**Informe AyED final 2016 Router**

**Integrantes:**

* Perez, Esteban Andres 39026980
* Malano, Leando Julian 38883701
* Orecchini Alem, Stefano Mauricio 36431237

**Clase NodoHash (<typename keyT, typename datoT>):**

Cada uno contiene una key, un value o dato y una direccion de memoria hacia otro NodoHash

**NOTA:** El método imprimir dara una dirección de memoria en caso de que key y dato sean objetos distintos de un tipo primitivo.

**Clase HashMap (<typename keyT, typename datoT>):**

Esta clase nos permite generar un objeto del tipo Hash map el cual tiene un key y un value. Para ello necesitamos un NodoHash el cual amacena un key y un value. El método más importante aca es el método **add** el cual crea un nuevo NodoHash con un determinado key y un determinado value, luego este nodo, apunta hacia la cabeza y la cabeza hacia el nuevo, haciendo que esto funcione como una estructura de pila.

**Recordar:** que en un principio el nodo cabeza apunta hacia un NULL.

**Clase NodoGrafo:**

Cada grafo puede apuntar a una cantidad determinada de nodos y con un determinado peso, para ello se hace un HashMap con direcciones de memoria a objetos de tipo NodoGrafo (seria nuestra key) y el value seria el peso que tiene esta coneccion (del tipo **int**). Para el algoritmo de dijkstra tuvimos que implementar un **bool** el cual nos indica si este nodo fue visitado o no.

**Nodo desde\* -> Comienzo**

**Nodo hasta\* -> Final**

**costoTotal -> Peso de las aristas**

**void** **inicializarColaDePaquetesDePaginas**()

Crea un hashmap que contiene un string y una cola de paquetes.

A ColaDePaquetesDePaginas le agrega un nodoGrafoPadre y el hashmap creado anteriormente.

**void** **setRutasRecomputadas**(HashMap<**int**, NodoGrafo\* >\* routRecomps)

Se agrega a rutasRecomputadas una llave y un value por índice.

**void** **addCola**(NodoGrafo\* n, Cola<Paquete\*>\* col){colas->add(n,col);}

**siendo “n” un NodoGrafo y “col” una cola de paquetes.**

**void** **enviarPaquetes**()

**Envia de a un paquete por vez a cada nodo.**

1. **Se obtiene el paquete de la cola. (esta en la cabeza de HM)**
2. **Se agrega el paquete encontrado a una nueva cola de espera.**
3. **Se elimina la cabeza de la cola. (el paquete ya fue enviado)**

**void** **recibirPaquetes**()

mientras hay paquetes en la cola de espera, obtengo el nombre del router de destino (numero entero).

Ademas, si el router de origen es diferente al router de destino, asignamos la cola de paquetes al router al que se dirigen.

Si esto no se cumple llamo a reacomodarPaquetesParaLasPCS()

Finalmente notifica si llegaron todos los paquetes.

**void** **reacomodarPaquetesParaLasPCS**()

Si los paquetes llegan a destino, agrego los paquetes a un HM y los identifico con un prefijo.

**void** **generarPaquetes**()

Se generan paquetes que contienen un dato de tipo char, router origen y destino, PC origen y destino, como dirección de memoria, un índice, un peso, y por ultimo una pagina a la que pertenecen.

Estos paquetes se introducen a una cola de espera.

Los paquetes son impresos.

Pagina\* **reconstruirPagina**(Cola<Paquete\*>\* colaDePaquetes)

Las paginas son similares a los paquetes, a diferencia de que estas contienen “mensajes” que son constituidos por los caracteres presentes en los mismos.

**Clase RutaD:**

Significa ruta dijkstra y se le asigno este nombre ya que se usa en el algoritmo de Dijkstra para calcular el peso o costo existente desde un nodo hacia otro. La característica de esta clase es que tiene 2 direcciones de memora del tipo Nodo Grafo una que se llama desde y otra que se llama hasta y por ultimo una valiable del tipo **int** que indica el costo total para viajar desde el nodo “desde” hasta el nodo “hasta”.

**Clase Grafo:**

int matrizAdyacencia[256][256]; es la matriz d adyacencia que se carga desde un archivo y nos dice que nodo se conecta con cual y con un peso de cuanto

HashMap<int,NodoGrafo\*>\* nodos; para acceder a los nodos de mi grafo los busco por su nombre en la key que en este caso es un entero… si el nodo se llama por ejemplo: int nombre=3; en el hash map va a ir como key el 3 y como value la dirección de memoria hacia este nodo con este nombre.

**void crearGrafo(){crearNodos();}**

En un principio leemos el archivo y sabemos cuántos nodos hay en la matriz, estos nodos los creamos y los metemos dentro del Hash y la cantidad a crear son “n” nodos siendo “n” el tamaño de nuestra matriz cuadrada leída desde el archivo de entrada. Quien crea los nodos es el método **void crearNodos()**  que al mismo tiempo llama al método **void crearAristasYEnlazar()**

Luego el método **void crearAristasYEnlazar()** lee la matriz de adyacencia y en función de los valores de los índices y el valor leído en la matriz de adyacencia enlazamos cada nodo a los otros nodos correspondientes con el peso leído en dicha matriz.

Dentro de la clase Grafo se encuentra el método Dijkstra que usa varios métodos mas que detallaremos a continuación

**Lista <RutaD\*>\* dijkstra(int numeroDeNodo,int numeroDeNodoFinal)**

Este método me devuelve el camino más corto desde un nodo hasta otro

Primero que nada me paro en un nodo el cual va a ser el nodo **actual=desde** siendo desde **NodoGrafo\* desde=getNodos()->get(numeroDeNodo);**

mi primer ruta de tipo Dijkstra va a ser la siguiente: **RutaD\* ruta= new RutaD(desde,desde,0);** y será agregada a la lista de rutas de tipo RutaD : **rutas->add(ruta);**

**calcularRutasD(actual, rutas,base);**

calcula las RutasD pero… lo hace de la siguiente forma… me paro en un nodo determinado, me fijo hacia que nodos esta mirando, calcula la distancia que es la base mas la distancia existente entre el mismo y los otros nodos luego se ejecuta….

**eliminarRepetidos(rutas);**

lo que hace este método es eliminar los elementos repetidos dentro de la variable **rutas** que contiene objetos del tiro RutaD pero.. solo elimina los elementos de mayor peso total por ejemplo si para llegar al nodo 2 exiten dos elementos repetidos 2-5, 2-9, se eliminarla el elemnto 2-9, si tuviesen el mismo peso se eliminaría cualqueira de los dos.

**Clase Paquetes y Pagina:**

Los Paquetes contienen caracteres que luego son puestos en una cola de Pagina para crear el mensaje compuesto por ellos.

**NodoGrafoPC:**

Apunta a un determinado Router y a una Pagina. Para identificarlo usamos un numero entero único para cada uno.

**void** **crearPagina**(**int** peso,NodoGrafoPC\* NGPCDestino)

Crea una nueva pagina con su propia cola de caracteres. Actualiza un contador de paginas generadas.