MÉTODO DE LA INGENIERÍA

INTEGRANTES:

Gabriel Suárez - A00368589 SIS
Alejandro Varela - A00369019 TEL & SIS
Luis Alfonso Murcia Hernández - A00369008 TEL

PROFESOR:
Aníbal Sosa Aguirre

ALGORITMOS Y ESTRUCTURAS DE DATOS

UNIVERSIDAD ICESI
2021-2

Contexto Problemático: Durante el semestre hemos abordado distintos temas orientados a los grafos, de los cuales se nos pidió realizar un proyecto final basado en ellos. Por lo que se pide un programa que implemente distintos algoritmos y resuelva un problema específico usando todo lo visto en las últimas semanas del semestre.

Desarrollo de la solución: Decidimos apoyarnos en el método de la ingeniería para resolver esta problemática. Al buscar en varias fuentes, fue posible darse cuenta de que se necesitan 7 pasos para una solución eficiente del problema.

- **Fase 1: Identificación del Problema**: Aunque no lo parezca, es la fase más importante del método de la ingeniería, se debe delimitar un buen problema para saber qué es lo que piden y con base en esto, comenzar a desarrollar soluciones creativas, efectivas y, sobre todo, innovadoras.
- **Fase 2: Recopilación de la Información Necesaria:** Se necesita saber la cantidad de tiendas que ofrecen el mismo servicio, la viabilidad de la incorporación de la tienda en el mercado y la ubicación.
- **Fase 3: Búsqueda de Soluciones Creativas:** Permite hacer una lluvia de ideas
- **Fase 4: Transición de la Formulación de Ideas a los Diseños Preliminares:** Colabora en el descarte de ideas en el paso anterior y profundización de las más viables.
- **Fase 5: Evaluación y Selección de la Mejor Solución:** Luego de un exhaustivo análisis se escoge la verdadera solución.
- **Fase 6: Preparación de Informes y Especificaciones:** Se documenta los aspectos más importantes del proyecto.
- **Fase 7: Implementación del Diseño:** Se procede a la etapa experimental para llevar a cabo la idea de los 6 pasos anteriores.

Con el fin de ser precisos en el proceso del desarrollo de este método, se explicará de manera clara cada paso:

Fase 1: Identificación del Problema

Ante el inminente cierre del semestre 2021 -2, el profesor **Anibal Sosa** quiso probar a sus estudiantes de **Algoritmos y Estructuras de datos**, con un proyecto basado en la teoría y práctica de los **grafos**, que como bien se sabe, es una estructura de datos que permite la representación simbólica de los elementos constituidos de un sistema o conjunto, mediante esquemas gráficos

Por esta razón, propuso una serie de **condiciones** para limitar a los estudiantes y guiarlos por el mejor camino posible para encontrar una solución.

- **Desarrollar** 2 versiones de Grafo (su solución debe funcionar sin problema con las dos versiones, es decir, el programa debe admitir el cambio de la implementación utilizada en cualquier momento y funcionar bien indistintamente de la que se esté usando). Cada grafo debe ser desarrollado desde el TAD hasta las pruebas unitarias automáticas.
- **Hacer** y documentar cada una de las fases del método de la ingeniería para la solución del problema planteado.
- **Documentar** apropiadamente las fases de análisis y diseño con el documento de especificación de requerimientos, el diseño del TAD, diagramas de clase y objetos, y el diseño de los casos de pruebas de las pruebas unitarias automáticas.
- **Manejar** una interfaz gráfica de usuario que permita utilizar las funcionalidades que respondan a los requerimientos del problema.
- **Implementar** todos los algoritmos de grafos vistos en clase así no sean utilizados en su proyecto para la solución del problema.

Fase 2: Recopilación de la información necesaria

Debido a que aún no se ha definido estrictamente qué se va a trabajar, es necesario conocer los conceptos, herramientas y aplicaciones de la teoría que se va a aplicar sobre el proyecto, por esa razón se ha hecho un glosario de todo lo indispensable para hacer este proyecto más ameno.

GitHub: La cual es una plataforma de desarrollo colaborativo de software para alojar proyectos utilizando el sistema de control de versiones Git. aloja un repositorio de código y te brinda herramientas muy útiles para el trabajo en equipo, dentro de un proyecto. Además de eso, puedes contribuir a mejorar el software de los demás. Una herramienta muy importante para trabajar con todos las personas que apoyarán en la creación del proyecto. Además de eso, nos proporciona ciertas características útiles, como:

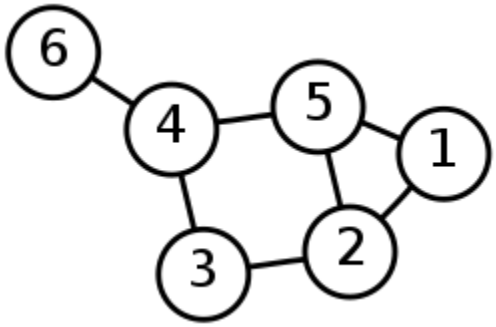
- Una wiki para el mantenimiento de las distintas versiones de las páginas.
- Un sistema de seguimiento de problemas que permiten a los miembros de tu equipo detallar un problema con tu software o una sugerencia que deseen hacer.
- Una herramienta de revisión de código, donde se pueden añadir anotaciones en cualquier punto de un fichero y debatir sobre determinados cambios realizados en un commit específico.
- Un visor de ramas donde se pueden comparar los progresos realizados en las distintas ramas de nuestro repositorio.

Dentro de los conceptos que se pueden encontrar en la fase de implementación, tenemos algunos como:

TAD: Un Tipo de dato abstracto (en adelante TDA) es un conjunto de datos u objetos al cual se le asocian operaciones. El TDA provee de una interfaz con la cual es posible realizar las operaciones permitidas, abstrayéndose de la manera en cómo estén implementadas dichas operaciones. Esto quiere decir que un mismo TDA puede ser implementado utilizando distintas estructuras de datos y proveer la misma funcionalidad.

Grafo: Un grafo es un conjunto de objetos llamados vértices o nodos unidos por enlaces llamados aristas o arcos, que permiten representar relaciones binarias entre elementos de un conjunto. Típicamente, un grafo se representa gráficamente como un conjunto de puntos (vértices o nodos) unidos por líneas (aristas o arcos).

Desde un punto de vista práctico, los grafos permiten estudiar las interrelaciones entre unidades que interactúan unas con otras. Por ejemplo, una red de computadoras puede representarse y estudiarse mediante un grafo, en el cual los vértices representan terminales y las aristas representan conexiones (las cuales, a su vez, pueden ser cables o conexiones inalámbricas).



Algoritmos de recorrido:

DFS: 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). Cuando ya no quedan más nodos que visitar en dicho camino, regresa al nodo predecesor, de modo que repite el mismo proceso con cada uno de los vecinos del nodo. Cabe resaltar que si se encuentra el nodo antes de recorrer todos los nodos, concluye la búsqueda.

BFS: Una búsqueda en anchura (BFS) es un algoritmo de búsqueda para lo cual recorre los nodos de un grafo, comenzando en la raíz (eligiendo algún nodo como elemento raíz en el caso de un grafo), para luego explorar todos los vecinos de este nodo. A continuación, para cada uno de los vecinos se exploran sus respectivos vecinos adyacentes, y así hasta que se recorra todo el grafo. Cabe resaltar que si se encuentra el nodo antes de recorrer todos los nodos, concluye la búsqueda.

Caminos de peso mínimo:

Algoritmo de Dijkstra: 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. La idea subyacente en este algoritmo consiste en ir explorando todos los caminos más cortos que parten del vértice origen y que llevan a todos los demás vértices; cuando se obtiene el camino más corto desde el vértice origen hasta el resto de los vértices que componen el grafo, el algoritmo se detiene. Se trata de una especialización de la búsqueda de costo uniforme y, como tal, no funciona en grafos con aristas de coste negativo (al elegir siempre el nodo con distancia menor, pueden quedar excluidos de la búsqueda nodos que en próximas iteraciones bajarían el costo general del camino al pasar por una arista con costo negativo).

Algoritmo de Floyd-Warshall:

El algoritmo de Warshall es un ejemplo de algoritmo booleano. A partir de una tabla inicial compuesta de 0`s (no hay correspondencia inicial en el grafo) y 1`s (hay una correspondencia, llamase “flecha”, entre nodos), obtiene una nueva matriz denominada “Matriz de Clausura Transitiva” en la que se muestran todas las posibles uniones entre nodos, directa o indirectamente. Es decir, si de “A” a “B” no hay una “flecha”, es posible que si haya de “A” a “C” y luego de “C” a “B”. Luego, este resultado se verá volcado en la matriz final.

El algoritmo de Floyd es muy similar, pero trabaja con grafos ponderados. Es decir, el valor de la “flecha” que representamos en la matriz puede ser cualquier número real o infinito. Infinito marca que no existe unión entre los nodos. Esta vez, el resultado será una matriz donde estarán representadas las distancias mínimas entre nodos, seleccionando los caminos más convenientes según su ponderación (“peso”). Por ejemplo, si de “A” a “B” hay 36 (km), pero de “A” a “C” hay 2(km) y de “C” a “B” hay 10 (km), el algoritmo nos devolverá finalmente que de “A” a “B” hay 12 (km).

Árbol de Recubrimiento Mínimo

Algoritmo Prim: Es un algoritmo perteneciente a la teoría de los grafos para encontrar un árbol recubierto mínimo en un grafo conexo, no dirigido y cuyas aristas están etiquetadas. En otras palabras, el algoritmo encuentra un subconjunto de aristas que forman un árbol con todos los vértices, donde el peso total de todas las aristas en el árbol es el mínimo posible. Si el grafo no es conexo, entonces el algoritmo encontrará el árbol recubridor mínimo para uno de los componentes conexos que forman dicho grafo no conexo.

Algoritmo Kruskal: Es un algoritmo de la teoría de grafos para encontrar un árbol recubierto mínimo en un grafo conexo y ponderado. Es decir, busca un subconjunto de aristas que, formando un árbol, incluyen todos los vértices y donde el valor de la suma de todas las aristas del árbol es el mínimo. Si el grafo no es conexo, entonces busca un bosque expandido mínimo (un árbol expandido mínimo para cada componente conexa)

Fase 3: Búsqueda de soluciones creativas

Debido a que es necesario tener varias propuestas de todos los integrantes para poder analizar cuál es la solución que se va a llevar a cabo, decidimos hacer una **lluvia de ideas** que no fueran tan elaboradas y pensadas en tan solo **5 minutos.**

Alejandro Varela:

Idea 1: Una aplicación que encontrara la mejor ruta para la recolección de basuras en una ciudad no tan grande.

Idea 2: Un juego tipo laberinto donde el jugador necesita atravesar varias puertas, con cada puerta una temática diferente, con una cantidad de intentos y con el objetivo de encontrar el final.

Gabriel Suárez:

Idea 1: Un Snake and Ladder modelado con grafos, donde si pisas una serpiente bajas, y si tocas una escalera, subes.

Idea 2: Un juego similar al buscaminas, pero modelado con grafos donde el jugador debe completar el cuadro sin hacer clic sobre ninguna mina.

Luis Murcia:

Idea 1: Un laberinto sencillo donde la persona pudiera ver transversalmente el juego para buscar la salida.

Idea 2: Una aplicación que ayudara a buscar el mejor camino para un camión de correspondencia.

Fase 4: Transición de la formulación de ideas a los diseños preliminares

Necesitamos encontrar la mejor solución para resolver el problema que estamos desarrollando, por esta razón es necesario revisar cada una de las propuestas de todos los integrantes que aportaron su granito de arena. Empezamos analizando cada una de las soluciones que se proponían, y en la mayoría se podía evidenciar como se encontraban búsqueda de caminos eficientes o juegos que se modelaran de una manera sencilla utilizando grafos.

Debido a la complejidad tan amplia de una aplicación enfocada a la obtención de la mejor ruta en una ciudad y debido a la cantidad tan grande de calles y carreteras que puede haber, el sistema de recolección de basura y el de correspondencia, **se descartan instantáneamente**. También tenemos que a pesar de que el famoso juego de las buscaminas puede ser modelado con grafos, no es muy claro cómo es posible abracar todos los algoritmos necesarios para hacer el juego de una manera completa y eficiente, por lo que **no es tenida en cuenta** para recurrir a la fase número 5.

Dentro de las ideas que **no se descartan**, tenemos la idea número 2 de **Alejandro**, la cual está basada en un sistema que permita calcular la mejor ruta para llegar al final de laberinto y así mismo será la cantidad de intentos que le dará al usuario para acabar el juego. Por otro lado, las ideas de **Gabriel (idea 1)** y **Luis (idea 1)** están basadas en un sistema de juegos tradicionales, sin cambio alguno.

De esa manera nos quedamos con **tres (3)** ideas posibles de reformulación para entrar a la siguiente fase:

Fase 5: Evaluación y selección de la mejor solución

Alejandro Varela (Idea 1):

- Laberinto con puertas

Como es un sistema de juegos poco tradicional, es una muy buena opción para que el público pueda probar algo nuevo.

Gabriel Suárez (Idea 2):

- Snake and Ladder

Un juego bastante querido por muchos y odiado por aquellos que les quedaba difícil ganar.

Luis Murcia (Idea 3):

- Laberinto visto desde arriba

Un laberinto tradicional que todos conocemos.

Criterios de Evaluación:

Criterio 1: Innovación de la aplicación

- [2] Innovador
- [1] Poco Innovador
- [0] Nada innovador

Criterio 2: Cumplimiento de los requisitos en el desarrollo del problema

- [2] Cumple con todos los requisitos
- [1] Cumple con algunos de los requisitos
- [0] No cumple con ninguno de los requisitos

Criterio 3: Facilidad de uso de la aplicación

- [3] Fácil
- [2] Normal
- [1] Difícil

	Criterio 1	Criterio 2	Criterio 3	Total
Idea 1	2	2	3	7
Idea 2	0	1	3	4
Idea 3	0	2	3	5

Gracias a estos criterios de evaluación, coincidimos en que la mejor solución a realizar es la 1, y tiene sentido, cumple con la mayoría de los lineamientos que se hicieron desde la primera etapa

Fase 6: Preparación de informes y especificaciones

Es necesario analizar los documentos en la carpeta docs. de este mismo proyecto.

Fase 7: Implementación del diseño

La implementación del diseño de nuestro proyecto está montada en GitHub con un enlace al repositorio.