PONTIFICIA UNIVERSIDAD JAVERIANA

ESTRUCTURAS DE DATOS

ANDREA DEL PILAR RUEDA OLARTE

INTEGRANTES

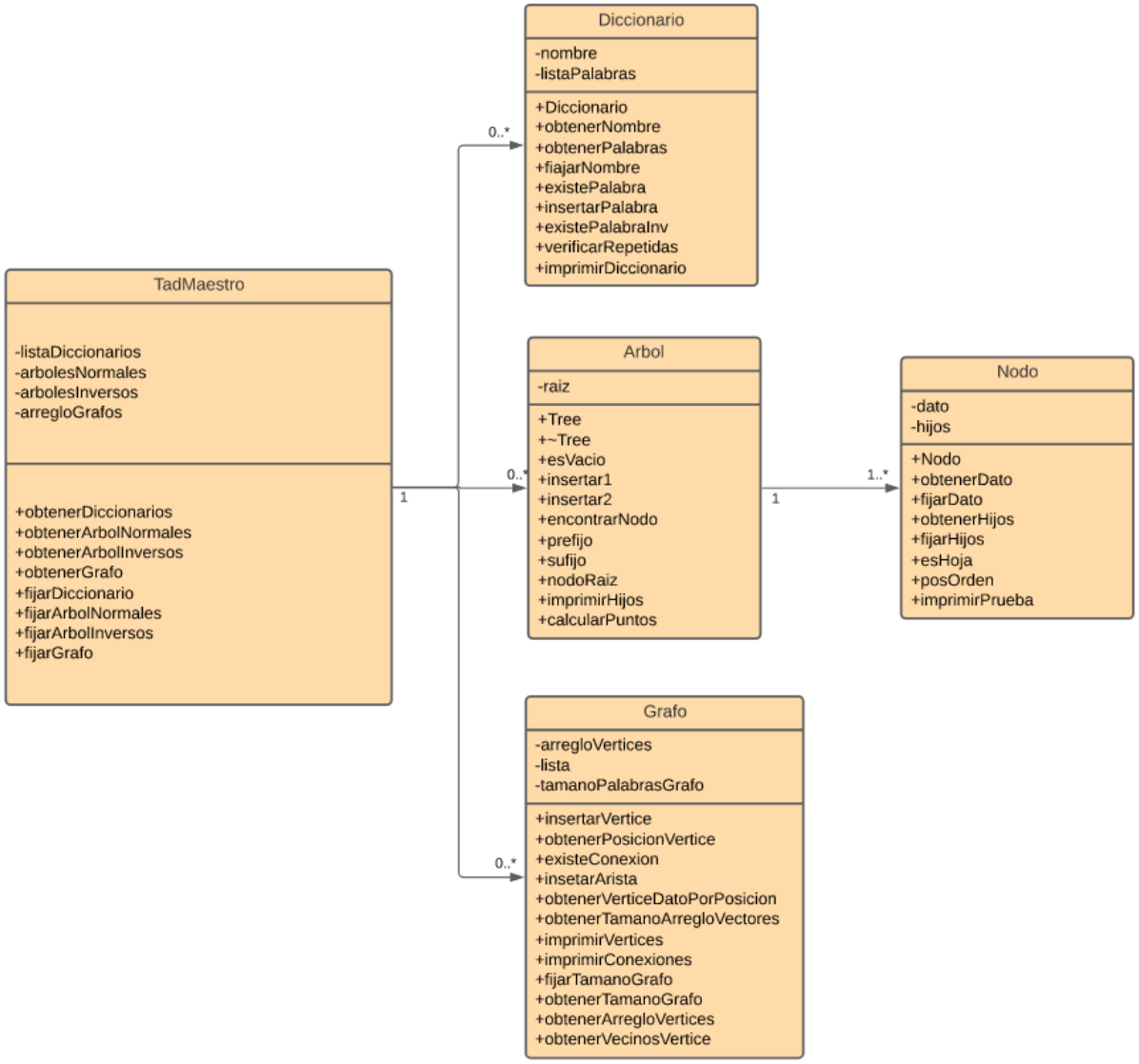
DAVID ALEJANDRO ANTOLÍNEZ SOCHA

DIEGO ALEJANDRO CARDOZO ROJAS

BRAYAN ESTIBEN GIRALDO LOPEZ

4 de Junio de 2021

DIAGRAMA DE RELACIÓN



DESCRIPCIÓN FUNCIONES

TAD diccionario

Datos mínimos:

- Nombre: cadena de caracteres, identifica el nombre del archivo diccionario que se abrió.
- Lista palabras: lista de una cadena de caracteres, almacena las palabras admitidas en el archivo diccionario.

Operaciones:

- FijarNombre(nombreP): reemplaza el nombre por la cadena nombreP.
- obtenerNombre(): retorna la cadena de caracteres almacenada en la variable nombre.

- insertarPalabra(palabraP), inserta una cadena de caracteres extraída del archivo diccionario en la lista palabras.
- obtenerPalabras(), retorna la variable lista palabras.
- imprimirDiccionario(), recorre la variable lista palabras e imprime los datos almacenados en esta.
- existePalabraInv(palabra), invierte la cadena de caracteres palabra, después de esto recorre la variable lista palabras comparando los datos almacenados en esta con la variable palabra si se encuentra un dato igual retorna un booleano.
- existePalabra(palabra), recorre la variable lista palabras comparando los datos almacenados en esta con la cadena de caracteres palabra si se encuentra un dato igual retorna un booleano.
- verificarRepetidos, función vacía que recorre el vector que almacena las palabras del archivo que se lee y las ordena de mayor a menor, una vez ordenado el arreglo se comprueba que solo exista una palabra y en caso contrario se eliminan las que están repetidas.

TAD árbol

Datos mínimos:

- Raíz, variable tipo nodo donde se guarda la raíz del árbol.

Operaciones:

- Tree, método constructor del tad árbol.
- ~Tree, método de eliminación del tad árbol.
- esVacio, función que retorna una variable booleana en caso de que la raíz sea nula.
- insertar1(lineaInsertar), funcion que retorna un booleano, esta función inicializa la raíz la cual será un carácter, este carácter inicial será un espacio, en caso de que la raíz ya fuera iniciada la función llama a la función insertar 2 y se le envía el nodo raíz junto con una cadena de caracteres que contiene la palabra que se desea agregar.
- insertar2(lineaInsertar,Nodo), función que es recurrente y asigna hijo por hijo, la función recibe una cadena de carácter con la palabra a ingresar, la cadena se recorre y se van ingresando sus datos, a medida que se ingresan los caracteres en la cadena se van eliminando hasta no quedar nada, en caso de que se vaya a ingresar un carácter ya existente este se borra directamente y no se agrega al arbol
- encontrarNodo(copiaPrefijo,nodo), función que es recurrente, recibe el prefijo y la raíz, la función busca en los hijos del nodo que se envía, un carácter que

corresponda al carácter en la primera posición de la cadena una vez encontrado se elimina la primera letra de la cadena y se crea un nuevo nodo con los valores que se encontraron y se envía nuevamente a la función encontrar nodo, una vez que llega al final retorna el último nodo.

- imprimirHijos(nod,prefijo,nombreDeArchivo),la función recibe un nodo el prefijo y el nombre del archivo función que lee el archivo para verificar cuántos nodos son necesarios para una palabra una vez obtenido este dato se calcula el puntaje de la palabra y se comienzan a recorrer los hijos del nodo para imprimir cada uno de los caracteres de la palabra.
- prefijo(prefijo,nombreArchivo), función que recibe dos cadenas de carácter,una con el prefijo y otra con el nombre del archivo, la función busca un nodo que contenga el carácter del prefijo para una vez hecho esto enviar los hijos e imprimir todas las palabras que se puedan hacer con este prefijo.
- sufijo,(sufijo, nombreArchivo), función que realiza el mismo proceso que el prefijo pero es aplicada en el árbol invertido pero usando el sufijo.
- nodoRaiz, función que retorna una variable de tipo nodo, la función retorna la raíz del árbol que se declaró
- calcularPuntos(letra),funcion que retorna un entero, recibe un carácter que al igual que en la función de puntos en el main compara este carácter y una vez lo encuentra lo retorna con su valor.

TAD nodo

Datos mínimos:

- Dato, variable tipo T, se almacena un carácter el cual corresponde a cada letra de la palabra que se ingrese del archivo diccionario.
- Hijos, variable de tipo vector, se almacena los nodos hijos <caracteres siguiente de una palabra>

Operaciones:

- Nodo, método constructor del tad nodo.
- NodoConDatos, método constructor del tad nodo que recibe un valor del dato que se va a ingresar en la estructura.
- ObtenerDato, funcion que retorna un dato tipo T que se almaceno en la variable dato del tad nodo.
- fijarDato(valor), función que recibe una variable de tipo T y la asigna a la variable dato.
- fijarHijos(hijos), función que recibe una variable de tipo vector la cual contiene todos los nodos que van a ser hijos y se asigna en la variable hijos del tad nodo
- esHoja, esta función retorna un booleano el cual indica si el nodo que se está iterando tiene hijos o no.

TAD Grafo

Datos mínimos:

- ArregloVertices, variable de tipo vector que almacena los diferentes vertices que va a tener el grafo.
- Lista, variable de tipo lista de listas, en esta variable se va a almacenar las conecciones que tiene cada uno de los vértices del grafo.
- TamanoPalabrasGrafo, variable de tipo entero en la cual se almacena el tamaño que tiene cada palabra que se va a almacenar el el grafo.

Operaciones:

- insertarVertice: función que inserta un nuevo vértice en el arreglo de vértices del grafo
- obtenerPosicionVertice: dado un vértice esta función retorna la posición dentro del arreglo de vertices
- existeConexion: variable que retorna un 1 en caso de que exista una conexión entre dos vertices del grafo, retorna -1 si esta no existe
- insetarArista: función que inserta una nueva arista en el arreglo de las conexiones
- obtenerVerticeDatoPorPosicion: dada una posición del arreglo de vértices esta función retorna el vertice de esa posicion
- obtenerTamanoArregloVectores: función que devuelve el tamaño del arreglo de vertices
- imprimirVertices: función que imprime los vértices de un grafo
- imprimirConexiones: función que imprime las conexiones de un grafo
- -fijarTamanoGrafo: función que fija el tamaño de las palabras que están dentó del grafo
- obtenerTamanoGrafo: función que retorna el tamaño de la longitud de las palabras que se almacenan en el grafo
- obtenerArregloVertices: función que retorna el arreglo de vértices
- obtenerVecinosVertice: retorna un arreglo con las palabras vecinas de un vértice

TAD Maestro

Datos mínimos:

- listaDiccionarios: arreglo que contiene la lista de diccionarios
- arbolesNormales: arreglo que contiene la lista de árboles
- arbolesInversos: arreglo que contiene la lista de árboles inversos
- arregloGrafos: arreglo que contiene la lista de grafos

Operaciones:

- obtenerDiccionarios: retorna el arreglo de los diccionarios

- obtenerArbolNormales: retorna el arreglo de los árboles
- obtenerArbolInversos: retorna el arreglo de los árboles inversos
- obtenerGrafo: retorna el arreglo de los grafos
- fijarDiccionario: asigna el arreglo de los diccionarios
- fijarArbolNormales: asigna el arreglo de los árboles
- fijarArbolInversos: asigna el arreglo de los árboles inversos
- fijarGrafo: asigna el arreglo de los grafos

Acta de evaluación

Corrección	Solución
El acta de evaluación debe incluir mis comentarios textuales y para cada uno una respuesta de parte de ustedes, así me queda más fácil recordar y revisar lo que había que ajustar.	Esta misma tabla con los comentarios realizados y su respectiva respuesta
El diseño es correcto pero quería sugerir que analizaran la necesidad de tener un TAD superior que pueda agrupar tanto al diccionario cómo al árbol, pensando también que en la tercera entrega debemos agregar el grafo y seguramente en su caso quedaría de nuevo como un TAD aislado de los otros.	En el código se implementó el TadMaestro el cual ahora almacena la lista de diccionarios, árboles y grafos que usa el programa principal
Comando palabras_por_prefijo: El comando presenta error de ejecución al momento de calcular el puntaje en algún sistema operativo, así que es importante revisar qué puede estar pasando.	Se encontró un error de iteración el la función puntos() del main se corrigió y corrió el programa en fedora para asegurarse de su correcta ejecución
Comando palabras_por_sufijo: Así como el comando anterior, genera problemas de acceso a memoria al momento de calcular el puntaje en algún sistema operativo, por lo que es importante revisar.	Misma corrección del punto anterior

El sufijo debe ingresarse al derecho, y las palabras deben mostrarse al usuario al derecho, de ser posible ordenado de forma que para el usuario sea más fácil revisarlas.	Se implementó una función que da vuelta al sufijo, lo cual permite que el usuario pueda ingresar el sufijo de manera correcta y el programa sea capaz de interpretarlo de la misma manera