****BENEMÉRITA UNIVERSIDAD AUTÓNOMA DE PUEBLA

FACULTAD DE CIENCIAS DE LA COMPUTACIÓN

**ANÁLISIS Y DISEÑO DE ALGORITMOS**

PROFESOR: MIGUEL ÁNGEL VARGAS LOMELI

“PROYECTO 2”

EQUIPO

APORTELA HERNÁNDEZ VICTOR HUGO

ORTIZ ORTEGA JOSE IGNACIO

PRIMAVERA 2019

Contenido

[Introducción 3](#_Toc7440990)

[Objetivos 3](#_Toc7440991)

[Generales 3](#_Toc7440992)

[Específicos 3](#_Toc7440993)

[Marco Teórico 4](#_Toc7440994)

[Origen 4](#_Toc7440995)

[El Algoritmo de Kruskal 4](#_Toc7440996)

[Desarrollo 5](#_Toc7440997)

[Impresiones 6](#_Toc7440998)

[Conclusiones 8](#_Toc7440999)

[Referencias 8](#_Toc7441000)

# Introducción

El presente Documento debe ser fiel reflejo de toda la funcionalidad del sistema, contiene toda aquella información necesaria para que el lector de este documento pueda entender claramente los objetivos y el funcionamiento del proyecto mencionado.

# Objetivos

## Generales

Desarrollar un proyecto que satisfaga todos los requerimientos que fueron pedidos a realizar

## Específicos

* Implementar una propuesta de algoritmo de Kruskal usando un lenguaje de programación.
* Crear un archivo con entradas múltiples para ser ocupadas con el programa creado.
* Ejecutar dicho programa en diferentes Sistemas Operativos

# Marco Teórico

## Origen

**Joseph B. Kruskal** fue un investigador del Math Center (Bell-Labs), que en 1956 descubrió un algoritmo para la resolución del problema del Árbol de coste total mínimo (minimum spanning tree - MST) al que más tarde, nombraría como “Algoritmo de Kruskal”.

El objetivo del algoritmo de Kruskal es construir un árbol (subgrafo sin ciclos) formado por arcos sucesivamente seleccionados de mínimo peso a partir de un grafo con pesos en los arcos.

## El Algoritmo de Kruskal

Dado un grafo G con nodos conectados por arcos con peso (coste o longitud): el peso o coste total de un árbol será la suma de pesos de sus arcos. Obviamente, árboles diferentes tendrán un coste diferente. El problema es entonces ¿cómo encontrar el árbol de coste total mínimo?

Una manera de encontrar la solución al problema del árbol de coste total mínimo, es la enumeración completa. Aunque esta forma de resolución es eficaz, no se puede considerar un algoritmo, y además no es nada eficiente.

Este problema fue resuelto independientemente por Dijkstra (1959), Kruskal (1956) y Prim (1957) y la existencia de un algoritmo polinomial (que todos ellos demostraron) es una grata sorpresa, debido a que un grafo con N vértices puede llegar a contener N^N-2 subárboles. A lo largo de la historia se ha hecho un gran esfuerzo para encontrar un algoritmo rápido para este problema. El algoritmo de Kruskal es uno de los más fáciles de entender y probablemente el mejor para resolver problemas a mano.

El algoritmo se basa en una propiedad clave de los árboles que permite estar seguros de si un arco debe pertenecer al árbol o no, y usar esta propiedad para seleccionar cada arco. Nótese en el algoritmo, que siempre que se añade un arco (u,v), éste será siempre la conexión más corta (menor coste) alcanzable desde el nodo u al resto del grafo G. Así que por definición éste deberá ser parte del árbol.

Este algoritmo es de tipo **Voraz**, ya que, a cada paso, éste selecciona el arco más barato y lo añade al subgrafo. Este tipo de algoritmos pueden no funcionar para resolver otro tipo de problemas, por ejemplo, para encontrar la ruta más corta entre los nodos a y b.

# Desarrollo

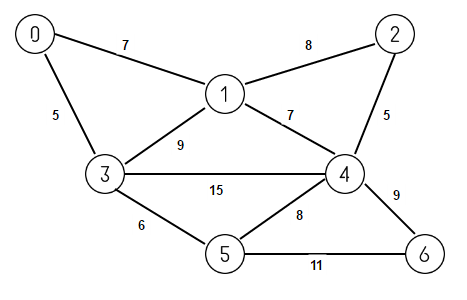
Para probar el correcto funcionamiento de nuestro programa hicimos pruebas con dos ejemplos diferentes.

Ejemplo 1:

Matriz de adyacencias del grafo de pruebas.

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
|  | A | B | C | D | E | F | G |
| A | 0 | 7 | 0 | 5 | 0 | 0 | 0 |
| B | 7 | 0 | 8 | 9 | 7 | 0 | 0 |
| C | 0 | 8 | 0 | 0 | 5 | 0 | 0 |
| D | 5 | 9 | 0 | 0 | 15 | 6 | 0 |
| E | 0 | 7 | 5 | 15 | 0 | 8 | 9 |
| F | 0 | 0 | 0 | 6 | 8 | 0 | 11 |
| G | 0 | 0 | 0 | 0 | 9 | 11 | 0 |

Grafo

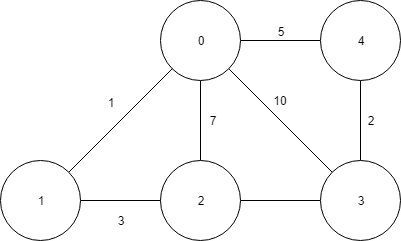


Ejemplo 2:

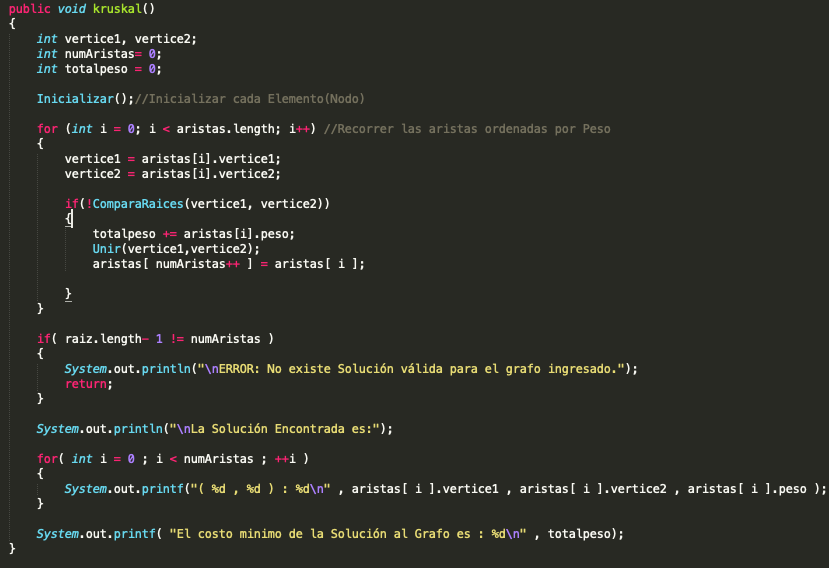
Matriz de adyacencias del grafo de pruebas.

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
|  | A | B | C | D | E |
| A | 0 | 1 | 7 | 10 | 5 |
| B | 1 | 0 | 3 | 0 | 0 |
| C | 7 | 3 | 0 | 4 | 0 |
| D | 10 | 0 | 4 | 0 | 2 |
| E | 5 | 0 | 0 | 2 | 0 |

Grafo



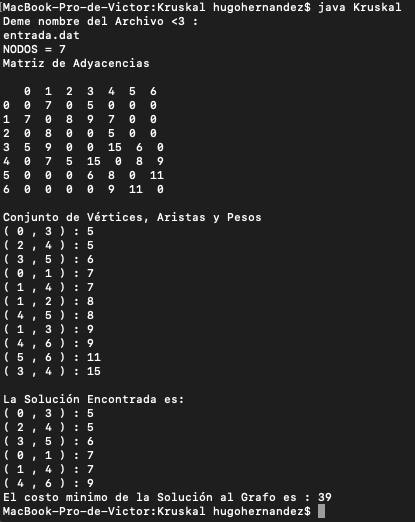
Código de Función Kruskal



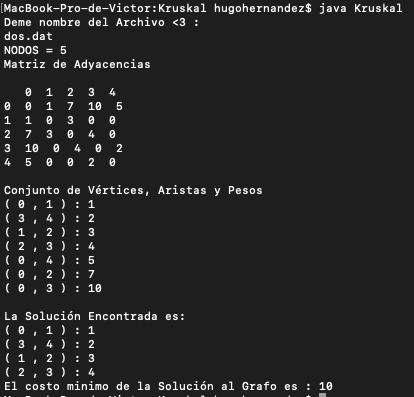
# Impresiones

Implementación del Programa realizado utilizando los datos de entrada de los ejercicios descritos en **Desarrollo**, dando como resultados:

* **Ejemplo 1**



* **Ejemplo 2**



# Conclusiones

El Algoritmo de Kruskal siempre tiene una solución óptima a este tipo de problemas donde se busca el árbol de expansión mínimo; eso quiere decir que este algoritmo pertenece a P porque se puede resolver de forma eficiente por una máquina determinista en tiempo polinomial.

Después de realizar el presente proyecto, los conocimientos adquiridos acerca del algoritmo de Kruskal aumentaron y se aclararon muchas dudas con respecto al nombrado anteriormente. Existieron algunas dificultades durante el proceso de desarrollo, sin embargo, se resolvieron de forma deseada.

# Referencias

Cormen, Thomas H. “Introducción a los Algoritmos”. 2001. 3ra Edición. Wikipedia. https://es.wikipedia.org/wiki/Algoritmo\_de\_Kruskal

Flores Alejo, Jhoel (2013). “Algorithms de Arboles”. SlideShare. https://es.slideshare.net/joeflores946517/algoritmo-de-kruskal.

Knuth, Donald (1998). «5.2.2: Ordenamiento por intercambio». *El arte de programar ordenadores, Volumen 3*. (segunda edición). Addison-Wesley. pp. 106-110.