
# Informe:
# Proyecto para examen libre

**El siguiente informe fue confeccionado con el fin de guiar y ayudar al usuario a entender la implementación y la utilización correcta del sistema.**
**Dicho sistema pertenece al proyecto de la materia Estructuras de Datos.**

**Autor:**

- **Jerez, Martin. LU: 118507**

# Consideraciones Previas

A la hora de realizar el proyecto se tuvieron en cuenta las siguientes consideraciones:

1. Se eliminó de la interfaz Graph los métodos removeEdge y removeVertex para así facilitar y evitar problemas de compilación a la implementar el TDAGrafo.
2. Se agregó en la interfaz Map el método entradasPorBucket el cual se solicitada implementar en la consigna del proyecto.
   Obs: Se tiene en cuenta que una interfaz no debería ejemplificar o forzar la implementación interna del TDA pero en este caso lo vi necesario para la facilidad a la hora de implementar el método en la clase MapeoHashAbierto.
3. Se agregó el paquete Super (el cual contiene únicamente la interfaz Position) al proyecto para facilitar la legibilidad y lectura del código, ya que la interfaz Position contenida en Super es usada por múltiples paquetes y de esta manera no se repite la misma clase en los diferentes paquetes.
4. Se agregó el TDALista al proyecto debido a que se requería del mismo para la implementación de algunos métodos y de los TDAGrafo, TDAMapeo solicitados.
5. Se asume que a la hora de utilizar la interfaz se posee conocimiento de los tipos genéricos que poseen el mapeo y el grafo generados, y que no se van a ingresar valores de otros tipos.
6. Se asume que el grafo generado no va a poseer dos vértices con el mismo rotulo.
7. TDAGrafo implementación Grafo no dirigido con lista de adyacencia.
8. TDAMapeo implementación Mapeo con hash abierto.

# Algoritmos principales

- public String entradasPorBucket():

  Este método implementado en la clase MapeoHashAbierto consiste en recorrer la tabla hash abierta y a medida que se pasa por los buckets correspondientes de la tabla se almacena en un String el índice del arreglo el cual representa ese bucket y la cantidad de elementos que posee el mismo.

- private Vertex<String>[] buscarVertices(String rotuloV1, String rotuloV2):

  Este método implementado en la clase Logica consiste en buscar en el grafo, que se tiene como atributo de instancia, los vértices que posean como rotulo a los parametrizados de la siguiente manera:

  Se recorren todos los vértices del grafo hasta encontrar los vértices con los rótulos parametrizados, una vez encontrados se deja de buscar y se retornan los mismo almacenados en un arreglo de la siguiente manera, array[0]= vertice1; array[1]= vértice2, siendo vertice1 y vertice2 los vértices que contienen a los rótulos parametrizados respectivamente.

- public String hallarCamino(String rotulov1, String rotulov2):

  Este método implementado en la clase Logica consiste en hallar, si es que existe, el camino que une a los vértices que posean los rótulos parametrizados y mostrar el mismo indicando el costo entre tales vértices. Para poder cumplir su función el método hace uso de métodos auxiliares los cuales algunos se nombraran a continuación.

- private boolean DFSst(Vertex<String> origen, Vertex<String> destino, PositionList<Vertex<String>> camino, Map<Vertex<String>, Boolean> map):

  Este método funciona haciendo uso del método para recorrer grafos llamado DFS(Depth-First Search) con la particularidad de que en este caso se parametriza una colección en la cual se almacenara el camino que se recorre para encontrar el vértice indicado, en este caso “destino”, empezando desde el vértice “origen”. También se utiliza un mapeo de vértices a boolean con la funcionalidad principal de marcado de visitado sobre los vértices del grafo.

  El método de búsqueda DFS funciona de la siguiente manera, su funcionamiento consiste en ir expandiendo todos y cada uno de los vertices que va localizando, de forma recurrente, en un camino concreto. Cuando ya no quedan más vertices que visitar en dicho camino, regresa (Backtracking), de modo que repite el mismo proceso con cada uno de los vertices adyacentes ya procesados. En este caso, esta forma de recorrer el

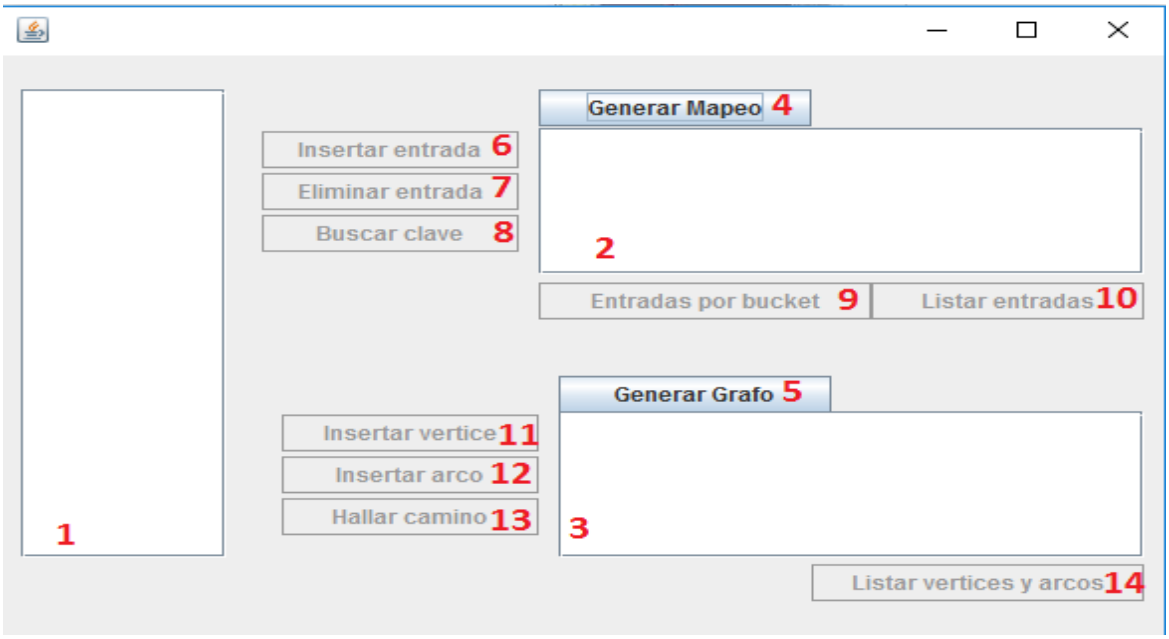
grafo se modifica de manera que cada vez que se regresa al vértice anterior (Backtracking), el vértice visitado se retirara de la colección en la cual se guarda el camino recorrido. También se modifica de manera que una vez llegado al vértice "destino" se termina la búsqueda del camino y se devuelve el camino recorrido.

He de aclarar que hago uso de este algoritmo para recorrer el grafo ya que en el enunciado dado no se indicó que se debía de hallar el camino mínimo entre los vertices.

# Funcionabilidad de la interfaz grafica

La función general de la interfaz es la simulación del manejo de las estructuras tales como Grafo y Mapeo, la cual permite ejecutar las siguientes operaciones:

I. Generar Mapeo.
II. Insertar entrada.
III. Eliminar entrada.
IV. Buscar clave.
V. Entradas por bucket.
VI. Listar entradas.
VII. Generar Grafo.
VIII. Insertar vértice.
IX. Insertar arco.
X. Hallar camino.
XI. Listar vertices y arcos.

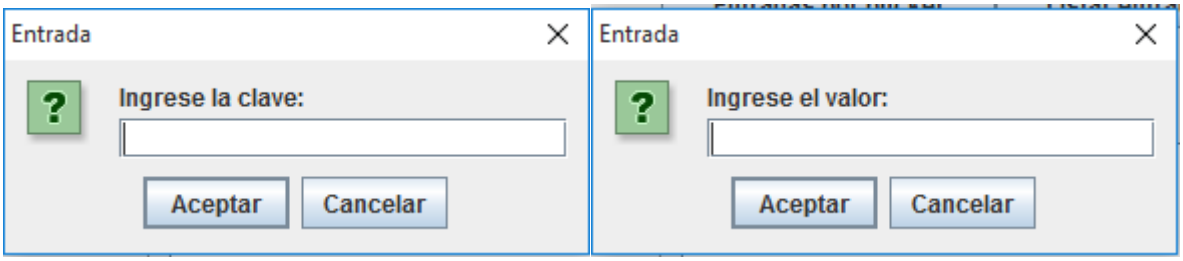


- En 1(Panel de acciones), se mostran las acciones que se van realizando a lo largo del uso del sistema.
- En 2(Panel de mapeo), se mostraran los resultados de las operaciones 9 y 10.
- En 3(Panel de grafo), se mostrara el resultado de la operación 14.

# Funcionalidad y uso de cada operación

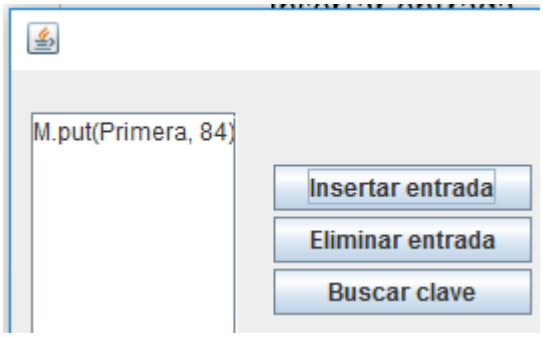
I. Generar Mapeo: Al presionar este botón se generara internamente el mapeo el cual almacenara entradas del tipo E= (K, V), siendo E: Entrada, K: Clave de tipo String, V: Valor de tipo Integer. También, una vez se presione el botón se habilitaran los botones "Insertar entrada", "Eliminar entrada", "Buscar clave", "Entradas por bucket" y "Listar entradas".

II. Insertar entrada: Al presionar este botón se abrirá una ventana en la cual se solicitara que ingresemos una clave y un valor para la nueva entrada.




Obs: En caso de no completar los campos y presionar en Aceptar la operación no se realiza;

Una vez completado ambos campos y presionado el botón aceptar, se insertara en el mapeo la entrada correspondiente y se indicara en el panel de acciones la acción realizada. En caso de insertar una clave ya existente con diferente valor, el valor de la entrada antigua será reemplazada en el mapeo por la ingresada.

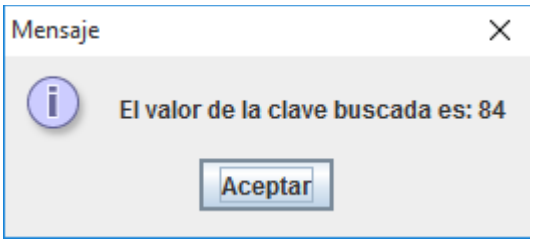



Siendo M: el mapeo generado, y (Primera, 84): la entrada ingresada.

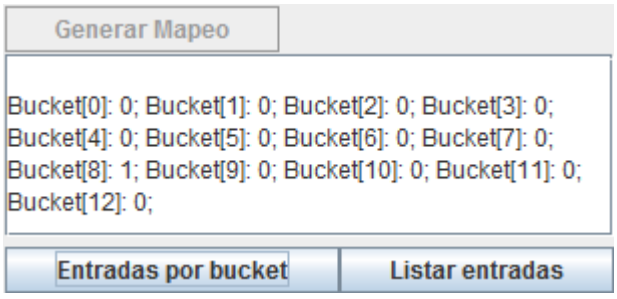
III. Eliminar entrada: Al presionar este botón se abrirá una ventana similar a la anteriormente mencionada en la cual se solicitara que ingresemos la clave que contiene la entrada a eliminar. Una vez ingresada la misma ocurrirán dos situaciones, en caso de que el mapeo posea una entrada con la misma clave ingresada entonces la misma será eliminada del mapeo y se dará por concluida la operación, por el contrario, si la clave

ingresada no pertenece a ninguna de las entradas del mapeo entonces el mapeo no se verá modificado.

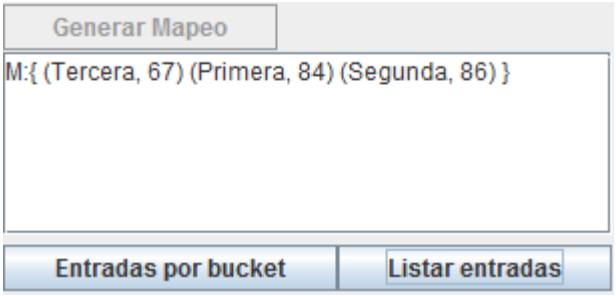
IV. Buscar clave: Al presionar este botón se abrirá una ventana en la cual se solicitara que ingresemos una clave la cual indicara la clave que se desea buscar en el mapeo, una vez ingresada la misma ocurrirán dos situaciones, en caso de que la clave no existe en el mapeo se mostrara un mensaje indicando que no se ha encontrado la clave deseada. Por otro lado, si se ingresa una clave que pertenezca al mapeo, entonces se mostrara un mensaje el cual indicara el valor asociado a dicha clave.




V. Entradas por bucket: Al presionar este botón se actualizara el panel de mapeo y se mostrara en el estado interno de cada bucket de la tabla hash, el cual indicara la cantidad de entradas que hay en cada uno de la siguiente manera.




VI. Listar entradas: Al presionar este botón se actualizara el panel de mapeo y se mostrada el estado actual del mapeo, indicando en el mismo las entradas que posee.



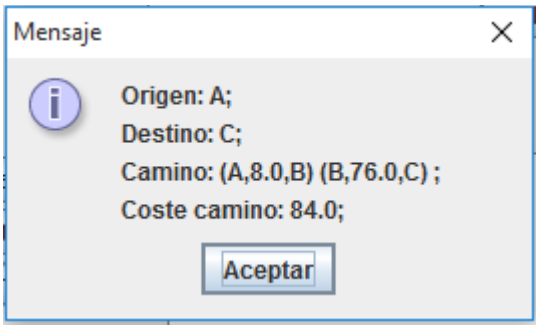

VII. Generar grafo: Al presionar este botón se generara internamente el grafo el cual almacenara vertices de tipo String y arcos con peso de tipo Real, y se habilitara el uso de los botones asociados a las operaciones de grafo.

VIII. Insertar vértice: Al presionar este botón se abrirá una ventana la cual nos solicitara que ingresemos el rotulo que poseerá el nuevo vértice.

IX. Insertar arco: Al presionar este botón se abrirá una ventana en la cual se nos solicitara que ingresemos los rótulos de los vertices de los cuales unirá el nuevo arco creado y el peso que poseerá el arco creado. En caso de ingresar algún vértice inexistente en los campos se mostrara un mensaje indicando cuál de los vertices fue el erróneo.

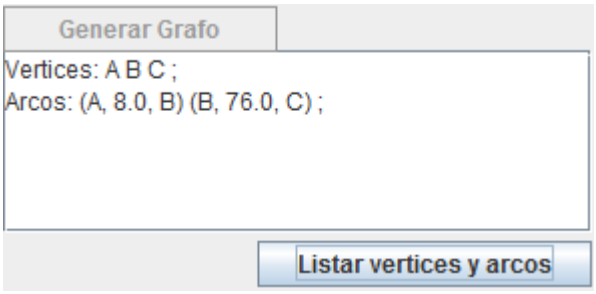

Ejemplo de cuando se ingresan vertices inexistentes.

X. Hallar camino: Al presionar este botón se abrirá una ventana en la cual se nos solicitara que ingresemos los vertices “origen” y “destino” los cuales serán los indicadores del camino que deseamos buscar. En caso de haber ingresado dos vertices existentes, los cuales estén unidos por un camino, entonces se desplegará un mensaje el cual nos indicara el camino recorrido y el coste de dicho camino, siendo este, el valor de los arcos que se recorre para realizarlo.




Siendo el camino: (verticeA, pesoArcoUnionEntreAyB, verticeB)

XI. Listar vertices y arcos: Al presionar este botón se listaran los vertices y arcos actuales en el panel de grafo.




Dadas las instrucciones de uso de cada uno de los botones se da por concluido informe sobre el proyecto.