Maria Alejandra Estacio
Santiago Figueroa

Método de la ingeniería

Contexto del problema

En la discoteca Son Latino el DJ quiere saber el género musical que a más personas le guste para priorizar esas canciones sobre las demás. Él tiene entendido que una persona no comparte los mismos géneros musicales con más de 3 personas y puede compartir varios con una misma persona, también que por definición una persona comparte consigo mismo todos sus gustos musicales así que eso no es tema de interés.
El Dj necesita una manera de registrar a estas personas y también saber con quienes comparten gustos musicales. También busca que las canciones más largas sean las que más gustan y que si no gustan tanto que sean de la menor duración posible, como para mantener una política de inclusión pero que no perjudique al negocio.
Por último, por políticas del negocio, se busca que el programa sea capaz de ofrecer una propuesta de ordenamiento del lugar según géneros compartidos, si muchas personas comparten un género, haya una zona donde probablemente ellos estén juntos por compartir este género.

1. **identificación del problema:** Principalmente hemos de encontrar el género a reproducir en la discoteca y proponer un orden del lugar según gustos en canciones compartidos. Para esto, tenemos que cumplir con:
    - Registrar el nombre, cédula, géneros y las canciones que les gusta de estos de los clientes que ingresan.
    - Crear una lista de reproducción para la noche.
    - Verificar que las canciones más largas sean las que más gustan.
    - Verificar que las canciones más cortas sean las que menos gustan.
    - Conocer las canciones que más se piden de un género especifico.
    - Ofrecer una posible distribución del lugar de acuerdo a canciones en común.

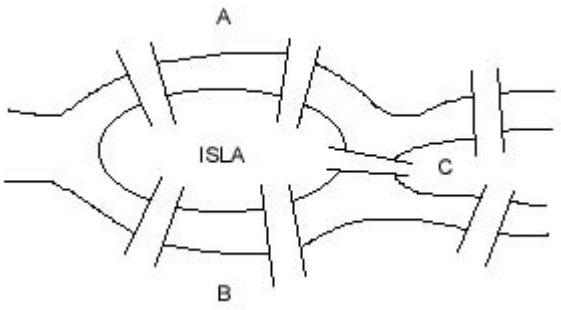
2. **Recopilación de la información**

Teoría de grafos

En matemáticas y en ciencias de la computación, la teoría de grafos (también llamada teoría de las gráficas) estudia las propiedades de los grafos. Un grafo es un conjunto, no vacío, de objetos llamados vértices (o nodos) y una selección de pares de vértices, llamados aristas (edges en inglés) que pueden ser orientados o no. Típicamente, un

grafo se representa mediante una serie de puntos (los vértices) conectados por líneas (las aristas).

El trabajo de Leonhard Euler, en 1736, sobre el problema de los puentes de Königsberg es considerado el primer resultado de la teoría de grafos. También se considera uno de los primeros resultados topológicos en geometría (que no depende de ninguna medida). Este ejemplo ilustra la profunda relación entre la teoría de grafos y la topología.



Tipos de grafos:

Terminología en teoría de grafos			
Tipos	***Aristas***	***¿Se admiten aristas múltiples?***	***¿Se admiten bucles?***
Grafo simple	No dirigidas	No	No
Multigrafo	No dirigidas	Sí	No
Pseudografo	No dirigidas	Sí	Sí
Grafo dirigido	Dirigidas	No	Sí
Multigrafo dirigido	Dirigidas	Sí	Sí

Estructuras de datos en la representación de grafos

Existen diferentes formas de almacenar grafos en una computadora. La estructura de datos usada depende de las características del grafo y el algoritmo usado para manipularlo. Entre las estructuras más sencillas y usadas se encuentran las listas y las matrices,

• **lista de incidencia:** Las aristas son representadas con un vector de pares (ordenados, si el grafo es dirigido), donde cada par representa una de las aristas.

• **lista de adyacencia:** Cada vértice tiene una lista de vértices los cuales son adyacentes a él. Esto causa redundancia en un grafo no dirigido (ya que A existe en

la lista de adyacencia de B y viceversa), pero las búsquedas son más rápidas, al costo de almacenamiento extra.

• **Matriz de adyacencia**

Grafos dirigidos.

G=(V,A) un grafo dirigido con |V|=n .Se define la matriz de adyacencia o booleana asociada a *G* como B_{nxn} con

$$b_{i,j} = \begin{cases} 1 & \text{si (i,j) pertenece a A} \\ 0 & \text{en otro caso} \end{cases}$$

Como se ve, se asocia cada fila y cada columna a un vértice y los elementos $b_{i,j}$ de la matriz son 1 si existe el arco (i,j) y 0 en caso contrario.

Grafos no dirigidos.

G=(V,A) un grafo no dirigido con |V|=n .Se define la matriz de adyacencia o booleana asociada a *G* como B_{nxn} con:

$$b_{ij} = b_{ij} \begin{cases} 1 \text{ si (i,j) pertenece a A.} \\ 0 \text{ en otro caso.} \end{cases}$$

La matriz B es simétrica con 1 en las posiciones *ij* y *ji* si existe la arista (i,j).

caminos y conexión:

Un recorrido en un grafo es una sucesión de vértices y aristas de la forma v0 a1 v1 a2...vk-1 ak vk donde la arista ai une los vértices vi-1 y vi. Éste es un recorrido de v0 a vk, de longitud k, siendo v1,..,vk-1 los vértices interiores del camino. Si v0=vk decimos que el recorrido es cerrado (en ocasiones se le llama circuito). Un camino en un grafo es un recorrido en el que no se repiten vértices ni aristas. Un ciclo es un camino cerrado.

DFS- recorrido en profundidad

Un Recorrido en profundidad (en inglés DFS o Depth First Search) es un algoritmo que permite recorrer todos los nodos de un grafo. Es una generalización del recorrido pre orden de un árbol.

La estrategia consiste en partir de un vértice determinado v 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.

Si existían vértices no alcanzables desde v el recorrido queda incompleto; entonces, se debe seleccionar algún vértice como nuevo vértice de partida, y repetir el proceso.

BFS - recorrido en amplitud

Recorrido en amplitud es otra 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.

La estrategia sería partir de algún vértice u, visitar u y después visitar cada uno de los vértices adyacentes a u. Hay que repetir el proceso para cada nodo adyacente a u, siguiendo el orden en que fueron visitados.

Algoritmos de Prim y Kruskal

Los algoritmos de Prim y Kruskal pertenecen al grupo de los algoritmos denominados "voraces". Estos algoritmos son aquellos que, para resolver un determinado problema, siguen un método que consiste en elegir la opción óptima en cada paso con el objetivo de de alcanzar una solución óptima general.

Un algoritmo voraz evalúa cada una de las opciones disponibles una sola vez. De este modo selecciona o rechaza la opción evaluada. Si selecciona la opción evaluada, entonces ésta pasa a formar parte de la solución y si es rechazada no pasa a formar parte de la solución y tampoco volverá a ser valorada para ello.

Utilizando algoritmos voraces no siempre es posible obtener una solución a un determinado problema. En estos casos se determina que el problema a resolver está fuera del alcance y por tanto, serán necesarios otros métodos para resolverlo.

Normalmente los algoritmos voraces se aplican a problemas de optimización.

Algoritmo Prim

El algoritmo de Prim, dado un grafo conexo, no dirigido y ponderado, encuentra un árbol de expansión mínima. Es decir, es capaz de encontrar un subconjunto de las aristas que forman un árbol que incluya todos los vértices del grafo inicial, donde el peso total de las aristas del árbol es el mínimo posible.

Funcionamiento del algoritmo de Prim

1. Se marca un vértice cualquiera. Será el vértice de partida.
2. Se selecciona la arista de menor peso incidente en el vértice seleccionado anteriormente y se selecciona el otro vértice en el que incide dicha arista.
3. Repetir el paso 2 siempre que la arista elegida enlace un vértice seleccionado y otro que no lo esté. Es decir, siempre que la arista elegida no cree ningún ciclo.
4. El árbol de expansión mínima será encontrado cuando hayan sido seleccionados todos los vértices del grafo.

Algoritmo de Kruskal

El algoritmo de Kruskal, dado un grafo conexo, no dirigido y ponderado, encuentra un árbol de expansión mínima. Es decir, es capaz de encontrar un subconjunto de las aristas que forman un árbol que incluya todos los vértices del grafo inicial, donde el peso total de las aristas del árbol es el mínimo posible.

Funcionamiento del algoritmo de Kruskal

1. Se selecciona, de entre todas las aristas restantes, la de menor peso siempre que no cree ningún ciclo.
2. Se repite el paso 1 hasta que se hayan seleccionado |V| - 1 aristas. Siendo V el número de vértices.

Algoritmo de Dijkstra

El *Algoritmo de Dijkstra*, también denominado *Algoritmo de caminos mínimos*, es un modelo que se clasifica dentro de los algoritmos de búsqueda. Su objetivo, es determinar la ruta más corta, desde el nodo origen, hasta cualquier nodo de la red. Su metodología se basa en iteraciones, de manera tal que, en la práctica, su desarrollo se dificulta a medida que el tamaño de la red aumenta, dejándolo en clara desventaja, frente a métodos de optimización basados en programación matemática.

Algoritmo de Floyd-Warshall (todos los caminos mínimos)

El problema que intenta resolver este algoritmo es el de encontrar el camino más corto entre todos los pares de nodos o vértices de un grafo. Esto es semejante a construir una tabla con todas las distancias mínimas entre pares de ciudades de

un mapa, indicando además la ruta a seguir para ir de la primera ciudad a la segunda. Este es uno de los problemas más interesantes que se pueden resolver con algoritmos de grafos.

Existen varias soluciones a este problema y los algoritmos a aplicar dependen también de la existencia de arcos con pesos o costes negativos en el grafo. En el caso de no existir pesos negativos, sería posible ejecutar V veces el algoritmo de Dijkstra para el cálculo del camino mínimo, donde V es el número de vértices o nodos del grafo. Esto conlleva un tiempo de ejecución de O (V^3) (aunque se puede reducir). Si existen arcos con pesos negativos, se puede ejecutar también V veces el Algoritmo de Bellman-Ford, una vez para cada nodo del grafo. Para grafos densos (con muchas conexiones o arcos) esto conlleva un tiempo de ejecución de O (V^4).

3. Búsqueda de soluciones creativas

Mediante una lluvia de ideas decidimos que las posibles soluciones son estas:

- **Usar multigrafos:** Mediante los multigrafos podemos registrar a los clientes y con los algoritmos de camino de estos resolver la problemática del género más escuchado y el orden de la lista de reproducción.
- **Enlistar los clientes:** Podemos usar una lista que contenga a los clientes junto a su género y canciones que quiere y mediante un algoritmo recorrer esta lista llenando la lista de reproducción y a su vez buscando el género que más gusta.
- **Permitir a los clientes elegir la canción:** Dejar el deber de organizar la lista a los mismos clientes, que ellos pidan una canción y con un algoritmo que obtenga la canción más pedida maneje la lista de reproducción. Además, que con esa información obtenida intuya cual es el género más requerido.
- **Reproducir géneros por horas:** Dado un estudio o una prueba piloto se podría conseguir un protocolo en el que se designe por horas a poner música de un mismo género. La parte del género que más gusta se obtendrá del estudio y la lista de reproducción se hará priorizando las que más gusten de las que no tanto.
- **usar grafo simple:** Dado que en el problema no es tema de interés que un cliente comparta consigo mismo sus gustos entonces no necesitamos bucles, por lo tanto, podríamos registrar a los clientes y mediante los algoritmos de estos resolver nuestros problemas.
- **usar matriz de adyacencia:** nos permite tener un mejor manejo de grafos y si el grafo es muy grande y contiene muchos vértices, la matriz de adyacencia es muy útil.

- **usar lista de adyacencia:** la lista de adyacencia nos puede ayudar a representar uno de nuestros grafos.

4. Transición de las Ideas a los Diseños Preliminares:

Gracias al análisis que realizamos a nuestras soluciones creativas decidimos desistir a las siguientes alternativas:

- **Usar grafo simple:** esta idea no es conveniente ya que un grafo simple no permite aristas múltiples, por lo tanto, más de dos clientes no tendrían más de un género musical en común, lo que no es cierto ya que más de un cliente pueden compartir gustos musicales.
- **Reproducir el mismo género:** no es una buena alternativa, ya que sí, con un algoritmo de búsqueda podemos encontrar el género que más guste, pero reproducir solo este género aburriría a los que no les gusta ya que puede haber a más de una persona que no les guste.
- **Permitir a los clientes elegir la canción:** No es una buena idea que los clientes elijan la canción, debido a que elegirán lo que a ellos les gusta sin importar la influencia que tengo sobre los demás.

Con las alternativas restantes tenemos que:

- **usar multígrafos:** Es una muy buena alternativa dado que la mayoría de los clientes tienen varios gustos en común y un multígrafo nos permite tener múltiples aristas. Por otro lado, un multígrafo no permite bucles y en nuestro problema no son necesarios.
- **Enlistar los clientes:** esta es una buena idea dado que al enlistar a los clientes podemos utilizar un algoritmo de grafos el cual nos permite buscar la canción que más guste.
- **reproducir los géneros por horas:** esta alternativa es muy buena dado que para reproducir las canciones por horas dependiendo de su tiempo podemos usar un algoritmo de grafos el cual nos permita encontrar los caminos mínimos, para así reproducir más canciones por hora.

- **usar matriz de adyacencia:** La matriz de adyacencia es muy útil dado que si hay muchos clientes y el grafo se vuelve muy grande la matriz es de muy buena utilidad ya que simplifica el problema.
- **usar lista de adyacencia:** la lista de adyacencia nos puede ayudar a representar uno de nuestros grafos, aparte de esto, las búsquedas son más rápidas.

5. Evaluación y selección de la mejor solución:

A continuación, se evalúan cada una de las alternativas, teniendo en cuenta que la de mayor puntaje es la más eficiente.

Criterios:

Criterio A: La alternativa es la adecuada para solucionar alguno de los problemas que plantea la situación.
- [2] Ayuda en la solución
- [1] No ayuda en la solución

Criterio B: La alternativa es eficiente a un nivel:
- [4] Constante.
- [3] Mayor a constante.
- [2] Logarítmica
- [1] Lineal o mayor

Criterio C: La alternativa permite combinarse con otras para generar una mejor solución
- [2] Mejora con otra alternativa
- [1] Es suficiente por ella misma

	Criterio A	Criterio B	Criterio C
Alternativa 1	2		2
Alternativa 2	2	3	2
Alternativa 3	2	3	2
Alternativa 4	2	3	2
Alternativa 5	2	3	2

Referencias

- http://decsai.ugr.es/~jfv/ed1/tedi/cdrom/docs/grafos.htm
- http://www.unipamplona.edu.co/unipamplona/portalIG/home_23/recursos/general/11072012/grafo3.pdf
- http://www.dma.fi.upm.es/personal/gregorio/grafos/web/caminos_minimos/teoria/teoria.htm
- https://www.icesi.edu.co/moodle/pluginfile.php?file=%2F45689%2Fmod_resource%2Fcontent%2F1%2FMDII-Grafos1.pdf
- http://163.10.22.82/OAS/recorrido_grafos/dfs__recorrido_en_profundidad.html
- https://sites.google.com/site/complejidadalgoritmicaes/kruskal
- https://www.ingenieriaindustrialonline.com/investigacion-de-operaciones/algoritmo-de-dijkstra/
- https://estructurasite.wordpress.com/algortimo-de-floyd-warshall/