Ejercicio 1

Inciso 1:

Partiendo de los conceptos vistos en clase, sabemos que una arista se puede representar como un par ordenado de vértices de un grafo. Con esto en mente, considero que la estructura más sencilla para el archivo que contiene la data es aquella en la cual cada línea del archivo representa un par ordenado de vértices. De esta manera, tendríamos la información de las conexiones entre cada vértice en el grafo. En caso de que haya un vértice sin conexiones, aparecerá como un solo elemento en el archivo. Un ejemplo de este formato utilizando un grafo de 4 vértices es:

$$1, 3$$

$$4, 1$$

$$1, 2$$

$$3, 2$$

Inciso 2:

Con esta estructura se cumplen los 3 requisitos solicitados. Cualquier humano que lea el archivo, asumiendo que conoce el contexto de la situación, puede entender fácilmente que, por ejemplo, los vértices $1, 3$ están conectados. Cuando se expresa la asunción de que la persona conoce el contexto se quiere decir que dicho individuo está consciente de que se está lidiando con grafos. Adicionalmente, es un archivo fácil de editar en el caso de que se requiera modificar una conexión o se quiera agregar una nueva; solo habría que escribir un nuevo par ordenado o modificar alguno existente. Sin embargo, entiendo que se puede hacer un poco tedioso si el grafo es muy grande ya que el archivo sería muy extenso. Finalmente, un programa puede cargar el *file* fácilmente tan solo leyendo cada línea y agregando los vértices que conforman el par ordenado.

Inciso 3: Algoritmo para cargar el archivo.

Entrada: $archivo\ F\ que\ contiene\ la\ data\ del\ grafo.$
Salida: $estructura\ representativa\ de\ un\ grafo\ G.$

1. ***hacer mientras*** $no\ sea\ el\ final\ del\ archivo$
2. $edge \leftarrow readLine()$
3. $vertices[\,] \leftarrow edge.splitByCommas()$
4. ***si*** $vertices.size() < 2$ ***entonces hacer***
5. ***si*** $G.exists(vertices[0])\ es\ falso$ ***entonces hacer***
6. $G.addVertex(vertices[0])$
7. ***fin si***
8. ***si no hacer***
9. ***si*** $G.exists(vertices[0])\ es\ falso$ ***entonces hacer***
10. $G.addVertex(vertices[0])$
11. ***fin si***
12. ***si*** $G.exists(vertices[1])\ es\ falso$ ***entonces hacer***
13. $G.addVertex(vertices[1])$
14. ***fin si***
12. $G.addEdge(vertices[0], vertices[1])$
13. ***fin mientras***
14. ***retorna*** G

Inciso 4.

Tomando en cuenta los datos que recibimos, tendríamos que:

$$T(m, n, a, v) = (m * v) + (n * a)$$