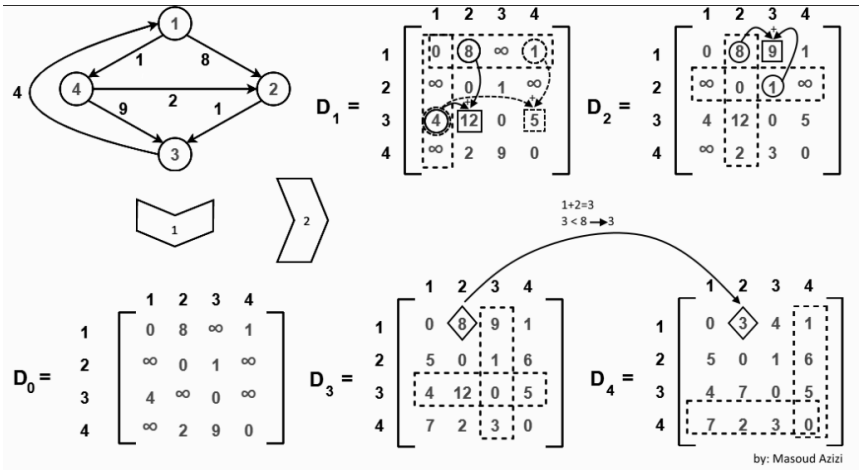
Etapa 4. Transición de las Ideas a los Diseños Preliminares.

La alternativa 4, 5 y 6 son descartadas, porque aunque encuentren una solución al problema, estas cuentan con una complejidad temporal y espacial muy elevada, ya que necesitan de procesos poco eficientes, y además, para poder resolver con la problemática necesitan de un mayor espacio dentro de la memoria, lo que las hace nada eficientes, es por todo esto que quedan eliminadas dentro de las opciones viables.

Luego de descartar las opciones no factibles para desarrollar el problema, se realizará una amplia revisión de las alternativas restantes, para así definir una de ellas como la solución pertinente al problema.

Alternativa 1. *Búsqueda del costo mínimo de un determinado viaje por medio del algoritmo Floyd Warshall usando grafos.*

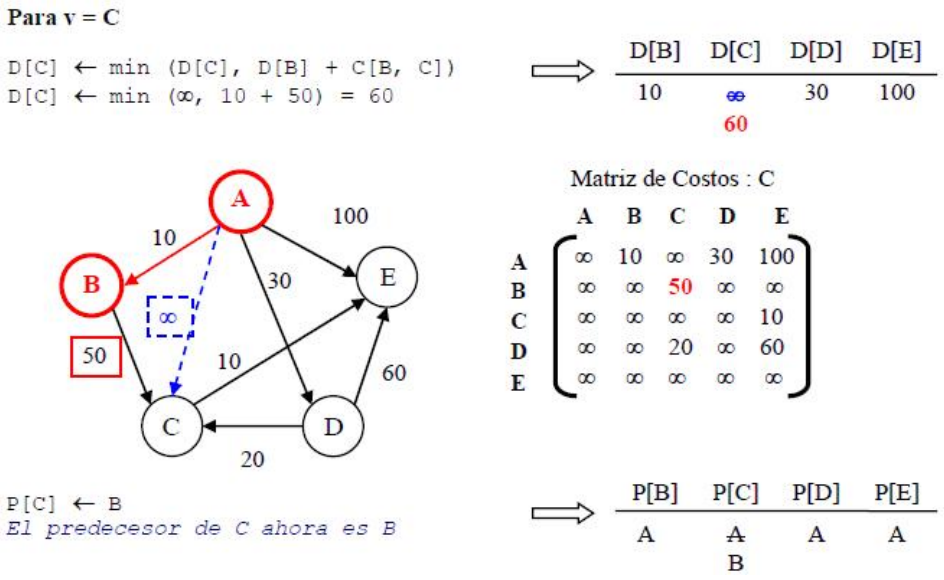
- La búsqueda del valor mínimo toma un tiempo $O(V^3)$ lo que hace a este algoritmo muy ineficiente.
- Toma un considerable espacio de memoria al implementar una matriz para los costos y los recorridos.
- Esta solución informa el recorrido a seguir para el viaje con un costo mínimo.



Algoritmo Floyd Warshall

Alternativa 2. *Búsqueda del costo mínimo de un determinado viaje por medio del algoritmo Dijkstra usando grafos.*

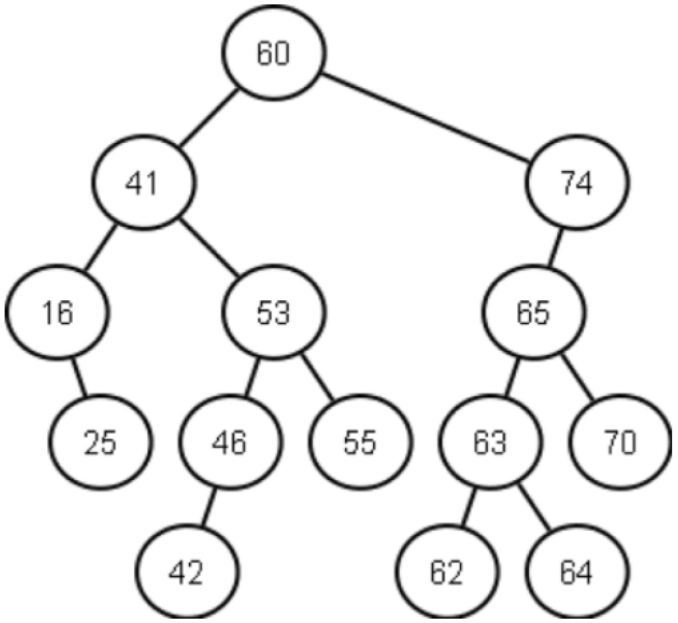
- Este algoritmo puede tomar una complejidad temporal de O(V^2) o O(E log V). La primera complejidad temporal se genera con una implementación normal del algoritmo. Con la segunda complejidad temporal, la cual es más eficiente, se requiere la implementación de un binary heap, lo que aumenta el nivel de complejidad de la implementación del algoritmo.
- Su implementación requiere el uso de colas y arreglos con un tamaño V, lo cual aumenta la complejidad espacial del sistema.
- Esta solución informa el recorrido a seguir para el viaje con un costo mínimo.



Algoritmo Dijkstra con matriz implementada (no es necesario).

Alternativa 3. *Búsqueda del costo mínimo de un determinado viaje por medio de un recorrido en Inorden usando árboles binarios.*

- Esta solución requiere de un tiempo O(log n), donde n es el número de hojas que contiene el árbol.
- Al implementar un árbol por cada viaje, en el peor de los casos se tendrían que implementar más de 90 árboles binarios, en donde cada uno se organizará por costos.
- La complejidad espacial de esta solución es muy alta, ya que se necesita una gran cantidad de árboles binarios para poder solucionar el problema.
- Esta solución no expresa el recorrido a seguir para obtener el costo mínimo en un viaje, solo expresa el costo mínimo.



Árbol binario

Etapa 5. Evaluación y Selección de la Mejor Solución.

Criterios:

Se van a definir criterios que permitirán evaluar con facilidad las alternativas propuestas para el desarrollo del problema previamente definido. Esto ayudará a definir las características que tienen prioridad al momento de desarrollar una solución para el problema. Cada característica será evaluada por medio de un valor, en donde el mayor valor significa que es muy importante para la solución y el último significa que es el que menos se espera para el desarrollo de la solución.

Criterio A. Eficiencia. Se busca una solución que tenga una complejidad temporal reducida. La complejidad temporal puede ser:

- [5] Constante.

- [4] Logarítmica.
- [3] Líneal
- [2] Cuadrática
- [1] Cúbica

Criterio B. Claridad. Se busca que la solución sea lo más intuitiva y clara posible para el usuario. La solución puede ser:

- [3] Intuitiva y clara.
- [2] Poco clara
- [1] Ambigua.

Criterio C. Optimización. Se busca que la solución sea lo más óptima posible, ocupando el menor espacio posible en memoria. La solución puede ser:

- [3] Óptima, ocupa el menor espacio posible en la memoria.
- [2] Regular, ocupa un espacio prudente en memoria, sin ser óptimo pero tampoco pésimo.
- [1] Pésima, ocupa mucho espacio en la memoria.

Evaluación:

Al evaluar las alternativas sobre los tres criterios previamente definidos, se obtuvo la siguiente tabla:

	Criterio A	Criterio B	Criterio C	Total
Alternativa 1	1	3	2	6
Alternativa 2	3	3	3	9
Alternativa 3	4	1	1	6

Selección:

A partir de los resultados obtenidos en la evaluación, podemos concluir que la alternativa 2 (Búsqueda del costo mínimo de un determinado viaje por medio del algoritmo Dijkstra usando grafos) es la mejor solución, ya que obtuvo el mayor puntaje cuando fue evaluado con los criterios previamente definidos.

TAD DE GRAFOS

TAD Grafo Dirigido			
$G = (V, E)\ , V = \{v_1, v_2, v_3, \dots v_n\} \wedge E = \left\{e_1\{v_i, v_f, w_1\}, e_2\{v_i, v_f, w_2\}, e_3\{v_i, v_f, w_3\}, \dots\ e_n\{v_i, v_f, w_n\}\right\}$			
$\{\text{inv} : \forall e \mid e \in E \wedge e\{v_i, v_f\} \in V \wedge e\{w\} \in \mathbb{Z}\}$			
Operaciones Primitivas			
CrearGrafo:			Constructora
AgregarVertice:	Valor	→ Booleano	Modificadora
AgregarAdyacencia:	Vértice x Vértice x Valor	→ Booleano	Modificadora
EliminarVertice:	Valor	→ Booleano	Modificadora
EliminarAdyacencia:	Vértice x Vértice	→ Booleano	Modificadora
ObtenerVertice:	Valor	→ Vértice	Analizadora
ObtenerArista:	Vértice x Vértice	→ Entero	Analizadora

CrearGrafo( )

"Crea un grafo vacío sin vértices ni aristas"

{*pre:* TRUE}
{*post:* Grafo = nulo}

AgregarVertice(e**)**

"Agrega un nuevo vértice en el grafo"

{*pre:* Grafo ≠ vacío, el grafo debe de estar inicializado}
{*post:* Grafo = {⟨e⟩, ⟨⟩} ∨ Grafo = {⟨e_1, e⟩, ⟨⟩}}

AgregarAdyacencia($\boldsymbol{v_1, v_2, w}$)

"Agrega una relación o adyacencia entre dos vértices dentro del grafo con su respectivo peso"

{*pre:* Grafo = {⟨v_1, v_2⟩}}
{*post:* Grafo = {⟨v_1, v_2⟩, ⟨e{v_1, v_2, w}⟩}}

EliminarVertice($\boldsymbol{v}$)

"Elimina un vértice dentro del grafo, destruyendo las aristas en las que esté relacionado dicho vértice"

{*pre:* Grafo = {⟨$\boldsymbol{v, v_2}$⟩, ⟨⟩} ∨ Grafo = {⟨$\boldsymbol{v, v_2}$⟩, ⟨e{v, v_2, w}⟩}}
{*post:* Grafo = {⟨v_2⟩, ⟨⟩}

EliminarAdyacencia($\boldsymbol{v_1}, \boldsymbol{v_2}$)

"Elimina la arista que se encuentra entre dos vértices"

{*pre:* Grafo = {⟨$\boldsymbol{v_1}, \boldsymbol{v_2}$⟩, ⟨e{$v_1, v_2, w$}⟩}
{*post:* Grafo = {⟨v_1, v_2⟩, ⟨⟩}

ObtenerVertice(s)

"Obtiene un vértice dentro del grafo"

{*pre:* Grafo = {⟨$\boldsymbol{v_1}, \boldsymbol{v_2}$⟩, ⟨e{$v_1, v_2, w$}⟩}
{*post:* si $v_1 = s$, entonces v_1, de lo contrario, nulo}

ObtenerArista($\boldsymbol{v_1}, \boldsymbol{v_2}$)

"Obtiene la adyacencia que hay entre dos vértices en el grafo"

{pre: Grafo = {⟨$\boldsymbol{v_1}, \boldsymbol{v_2}$⟩, ⟨e{$v_1, v_2, w$}}
{post: e{v_1, v_2, w}}