**DFS y DFS\_FOREST**

*Estructura de datos utilizadas*

GrafoMA(matriz de adyacencia):

* Tuve la necesidad de agregar 2 métodos más que son Adyacentes() y getCantNodos().
* Utiliza la estructura de datos ConjuntoLD que implementa ConjuntoTDA, se usa para pedir el conjunto de vértices y para pedir los adyacentes de vértice V

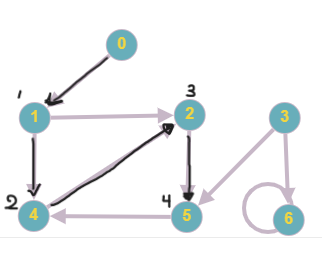
Tenemos dos métodos para llamar con DFS, DFS y DFS\_FOREST.

DFS iniciara su recorrido desde un origen recorriendo los adyacentes de manera recursiva. Mientras que DFS\_FOREST va a recorrer por todos los vértices sus adyacentes

Seguimiento de ejemplo:

Imagen que contiene Diagrama

Descripción generada automáticamente

Si empezamos el recorrido con DFS y como inicio el 0 seria de esta forma:

Se empieza desde el 0 y va a recorrer los adyacentes del 0, después pasa al 1, como los adyacentes están ordenados de mayor a menor se dirige a el 4, luego va hacia el 2, por último, hacia el 5 y como el adyacente de 5 ya fue visitado no entra.

Lo que imprime por consola:

Imagen que contiene teclado, computadora, sostener

Descripción generada automáticamente

Para DFS\_FOREST visita todos y se imprime de esta forma:

Imagen que contiene teclado, computadora

Descripción generada automáticamente

Esta es la lista de predecesores con respecto a la lista de los vértices:

Tabla, Calendario

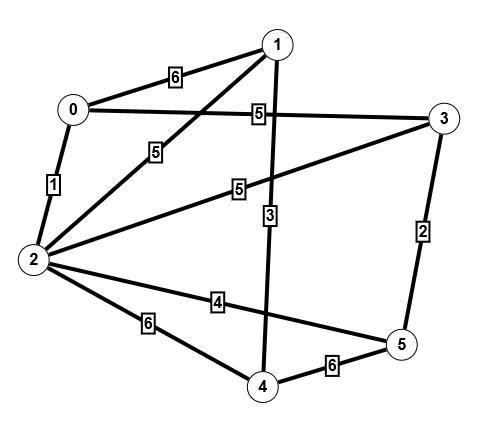
Descripción generada automáticamente

P son los predecesores y V los vértices del grafo.

PRIM

el grafo utilizado es GrafoND que es de un grafo ponderado no dirigido y se maneja con matriz de adyacencia

utilizo este grafo :

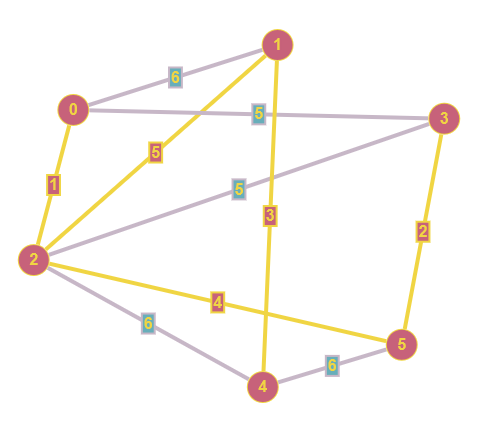
Para el algoritmo de Prim, utilizo arreglos para almacenar la información sobre los nodos padres y si fueron visitados o no. También empleo un ArrayList de aristas para almacenar el árbol mínimo. Inicializo todos los pesos de mi arreglo peso con un valor máximo y marco a todos los nodos como no visitados. Establezco el punto de inicio del recorrido del grafo con el valor 0 y con -1 para indicar que es el nodo raíz.

Luego, ejecuto el algoritmo de Prim que recorre todos los nodos, encuentra el de menor peso y lo marca como visitado. Después, visita los nodos adyacentes y, si encuentra uno con un peso menor, actualiza su valor.

La función minPeso busca el peso mínimo de un nodo no visitado.

Después, creo el árbol de recubrimiento mínimo y lo cargo en el ArrayList de aristas.

Y por ultimo llamo a la función que imprime y muestra por pantalla el resultado.



Su peso total es de 15