**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

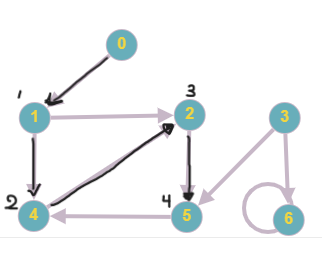
Tenemos dos métodos para llamar con DFS, DFS y DFS\_FOREST.

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

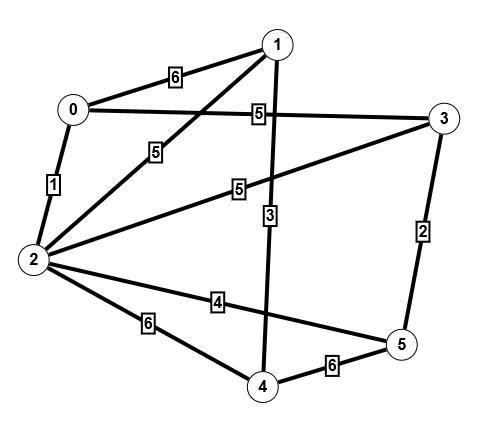
Descripción generada automáticamente

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 una variable para almacenar la cantidad de nodos que tiene mi grafo. Con esta cantidad, inicializo los diferentes arreglos que voy a utilizar: Padre[ ], Peso[ ] y Visitado[ ]. También empleo un ArrayList de aristas para almacenar el árbol mínimo.

Inicializo todos los pesos de mi arreglo Peso[ ] con un valor máximo y mi arreglo Visitado[ ] en false. Establezco desde dónde se va a comenzar a recorrer el grafo con Padre[0] = -1, indicando que este será el nodo raíz, y Peso[0] = 0, ya que es el nodo raíz.

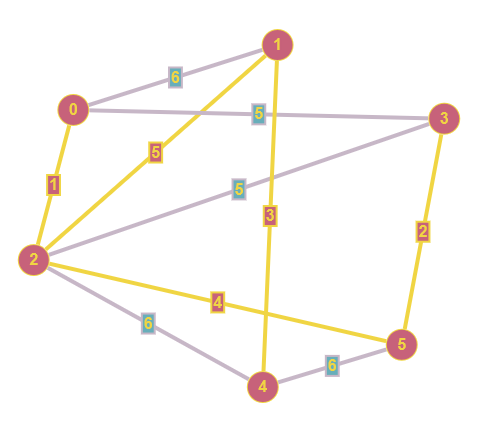
Luego, se inicia el algoritmo de Prim, que recorre el grafo estableciendo, a través de la función minPeso, el siguiente nodo a visitar basándose en el recorrido con el menor peso. Se marca el nodo como visitado y se buscan los pesos de los nodos adyacentes. Si el peso es menor, se actualizan los arreglos.

A medida que avanza el algoritmo, cargo en el ArrayList los datos de las aristas y nodos para crear el árbol de recubrimiento mínimo. Finalmente, se imprime el árbol por pantalla junto con el peso total del recorrido.

Principio del formulario

Final del formulario

El árbol resultante de mi grafo es:



Su peso total es de 15