Práctica 30

Datos generales:

Nombre de la Práctica	Grafos
Nombre de la carrera	Ingeniería de Software
Nombre de la materia	Laboratorio de Estructuras de Datos
Número y nombre de Unidad(es) temática(s)	IV. Estructuras no lineales.
Docente que imparte la materia	Aldonso Becerra Sánchez
Fecha de entrega para los alumnos	17-noviembre-2021
Fecha de entrega con penalización	18-noviembre-2021
Fecha de elaboración:	17-noviembre-2021

Objetivo de la Práctica	Practicar con las operaciones sobre grafos
Tiempo aproximado de realización	2.5 horas
Introducción	Los grafos son útiles en el manejo de mapas y planos donde se defina un amplia gama de posibilidades de conexión entre diferentes entidades con el fin de estar comunicadas o por el hecho de ser dependientes unas de otras

Referencias que debe consultar el alumno (si se requieren):

Referencia 1:

1.Cairo, Osvaldo; Guardati, Silvia. Estructura de Datos, Tercera Edición. McGraw-Hill, México, Tercera Edición, 2006.

Referencia 2:

2.Mark Allen Weiss. Estructura de datos en Java. Ed. Addison Wesley.

Referencia 3:

3. Joyanes Aguilar, Luis. Fundamentos de Programación. Algoritmos y Estructuras de Datos. Tercera Edición, 2003. McGraw – Hill.

Actividades que debe realizar el alumno:

Actividad inicial:

Generar el reporte en formato IDC.

Actividad 1:

Primero genere la **Introducción**.

Actividad 2:

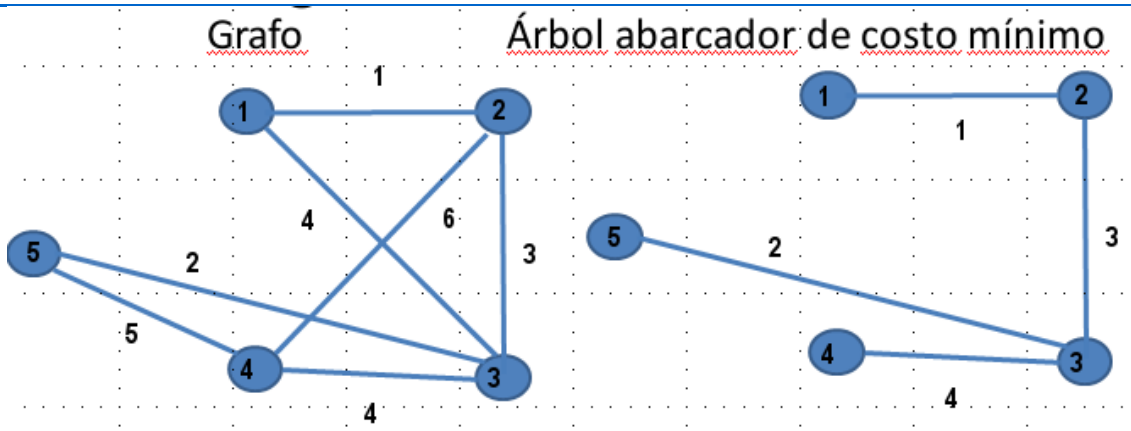
El algoritmo de Prim y el algoritmo de Kruskal pertenecen a la teoría de grafos, donde ambos permiten obtener el árbol abarcador de costo mínimo. El árbol abarcador de costo mínimo es un árbol formado por las aristas de G que conectan todos los vértices con un costo total mínimo. Los algoritmos encuentran 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.

Algoritmo de Prim.

Permite encontrar el árbol abarcador de costo minimo de un grafo. Para ello utilizan dos componentes: V, conjunto de todos los vértices, y U, conjunto auxiliar iniciado con el primer vértice. Los vértices mínimos se van agregando a L, que originalmente está vacío.

1. Mientras V<>U Repetir
   Elegir una arista (u,v) ∈ A(G) tal que el costo
   sea el mínimo, u ∈ U y v ∈ (V - U)
   Agregar el arista (u,v) a L
   Agregar el nodo v a U
2. {fin del ciclo del paso 1}

Ejemplo:



Iteración	U	(u,v)	Costo	v	L	U actualizado
1	{1}	(1,2)	1	2	{(1,2)}	{1,2}
2	{1,2}	(2,3)	3	3	{(1,2),(2,3)}	{1,2,3}
3	{1,2,3}	(3,5)	2	5	{(1,2),(2,3),(3,5)}	{1,2,3,5}
4	{1,2,3,5}	(3,4)	4	4	{(1,2),(2,3),(3,5),(3,4)}	{1,2,3,5,4}

Algoritmo de Kruskal

Al igual que el de Prim, permite encontrar el árbol abarcador de costo mínimo de un grafo. La construcción se lleva acabo seleccionando la arista de menor costo y agregándola al árbol abarcador. Se debe generar una serie de particiones a partir del conjunto de vértices V. Inicialmente las particiones tienen el tamaño de 1.

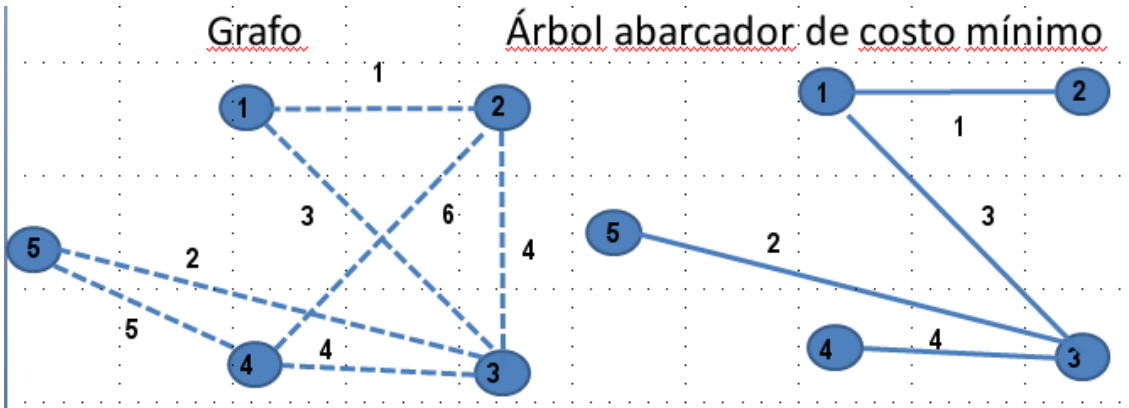
P0={{V1},{V2},{V3},..,{Vn}}

donde P0 indica la partición inicial.

A partir de ahí, se busca la arista de menor costo, y si está en dos particiones diferentes, estas particiones se reemplazan por su unión. El proceso termina hasta que se tenga una sola partición formada por los vértices V.

1. *Mientras* haya vértices en P que pertenezcan a particiones distintas *Repetir*
   De L, seleccionamos la arista (u,v) que tenga el menor costo.
   1.1 Si (u y v se encuentran en particiones diferentes) entonces
   Unir las particiones a las cuales pertenezcan u y v
   1.2{fin del condicional del paso 1.1}
2.{fin del condicional del paso 1}

Ejemplo:



Iteración	Arista elegida	Costo	Particiones
0			P0={{1},{2},{3},{4},{5}}
1	(1,2)	1	P1={**{1,2}**,{3},{4},{5}}
2	(3,5)	2	P2={**{1,2}**,**{3,5}**,{4}}
3	(1,3)	3	P3={**{1,2,3,5}**,{4}}
4	(3,4)	4	P4={**{1,2,3,5,4}**}

Como parte del **Desarrollo**, hacer un método que permita realizar el proceso de Prim o Kruskal (usted elija cuál de los dos) utilizando la clase **GrafoMatriz**.

Actividad 3:

Pruebe el funcionamiento del programa de las actividades con todo y sus capturas de pantalla.

Actividad 4:

Realice la sección de **Código agregado** (diagrama de clases UML).

Actividad 5:

Realice la sección de **Pre-evaluación** (use los lineamientos establecidos).

Actividad 6:

Finalmente haga las **Conclusiones**.

Actividad 7:

Enviar en http://ingsoftware.reduaz.mx/moodle

Archivo anexo que se requiere para esta tarea (opcional):

Dudas o comentarios: a7donso@gmail