ALGORITMO DE DIJKSTRA EN INVESTIGACIÓN OPERACIONAL

Leydy J. Pachón A.
leypachon@uan.edu.co
Msc. Tania Rodríguez Quiñones

***Resumen*—One of the fields of operational research study is the network model, which includes several algorithms to optimize the operation of a network. In this case emphasis on Dijkstra's algorithm, where its operation shown and as practical work is designed and implemented an applet for troubleshooting it is done.**

***Index Terms*—Networking, algorithm, optimize, investigation, applet.**

I. INTRODUCCIÓN

LA investigación operacional abarca diversos campos de acción, uno de ellos es el modelo de redes. Este busca optimizar el flujo de una red mediante una representacin grfica; todo problema que se pueda representar por medio de un grafo se puede tratar usando modelo de redes. Para resolver problemas de la vida real se facilita su estudio al abstraerlo en un grafo.

La aplicación de este modelo es amplia, por ejemplo para problemas de redes de transporte, negocios, salud, puntos geográficos, distribución de elementos; todo lo relacionado con flujo, donde la función generalmente se minimiza, cuya finalidad es reducir costos.

II. OBJETIVOS

II-A. OBJETIVO GENERAL

Crear una aplicaciń que contribuya en el proceso de soulución de problemas de redes de la ruta más corta utilizando el algoritmo de Dijkstra.

II-B. OBJETIVOS ESPECÍFICOS

- Aplicar los conocimientos adquiridos en la asignatura Investigación operacional.
- Modelar la solucin al problema a partir del algoritmo de Dijkstra.
- Implementar una GUI accesible que facilite la interacción usuario- software.

III. ANTECEDENTES

III-A. MODELO DE REDES

El modelo de redes se aplica para problemas de flujo en redes. A continuación se definen los conceptos principales de la teoría de redes:

- Nodo: es un punto o vértice de dónde o hacia se transporta el flujo.
- Nodo Orígen: es el nodo en donde inicia el flujo.
- Nodo destino: es el nodo a donde tiene que llegar por último el flujo.
- Arista: es la rama o conexión que hay entre dos nodos, por la cual pasa el flujo.
- Red o Grafo: conjunto de nodos y aristas los cuales están comunicados total o parcialmente.
- Grafo dirigido: es una red cuyas aristas tienen una dirección que indica hacia donde es posible el paso de flujo.
- Grafo no dirigido: la arista no indica dirección, esto quiere decir que el flujo es posible hacia cualquiera de lo dos nodos.
- Peso o Costo: es el valor que conlleva transportar de un nodo a otro, este se indica en la respectiva arista.
- Ruta: Conjunto de nodos que forman un camino.
- Ciclo: es una ruta que tiene como nodo inicial y nodo final al mismo.

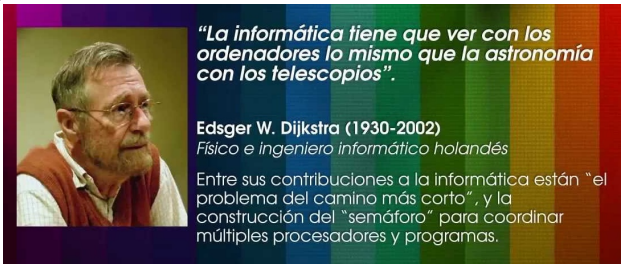
El modelo de redes presenta varios algoritmos de solución, dentro de los cuales estos son los principales:

- Árbol de mínima expansión
- Algoritmo de la ruta más corta
- Algoritmo de flujo máximo
- Algoritmo de la ruta crítica

En este caso el estudio es acerca de los algoritmos de la ruta más corta y en específico del algoritmo de Dijkstra.

III-B. ALGORITMO DE DIJKSTRA

Figura 1. El algoritmo toma este nombre gracias a su creador Edsger W. Dijkstra en 1956.



III-B1. FUNCIONAMIENTO: Este algoritmo se vale de etiquetas para encontrar la ruta más corta de un único nodo inicial a cada uno de los nodos del sistema. El grafo puede ser dirigido o no, pero si es necesario que el grafo sea ponderado.

Figura 2. Estructura de una etíqueta

[Peso acumulado , Nodo predecesor] número de iteración

PASOS DEL ALGORITMO:

- 1. Etiquetar el nodo inicial con [0 , -] porque no tiene peso acumulado ni nodo predecesor.
- 2. Proceder a etiquetar los nodos adyacentes de acuerdo a la estructura de la etíqueta.
- 3. Se escoge el nodo donde la etíqueta tenga el menor valor del peso acumulado, si hay empate se elige arbitrariamente.
- 4. Se repite el paso 2, hasta que todos los nodos tengan etíqueta.
- 5. Dado el caso de que la nueva etíqueta sea más eficiente que la anterior, lo que se hace es eliminar la no óptima.
- 5. Se puede ver que cada nodo tiene un costo de flujo que viene desde el nodo inicial, por tanto ya está evaluada la ruta desde el nodo inicial a cada uno de los otros.

IV. APLICACIONES

El algoritmo de Dijkstra se usa para conocer de forma precisa cual es la ruta óptima desde un nodo a los demás de la red, llámese red a la representación de un problema, de flujo, mediante un grafo. Por tanto las aplicaciones que este tiene son diversas.

EJEMPLOS:

- Negocios, transporte, asignación, distribución
- Encaminamiento de paquetes por los routers
- Aplicaciones para sistemas de información geográficos
- Extracción de características curvilíneas de imágenes usando técnicas de minimización del camino
- Reconocimiento de lenguaje hablado
- Enrutamiento de aviones y tráfico aéreo
- Tratamiento de imágenes médicas

CASOS REALES:

- Conocer el camino más rápido que sigue la información a través de las neuronas
- Cómo rodear una montaña por el camino más corto
- Estudios sobre la probabilidad, por ejemplo, frases más usadas por una población
- Llegar desde un punto de una vía hasta otro por la ruta más rápida

Estos son unos pocos ejemplos de la dimensión de aplicaciones del algoritmo.

V. HERRAMIENTA

Existen herramientas tecnológicas para resolver este tipo de problemas como Tora y WinQSB, sin embargo, estas no facilitan al usuario el proceso de introducción de datos, ya que lo hacen por medio de una matriz de adyacencia donde se ingresar los nodos y los costos.

A continuación se muestra la herramienta diseñada e implemntada para encontrar la ruta más corta de acuerdo al algoritmo de Dijkstra.

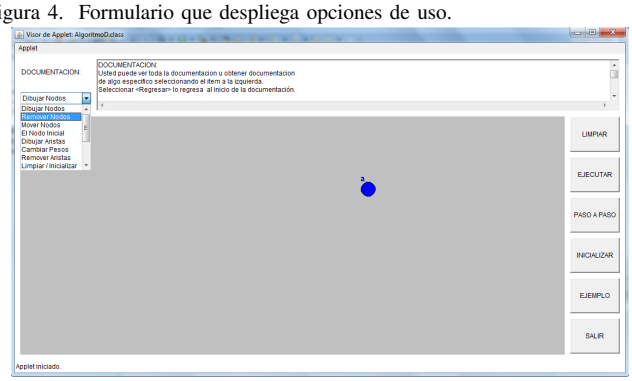
V-A. GUI (Interfaz Gráfica de Usuario)

A simple vista se detalla que la GUI es de facil manejo, como un formulario de internet donde solo hay que moverse en la pantalla y seleccionar opciones. La gran ventaja es el uso del mouse para dibujar los nodos y las aristas.

Figura 3. Interfaz gráfica de usuario.

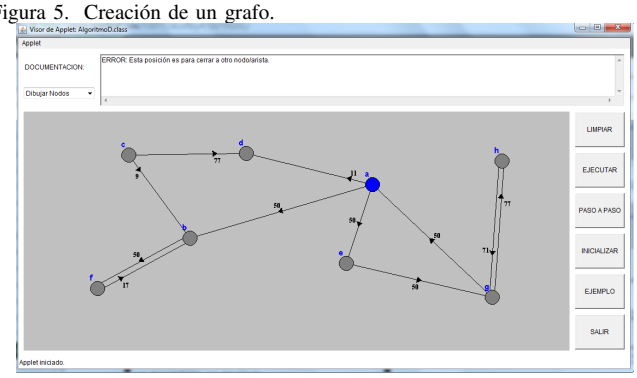
En la figura 3. se muestra la pantalla inicial, la parte gris es el espacio donde se va a dibujar el grafo, en la parte derecha están los botones de las opciones a aplicar al grafo y en la parte superior está lo referente a toda la información sobre cómo usar esta herramienta.

Figura 4. Formulario que despliega opciones de uso.



En la figura 4. se ve como se despliega el formulario donde está la documentación, es decir el manual de uso de la herramienta.

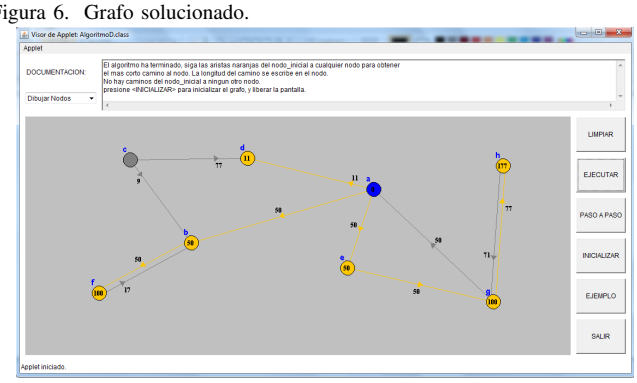
Figura 5. Creación de un grafo.



En la figura 5. se crea un grafo, haciendo click en el mouse se dibujan los nodos, el inicial siempre va a ser de color azul y los demás de color gris. Para dibujar las aristas se presiona sobre el nodo orígen y se arrastra con el mouse en dirección al nodo adyacente.

La forma del grafo se da cuando se dibuja, es recomendable no dibujarlos alineados.

Figura 6. Grafo solucionado.



En la figura 6. se muestra la solución óptima para este grafo. Los nodos de color naranja marcan la ruta óptima y el valor acumulado está en cada nodo. Los nodos que quedan en color gris es debido a que no hay una ruta del nodo inicial hasta él.

V-B. DESARROLLO

Este programa está desarrollado en lenguaje java y el editor de código es eclipse.

Dentro del código hay una clase **DocText** dedicada exclusivamente al manejo de la información detallada, lo que contituye el manual.

Se inserta en la GUI un formulario para desplegar el manual, además se adiciona una ventana donde se tiene acceso a la información sin necesidad de desplegar las opciones.

Se agregaron los botones de opción, los cuales son activados y ejecutan la respectiva acción dsescrita en el manual.

Se trabajó con seis clases : **AlgoritmoD** (es la clase principal), **DocOpciones** (maneja las opciones de la documentación), **DocText** (es la clase que contiene la información que describe como usar la herramienta), **Documentacion** (maneja toda la información, la que aparece de forma completa), **GrafoJohana** (Construye el grafo que el usuario dibuja), **Opciones** (maneja la funcionalidad de los botones de opciones).

Se elaboró una sola clase llamada **AlgoritmoD**, en esta se incluye a las demás con el propósito de evitar importar de las otras clases, también tiene como finalidad ahorrar recurso humano en la elaboración del código. Esto conlleva a que no se privaticen algunos métodos y variables.

Es posible que se refute diciendo que el código es desordenado, pero si se visualiza desde el explorador de paquetes se evidencia todo lo contrario.

Además de emplear el mouse, se hace uso de las teclas **CTRL** y **SHIFT** para eliminar, mover y cambiar nodos. Para demostrar el perfecto funcionamiento de la applet se incluye un ejemplo, el cual se expone maś adelante en la sección de **EJEMPLO**.

V-C. ALCANCES

- La herramienta está en capacidad de resolver problemas de redes con una magnitud de 20 nodos.
- Se puede eliminar nodos
- Se puede mover nodos
- Crear y eliminar aristas
- Se puede cambiar el nodo inicial
- Cambiar el costo de las aristas
- Ejecutar el ejercicio de forma directa
- Ejecutar el ejercicio de forma paso a paso
- Se puede inicializar el ejercicio
- Borrar el ejercicio
- Salir del ejercicio
- El visualizador de la applet se acomoda al tamaño de la pantalla

En la documentación está explicado como llevar a cabo cada una de estas acciones.

V-D. LIMITACIONES

- Toma la primera ruta más corta que encuentre, es decir si hay empate esta herramienta toma la primera encontrada.
- Cantidad limitada de nodos: Por ahora se colocó a la variable **final int MAXNODOS = 20.**
- Costo de las aristas: Las aristas solo se pueden asignar con un valor de 1 a 100, para esto se debe colocar las unidades en estos términos.
- Base de datos: No esta conectada a una base de datos.
- Edición: no se puede editar el nombre de los nodos

V-E. EJEMPLO

A modo de prueba se incluye un ejercicio ya resuelto en la aplicación y se muestra cuando se elige la opción EJEMPLO ubicada al lado derecho de la pantalla. Este ejemplo también se puede editar.

A medida que se realiza un procedimiemto, en la pantalla de arriba aparece el detalle de cada movimiento.

A continuación se explica el ejemplo: Se tiene una red con 10 nodos (a - j), es una red dirigida y algunos nodos tienen flujo en ambos sentidos. Los costos se ven en las aristas. El nodo inicial es el **a** y está resaltado en azul.

Figura 7. Ejercicio de ejemplo

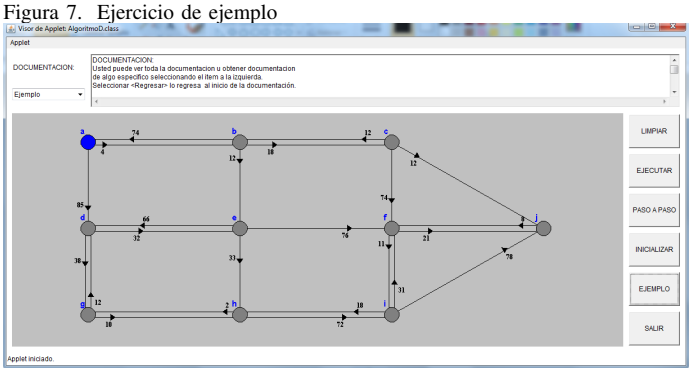
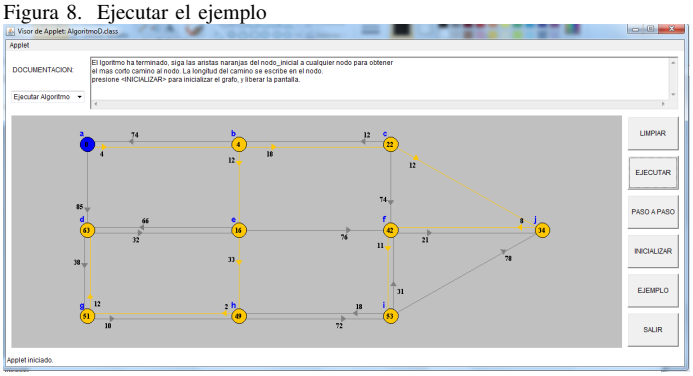


Figura 8. Ejecutar el ejemplo



Se da click en el botón de **EJEMPLO** y aparece como se ve en la figura 7.

Si se da click en el botón de **EJECUTAR** (se resuelve de forma directa, aunque tiene un retardo de 1 segundo), solo se tiene que esperar a que termine y la solución se observa como en la figura 8.

Figura 9. Ejecutar paso a paso

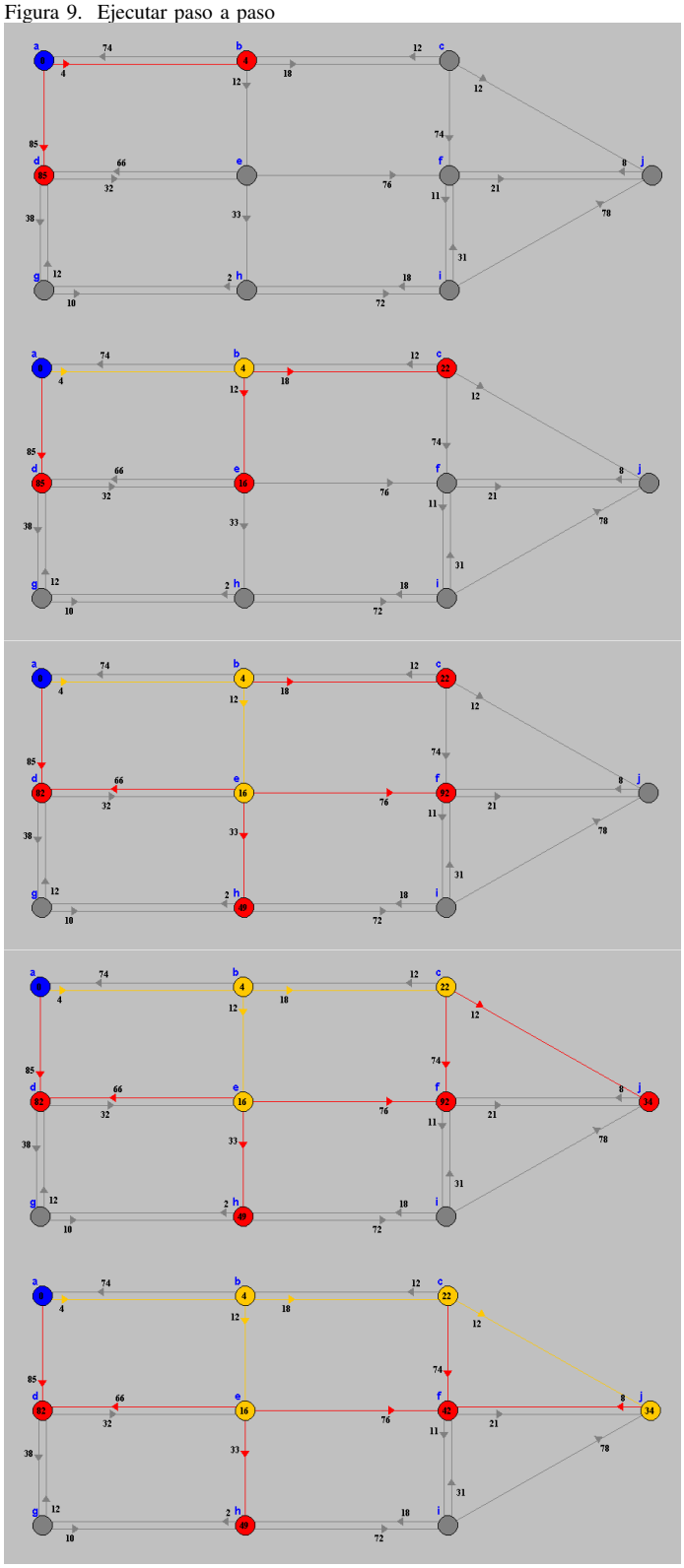
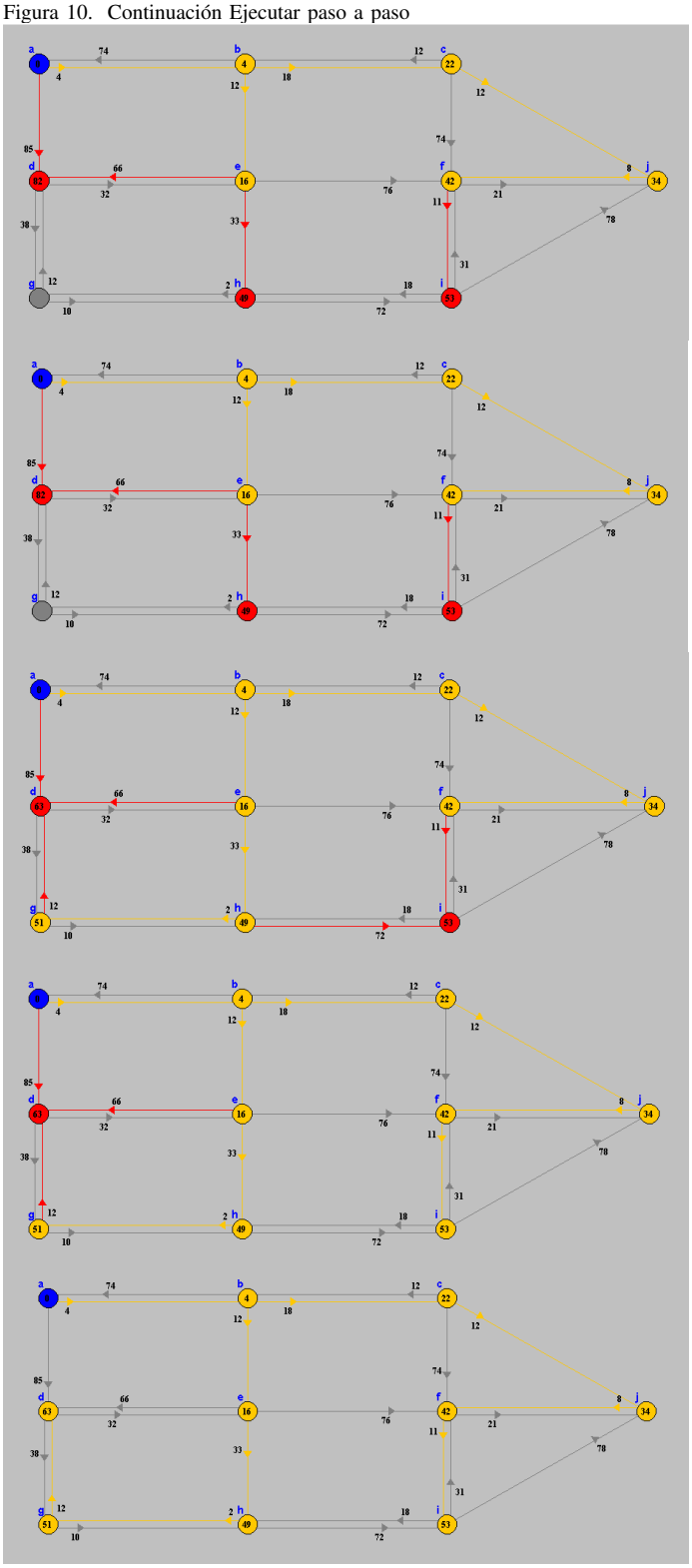


Figura 10. Continuación Ejecutar paso a paso



Las figuras 9. y 10. están conformadas por diez grafos, cada uno de ellos representa un paso del algoritmo.

El nodo de color azul es el inicial.

Los nodos de color amarillo son los que tienen las etiquetas de menor peso acumulado.

Los nodos de color rojo son los adyacentes al nodo de color amarillo en proceso de elegir el próximo nodo más cercano.

En este ejemplo no quedo ningún nodo en color gris porque el nodo **a** tiene acceso a todos.

El grafo de la figura 8. es exactamente igual al grafo de la parte final de la figura 9., esto porque es el mismo ejercicio y el resultado debe ser el mismo, si se cumple que el algoritmo está bien implementado.

Esta es la solución al problema de la ruta más corta del nodo **a** a los demás:

- a —-> b = 4
- a —-> c = 22
- a —-> d = 63
- a —-> e = 16
- a —-> f = 42
- a —-> g = 51
- a —-> h = 49
- a —-> i = 53
- a —-> j = 34

VI. CONCLUSIONES

- Comparando el algoritmo de Dijkstra con el de Floyd que también es para encontrar la ruta más corta, se sabe que el de Floyd incluye al de Dijkstra, sin embargo, el de Dijkstra requiere menos carga computacional y es el óptimo cuando el objetivo es hallar la ruta más corta de un único nodo a los demás de la red.
- El programa desarrollado cumple con los objetivos de este trabajo, además proporciona una Interfaz de usuario más accesible y de fácil manejo comparada con Tora y WinQSB, gracias a que no hay necesidad de llenar la matriz de adyacencia, lo cual a su vez agiliza el tiempo de respuesta.

VII. AGRADECIMIENTOS

Este artículo se llevó a cabo gracias a la instrucción de la ingeniera Tania Andrea Rodríguez, dada en la asignatura de Investigación operacional. Quien brindó colaboración de tipo académico y siempre estuvo disponible para resolver inquietudes acerca del tema.

Agradezco a mis compañeros de clase por su apoyo en las ocasiones que lo necesité.

REFERENCIAS

[1] Hamdy A. Taha Investigación de operaciones 9na ed. University of Arkansas, Fayetteville.

[2] Investigación de operaciones (2016, 05, 31)[En linea]. Disponible en: https://jrvargas.files.wordpress.com/2009/01/investigacic3b3n-de-operaciones-9na-edicic3b3n-hamdy-a-taha-fl.pdf

[3] Investigación de operaciones en la ciencia administrativa (2016, 05, 31), Disponible en: https://jrvargas.files.wordpress.com/2009/01/investigacic3b3n-de-operaciones-en-la-ciencia-administrativa-5ta-edicic3b3n.pdf

[4] Método void init en java (2016, 05, 31)[En linea]. Disponible en: http://userpages.umbc.edu/ emurian/learnJava/swing/tutor/v2/explanations/Explain21.html.

[5] LatexNine goole sites (2016, 05, 31). Disponible en: https://sites.google.com/site/latexnine/aplicaciones-de-dijkstra