MANUAL DE SISTEMAS

SKETCHER

SEBASTIAN CABARCAS
CRISTHYAN DE MARCHENA

Planteación del problema

El problema planteado fue que se tienen un numero N de ciudades y cada ciudad tiene un coste propio para construir un aeropuerto, los precios de los aeropuertos entre ciudades puede diferir o ser el mismo, aparte a eso cada ciudad tiene la posibilidad de conectarse a otra ciudad mediante un camino o no, si hay posibilidad de conexión entonces dicho camino tendrá un costo, todos los caminos pueden o no variar de costo; todo lo mencionado previamente es información la cual se le pide al usuario del programa. Con todo eso aclaro, se le pide al programa que halle la mejor solución o bosquejo para que todas las ciudades estén a un camino de distancia 0 o 1 de un aeropuerto con el costo mínimo. Solo se pueden usar nodos, vectores y matrices.

Introducción a la solución del problema

Ya que las estructuras disponibles a usar fueron limitadas a nodos matrices y vectores y estamos hablando del uso de un grafo para hacer la representación de las ciudades y sus conexiones entre sí, se optó por usar una ventana que tenga el entorno grafico necesario para la toma de datos por parte del usuario y otra ventana que muestre la solución del problema presentado mediante la información del usuario. Antes de hablar de dichas ventanas hablaremos de la clase del nodo, o sea la clase llamada Place que pertenece al package de Structures.

Clase Structures.Place

Esta clase nodo contiene la información de lo que vamos a denominar un “lugar”, dicho lugar tiene información propia como:

- Su nombre.
- El coste de construir un aeropuerto.
- Las posibles conexiones que puede tener con los otros lugares.
- El costo de las posibles conexiones con los otros lugares.
- Las conexiones con otros lugares que actualmente están siendo usadas.
- Su número de identificación o ID.
- Su posición en (X, Y) para ser dibujado en un “Display”.
- Su imagen haciendo referencia a un estado.
- Los diferentes estados como ser un lugar “Vacío”, “Camino” o “Aeropuerto”.

La mayoría de esta información la brinda el usuario la única que realmente se produce mediante cálculos u operaciones son las conexiones que están siendo actualmente usadas. Todas las variables de esta clase o son vectores booleanos, enteros, variables tipo cadena o de tipo imagen.

Clase Util.Radix

Como indica su nombre es una clase que hace referencia al método de ordenamiento radix, el cual se usa para hacer las cosas un más agiles. La cual posee los siguientes métodos:

- El método de radix el cual es el principal, que es el que se encarga de llamar al método de hallar el número máximo y al método que se encarga de hacer los ordenamientos por una cantidad N de dígitos dentro de un ciclo para.
- El método del máximo número que dado un vector halla el número máximo que este contenga.
- El método de ordenar un vector según la cantidad de dígitos de los números, método el cual solo toma en cuenta un numero N de dígitos para ordenar a sus números, teniendo estos un numero cuya cantidad dígitos sea menor a N.

Clase Assets.ImageLibrary

Como indica su nombre es una clase sé que se encarga de ser exactamente nada más y nada menos que una librería de imágenes que el programa tendrá a su disposición. Con un único método que se consiste en cargar las imágenes al programa.

Clase Visual.Display

Como se mencionó previamente, los nodos o lugares serán pintados o dibujados en un display el cual hace referencia a esta clase que extiendes a la clase del canvas de java (java.awt.Canvas), que a su vez implementa Runnable para poder ser ejecutada en un hilo aparte sin tener nada que ver con lo que pasa en las otras ventanas. Posee un único método el cual es “Run” y se encarga de inicializar el canvas con un fondo verde pastel y además también inicializa a un Listener con un (java.awt.event.MouseAdapter) el cual recibirá los clicks que se efectúen en este por parte del usuario. Cada click que el usuario efectué sobre este generara un evento el cual se encargara de añadir o reposicionar a un nodo con índice *C* en cuestión. Aparte a esto mientras que el Run se efectué el canvas entrara en un ciclo While(true), en el cual se pintara la información que se tenga disponible como si la ciudad tiene un aeropuerto o si solo está conectada con otra ciudad, o en el caso base que no sea ninguna de las 2; o sea un lugar vacío. Cabe destacar que cada nodo o lugar es pintado con su estado en un (X, Y), respectivo al que el usuario cliqueo. Aparte a esto todos los nodos o lugares poseen un rectángulo negro con letras blancas muy por debajo del centro que indica su nombre para diferenciarlos.

Clase Visual.InfoWindow

La clase Info Window, como su nombre indica hace una referencia a que será una ventana que extiende a la clase de la clase JFrame (javax.swing.JFrame), la cual poseerá las siguientes variables:

- Dos JLabels, las cuales harán referencia a la información que se deberá ingresar en el JTextField contiguo a cada uno de estas.
- Dos JTextFields, en los cuales se deberá ingresar información referente al nombre de la ciudad y el coste de construir un aeropuerto en esta.
- Un JTextArea, el cual será usado para mostrarle unas indicaciones especificas al usuario.
- Un JTable, en el cual se podrá modificar información referente a una ciudad como lo es el coste de su aeropuerto y el valor de sus conexiones con las otras ciudades; aparte a eso todas las celdas que pertenezcan o estén por encima a la diagonal formada por la esquina inferior derecha y la esquina superior derecha tendrá un acceso bloqueado a lo mismo que el nombre de las ciudades que han sido añadidas.
- Tres JButtons, los cuales cumples funciones que indican sus textos como agregar una ciudad a la tabla en el caso del botón con el texto “Agregar”, remover una fila seleccionada en la tabla en el caso del botón con el texto “Remover” y por ultimo solucionar el problema que actualmente este en la tabla para el caso del botón con el texto “Sketch”.

Aparte a esto esta clase tiene cuatro métodos a destacar los cuales serían:

- Agregar una ciudad a la tabla, el cual hace exactamente lo que dice.
- Remover una ciudad de la tabla, la cual funciona seleccionando una fila y según esa fila se removerá la información correspondiente a ese lugar en la tabla.
- La función Sketch o bosquejar que se encargara de solucionar el problema del usuario con la información que poseía la tabla en ese momento.
- La función createLandscape, la cual se encarga de rellenar la tabla con los valores restantes y luego pasarlos a una matriz de tipo entero para finalmente crear un vector de tipo Place con la información necesaria el cual al estar listo será copiado por la clase Visual.VisualWindow para llevar a cabo todos los procesos que necesite y finalmente regresar la tabla a su estado original. Cabe destacar que esta función es llamada al final de las funciones de Agregar y Remover, y también es llamada al principio de la función Sktech, para evitar posibles errores.

Clase Visual.VisualWindow

La clase Visual Window, posee las siguientes variables:

- El Display donde muestra a los nodos.
- El vector de tipo Place para llevar a cabo los pasos de la solución al problema.
- Los vectores de tipo entero como lo son los aeropuertos (en el cual los aeropuertos están ordenados de menor a mayor), las ciudades (en el cual hace referencia a las ciudades del vector de aeropuertos) y las posibilidades el cual es un vector especial al que se le da mucho uso para recargar las posibilidades de acciones durante todo el proceso de la solución del problema, básicamente es un vector que tiene el costo más óptimo haciendo referencia al vector de acciones.
- Un vector tipo String llamado acciones el cual hace referencia a la acción más óptima tomando como referencia el precio de esta el cual es perteneciente al vector de posibilidades (estos 2 vectores están ordenados a la par para mantener la información estable).
- Dos variables tipo entero que se llaman airportIndex el cual hace referencia al índice del próximo aeropuerto que se debe de poner siguiendo al vector de las ciudades, y totalCost el cual vendría siendo el resultado de toda la operación, o sea el costo total de solucionar el problema planteado por el usuario.
- Una variable de tipo booleano que indica cuando una ciudad en específico no tienes más acciones posibles, esencial para hacer la diferencia entre que un paso A ocurra o no y a su vez ocurra o no un pasó B.

Ahora los métodos que destacarían:

- El Paso #0 o set up, el cual se encarga dar valores a los vectores aeropuertos y ciudades, y a su vez se encarga de ordenar los aeropuertos y ciudades por igual para tener toda la información lista y ordenada para los pasos contiguos.
- checkAllAirports_ReturnBestOption, el cual se encarga de encontrar el primer mejor aeropuerto que se pueda construir para llevar a cabo los otros pasos.
- Paso #1, el cual se encarga de según las ciudades con aeropuertos crear una lista de acciones y posibilidades las cuales se crean con pares de posibilidades y acciones que se van filtrando hasta solo quedar las mejores del par, una vez estos vectores son llenado, se ordenan para tenerlos listos para el paso #2.

- Paso #2, el cual se encarga de escoger la primera acción del vector de acciones si es que dicho vector no es nulo o tiene tamaño mayor a cero, aparte a eso se encarga de decir si hizo o no alguna acción o algo a lugar retornando un booleano de verdadero si efectuó algún cambio y de falso si no; hay que tener en cuenta de que si este caso retorna falso es porque o bien no se pudo hacer ninguna acción o bien ya todas las acciones que están disponibles han sido efectuadas. Razón por la cual hay una variable booleana que toma esto en cuenta.
- Paso #3, si el paso #2 retorno falso y la variable booleana de esta clase es falsa, este caso se lleva acabo. Este paso se encarga de realizar una función similar al Paso #1, solo que las posibilidades y acciones que se toman en cuenta son distintas.
- Paso #4, su funcionamiento es similar al de Paso #2 solo que la diferencia en el vector de acciones da cabida para más variedad de acciones disponibles dentro de una acción, el cual también retorna verdadero si alguna acción se llevó a cabo o falso si no.
- Paso #5, un paso que raras veces se ejecuta ya que casi nunca se dan las condiciones para que el Paso #4 retorne falos ya que este paso solo se ejecuta dada esa situación, este paso se encarga de revisar cuál de las ciudades no tiene posibilidad de conectarse con nadie más para construirle un aeropuerto encima.
- Paso #6 o paso de chequeo ya que solo se ejecuta una vez y es un paso que se encarga de ver de que todo esté en su lugar y si no, lo corrige mediante los pasos #3, #4 y #5.
- buildLandscape o Solución del problema, el cual es un ciclo while que se dura lo que se tarde en recorrerse el vector de los lugares o nodos y durante ese lapso, alterna en todos los métodos descritos previamente para llegar a la solución ideal, cuando ya llega al tamaño del vector de los lugares o nodos, es que se lleva a cabo el paso #6, dando así una respuesta contundente.

Para más información leer el código que esta documentado…