Nombre y Apellido: ..... N° Legajo: .....

Segundo Parcial de Estructuras de Datos y Algoritmos

Primer Cuatrimestre de 2012 – 25/06/2012

Ejercicio 1	*Ejercicio 2*	*Ejercicio 3*	*Nota*

Condición Mínima de Aprobación: Tener por lo menos dos ejercicios con B-

Ejercicio 1

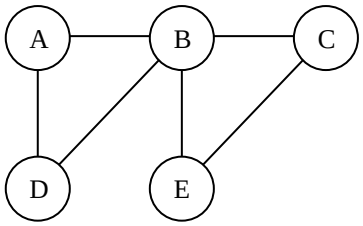
Se cuenta con la siguiente implementación parcial de un grafo no dirigido con peso en las aristas:

```
public class Graph<V> {

	private HashMap<V, Node> nodes = new HashMap<V, Node>();
	private List<Node> nodeList = new ArrayList<Node>();

	public void addVertex(V vertex) {
		if (!nodes.containsKey(vertex)) {
			Node node = new Node(vertex);
			nodes.put(vertex, node);
			nodeList.add(node);
		}
	}
	public void addArc(V v, V w, int weight) {
		Node origin = nodes.get(v);
		Node dest = nodes.get(w);
		if (origin != null && dest != null && !origin.equals(dest)) {
			for (Arc arc : origin.adj) {
				if (arc.neighbor.info.equals(w)) {
					return;
				}
			}
			origin.adj.add(new Arc(weight, dest));
			dest.adj.add(new Arc(weight, origin));
		}
	}

	private class Node {
		V info;
		boolean visited = false;
		List<Arc> adj = new ArrayList<Arc>();

		public Node(V info) {
			this.info = info;
		}
		public int hashCode() {
			return info.hashCode();
		}

		public boolean equals(Object obj) {
			if (obj == null || obj.getClass() != getClass()) {
				return false;
			}
			return info.equals(((Node)obj).info);
		}
	}
	private class Arc {
		int weight;
		Node neighbor;

		public Arc(int weight, Node neighbor) {
			this.weight = weight;
			this.neighbor = neighbor;
		}
	}
}
```

Agregar a la implementación de grafos un método que reciba una lista de objetos de tipo V y determine si dicha lista se corresponde con un recorrido DFS del grafo.

```
public boolean isDFS(List<V> values)
```

Ejemplo: para el siguiente grafo, el algoritmo retornaría **true** para las siguientes listas {A, D, B, C, E}, {B, D, A, E, C}, {B, C, E, D, A}; y retornaría **false** para estas listas: {A, B, E, D, C}, {B, E, D, A, C}, {A, F}, {B, E, C, E, C}.

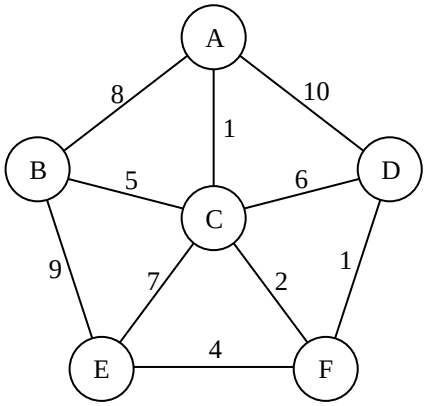


Ejercicio 2

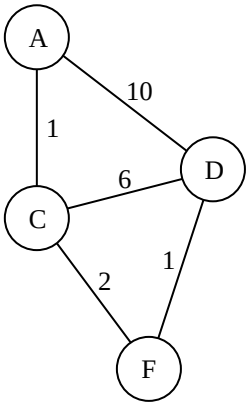
Se tiene un grafo con pesos enteros positivos en las aristas. Implementar un algoritmo que dado un nodo **v** y cierto valor entero **n**, retorne un subgrafo formado por todos los nodos alcanzables desde **v** por al menos un camino de peso menor o igual a **n**, y todas las aristas existentes entre estos nodos en el grafo original.

```
public Graph<V> subgraph(V v, int n)
```

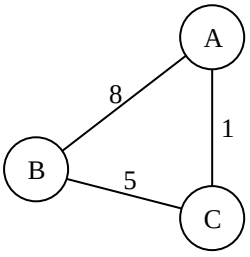
Ejemplo: se tiene el grafo de la figura (a). Si se ejecuta el algoritmo con v=A y n=5 se obtiene el grafo de la figura (b). Si se ejecuta con v=B y n=6 se obtiene el grafo de la figura (c).



(a)



(b)



(c)

Ejercicio 3

Se quiere implementar un algoritmo para determinar el número cromático de un grafo (cantidad mínima de colores con los que se pueden pintar los nodos de manera tal que dos nodos adyacentes no tengan el mismo color).

Agregarle a la implementación de grafos un método que utilice un algoritmo de tipo **hill climbing** para encontrar una solución aproximada.