Universidad de San Carlos de Guatemala
Centro Universitario de Occidente División de Ciencias de la Ingeniería
Estructura de Datos
Aux. Yefer Rodrigo Alvarado Tzul
Sección: A

Manual Técnico

Carlos Raúl Alberto López Peláez 202031871

Quetzaltenango,5 de Mayo del 2024

Clase Arbol

Atributos:

- NodoArbolB raiz: Representa el nodo raíz del árbol B.
- int nClaves: Define el número máximo de claves que puede contener un nodo en el árbol B.

Constructor:

- public ArbolB(int t): Constructor de la clase que inicializa un árbol B con un parámetro t, que indica el grado mínimo del árbol.

Métodos públicos:

1. buscarClaveMayor(): Busca la clave mayor en el árbol B.
2. mostrarClavesNodoMinimo(): Muestra las claves del nodo mínimo en el árbol B.
3. buscarNodoPorClave(int num): Busca un nodo en el árbol B que contenga la clave especificada.
4. insertar(int key,List<NodoGrafo> nodografo): Inserta una nueva clave en el árbol B.
5. showBTree(): Muestra el contenido del árbol B.

Métodos privados:

1. getClaveMayor(NodoArbolB actual): Obtiene la clave mayor de un nodo dado.
2. claveMayorPorNodo(NodoArbolB current): Obtiene la clave mayor de un nodo hoja.
3. buscarNodoMinimo(NodoArbolB nodoActual): Busca el nodo mínimo en el árbol B.
4. search(NodoArbolB actual, int key): Busca una clave en el árbol B.
5. split(NodoArbolB x, int i, NodoArbolB y): Divide un nodo en dos cuando está lleno durante una inserción.
6. nonFullInsert(NodoArbolB x, int key, List<NodoGrafo> nodoGrafo): Inserta una clave en un nodo que no está lleno durante una inserción.
7. print(NodoArbolB n): Imprime el contenido de un nodo y sus hijos recursivamente.

Clase Nodo Árbol

Atributos:

- int clavesEnNodo: Número de claves almacenadas en el nodo.
- boolean hoja: Indica si el nodo es una hoja o no.
- int[] key: Arreglo de claves enteras almacenadas en el nodo.
- Key[] keys: Arreglo de objetos Key asociados a las claves del nodo.
- NodoArbolB[] hijos: Arreglo de nodos hijos del nodo actual.

Constructor:

- public NodoArbolB(int t): Constructor que inicializa un nodo del árbol B con un parámetro t, que indica el grado mínimo del árbol.

Métodos públicos:

1. imprimir(): Imprime las claves y sus nodos grafo asociados en el nodo.
2. find(int k): Busca una clave en el nodo y devuelve su índice si se encuentra, de lo contrario devuelve -1.
3. Métodos getter y setter para acceder y modificar los atributos del nodo.

Clase Cargar Datos

Atributos:

- List<Grafo> listadoGrados: Lista que almacena los grafos creados a partir de los datos leídos del archivo CSV.

Constructor:

- public CargarDatos(): Constructor por defecto.

Método leerCSVDatos(File Valor):

- Este método recibe un archivo CSV como parámetro y procesa su contenido para crear grafos.
- Lee cada línea del archivo y divide los datos utilizando el separador |.
- Crea un objeto Grafo para cada origen encontrado en el archivo.
- Para cada origen, verifica si ya existe un grafo en el listadoGrados. Si existe, agrega un nodo a ese grafo. Si no existe, agrega un nuevo grafo al listadoGrados y luego agrega un nodo a ese grafo.
- Verifica si el destino ya existe en los nodos del grafo. Si existe, actualiza los datos del nodo. Si no existe, crea un nuevo nodo y lo agrega al grafo.
- Retorna la lista de grafos creados.

Método leerCSVTrafico(File Valor):

- Este método recibe un archivo CSV que contiene datos de tráfico y actualiza los nodos de los grafos en el listadoGrados.
- Lee cada línea del archivo y divide los datos utilizando el separador |.
- Busca los nodos correspondientes en los grafos almacenados en el listadoGrados y actualiza los datos de tráfico en esos nodos.
- No retorna ningún valor.

Método mostrarGrafos(Grafo grafo):

- Este método muestra los nodos de un grafo recursivamente.
- Imprime los datos de cada nodo del grafo y luego verifica si el nodo tiene un subgrafo y lo muestra también.

Método agregarSubgrafos(Grafo grafo, String[] datos):

- Este método agrega subgrafos a un grafo existente si los datos coinciden con algún nodo del grafo.

- Verifica si el origen de los datos coincide con algún origen de los nodos del grafo. Si coincide, agrega un nuevo nodo al grafo como subgrafo.
- No retorna ningún valor.

Método agregarEnSubgrafos(Grafo grafo, String[] datos, NodoGrafo nodoGrafox):

- Este método agrega subgrafos a un grafo existente si los datos coinciden con algún nodo del grafo, pero también actualiza el nodo existente con el nuevo nodo proporcionado.
- Verifica si el origen de los datos coincide con algún origen de los nodos del grafo. Si coincide, agrega el nuevo nodo como subgrafo y actualiza el nodo existente con el nuevo nodo proporcionado.
- No retorna ningún valor.

Método argegarNodo():

- Este método está incompleto y no tiene implementación.

Método main(String[] args):

- Este método es el punto de entrada del programa.
- Crea un objeto File con la ruta del archivo CSV.
- Llama al método leerCSVDatos(File Valor) para leer los datos del archivo CSV y crear grafos.
- Llama al método Graficar.Graficar.generarGraficosListado(grafos) para generar gráficos basados en los datos leídos.

Clase Graficar

Método generarGraficos(Grafo grafo):

- Este método genera un gráfico a partir de un solo grafo.
- Crea una lista de gráficos listadoDeGRafos.
- Llama al método graficar para generar el gráfico del grafo pasado como argumento y agregarlo a listadoDeGRafos.
- Crea un gráfico final combinando todos los gráficos de listadoDeGRafos.
- Renderiza el gráfico final como una imagen PNG y lo guarda en un archivo.

Método generarGraficosListado(List<Grafo> grafos):

- Este método genera gráficos para una lista de grafos.
- Crea una lista de gráficos listadoDeGRafos.
- Por cada grafo en la lista grafos, llama al método graficar para generar el gráfico y agregarlo a listadoDeGRafos.
- Combina todos los gráficos de listadoDeGRafos en un gráfico final.
- Renderiza el gráfico final como una imagen PNG y lo guarda en un archivo.

Método graficar(List<Graph> listadoDeGrafos, Grafo grafo, String nodoPadre):

- Este método genera gráficos recursivamente para un grafo y sus subgrafos.
- Para cada nodo en el grafo, si tiene un subgrafo, crea un nuevo gráfico para ese subgrafo y lo agrega a listadoDeGrafos.
- Llama recursivamente a graficar para generar gráficos para los subgrafos del nodo.

Método createGraph(String name, String node1, String node2):

- Este método crea un gráfico simple que representa una conexión entre dos nodos.
- El nombre del gráfico es dado por name.
- El nodo node1 está conectado al nodo node2.
- Configura el estilo y atributos del gráfico.

Método main(String[] args):

- Punto de entrada del programa. Está vacío ya que no se ha implementado ninguna funcionalidad en el método main.

Clase Grafo

Atributos:

- valor: Una cadena que representa el valor asociado al grafo.
- nodos: Una lista de nodos del tipo NodoGrafo que pertenecen al grafo.

Constructores:

1. Constructor sin argumentos: No realiza ninguna operación especial.
2. Constructor con parámetros: Inicializa el valor del grafo y la lista de nodos con los valores proporcionados.

Métodos:

1. imprimirNodos(): Imprime el valor del grafo y los destinos de todos los nodos pertenecientes al grafo.
2. getValor(): Devuelve el valor asociado al grafo.
3. setValor(String valor): Establece el valor asociado al grafo.
4. getNodos(): Devuelve la lista de nodos del grafo.
5. setNodo(NodoGrafo nodo): Agrega un nodo a la lista de nodos del grafo.
6. setNodos(List<NodoGrafo> nodos): Establece la lista de nodos del grafo.

Clase NodoGrafo

Atributos:

- Origen: La ubicación de origen del nodo.
- Destino: La ubicación de destino del nodo.
- tiempo_promedio_vehiculo: Tiempo promedio en vehículo para llegar al destino.
- tiempo_aproximado_a_pie: Tiempo aproximado a pie para llegar al destino.
- gasto_gasolna: Gasto de gasolina para llegar al destino.
- gasto_fisico: Gasto físico para llegar al destino.
- distancia: Distancia al destino.
- horaInicial: Hora inicial de referencia.
- horaFinal: Hora final de referencia.
- Color: Color del nodo.
- puntuacio: Puntuación del nodo.
- probabilidadTrafico: Probabilidad de tráfico en el nodo.
- nodo: Grafo asociado al nodo.

Constructores:

1. Constructor sin argumentos: Inicializa algunos atributos con valores predeterminados.
2. Constructor con parámetros: Inicializa todos los atributos con los valores proporcionados.

Métodos:

1. Métodos get y set para todos los atributos, para acceder y modificar sus valores.
2. toString(): Devuelve una cadena que representa el estado actual del nodo.
3. destinos(): Devuelve una cadena con el destino del nodo.

Clase UI funciones principales

1. Carga de Datos:

   CargarDatos.leerCSVDatos(selectedFile): Esta función lee un archivo CSV que contiene datos sobre los grafos (nodos, distancias, etc.). Es crucial explicar el formato esperado del archivo CSV y los datos que se deben incluir.

   CargarDatos.leerCSVTrafico(selectedFile): Similar a la anterior, pero leerdatos de tráfico. Se necesita aclarar qué información de tráfico se espera y cómo se utiliza.

2. Gestión de Grafos:
   enlistarDestinos(selectedValue): Esta función parece poblar el jComboBox2 con los destinos posibles desde un origen seleccionado.

   graficarNodos(grafos, grafo): Construye una lista de grafos conectados a un grafo inicial.

   siguienteNodo(nodo) y nodoFinal(nodo): Estas funciones recorre los nodos de un grafo.

   enlistar(listadoDeGrafos, nodoGrafos , grafo, origen, destino): Es una función recursiva que busca caminos entre un origen y un destino.

3. Cálculos:
   Calculo.calcularRapidezCaminando(distancia, tiempoAPie): Calcula el acumula de la rapidez de una ruta a pie.
4.
   Calculo.calcularRapidezVehiculo(distancia, tiempoVehiculo, probabilidadTrafico): Calcula el acumula de la rapidez de una ruta en auto.