

FASE 1: IDENTIFICACIÓN DEL PROBLEMA

Definición del problema

La comunidad universitaria requiere desarrollo de un Software que facilite la conexión social entre un grupo de personas, dado que actualmente por la pandemia las personas no pueden interactuar presencialmente y por esto las conexiones sociales se han visto disminuidas entre la comunidad universitaria, afectando de esta forma el aura social.

Identificación de necesidades y síntomas

- Encontrar la forma más eficiente de conocer a una persona mediante una secuencia de lasos sociales que empieza en el individuo y finaliza en la persona que desea conocer (Dijkstra)
- Verificar si es posible contactar a alguien mediante un grafo de relaciones, es decir, si alguno de mis amigos o conocidos tiene relación con la persona que deseo conocer (BFS)
- Listar las relaciones sociales de una persona por categorías tales como mejores amigos, amigos y conocidos
- Mostrar todos los contactos con la información de cada uno tal como correo, nombre de usuario, nombre de la persona, número telefónico, genero, genero de interés, edad y facultad
- Sugerir personas de manera aleatoria a las personas que apenas se registren y no cuenten con contactos aún
- Buscar entre los contactos una persona y mostrar la información de esta
- Eliminar una relación de una persona con otra persona que estaba dentro de sus relaciones.
- Modificar la información del perfil propio del usuario
- Crear una interfaz de registro e inicio de sesión de los usuarios
- Registrar la cuenta de una nueva persona por los campos requeridos (nombre completo, nombre de usuario, número de teléfono, genero, genero de interés, edad y facultad)
- Eliminar la cuenta definitivamente, esto quiere decir que la persona pierde los contactos que tenía y no podrá volver a ingresar nunca más a esta cuenta.

FASE 2: RECOPIACIÓN DE LA INFORMACIÓN NECESARIA

Requerimientos para la solución del problema

RF1. Encontrar la secuencia que conlleve a un individuo y que sea la secuencia más prometedora.

RF2. Modelar la fuerza que une a un individuo con otro, esto se refiera a que puede ser conocido, amigo y mejores amigos.

RF3. Mostrar todos los contactos con la información de cada uno tal como correo, nombre de usuario, nombre de la persona, número telefónico, genero, genero de interés, edad y facultad.

RF4. Sugerir de manera aleatoria un usuario a los usuarios que recién crean sus cuentas y aún no siguen a nadie.

RF5. Realizar la búsqueda entre los seguidores y mostrar la información del seguidor solicitado

RF6. Permitir que el usuario deje de seguir a otro usuario

RF7. Modificar la información del perfil propio del usuario, tal como contraseña o nombre de usuario

RF8. Registro de un nuevo usuario

RF9. Permitir inicio de sesión de los usuarios ya registrados.

RF10. Eliminar la cuenta definitivamente, esto quiere decir que la persona pierde los contactos que tenía y no podrá volver a ingresar nunca más a esta cuenta eliminada.

Restricciones para la solución del problema

- *La solución debe poseer una interfaz gráfica de usuario.*
- *El modelamiento de la base de datos de las personas debe estar implementada con una estructura de dato tipo Grafo.*
- *Debe implementarse al menos dos métodos funcionales de la estructura de datos tipo Grafo*
- *Debe implementarse todos los métodos, aunque no sean funcionales de la estructura de datos tipo Grafo*
- *Se debe tener dos implementaciones de las estructuras de datos tipo Grafo, una de lista y matriz de adyacencia*
- *Entre cada par de vértices existe una única arista con peso variable de 1 a 3.*

Definiciones

1. Vértice

Son puntos o nodos con los que están conformado los grafos. Llamaremos grado de un vértice, al número de aristas de las que es extremo. Se le dice vértice “par” o “impar” según sea su grado.

2. Arista

Una arista es una relación entre dos vértices de un grafo.

3. Grafo

Conjunto de objetos llamados vértices o nodos unidos por enlaces llamados aristas o arcos, que permiten representar relaciones binarias entre elementos de un conjunto.

4. Caminos

En Teoría de Grafos, se llama camino a una secuencia de vértices dentro de un grafo tal que exista una arista entre cada vértice y el siguiente. Se dice que dos vértices están conectados si existe un camino que vaya de uno a otro, de lo contrario estarán desconectados. Dos vértices pueden estar conectados por varios caminos. El número de aristas dentro de un camino es su longitud

5. Dijkstra

Algoritmo de 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.

6. Deep first search

Una Búsqueda en profundidad (en inglés DFS o Depth First Search) es un algoritmo de búsqueda no informada utilizado para recorrer todos los nodos de un grafo o árbol (teoría de grafos) de manera ordenada, pero no uniforme. Su funcionamiento consiste en ir

expandiendo todos y cada uno de los nodos que va localizando, de forma recurrente, en un camino concreto. Cuando ya no quedan más nodos que visitar en dicho camino, regresa (Backtracking), de modo que repite el mismo proceso con cada uno de los hermanos del nodo ya procesado.

7. Breadth first search

En Ciencias de la Computación, Búsqueda en anchura (en inglés BFS - Breadth First Search) es un algoritmo de búsqueda no informada utilizado para recorrer o buscar elementos en un grafo (usado frecuentemente sobre árboles). Intuitivamente, se comienza en la raíz (eligiendo algún nodo como elemento raíz en el caso de un grafo) y se exploran 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 árbol.

8. Algoritmo de Floyd-Warshall

En informática, el algoritmo de Floyd-Warshall, descrito en 1959 por Bernard Roy, 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.

9. Árbol de recubrimiento mínimo

Dado un grafo conexo y no dirigido, un árbol recubridor, árbol de cobertura o árbol de expansión de ese grafo es un subgrafo que tiene que ser un árbol y contener todos los vértices del grafo inicial. Cada arista tiene asignado un peso proporcional entre ellos, que es un número representativo de algún objeto, distancia, etc.; y se usa para asignar un peso total al árbol recubridor mínimo computando la suma de todos los pesos de las aristas del árbol en cuestión. Un árbol recubridor mínimo o un árbol de expansión mínimo es un árbol recubridor que pesa menos o igual que todos los otros árboles recubridores. Todo grafo tiene un bosque recubridor mínimo.

10. Algoritmo de Prim

El 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.

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.

11. Algoritmo de Kruskal

El algoritmo de Kruskal es un algoritmo de la teoría de grafos para encontrar un árbol recubridor 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).

12. Grado del vértice

Es el número de aristas que inciden sobre un vértice.

FASE 3: BÚSQUEDA DE SOLUCIONES CREATIVAS

Para la generación de ideas se realizó una revisión secuencial a conceptos previamente adquiridos relacionados con las necesidades del problema anteriormente expuesto. Partiendo de lo anterior, las posibles soluciones encontradas se exponen a continuación:

Implementación estructura de datos

- a. Grafo ponderado
- b. Grafo no ponderado
- c. LinkedList
- d. Árbol n-ario

Tipo de estructura de datos

- a. Grafo dirigido
- b. Grafo no dirigido
- c. Multígrafo dirigido
- d. Multígrafo no dirigido
- e. Ninguno de los anteriores (en caso de que no se elijan los grafos, si no la opción c o d)

Forma de implementación de estructura de datos

- a. Matriz de adyacencias
- b. Lista de adyacencias
- c. Ninguna de las anteriores (en caso de que no se elijan los grafos, si no la opción c o d)

FASE 4: TRANSICIÓN DE LA FORMULACIÓN DE IDEAS A LOS DISEÑOS PRELIMINARES

En esta fase hemos descartado las peores alternativas que no brindan una solución adecuada a los requerimientos, tales como: LinkedList, árbol n-ario, grafo dirigido y multígrafo dirigido.

- **Implementación estructura de datos**

Analizando a profundidad lo requerido, las estructuras tales como LinkedList y árbol n-ario no son las más eficientes ni nos permiten resolver los requerimientos que se piden en el enunciado.

- **Tipo de estructura de datos**

En este caso, no seleccionamos la opción ninguno de los anteriores dado que la estructura LinkedList y árbol n-ario se descartaron. Se descarta además el grafo dirigido y multígrafo dirigido dado que el tipo de grafo que mejor se acomoda a nuestros requerimientos son los grafos no dirigidos.

- **Forma de implementación estructura de datos**

En este caso, no seleccionamos la opción ninguno de los anteriores dado que la estructura LinkedList y árbol n-ario se descartaron. Sin embargo, la opción a y b siguen siendo muy factibles por lo que no se descarta ninguna de las dos.

FASE 5: EVALUACIÓN Y SELECCIÓN DE LA MEJOR SOLUCIÓN

- **Criterios implementación de estructura de datos**
 - a. Permita asignar los tipos de relación entre los seguidores del usuario
 - b. Permita encontrar el camino más conveniente a el usuario para conocer a una persona (esto se realiza por medio de los tipos de relación)
- **Criterios tipo de estructura de datos**
 - a. Permita modelar de la mejor manera el programa por medio de las funciones seguir (tanto del usuario hacía del seguidor, como del seguidor hacia el usuario, es decir que ambos pueden seguirse mutuamente)
- **Forma de implementación de estructuras de datos**

No tiene criterios, dado que se decide realizar la implementación de ambos tipos de grafo.

Selección de alternativas de implementación de estructuras de datos

	Criterio a	Criterio b	Total
Alternativa a	5	5	10
Alternativa b	0	0	0

Dado que obtiene el mayor puntaje la alternativa a, descartamos la alternativa b.

Selección de alternativas de tipo de estructuras de datos

	Criterio a	Total
Alternativa b	2	2
Alternativa d	5	5

Dado que obtiene el mayor puntaje la alternativa b, descartamos la alternativa a.

Webgrafía

- Definición de vértice tomada de: <https://sites.google.com/site/matematicasmoralesgalindo/6-1-elementos-y-caracteristicas-de-los-grafos/6-1-1-componentes-de-un-grafo-vertices-aristas-lazos-valencia>
- Definición de arista tomada de: <https://sites.google.com/site/matematicasmoralesgalindo/6-1-elementos-y-caracteristicas-de-los-grafos/6-1-1-componentes-de-un-grafo-vertices-aristas-lazos-valencia>
- Definición de vértice tomada de: <https://sites.google.com/site/matematicasmoralesgalindo/6-1-elementos-y-caracteristicas-de-los-grafos/6-1-1-componentes-de-un-grafo-vertices-aristas-lazos-valencia>
- Definición de grado de vértice tomada de: <https://sites.google.com/site/matematicasmoralesgalindo/6-1-elementos-y-caracteristicas-de-los-grafos/6-1-1-componentes-de-un-grafo-vertices-aristas-lazos-valencia>
- Definición de grafo tomada de: <https://es.wikipedia.org/wiki/Grafo>
- Definición de camino tomada de: [https://es.wikipedia.org/wiki/Camino %28teor%C3%ADa de grafos%29](https://es.wikipedia.org/wiki/Camino_%28teor%C3%ADa_de_grafos%29)
- Definición de Dijkstra tomada de: [https://www.ecured.cu/Algoritmo de Dijkstra](https://www.ecured.cu/Algoritmo_de_Dijkstra)
- Definición DFS tomada de: [https://es.wikipedia.org/wiki/B%C3%BAsqueda en profundidad](https://es.wikipedia.org/wiki/B%C3%BAsqueda_en_profundidad)

- Definición de BFS tomada de: https://es.wikipedia.org/wiki/B%C3%BAsqueda_en_anchura
- Definición de algoritmo de Floyd-Warshall tomada de: https://es.wikipedia.org/wiki/Algoritmo_de_Floyd-Warshall
- Definición de árbol de recubrimiento mínimo tomada de: https://es.wikipedia.org/wiki/%C3%81rbol_recubridor_m%C3%ADnimo
- Definición algoritmo de Prim tomada de: https://es.wikipedia.org/wiki/Algoritmo_de_Prim
- Definición de algoritmo de Kruskal tomada de: https://es.wikipedia.org/wiki/Algoritmo_de_Kruskal