# 

# El mejor influencer:

# 

# 

# Geovanny Quintero Velez – A00378039

# Johan Felipe Jojoa - A00377905

# Juan David Garzón - A00377983

# 

# 

# 

# Computación y estructuras discretas I

# 2022

# Método de la ingeniería

## Identificación del problema

Se requiere implementar correctamente una solución que implemente grafos como estructura para almacenar y disponer de los datos. Los obstáculos principales residen en generar la estructura, que funcione correctamente y formular una aplicación en la que podría resultar de utilidad. Con los conocimientos sobre grafos adquiridos en el curso se ideó entonces una simulación de una red social, en donde se necesita transmitir publicidad y la persona ideal para transmitirla será quien más seguidores tenga, además al ser una red social también tendrá la opción de amigos con la cual buscaremos el camino más corto desde un conocido hasta otro, como lo puede ser la persona con más seguidores o amigos directos.

Situación problema:

Decidimos simular el ambiente de una red social para implementar los grafos y los métodos que contienen. La solución debe permitir localizar a la persona con más amigos tenga. Por ejemplo, para esparcir publicidad a través de sus amigos y los amigos de sus amigos, etc. hasta llegar a la mayor cantidad de personas posible. Por otro lado, también se debe conocer la ruta más corta desde una persona hasta otra, que idealmente sería la persona con más amigos directos.

# Especificación de requerimientos funcionales

* El programa debe permitir buscar a la persona que más amigos tenga.
* El programa debe permitir conocer el camino más corto desde una persona hasta otra.
* El programa debe conocer el camino más corto a la persona con más amigos directos (Nodo con mayor grado).

## Recopilación de información

* Grafo:

“Un grafo es una entidad matemática introducida por Euler en 1736 para representar entidades (vértices) que pueden relacionarse libremente entre sí, mediante el concepto de arista. Se puede definir un TAD Grafo basado en estas ideas, el cual contiene elementos sobre los que está definida una relación de vecindad o adyacencia. Un vértice puede relacionarse con cualquier otro vértice y establecer cualquier número de relaciones.”

Recuperado de: <https://www.infor.uva.es/~cvaca/asigs/estr0506apg.pdf>

* BFS

Es un algoritmo para buscar en una estructura de datos de árbol un nodo que satisfaga una propiedad determinada. Comienza en la raíz del árbol y explora todos los nodos de la profundidad actual antes de pasar a los nodos del siguiente nivel de profundidad. Se necesita memoria adicional, normalmente una cola, para llevar la cuenta de los nodos hijos que se han encontrado pero que aún no se han explorado.

Recuperado de: <https://en.wikipedia.org/wiki/Breadth-first_search>

* DFS

Es un algoritmo para recorrer o buscar estructuras de datos en forma de árbol o grafo. El algoritmo comienza en el nodo raíz (seleccionando algún nodo arbitrario como nodo raíz en el caso de un gráfico) y explora lo más lejos posible a lo largo de cada rama antes de retroceder.

Recuperado de: <https://en.wikipedia.org/wiki/Depth-first_search>

* 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. Su nombre se refiere a Edsger Dijkstra, quien lo describió por primera vez en 1959.

Recuperado de: <https://www.ecured.cu/Algoritmo_de_Dijkstra>

* 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.

Recuperado de: [https://es.wikipedia.org/wiki/Algoritmo\_de\_Floyd-Warshall](https://es.wikipedia.org/wiki/Algoritmo_de_Floyd-Warshall#:~:text=En%20inform%C3%A1tica%2C%20el%20algoritmo%20de,v%C3%A9rtices%20en%20una%20%C3%BAnica%20ejecuci%C3%B3n.)

* Algoritmo de Prim

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.

Recuperado de: <https://es.wikipedia.org/wiki/Algoritmo_de_Prim#:~:text=El%20algoritmo%20de%20Prim%20es,y%20cuyas%20aristas%20est%C3%A1n%20etiquetadas>.

* 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 conexo). Este algoritmo toma su nombre de Joseph Kruskal, quien lo publicó por primera vez en 1956.

Recuperado de: <https://es.wikipedia.org/wiki/Algoritmo_de_Kruskal#:~:text=El%20algoritmo%20de%20Kruskal%20es,del%20%C3%A1rbol%20es%20el%20m%C3%ADnimo>.

## Búsqueda de soluciones creativas:

Luego de recopilar la información suficiente sobre los grafos, la búsqueda de la mejor implementación de los mismos al problema escogido se convertía en una prioridad principal, en base a ello, se dispuso una lluvia de ideas en la cual se generarían sugerencias con respecto a la construcción de los grafos y a su implementación.

### Estructura de los grafos:

#### Alternativa 1:

Simple.

#### Alternativa 2:

Simple dirigido.

#### Alternativa 3:

Multigrafo.

#### Alternativa 4:

Multigrado dirigido.

### Métodos a utilizar:

#### Alternativa 1:

BFS

#### Alternativa 2:

DFS

#### Alternativa 3:

Algoritmo de Dijkstra

#### Alternativa 4:

Algoritmo de Floyd - Warshall

#### Alternativa 5:

Algoritmo de Prim

#### Alternativa 6:

Algoritmo de Kruskal

## Transición de la formulación de ideas a diseños preliminares:

Lo primero que hicimos en este paso fue descartar las ideas que no son tan fáciles de implementar y cumplen los mismos requerimientos que otras soluciones. En este sentido descartamos como estructura a los multígrafos, debido a que aumentan en gran medida la complejidad de la solución. En cuanto a los métodos utilizados, descartamos los métodos de árboles generadores mínimos puesto que estos no aportan nada a la solución de las problemáticas planteadas. Finalmente, los candidatos a solución restantes son:

### Estructura de los grafos:

#### Alternativa 1:

Simple.

#### Alternativa 2:

Simple dirigido.

Métodos a utilizar:

#### Alternativa 1:

Algoritmo de Dijkstra.

#### Alternativa 2:

Algoritmo de Floyd - Warshall.

#### Alternativa 3:

BFS.

#### Alternativa 4:

DFS.

## Evaluación y selección de la mejor solución:

Se evaluarán las estructuras teniendo en cuenta la siguiente rúbrica:

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Criterios | Éxito total (3) | Éxito alto (2) | Éxito suficiente (1) | Éxito Nulo (0) |
| Número de conexiones entre los vértices | Permite conectar de uno a muchos vértices entre sí | Permite conectar dos o más vértices entre sí | Solo permite una arista por vértice | No permite conexiones |
| Navegación entre los vértices | Permite navegar libremente de un vértice al otro | Permite solo navegar en una de las dos direcciones posibles | Solo permite navegar en una dirección | No permite navegar entre vértices |
| Adaptación a la solución | La estructura utilizada modela adecuadamente la solución requerida | La estructura utilizada modela el 80% de la solución | La estructura utilizada modela el 50% de la solución | La estructura utilizada modela menos del 50% de la solución |

Se evaluarán los métodos a usar de acuerdo a las siguiente rúbrica:

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Criterios | Éxito total (3) | Éxito alto (2) | Éxito suficiente (1) | Éxito Nulo (0) |
| Tiempo de ejecución | O(1) | O() | O() | O() o superior |
| Recursos utilizados | O(1) | O() | O() | O() o superior |
| Utilidad frente al problema | La utilidad del método resulta indispensable para la solución | El método resulta muy útil, pero no es el mejor | El método puede ayudar en solo pequeñas partes de la solución | El método es totalmente dispensable |

### Estructuras:

|  |  |  |  |
| --- | --- | --- | --- |
| Opciones | Criterio 1 | Criterio 2 | Criterio 3 |
| Alternativa 1 | 3 | 3 | 2 |
| Alternativa 2 | 3 | 2 | 3 |

### Métodos a usar:

|  |  |  |  |
| --- | --- | --- | --- |
| Opciones | Criterio 1 | Criterio 2 | Criterio 3 |
| Alternativa 1 | 1 | 1 | 3 |
| Alternativa 2 | 0 | 2 | 2 |
| Alternativa 3 | 3 | 2 | 3 |
| Alternativa 4 | 3 | 2 | 1 |

Puntuaciónes:

Para las estructuras:

Alternativa 1 = 7.

Alternativa 2 = 7.

Para los métodos:

Alternativa 1 = 5.

Alternativa 2 = 6.

Alternativa 3 = 6.

Alternativa 4 = 8.

Los métodos a implementar serán el algoritmo de Dijkstra que nos ayudara a encontrar los caminos mínimos y el de BFS que usaremos para encontrar el que tenga mayores seguidores.

En cuanto a las estructuras se decidieron usar ambas: Grafos simple y dirigido. Se pensó que el algoritmo BFS sería más útil en el grafo dirigido, bajo un contexto de seguidores, solucionando el problema a encontrar a cuantas personas puede acceder una, expandiendo publicidad y siendo el mas influyente. Por otra parte, Dijkstra es más útil en un contexto de amigos, si se busca conocer a una persona con el menor número de intermediarios posible.