Facultad de Ingeniería
Universidad de Concepción

Universidad de Concepción

# Tarea Computacional Matemáticas Discretas (501252-1).


## Integrantes

Bryan Aguirre (Departamento de Ingeniería Informática y Ciencias de la Computación).

Máximo Beltrán (Departamento de Ingeniería Informática y Ciencias de la Computación).

Javier Castillo (Departamento de Ingeniería Informática y Ciencias de la Computación).

Francisco Fuentealba (Departamento de Ingeniería Informática y Ciencias de la Computación).

## Docentes

Pierluigi Cerulo.

Waldo Gálvez.

# Introducción

Esta tarea consiste en crear un algoritmo de Dijkstra en lenguaje C/C++, es decir, implementar un código que permita encontrar el camino más corto entre un par de vértices de un grafo. Esto para aplicar los conocimientos obtenidos en el curso y sumarlo con nuestras habilidades de programación.

El algoritmo debe ser capaz de encontrar un camino entre un vértice de origen y un vértice de destino dados, en caso de no existir dicha ruta, el programa debe indicar mediante una impresión en pantalla que no es posible encontrar un camino que conecte ambos vértices.

Para abordar esta situación se utilizará un algoritmo basado “en colas”, es decir, se planea introducir múltiples nodos a una lista a modo de cola, y así revisar sus nodos adyacentes para comprobar a que nodos se puede avanzar, para de esta forma ir construyendo el camino poco a poco, como cada peso tiene el valor de 1 el algoritmo netamente se reduce a búsqueda por anchura, por lo que cada que se avanza a un nodo nuevo, se incrementa en uno el largo del camino actual, para posteriormente comparar los múltiples caminos y encontrar el más óptimo.

Todos los elementos por utilizar se definirían como estructuras, que guardaran diversos tipos de datos e información, como nodos, tamaños de los caminos, estados de ejecución, etc.

La documentación de este proyecto estará estructurada de la siguiente forma:

- 2° sección: Objetivos propuestos para el desarrollo del programa.
- 3° sección: Explicación sobre la entrada del programa y cómo se trabajan y procesan los nodos, aristas y matriz de adjacencia del grafo.
- 4° sección: Salida y resultados del programa, diferencias entre grafos dirigidos y no dirigidos en el programa.
- 5° sección: Conclusión, principales observaciones del proyecto.

# Objetivos

## Objetivo general.

Desarrollar en conjunto un programa capaz de encontrar el camino más corto entre dos vértices mediante el algoritmo de Dijkstra, aplicando conceptos de programación en C y teoría de grafos, con el fin de comprender el algoritmo de Dijkstra y como aplicarlo en programación en el caso asociado.

## Objetivos específicos.

- Indagar sobre el algoritmo de Dijkstra: Leer y entender su funcionamiento, descifrar una implementación acorde a la problemática asociada, y lograr situarlo dentro del código.

- Asociar conocimientos adquiridos en programación en C y Teoría de Grafos, para lograr un algoritmo eficiente y estructurado que cumpla con los requisitos establecidos.

- Observar en la medida del desarrollo del código, posibles fallas en la funcionalidad del programa, y una rápida incorporación de soluciones óptimas para un funcionamiento eficiente y preciso.

- Distribuir roles de trabajo dentro del proyecto para desarrollar un buen entorno de colaboración entre pares, promoviendo el trabajo colectivo y el buen trato entre futuros compañeros de profesión.

# Modelo

## Definición y construcción del grafo a partir de la entrada del programa.

Los distintos tipos de grafos se almacenan en distintos archivos de texto (.txt), diferenciados por sus respectivos nombres.

Los vértices se definen mediante un carácter, puede ser cualquier letra, número y/o símbolo a excepción de la coma (,) ya que esta se usa como separador de los vértices.

Ejemplo de entrada para vértices:
V = {A, B, C, D, E, R, Z}.

Las aristas se leerán como un conjunto de pares de vértices, separados por coma (,).

Ejemplo de entrada para aristas:
E = {AB, DZ, ER, DE, AC}.

Es importante que las aristas NO contengan vértices que no estén en el grafo, es decir, para un conjunto de vértices V, si para un vértice v, $v \notin V$, entonces

$\forall k \in V$, $vk \notin E$ o bien $\forall v,k \notin V$, $vk \notin E$

Construcción del grafo:

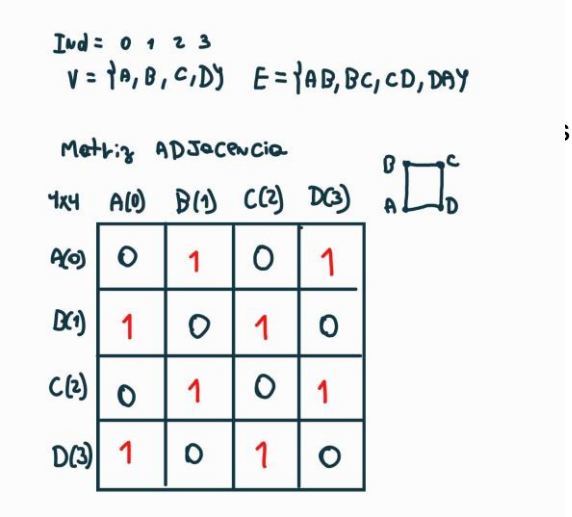
Función de los vértices en la construcción del grafo

- Se usarán para identificar los índices para crear y modificar la matriz de adjacencia del grafo.

Función de las aristas en la construcción del grafo:

- Se usarán para crear la matriz de adjacencia agregando un 1 si los vértices están conectados o un 0 si no.

Notar que: si 0 entonces no son adyacentes



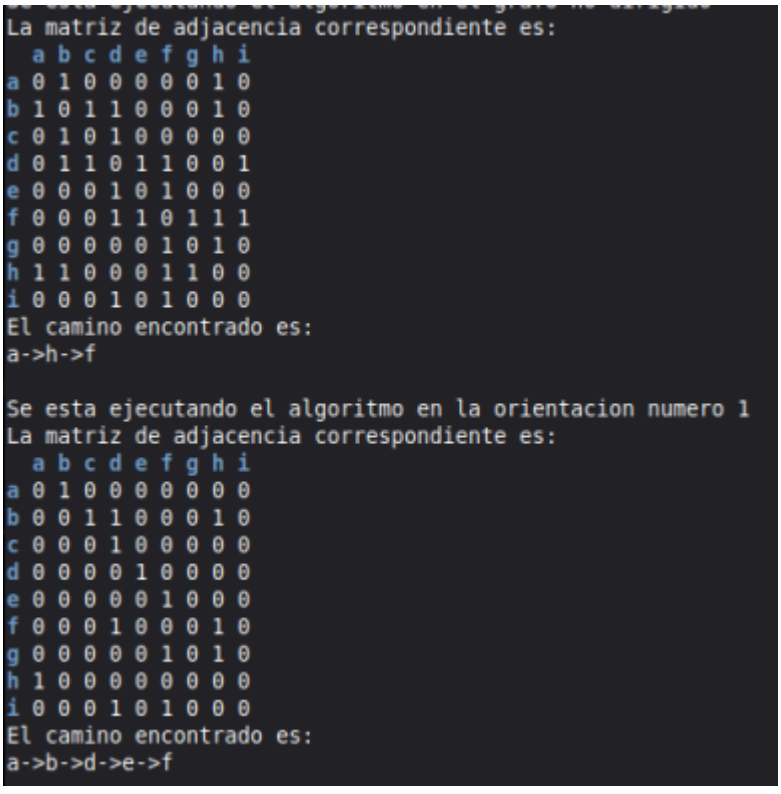
Se trabajará el algoritmo como estructura de datos (FCFS), se implementó un sistema de cola, en el cual se irán obteniendo valores de vértice periódicamente, se obtendrán los adyacentes a los cuales se les pueda cambiar su distancia (valor) asignada por defecto al inicio del algoritmo, y luego se seleccionará uno de estos arbitrariamente para continuar la secuencia/recorrido. Se visitarán todos los caminos posibles. Se guardarán de uno en uno, y para comparar si agregarlos o no a la lista de caminos de menor longitud, se verificará la longitud del/los guardado/s. Si es menor, elimina los que estaban en la lista y coloca al nuevo en ella. Si es igual, se agrega a la lista existente, y si es mayor no se agrega y se mantiene la lista actual.

Al finalizar la lista de caminos posibles, imprime la lista guardada de los más cortos.

## Resultados

El algoritmo logró el objetivo de crear la matriz de adjacencia e implementar el Dijkstra. Los resultados fueron satisfactorios en el caso de grafos y dígrafos, cumpliendo así con el objetivo propuesto.
El programa distingue entre grafo y dígrafo con la matriz de adjacencia ya que en caso de los dígrafos la arista AB != BA y en caso de los grafos AB == BA, dando así orientación a los dígrafos, pero al momento del algoritmo para aplicar Dijkstra trabaja a los grafos y dígrafos de la misma manera.

imagen de los resultados obtenidos



## Conclusión

Este trabajo demandaba una buena capacidad de análisis y síntesis del problema en cuestión. Gracias a las indicaciones dadas, nos permitió simplificar mucho la funcionalidad en cuestión. Se puede tomar de referencia el caso del algoritmo de Dijkstra, pues, aunque en su totalidad ocupa variables de anchura y peso, al notar en el problema la asignación igualitaria de este a cada vértice, permitió simplificarlo a solo un parámetro (anchura). Debido a este tipo de visión dentro del equipo, cumplimos el objetivo de entender el algoritmo y aplicarlo.

En el trabajo colectivo, pudimos asignar rápidamente roles dentro del equipo. Dependiendo de la etapa del problema, se dividieron roles de creación del algoritmo y programación, y el de avance y desarrollo del informe. Si bien hay integrantes que tenían mayor afinidad con alguna de las dos tareas, se logró un trabajo equitativo, cumpliendo con lo estipulado.

Durante el desarrollo del proyecto, surgieron diversos contratiempos, debido a errores de compilación y testing. El que más se repitió fue “Segmentation fault”, el cual ocurre en el manejo de arreglos. Si bien atrasaron un poco el trabajo, se arreglaron rápidamente, por lo que se cumplió con lo esperado.

Se adjuntaron los detalles de compilación y ejecución del programa en plataforma de trabajo GitHub, en el README asociado.

Link: https://github.com/pandita45/MatematicasDiscretasTarea.git