Informe del las funciones TDA Lista

* estaEnLista(self, elemento): funcion que retorna si el elemento pasado por parametro se encuentra en la lista

  -primero ser crea un variable para guardar el resultado
  - variable se incia en 'False'
  - pregunta si la lista no esta vacia
  - llamada recursiva con el nodo primero de la lista
  - cambia el valor de la variable
  -si la lista esta vacia lanza una exception
  - retorna el valor de la variable 'true' o 'false'

* elimimarDuplicados: elimina los elemento duplicados que hay en la lista. Retorna una nueva lista sin los elemento duplicados

  -variable nodoActu indica el nodo actual, se incializa en el primer nodo
  - variable nodoPrev indica el nodo previo al nodo actual. Se incia en el nodo primero
  - nuevaLista lista que se retorna, si incia vacia
  - incia el recorrido hasta que el puntero actual no tenga siguiente
  - condicional para saber si el elemento del nodo actual esta en la lista
  - si esta cambia los valores de las variable, nuevaLista pasa apuntar al nodoActu y nodoPrev siguiente pasa apuntar al nodoActu siguiente
  -si el elemento no esta, nuevaLista apunta al nodoLista pasa a hacer el nodoActu y nodoPrev pasa al siguiente del nodoActu
  -retorna la nueva lista

* agregarLista(lista): resive por parametro una lista y agrega al final de la lista
  -crea un nodoAux , si incia en el primero del nodo de la lista
  - entra al while hasta que el nodoAux no tenga siguiente
  - sale del while y agrega el primero de la lista que viene por parametro

funcion TDA ABB

* constructor
  - raiz del ABB se incia en None

* insertaPalabra() resibe por parametro una palabra y un web y la agrega al ABB
  -se crea un variable nuevoNodo
  - pregunta si esta vacia
  - cambia la raiz a nuevoNodo
  - si el ABB no esta vacio llama a la funcion recursiva

* insetarPagina() resibe una lista de paginas y la inserta en ABB
  - se verifica que la lista que se pasa por parametro no esta vacia.
  - llama al la funcion insertaPalabra(), se le paso por parametro un el dato que se obtiene de llamar a la funcion getDato del TDA lista

- despues llama a la funcion de forma recursiva
- a la poscion pasada por parametro se la incremena en 1

* buscarPalabras() resibe un lista de palabras y busca la palabra en el ABB, retorna una lista con las direcciones web de la cada palabra de la lista
- crea una lista
- pregunta si el ABB no esta vacio
- llama con el nodo raiz a la funcion recursiva
- si esta vacio lanza una exception

* listaWebPalabras() resibe por parametro una palabra y retorna una lista con todas la direcciones web de la misma
-crea una lista aux vacia
- pregunta si el ABB no esta vacio
- llama a la funcion recursiva con el nodo raiz
-si esta vacio lanza una exception

* plabrasDePagina() recibe por parametro un pagina y retorna una lista con todas las palabra del ABB
-crea una lista vacia
-pregunta si el ABB no esta vacio
-llama a la funcion recursiva con el nodo raiz
- si esta vacio lanza una exception

*eliminarPalabra() recibe por parametro una palabra y la elimina del ABB
-primero pregunta si no esta vacio
-pregunta si la palabra a borrar no es la del nodo raiz
-verifica el grado de la raiz sea mayor a 2
- crea un nodoPre con la funcion predecesor, llamada desde la raiz
-llama a la funcion del ABB eliminarPalabra
-cambia la variable nodPrec izquierdo y derecho por los nodos que estan a la izquierda y derecha del nodo raiz respectivamente
-cambia el nodo raiz al nodoPre
-si el grado del nodo raiz es menor a 2
- pregunta si tiene izquierdo y cambia la raiz al nodo izquierdo
- si no pregunta si tiene derecho y cambia la raiz al nodo derecho
-si la palabra no es la raiz llama a la funcion recursiva
-si el ABB no esta vacio lanza una exception

* eliminarPagina() funcion que elimina la pagina pasada por parametro, del ABB
-pregunta si ABB esta vacio
-si no esta llama a desde la raiz al la funcion eliminarPagina()

* cantidaTotalPalbras() recibe una cantidad de letras por parametro
y retorna la cantidad de palabras que tienen esa cantida o mas de letras
-incia una variable en cero
-pregunta si esta vacio
- sino esta vacio llama desde la variable al la funcion recursiva
- si esta vacio lanza una exception

* estaBalanceado() retorna 'true' o 'false' si el ABB esta estaBalanceado
  - inicia una variable en None
  -pregunta si esta vacio
  - llama desde la variable a al funcionn recursiva
  - si esta vacio lanza una exception

*paginasEnNivel() recibe por parametro un nivel y retorna la cantidad de paginas qu
e hay en ese nivel
  -crea un lista vacia
  -pregunta si esta vacio
  - sino esta vacio llama desde la variable al la funcion recursiva
  - si esta vacio lanza una exception

* cantidadDePalabrasMasUsadas() recibe por parametro una cantidad de paginas y reto
rna la cantidad de palabra que hay en el arbol con esa cantidad de paginas
  -
  -crea un lista vacia
  -pregunta si esta vacio
  - sino esta vacio llama desde la variable al la funcion recursiva
  - si esta vacio lanza una exception

* internasMayusculasAlfabetico() retorna una lista de las palabras que tiene comien
za en mayusculas y la retorna en orden internasMayusculasAlfabetico
  -crea un lista vacia
  -pregunta si esta vacio
  - sino esta vacio llama desde la variable al la funcion recursiva
  - si esta vacio lanza una exception

Funcione del NodoArbol

* constructor
-recibe por parametro una palabra y una pagina (web)
-crea un puntero izquierdo
-crea un puntero derecho
- crea una lista vacia de paginas
- agrega a la lista la web que resibe por parametro

* insertaPalabra() resibe del nodo raiz una palabra con una web por parametro y la
ubica en el ABB
  -Pregunta si la palabra es igual a la del nodo que recibe
  - si es igual insetar solo la pagina web a la lista
  -si no pregunta si la palabra del nodo que resibe es mayor a la del nodo actual
  - si tiene izquierdo llama a la recursiva desde el izquierdo
  - si no desde el nodo llama a la funcion insertarWeb()
  - si la palabra es mayor hace la misma comparacion pero desde el derecho

* insetarWeb() recibe una web y la agrega al la lista de paginas
  -usa la funcion estaEnLista() para saber si la web que recibe por parametro esta
en la lista de paginas

- sino esta la agrega a la lista

* listaWebPalabras() es llama desde el nodo raiz
- crea una lista vacia
-pregunta si la palabra del nodo actual es igual a la palabra que recibe por para metro
-si es igual clona la lista paginas del nodo actual
-sino pregunta la palabra es mayor a la palabra del nodo actual
-pregunta si tiene izquierdo y llama a la funcion recursivamente
- sino hace los mismo con el nodo derecho
-retorna la lista

* palabrasDePagina() recibe una lista de palabras desde el nodo raiz
- usa la funcion estaEnLista() para saber si la web esta en la lista-
-agrega a la listaDePalabras la palabra del nodo
- llama a recursivamente con el nodo izquierdo y derecho

* eliminarPalabra() elimina la palabra del ABB
- crea variable palabraEliminar, padre , lado
- usa la funcion busarElPadre
-preguna si la palabra es distana a None
- si el nodo de la palabra a eliminar tiene grado 2
-crea el nodoPre usando la funcion palabraEliminar para eliminar al nodo predecesor
- llama a la funcion eliminarPalabra pasando por parametro el nodoPre
-llama recursivamente a la funcion con el nodo izquierdo y derecho
-si la palabra a eliminar esta del lado 'izq' cambia el valor de padre.izquierdo al nodoPre, sino la cambia padre.derecho por nodoPre
-pregunta si la variable palabraEliminar tiene izquierdo
-si tiene cambia el valor del lado a 'izq'
- cambia el valor de padreIzquierdo a palabraEliminar.izquierdo
- la variable palabraEliminar tiene derecho hace lo mismo pero cabmia el valor del padreIzquierdo a derecho

* eliminarPagina()
-pregunta si tiene izquierdo
-llama a la recursiva del lado izquierdo
-lo mismo para el lado derecho
-cuando vuelve la las llamas recursivas preguna si las paginas esta en la lista
si es la elimina

*cantidaTotalPalabras()
-inica un contador en cero
- pregunta si tiene izquierdo
- desde la variable llama a la recursiva con el izquierdo
- hace lo mismo del lado derecho
- cuando vuelve de las llamdas recursiva compara las la cantidadLetras que recibe por parametro con la resultado de la funcion totalDeLetras() si es menor o igual su

ma 1 a la variable total
  - retorna la variable total

* paginasEnNivel() por parametro recibe un nivel
  -recibe un nivelNodo en cero para saber donde esta ubicado
  -crea una lista vacia
  -pregunta si el nivel que recibe por parametro es igual al nodo actual
  -si es agrega las paginas del nodo a la lista
  - sino hace la llamda recursiva desde el lado izquerdo y derecho
  sumando 1 la nivelNodo

* cantidadDePalabrasMasUsadas()
  -inica una variable en cero
  -llama a la recursiva desde el lado izquerdo
  - compara la cantidad de paginas con la el tamaño de la lista de las paginas
  - suma 1 a la variable
  - llama a la funcion desde el lado derecho
  - hace la misma comparacion

* internasMayusculasAlfabetico()
  - llama a la funcion recursiva desde el lado izquierdo y derecho
  -recorre el arbol en orden
  - cuando sale de llama izquierda hace la comparacion si es hoja y uas la funcion empiezaConMayusculas, agraga al la lista la palabra-
  -hace lo mismo del lado derecho