Pontificia Universidad
JAVERIANA
Bogotá

Documento de Diseño de Proyecto

Manuel Alejandro Rios Romero

&

Andrés José Rodríguez Ortega

Estructuras de Datos

Andrea Rueda

2021-01

Pontificia Universidad Javeriana

Bogotá D.C

Colombia

Contents

TADS

TAD Diccionario:

Conjunto mínimo de datos:

-nombreUlt, cadena de caracteres, representa el nombre del diccionario.

-palabras, lista de Palabra, representa las palabras guardadas en orden.

-palabrasInv, lista de Palabra, representa las palabras guardadas en orden inverso.

-totalPalabras, numero entero, representa cuantas palabras hay en el diccionario.

-arbolPal, árbol general de Palabra, representa las palabras guardadas en un árbol general.

-arbolPalInv, árbol general de Palabra, representa las palabras guardadas en orden inverso en un árbol general.

Operaciones:

Diccionario (), constructor del TAD que empieza con todos los datos vacíos.

Diccionario (nombre, pala, palInv, total), constructor del TAD que usa los parámetros enviados para asignar a los datos.

obtenerNombre(), retorna el nombre.

obtenerPalabras(), retorna lista con Palabras.

obtenerPalabrasInv(), retorna lista con Palabras inversas.

obtenerTotal(), retorna cantidad total de palabras.

obtenerArbolPal(), retorna el árbol general de Palabra.

obtenerArbolInv(), retorna el árbol general de Palabra inverso.

fijarArbolPal(arbol), fija el árbol general usando el parámetro.

fijarArbolInv(árbol), fija el árbol general inverso usando el parámetro.

fijarNombre(nombre), fija el nombre usando el parámetro.

fijarPalabras(pala), fija las Palabras usando el parámetro.

fijarPalabrasInv (palInv), fija las Palabras inversas usando el parámetro.

fijarTotal(total), fija la cantidad de palabras usando el parámetro.

TAD Palabra:

Conjunto mínimo de datos:

-pal, cadena de caracteres, los caracteres que forman la palabra.

Operaciones:

Palabra(), Constructor del TAD que empieza con la cadena de caracteres vacia.

Palabra(pala), Constructor del TAD que usa el parámetro enviado para asignar el dato.

puntaje(), calcula los puntos de la palabra y los imprime en pantalla.

obtenerPalabra(), retorna la palabra.

fijarPalabra (pala), fija la palabra.

TAD ArbolGeneral

Conjunto mínimo de datos:

-raiz, apuntador a nodo de tipo abstracto, indica el nodo que corresponde a la raíz del árbol

Operaciones:

ArbolGeneral(), constructor de TAD que empieza con todos los datos vacíos.

ArbolGeneral(val), constructor del TAD que empieza con raíz de valor del parámetro.

~ArbolGeneral(), destructor del árbol.

esVacio(), booleano, retorna un verdadero si la raíz del árbol es nula.

obtenerRaiz(), retorna el nodo de la raíz del árbol.

fijarRaiz(nraiz), asigna la raíz del árbol a partir del nodo recibido.

insertarNodo(padre, n), booleano que confirma con verdadero si se inserto el nodo “n” como hijo del nodo “padre”.

eliminarNodo(n), boleano que confirma con verdadero si se elimino el nodo “n”.

buscar(n), boleano que confirma con verdadero si se encuentra el nodo “n”.

altura(), retorna un entero equivalente a la altura del árbol.

tamano(), retorna un entero equivalente a la cantidad de nodos en el árbol.

preOrden(), imprime el árbol en pre-orden.

posOrden(), imprime el árbol en pos-orden.

nivelOrden(), imprime el árbol en nivel-orden.

TAD Nodo General

Conjunto mínimo de datos:

-dato, de tipo abstacto, es el contenido del nodo.

-desc, lista de nodos, contiene los apuntadores a los nodos hijos.

Operaciones:

NodoGeneral(), crea un nodo vacio.

~NodoGeneral(), destruye el nodo y sus contenidos.

obtenerDato(), retorna el dato contenido por el nodo.

obtenerDesc(), retorna la lista que contiene los hijos del nodo.

fijarDato(val), ingresa el dato val en el nodo.

limpiarLista(), limpia la lista de hijos del nodo.

adicionarDesc(val), adiciona a la lista de hijos el parámetro val.

eliminarDesc(val), boleano que retorna verdadero si se elimina el nodo con valor igual a val de los descendientes.

esHoja(), boleano que retorna verdadero si un nodo es hoja.

insertarNodo(padre, n), boleano que retorna verdadero si se inserta un nodo “n” a un nodo “padre”.

eliminarNodo(n), boleano que retorna verdadero si se elimina el nodo “n”.

buscar(n), boleano que retorna verdadero si se encuentra el nodo “n”.

altura(), retorna un entero igual a la altura del nodo.

tamano(), retorna un entero igual a la cantidad de nodos dentro del árbol.

preOrder(), imprime el árbol en pre-orden.

posOrden(), imprime el árbol en pos-orden.

nivelOrden(nivel, lvActual), imprime el árbol en nivel-orden, con los parámetro verifica el nivel en el que se encuentra.

TAD Grafo

Conjunto mínimo de datos:

-vertices, lista con todos los vértices del grafo.

-conexiones, lista de lista de aristas de todos los vértices del grafo

Operaciones:

-grafo(), crea un grafo vacio.

-setVertices(vértices), recibe una lista de vértices y la guarda en el grafo.

-setConexiones(conexiones), recibe la lista de lista de aristas y las guarda en el grafo.

-getVertices(), retorna todos los vértices del grafo en forma de lista.

-getConexiones(), retorna la lista de lista de conexiones.

-esVacio(), retorna verdadero si el grafo es vacío, de lo contrario retorna falso.

-cantidadVertices(), retorna la cantidad de vértices en el grafo.

- cantidadAristas(), retorna la cantidad de aristas en el grafo.

- insertarVertice(vertice), inserta un vértice en el grafo si este no existe.

- insertarVertice(dato), con el dato ingresado crea un vértice y lo inserta en el grafo.

- estaVertice(vertice), retorna verdadero si un vértice está en el grafo, falso si no.

- estaVertice(dato), crea un vértice con el dato ingresado y verifica si este nuevo vértice ya esta en el grafo.

- insertarArista(ver_org, ver_des, peso), dado un vértice de origen, un vértice de destino y un peso, crea e ingresa una arista al grafo.

- obtenerConexionesVertice(indice), retorna el apuntador de la arista dado el índice indicado.

- obtenerIndice(ver_org), retorna la posición del vértice en la lista de vértices.

-buscarVertice(ver_des), retorna el apuntador del vértice de la lista de vértices a partir del parámetro.

-insertarConexion(conex, verDes, peso), inserta una arista nueva a partir del vértice de destino y el peso ingresado.

- eliminarConexionesEnEliminarVertice(c, dato), elimina una arista a partir del dato y del apuntador.

- eliminarVertice(dato), elimina el vertice que contenga el dato ingresado.

-imprimirAristas(), imprime todas las aristas presentes en el grafo.

- imprimirVertices(), imprime todos los vértices del grafo.

- eliminarConexion(conex, verDes), elimina una conexión utilizando el vertice descendiente como parámetro.

- eliminarArista(ver_org, ver_des), elimina una arista a partir del vértice de origen y del vértice de destino.

- buscarArista(ver_org, ver_des), retorna la arista que comparta el vértice de origen y destino ingresados.

- verticeVisitado(lista, dato), a partir de la lista ingresada verifica si el vértice que contiene el dato ingresado ha sido visitado.

- busquedaEnProfundidad(ver_org, lista), realiza un recorrido de profundidad a partir del vértice de origen y la lista que se ingresa.

- busquedaEnAnchura(ver_org, lista), realiza un recorrido en anchura a partir del vértice de origen y la lista que se ingresa.

- mindistance(todosLosVertices, distancias), retorna el vértice de menor distancia del grafo.

- recorridoDijkstra(distancias, prede, dato) realiza el recorrido por Dijkstra del grafo.

- imprimirPredecesores(lista), imprime todos los predecesores de la lista de vectores ingresada.

TAD Vértice

Conjunto mínimo de datos:

-dato, dato de tipo abstracto

Operaciones:

-Vértice(), crea un vértice de datos vacíos.

-Vértice (dato), crea un vértice con el dato ingresado.

-getDato(), retorna el dato del vértice.

-setDato(dato), guarda el dato ingresado en el vértice.

-eliminarArista(peso), elimina la arista que tenga un peso igual al ingresado.

- <(ver), compara el dato ingresado con el del vértice, si es menor retorna verdadero, de lo contrario retorna falso.

TAD Arista

Conjunto mínimo de datos:

-peso, dato de tipo abstracto que representa el peso del arista.

-sucesor, apuntador al vértice que le sigue en la lista.

Operaciones:

-arista(), crea una arista vacia.

-arista(verDes, peso), crea una arista con el peso indicado y el sucesor ingresado en verdes.

-setPeso(peso), asigna el peso de la arista.

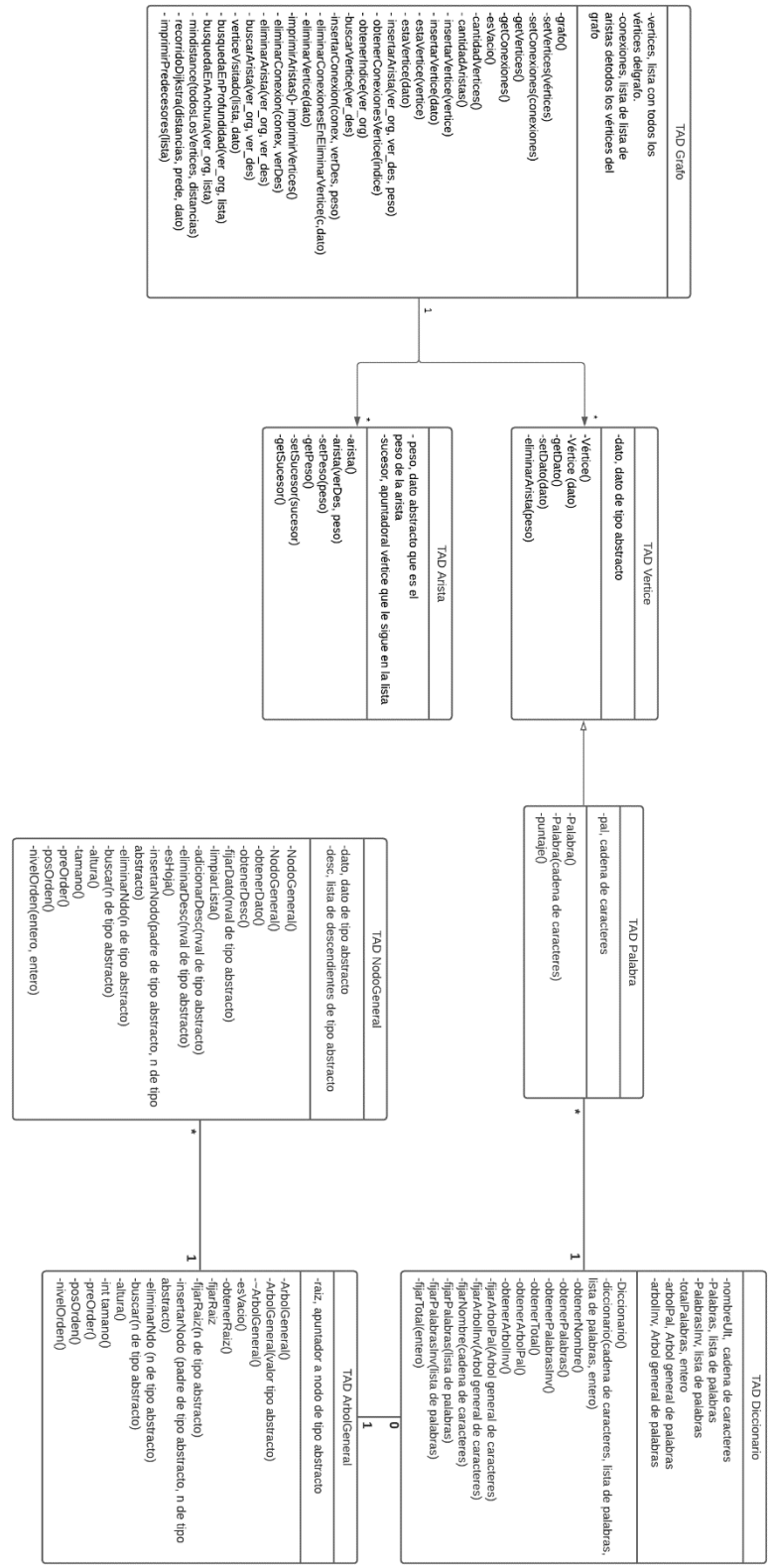
-getPeso(), retorna el peso de la arista.

-setSucesor(sucesor), asigna un sucesor a la arista.

-getSucesor(), retorna el vertice sucesor de la arista.

- <(ar), si el sucesor de la arista ingresado es mayor que el de la arista retorna verdadero, de lo contrario retorna falso.

Diagrama de relación entre TADs



Acta de Evaluación

Elementos Faltantes de la Entrega 2

- Completar el recorrido del árbol para poder identificar las palabras a partir del prefijo y del sufijo.
- La implementación del árbol ya estaba únicamente hacia falta recorrerlo de la manera más adecuada para obtener las palabras.

Como se completaron

- Se escribió el codigo para realizar el recorrido por el árbol.
- Una vez este estaba completo, se realizaban pruebas de ensayo y error para verificar el correcto funcionamiento del recorrido del árbol.
- Dado un error se repiten las pruebas de ensayo y error hasta encontrar el posible error y corregirlo.
- Para asegurar que la impresión de las palabras sea más simple, toda palabra que comienza con el prefijo o el sufijo se agrega a una lista. De está manera una vez halladas las palabras, solo es necesario recorrer la lista y no el árbol.

Funciones Principales:

Main:

Entradas:

- Ninguna.

Salidas Posibles:

- El comando no es válido.
- Se llama la función correspondiente al comando y opera a partir del parámetro si es necesario.

Proceso:

- Se solicita que el usuario ingrese el comando.
- El comando ingresado por el usuario se divide en distintas cadenas de caracteres.
- A partir de la primera cadena de caracteres, se hace llamado a la función correspondiente. Si la función requiere de la segunda cadena de caracteres, esta tambien es utilizada como un parámetro adicional.

Diseño:

Anexo 1

Inicializar Diccionario:

Entradas:

- Cadena de caracteres con el nombre del archivo.

Salidas Posibles:

- Se imprime en pantalla error al cargar archivo.
- Se informa que el diccionario ya está cargado.
- Se carga el diccionario al programa.

Proceso:

- Se crea un stream con el que se intenta abrir el archivo con el nombre recibido. Si esto no es posible, se imprime el error y termina la función.
- Si ya se ha leído el diccionario, se informa que este diccionario ya está cargado y termina la función.
- Al abrir el archivo se revisa linea por linea las palabras, si la palabra es válida esta se guarda en una lista de la variable Palabra.

Diseño:

Anexo 2

Inicializar Diccionario Inverso:

Entradas:

- Cadena de caracteres con el nombre del archivo.

Salidas Posibles:

- Se imprime en pantalla error al cargar archivo.
- Se informa que el diccionario ya está cargado.
- Se carga el diccionario al programa.

Proceso:

- Se crea un stream con el que se intenta abrir el archivo con el nombre recibido. Si esto no es posible, se imprime el error y termina la función.
- Si ya se ha leído el diccionario, se informa que este diccionario ya esta cargado y termina la función.
- Al abrir el archivo se revisa linea por linea las palabras, si la palabra es válida esta se invierte el orden usando una función strrev. Se guarda en una lista de variable Palabra.

Diseño:

Anexo 3

Puntaje Palabra:

Entradas:

- Palabra a revisar puntaje.

Salidas Posibles:

- Si la palabra contiene símbolos inválidos, se informa del error y termina la función.
- Si la palabra no está registrada en el diccionario, se informa del error y termina la función.
- Si la palabra es válida, se imprime en pantalla los puntos que vale la palabra.

Proceso:

- Se verifica que la palabra sea válida.
- Se verifica que la palabra este en el diccionario cargado.
- Si la palabra falla una de las verificaciones, se informa a cerca del error.
- Si la palabra cumple con todas las verificaciones, la palabra sigue a calcular su valor dependiendo de los caracteres que presente.

Diseño:

Anexo 4

Inicializar Arbol:

Entradas:

- Cadena de caracteres con el nombre del archivo.

Salidas Posibles:

- Se imprime en pantalla error al cargar el archivo.
- Se informa que el diccionario elegido ya está cargado.
- Se carga el diccionario al programa y se crea el árbol.

Proceso:

- Se crea un stream con el que se intenta abrir el archivo con el nombre recibido. Si esto no es posible, se imprime el error y termina la función.
- Si ya se ha leído el diccionario, se informa que este diccionario ya está cargado y termina la función.
- Al abrir el archivo se revisa linea por linea las palabras, si la palabra es valida esta se guarda como un camino en el árbol general.

Diseño:

Anexo 5

Inicializar Árbol Inverso:

Entradas:

- Cadena de caracteres con el nombre del archivo.

Salidas Posibles:

- Se imprime en pantalla error al cargar el archivo.
- Se informa que el diccionario elegido ya está cargado.
- Se carga el diccionario al programa y se crea el árbol.

Proceso:

- Se crea un stream con el que se intenta abrir el archivo con el nombre recibido. Si esto no es posible, se imprime el error y termina la función.
- Si ya se ha leído el diccionario, se informa que este diccionario ya está cargado y termina la función.
- Al abrir el archivo se revisa linea por linea las palabras, si la palabra es válida esta se guarda como un camino en el árbol inverso. Las palabras se invierten usando dos cadenas de caracteres y cambiando la posición del primer carácter de la cadena 1 por el ultimo de la cadena dos, aumentando de posición en la cadena 1 y decrementando en la cadena 2 hasta invertir la palabra.

Diseño:

Anexo 6

Palabras por Prefijo:

Entradas:

- Prefijo de las palabras a buscar.

Salidas Posibles:

- Se imprime en pantalla error no se encuentran palabras con ese prefijo.
- Se imprime en pantalla las palabras del prefijo con sus puntajes respectivos y el tamaño de cada una.

Proceso:

- Se recibe el prefijo con el que se buscan las posibles palabras en el árbol.
- Se busca el prefijo en el árbol, si el prefijo no equivale a ninguna palabra en el árbol se imprime el error y se termina la función. Si se encuentra el prefijo se recorre el subárbol del ultimo nodo encontrado.
- Al recorrer el subárbol, se almacenan las palabras en una lista y se retornan.
- Con la lista de palabras se calcula el tamaño de las palabras y su puntaje, finalmente esto se imprime en pantalla.

Diseño:

Anexo 7

Palabras por Sufijo:

Entradas:

- Sufijo de las palabras a buscar.

Salidas Posibles:

- Se imprime en pantalla error no se encuentran palabras con ese sufijo.
- Se imprime en pantalla las palabras del sufijo con sus puntajes respectivos y el tamaño de cada una.

Proceso:

- Se recibe el sufijo con el que se buscan las posibles palabras en el árbol inverso.
- Se busca el sufijo en el árbol inverso, si el sufijo no equivale a ninguna palabra en el árbol inverso se imprime el error y se termina la función. Si se encuentra el sufijo se recorre el subárbol inverso del ultimo nodo encontrado.
- Al recorrer el subárbol inverso, se almacenan las palabras en una lista y se retornan.
- Con la lista de palabras se calcula el tamaño de las palabras y su puntaje, finalmente esto se imprime en pantalla.

Diseño:

Anexo 8

Grafo de Palabras:

Entradas:

- Ninguna

Salidas Posibles:

- Se imprime en pantalla error de no tener ningún diccionario inicializado.
- Se imprime en pantalla el numero vértices y conexiones insertados con mensajes para saber cuando se están insertando los vértices y las conexiones al grafo.

Proceso:

- Se recorre la lista de palabras del diccionario de juego y se insertan uno a uno al grafo.
- Se comparan las palabras insertadas en la lista de vértices y si la diferencia entre las palabras es de una sola letra se les agrega una arista entre ellas de peso 0 y sin dirección.

Diseño:

Anexo 9

Posibles Palabras:

Entradas:

- Letras por usar como parámetro de búsqueda.

Salidas Posibles:

- Se imprime en pantalla error si la cadena de letras contiene símbolos inválidos.
- Se imprime en pantalla error si no se ha construido un grafo de palabras.
- Se imprime en pantalla error si no es posible construir ninguna palabra con las letras ingresadas.
- Se imprime en pantalla todas las palabras posibles, sus puntajes y sus longitudes.

Proceso:

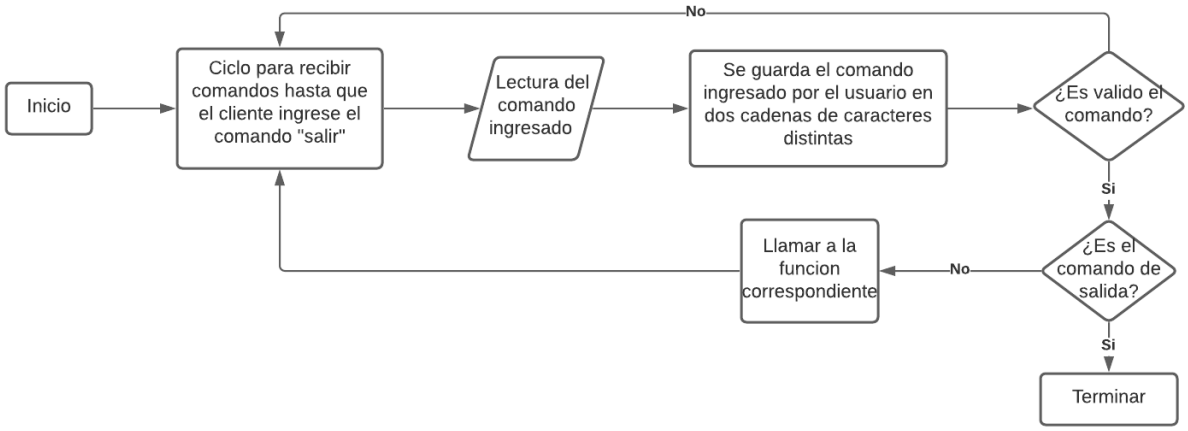
- Recorrer la lista de vértices y de aristas de cada vértice del grafo.
- Se comparan las letras de las palabras con las letras ingresadas como parámetro.
- Si la palabra contiene las letras de parámetro, esta palabra es agregada a una lista para almacenar todas las posibles palabras.
- Se imprime la lista de posibles palabras, se saca el puntaje y tamaño de cada palabra a medida que estas se imprimen.

Diseño:

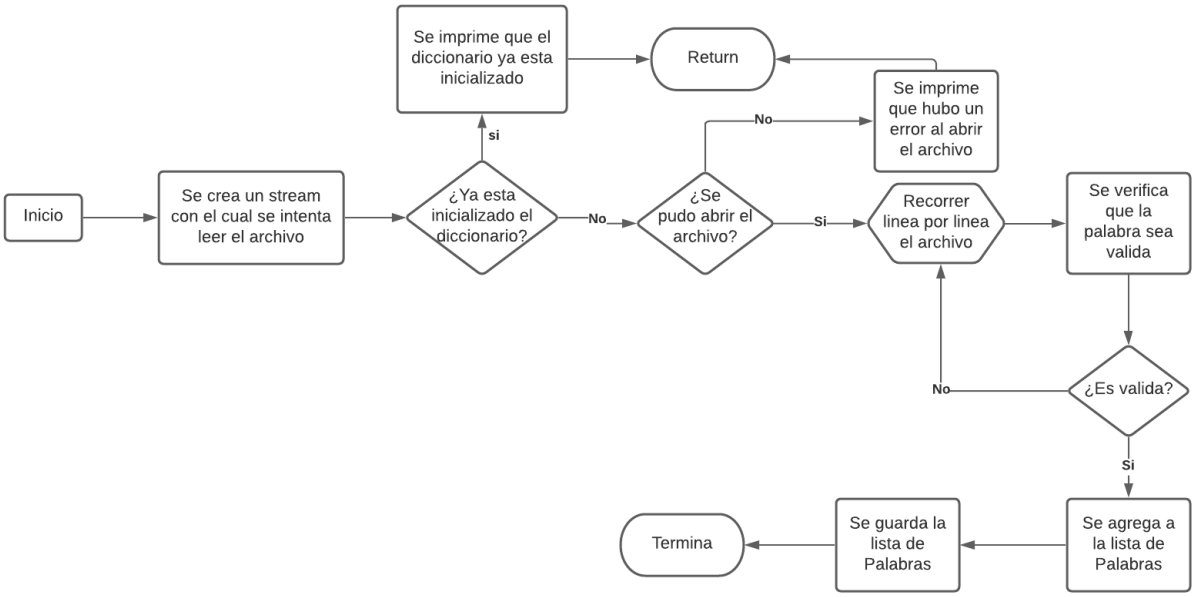
Anexo 10

Anexos:

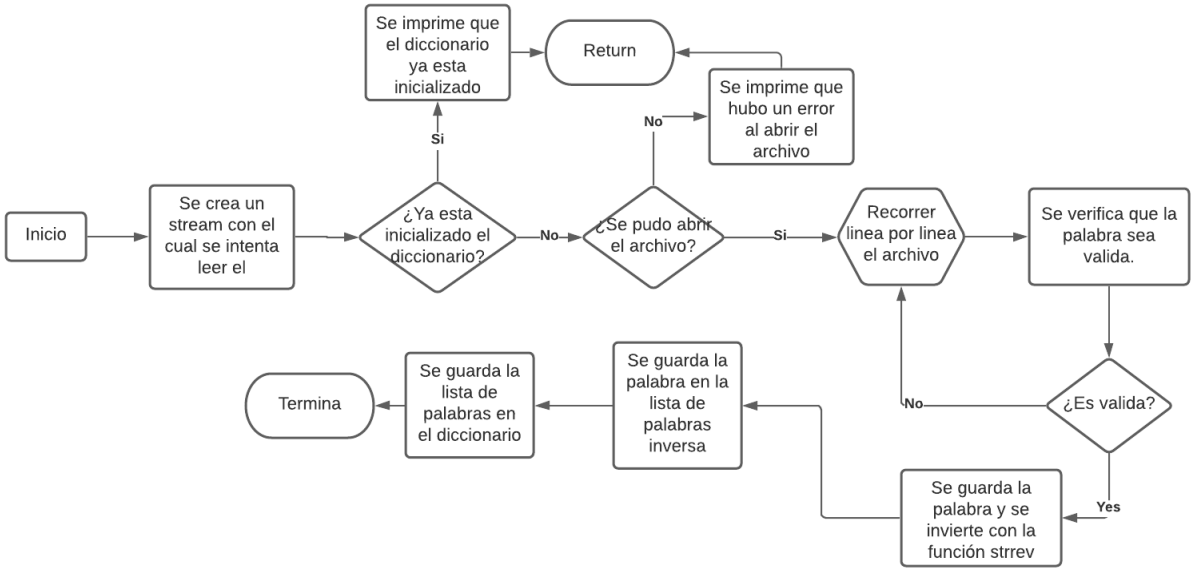
Anexo 1: Main



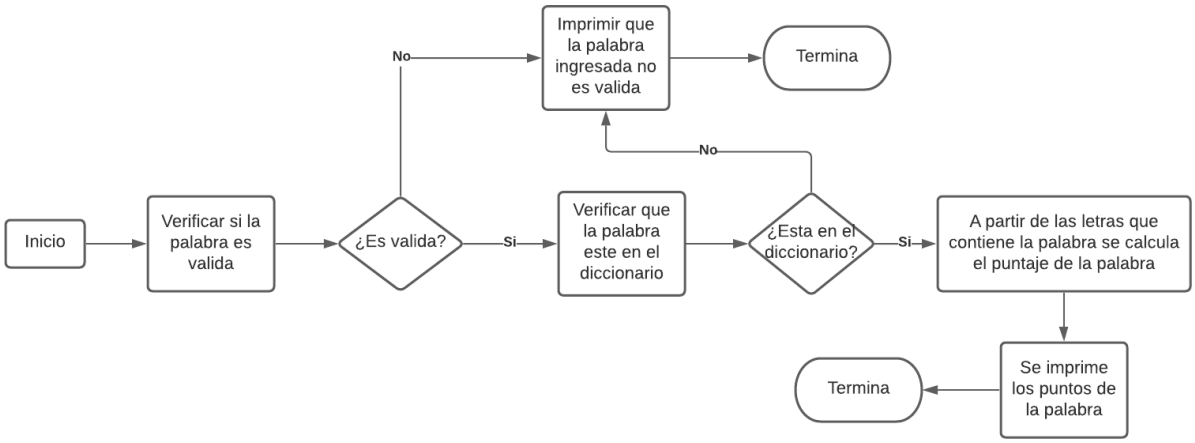
Anexo 2: Inicializar Diccionario



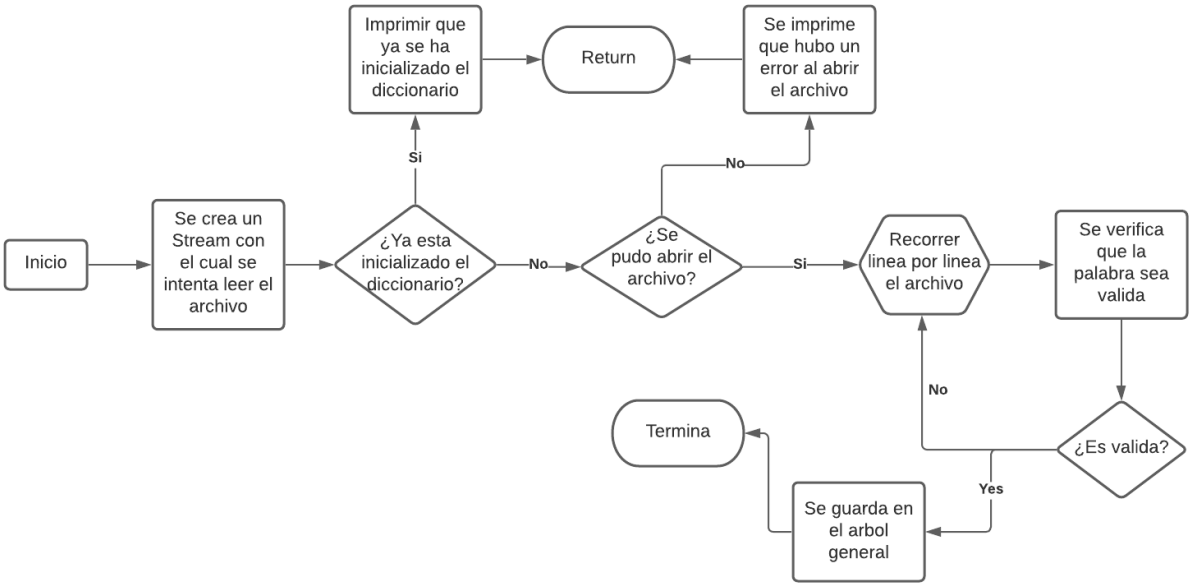
Anexo 3: Inicializar Diccionario Inverso



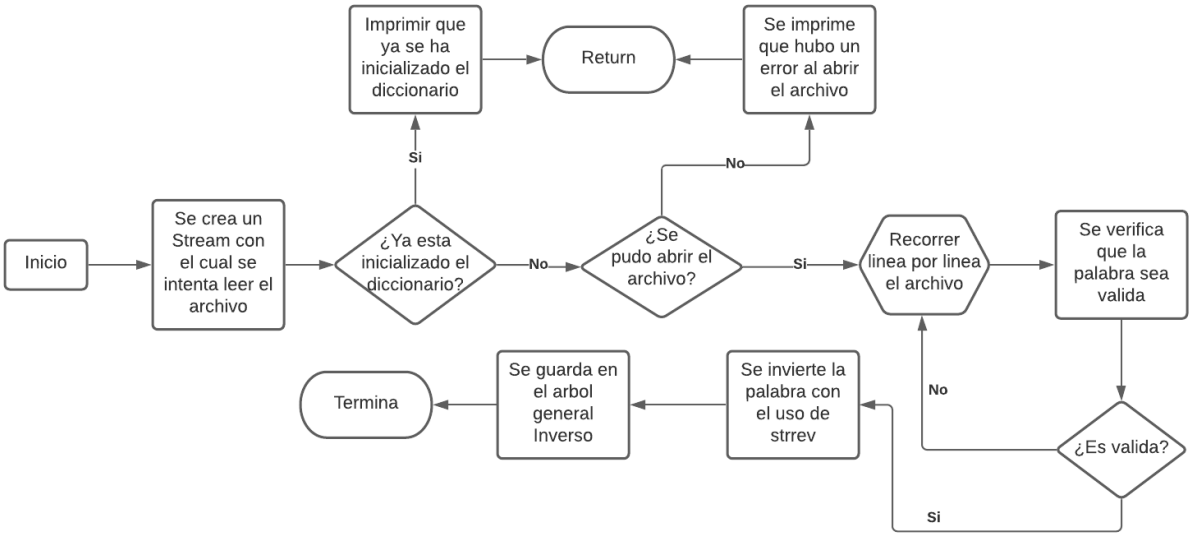
Anexo 4: Puntaje Palabra



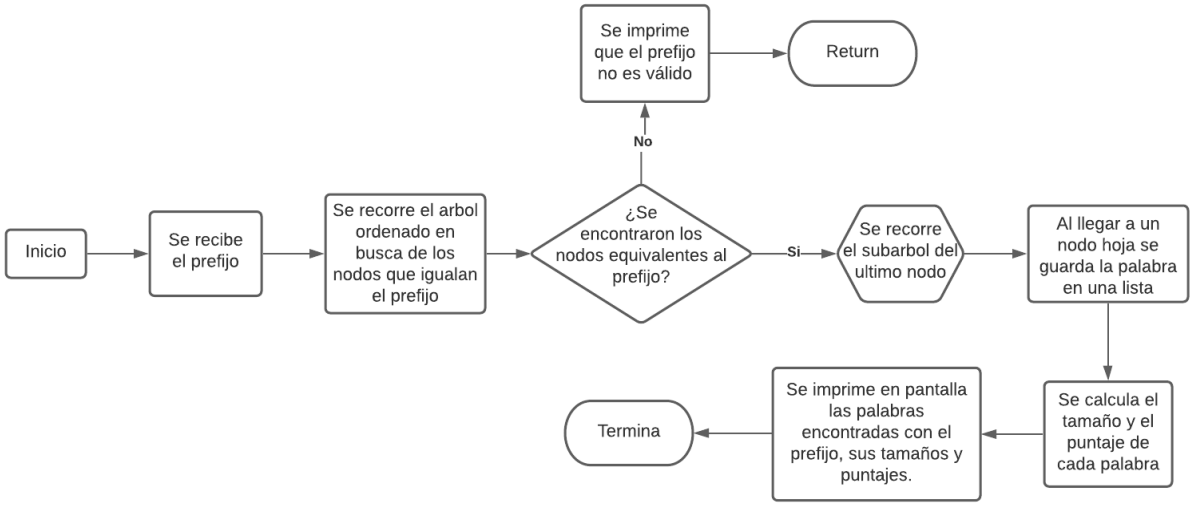
Anexo 5: Iniciar Árbol



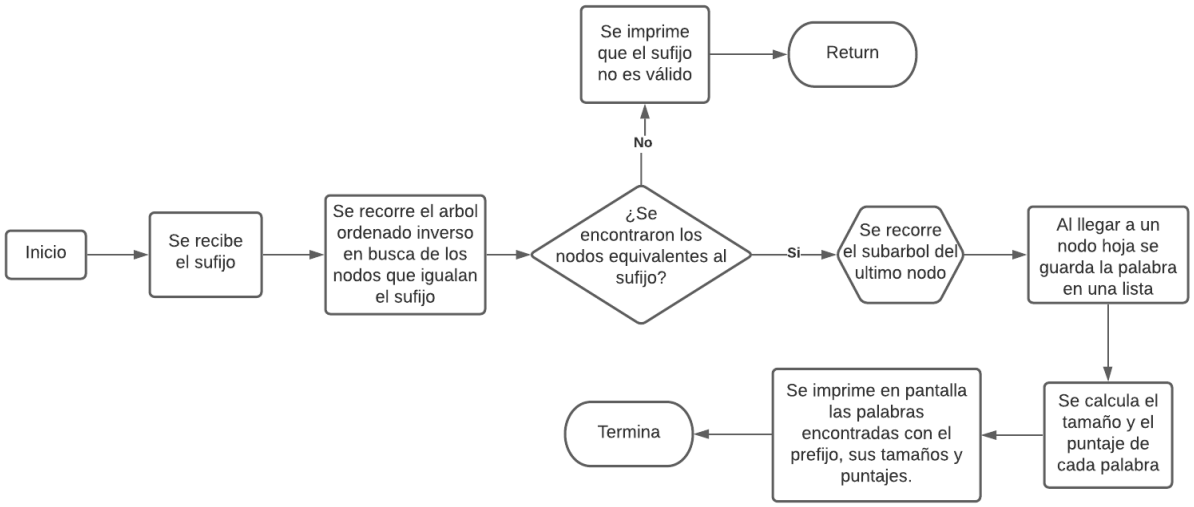
Anexo 6: Iniciar Árbol Inverso



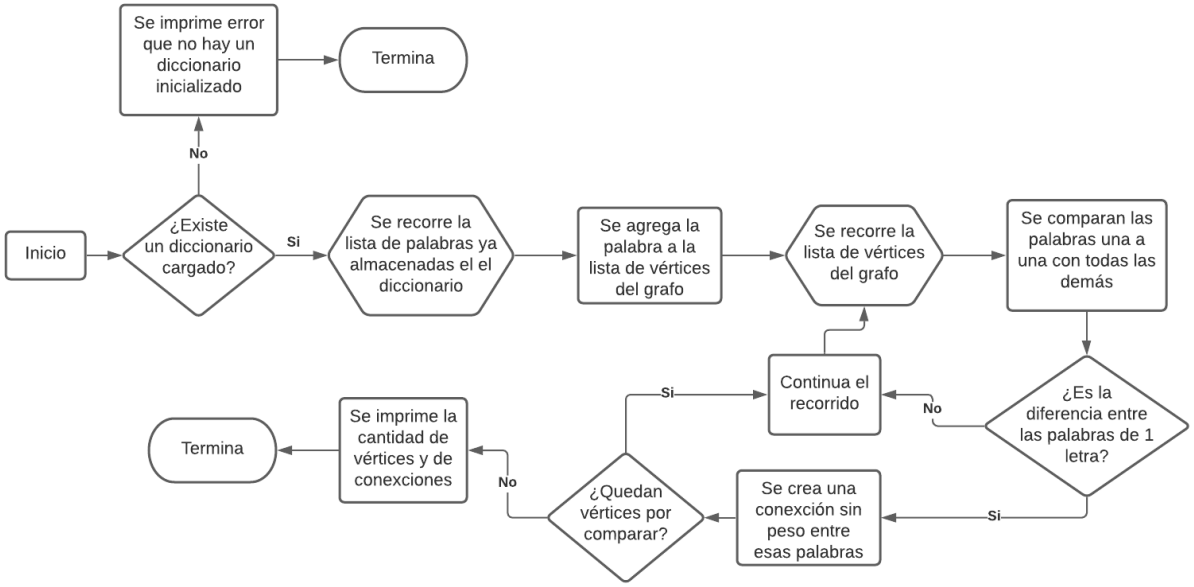
Anexo 7: Palabras por Prefijo



Anexo 8: Palabras por Sufijo



Anexo 9: Grafo de Palabras



Anexo 10: Posibles Palabras