Nombre y Apellido: ….…… N° Legajo: ……

***Segundo Parcial de Estructuras de Datos y Algoritmos 72.34***

***Segundo Cuatrimestre de 2013 – 19/11/2013***

| *Ejercicio 1* | *Ejercicio 2* | *Ejercicio 3* | *Nota* |
|---|---|---|---|
| | | | |

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

**Consideraciones a tener en cuenta. MUY IMPORTANTE**

- **El ejercicio que no respete estrictamente el enunciado será anulado.**
- **Se puede entregar el examen escrito en lápiz.**
- **Se tendrán en cuenta la eficiencia y el estilo de programación.**
- **Los teléfonos celulares deben estar apagados.**
- **Entregar los ejercicios en hojas separadas**

## *Ejercicio 1*

Se cuenta con la siguiente implementación parcial de un grafo no dirigido:

```
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, Double d) {
		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(dest, d));
			dest.adj.add(new Arc(origin, d));
		}
	}

	private class Node {
		V info;
		boolean visited = false;
		int tag = 0;
		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 {
		Node neighbor;
		Double weight;

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

Un "ciclo toniano" en un grafo G es un ciclo simple (no repite nodos) que pasa al menos por la mitad de los vértices de G. Agregar a la clase de grafos un método que determine si tiene un "ciclo toniano". Nota: el grafo puede estar vacío y puede no ser conexo.
El método debe retornar el ciclo hallado o null si no hay un "ciclo toniano".

***Ejercicio 2***

Agregar a la clase de grafos del ejercicio anterior un método que determine en forma aproximada el peso del camino con mayor peso entre dos nodos. Los pesos pueden ser negativos. El método a utilizar para encontrar este camino en forma aproximada debe ser Hill Climbing "puro" (no Hill Climbing Estocástico ni ninguna otra adaptación al método original).
El método debe retornar únicamente el peso del camino.

***Ejercicio 3***

La excentricidad ( notado como $\epsilon(v)$ ) de un nodo $v$ de un grafo es la mayor distancia entre $v$ y cualquier otro nodo. Puede ser pensado como cuán lejos se puede llegar desde ese nodo hasta el nodo más distante del grafo.

El radio $r$ de un grafo es la excentricidad mínima para todos los nodos, en símbolos sería

$$r = \min_{v \in V} \epsilon(v)$$

donde V es el conjunto de nodos del grafo.

Agregar al grafo un método que retorne el valor del radio.