FACULTAD DE INGENIERÍA

CARRERA DE INGENIERÍA DE SISTEMAS

ESTRUCTURAS DE DATOS

JORGE OSWALDO VALENZUELA BUITRAGO.

PROYECTO ENTREGA #3

SANTIAGO ANDRÉS LEAL OMAÑA

SANTIAGO ZUÑIGA MARTÍNEZ

NICOLÁS SÁNCHEZ LAITON

26/10/2022

Introducción:

Llevar el control de las distintas formas de organización de un país siempre ha sido importante para poder y administrar los distintos recursos que circula por medio de estas, en caso de Colombia según la división político-administrativa, esto se comprende en un sistema de codificación de los departamentos, municipios y centros población que conforman de forma jerárquica el territorio de Colombia, puede ser importante llegar a manejar ciertos datos y operaciones con estas operaciones políticas, el propósito del proyecto es poder realizar un programa que permita hacer estas operaciones de manera sencilla usando el programa de C++ y los conceptos que se han ido aprendidos durante las clases de Estructuras de Datos, para así poder crear este programa de una manera mas sencilla, además de con una complejidad baja para de esa manera no tener que ocupar tanto espacio en la memoria junto con una bajo tiempo de ejecución.

1)Commands:

En este hpp se incluye las librerías map, fstream, string, vector y iostream, además de incluir unos .hpp como structures.hpp y utils.hpp.

Funciones:

a) carga_divipola

Aspectos sintácticos:

-Nombre: carga_divipola.

-Parámetros de entrada:

-filename (nombre del archivo a cargar).

-dpto (mapa de llaves de departamento con identificador).

-Resultado devuelto: No retorna nada.

Semánticos: Carga en una o más estructuras de datos en memoria el contenido del archivo identificado por filename.

Complejidad: O(n)

b) listar_departamentos

Aspectos sintácticos:

-Nombre: listar_departamentos

-Parámetros de entrada:

-dpto (mapa de llaves de departamento con identificador).

-Resultado devuelto: No retorna nada.

Semánticos: Imprime en n líneas (una para cada departamento) la información básica del departamento que se cargó de la Divipola.

Complejidad: O(n).

c) listar_municipios

Aspectos sintácticos:

-Nombre: listar_municipios.

-Parámetros de entrada:

-code (codigo del departamento).

-dpto (mapa de llaves de departamento con identificador).

Resultado devuelto: No retorna nada.
Semánticos: Imprime en n líneas (una para cada municipio) la información básica del departamento que se cargó de la Divipola.

Complejidad: O(n long(n))

d) listar_poblaciones

Aspectos sintácticos:

-Nombre: listar_poblaciones.

-Parámetros de entrada:

-code (codigo del departamento).

-dpto (mapa de llaves de departamento con identificador).

-Resultado devuelto: No retorna nada.
Semánticos: Imprime en n líneas (una para cada población) la información básica del departamento que se cargó de la Divipola.

Complejidad: O(n long(n))

e) info_sumaria

Aspectos sintácticos:

-Nombre: info_sumaria.

-Parámetros de entrada:

-code (codigo del departamento).

-dpto (mapa de llaves de departamento con identificador).

-Resultado devuelto: No retorna nada.

Semánticos: Imprime en una línea la información básica del nombre del departamento que corresponde al código dado como parámetro junto con la cantidad de municipios y centros poblados que lo conforman.

Complejidad: O(n long(n))

f) carga_SC

Aspectos sintácticos:

-Nombre: carga_SC.

-Parámetros de entrada:

-file_name (nombre del archivo a cargar).

-dpto (mapa de llaves de departamento con identificador).

-Resultado devuelto: No retorna nada.

-Semánticos: Carga en una o más estructuras de datos en memoria el contenido del archivo identificado por file_name con la información básica para representar el sistema de ciudades.

Complejidad: O(n long(n))

g) esta_en_sistema

Aspectos sintácticos:

-Nombre: esta_en_sistema

-Parámetros de entrada:

-divipola (codigo de un municipio).

-dpto (mapa de llaves de departamento con identificador).

-Resultado devuelto: No retorna nada

Semánticos: Determina si un municipio, con código dado por el usuario, existe dentro de los municipios definidos en el Sistema de Ciudades y cargados desde el archivo correspondiente.

Complejidad: O(n long(n))

Commands2:

En este hpp se incluye las librerías map, fstream, string, vector y iostream, además de incluir unos .hpp como structures.hpp, utils.hpp y huffman.hpp.

Funciones:

a) aglomeración

Aspectos sintácticos:

-Nombre: aglomeración

-Parámetros de entrada:

-SC (Estructura del sistema de ciudades).

-dptos (Mapa de departamentos).

-Resultado devuelto: No retorna nada.

Semánticos: El comando debe crear los componentes C1, aglomeraciones urbanas, como componentes del sistema de Ciudades, de acuerdo con los datos del archivo SC_2018 guardados en memoria y los criterios definidos con anterioridad.

Complejidad: O(n)

b) uninodal

Aspectos sintácticos:

-Nombre: uninodal

-Parámetros de entrada:

-SC (nombre del archivo a cargar).

-dptos (mapa de llaves de departamento con identificador).

-Resultado devuelto: No retorna nada.

Semánticos: El comando debe crear los componentes C2, ciudades uninodales, como componentes del sistema de Ciudades, de acuerdo con los datos del archivo SC_2018 guardados en memoria y los criterios definidos con anterioridad.

Complejidad: O(n)

c) capital_menor

Aspectos sintácticos:

-Nombre: capital_menor

-Parámetros de entrada:

-SC (nombre del archivo a cargar).

-Resultado devuelto: No retorna nada.

Semánticos: El comando debe crear los componentes C3, ciudades capitales menores de cien mil habitantes, como componentes del sistema de Ciudades, de acuerdo con los datos del archivo SC_2018 guardados en memoria y los criterios definidos con anterioridad.

Complejidad: O(n)

d) reporte

Aspectos sintácticos:

-Nombre: reporte

-Parámetros de entrada:

-SC (nombre del archivo a cargar).

-dpto (mapa de llaves de departamento con identificador).

-Resultado devuelto: No retorna nada.

Semánticos: El comando debe crear un reporte con los datos del sistema, así:

Sistema de Ciudades	Municipios	Población Total
Ciudades que hacen parte del Sistema		
Aglomeraciones urbanas (X1)	X2	X3
Ciudades uninodales (Y1)	Y2	Y3
Ciudades fuera del Sistema		
Ciudades con más de cien mil habitantes (Z1)	Z2	Z3
Ciudades con menos de cien mil habitantes (W1)	W2	W3
Total Sistema de ciudades (T1)	T2	T3
% con respecto a Colombia (P1)	P2	P3
Total Colombia	TC1	TC2

Los datos X1, X2… hasta TC2 deben mostrar los valores correspondientes calculados del archivo leído y guardado en memoria.

Complejidad: O(n)

e) codificar nombre_archivo.icmbin nombre_archivo.csv

Aspectos sintácticos:

-Nombre: codificar nombre_archivo.icmbin nombre_archivo.csv

-Parámetros de entrada:

-filename (nombre del archivo a crear).
-compress (nombre del archivo a comprimir)

-Resultado devuelto: No retorna nada.

Semánticos: El comando debe cargar en memoria la información completa del Índice de Ciudades Modernas (ICM) y generar el archivo binario con la correspondiente codificación de Huffman con toda la información que lo compone, almacenándolo en disco bajo el nombre: nombre_archivo.icmbin .

Complejidad: O(n^2)

f) decodificar nombre_archivo.icmbin

Aspectos sintácticos:

-Nombre: decodificar nombre_archivo.icmbin

-Parámetros de entrada:

-filename (nombre del archivo a descomprimir).

-Resultado devuelto: No retorna nada.

Semánticos: El comando debe cargar en memoria los datos contenidos en el archivo binario nombre_archivo.icmbin , que contiene una codificación Huffman de toda la información que compone el Índice de Ciudades Modernas y debe mostrarlo decodificado en pantalla.

Complejidad: O(n)

Commands3:

a) Distancia

Aspectos sintácticos:

-Nombre: Distancia

-Parámetros de entrada:

-aglo (string del nombre de la aglomeracion).

-gsc (map de aglomeraciones).

-Resultado devuelto: No retorna nada.

Semánticos: El comando debe Calcular la distancia en kilómetros entre el centro de centro de la aglomeración y cada una de las ciudades en la aglomeración utilizando los datos de latitud y longitud que tienen los municipios en la Divipola y guardarlos en la estructura del Sistema de Ciudades.

Complejidad: O(n^2)

b) ruta_mas_corta_de_Aglomeracion

Aspectos sintácticos:

-Nombre: ruta_mas_corta_de_Aglomeracion.

-Parámetros de entrada:

- aglo (string del nombre de la aglomeracion).

-gsc (map de aglomeraciones).

-Resultado devuelto: No retorna nada.

Semánticos: El comando debe imprimir en pantalla la secuencia de vértices del grafo que describen la ruta más corta entre el centro de la aglomeración y cada una de las ciudades en la aglomeración. Así mismo, debe imprimir el costo total de la ruta en kilómetros.

Complejidad: O(n^2)

c) ciudad_remota

Aspectos sintácticos:

-Nombre: ciudad_remota.

-Parámetros de entrada:

- aglo (string del nombre de la aglomeracion).

-gsc (map de aglomeraciones).

-Resultado devuelto: No retorna nada.

Semánticos: Para la ciudad centro de la aglomeración el comando busca la ciudad en la aglomeración más remota, y debe imprimir en pantalla la secuencia de vértices que describen la ruta entre la centro de aglomeración y la ciudad en la aglomeración más remota, y el costo total de la ruta en kilometros.

Complejidad: O(n^2)

2)Utils

En este hpp se incluye las librerías string, vector y algorithm.

Funciones:

a) trim

Aspectos sintácticos:

-Nombre: trim

-Parámetros de entrada:

-input (cadena de texto la cual será editada).

-Resultado devuelto: Retorna el string con los espacios extra removidos.

Semánticos: Quita los espacios adicionales que puede haber en un string, ya sea por error del usuario al ingresar el string o por temas del mismo programa.

b) tokenize

Aspectos sintácticos:

-Nombre: tokenize

-Parámetros de entrada:

-s (Cadena de texto que será tokenizada).

-del (cadena de texto constante).

-dlim (delimitador).

-Resultado devuelto: Regresa un vector en donde se encuentra el string tokanizado.

Semánticos: Esta función busca en el string el delimitador que se pasó por parámetro para ir dividiendo el strings en partes, y estas partes las va guardando en un vector llamado tokens el cual devolverá el string, pero ya separado. Para los casos en los que el vector esta entre comillas, la función revisa si hay comillas para saltar y luego dividir el string.

Complejidad: O(n long(n))

3)Help

Este hpp incluye las librerías iostream y string.

Funciones:

a) help

Aspectos sintácticos:

-Nombre: help

-Parámetros de entrada:

-command (cadena de texto).

Resultado devuelto: No retorna nada.

Semánticos: Con esta función lo que se busca es dar información sobre el comando que se puede usar, para eso el usuario ingresa en la terminal el comando help seguido del comando del cual quiere saber su uso, en caso de que el comando sea desconocido, se mostrara en pantalla un mensaje de error.

Complejidad: O(1)

b) help

Aspectos sintácticos:

-Nombre: help.

-Parámetros de entrada: No recibe parámetros.

-Resultado devuelto: No retorna nada.

Semánticos: Esta función le muestra al usuario los distintos tipos de comandos que hay, para acceder a esta función toca colocar el comando help sin ningún comando extra, para que se despliegue una lista de todos los comandos posibles para usar.

Complejidad: O(1)

4) Interpreter

a) interpreter

Aspectos sintácticos:

-Nombre: interpreter

-Parámetros de entrada: No recibe parametros.

-Resultado devuelto: No retorna nada

Semánticos: Es la función la cual toma los datos suministrados por el usuario en pantalla y luego los transforma en la información que se necesita.

Complejidad: O(1)

b) executeCommand

Aspectos sintácticos:

-Nombre: executeCommand

-Parámetros de entrada:

-tokens (vector constante).

-dpto (mapa de llaves de departamento con identificador).

-Resultado devuelto: No retorna nada.
Semánticos: Esta función va comparando la primera posición del vector para saber cual es el comando que se pide y así llevar al usuario a la función pertinente.

Complejidad: O(1)

Estructuras:

a) CP

Aspectos sintácticos:

-nombre: CP.

-Modelo: name, latitude, longitude.

Para cada operación:

-Nombre: CP

-Parámetros de entrada:

-name (cadena de texto).

-latitude (número decimal).

-longitude (número decimal).

-Resultado devuelto: No retorna nada.

Semánticos: Es una estructura en la que se guarda la información de los centros poblados.

b) ANM

Aspectos sintácticos:

-nombre: ANM

-Modelo: name, anm, cp, sc.

Para cada operación:

-Nombre: ANM

-Parámetros de entrada:

-n (cadena de texto constante).

-Resultado devuelto: No retorna nada.

Semánticos: Es una estructura en la que se guarda la información de las áreas no municipalizadas, que son clasificadas como municipios, pero no cumplen con los requisitos mínimos.

c) CM

Aspectos sintácticos:

-nombre: CM

-Modelo: name, cp, latitude, longitude, population, sc.

Para cada operación:

-Nombre CM

-Parámetros de entrada:

-n (cadena de texto constante).

-L1 (número decimal constante).

-L2(número decimal constante).

-Resultado devuelto: No retorna nada.

Semánticos: Es una estructura en la que se guarda la información de los municipios.

d) Department

Aspectos sintácticos:

-nombre: Department.

-Modelo: name, cm, anm.

Para cada operación:

-Nombre: Department.

-Parámetros de entrada:

-n (cadena de texto constante).

-Resultado devuelto: No retorna nada.

Semánticos: Es una estructura en la que se guarda la información de los departamentos.

e) Huffman_bin

Aspectos sintácticos:

-nombre: Huffman_bin.

-Modelo: chuff, bin.

Para cada operación:

-Nombre: toBinary.

-Parámetros de entrada:

-n (Entero).

-Resultado devuelto: r(cadena de texto constante).

Semánticos: Es una estructura en la cual se guardan los códigos de Huffman y el texto comprimido de Huffman.

f) Huffman_binAuxiliar

Aspectos sintácticos:

-nombre: Huffman_binAuxiliar.

-Modelo: chuff, bitsetbin, chuffsize, bitsetbinsize.

Para cada operación:

-Nombre: bin2Auxiliar.

-Parámetros de entrada:

-bin (Cadena de texto).

-Resultado devuelto: No retorna nada.

Semánticos: Es una estructura en la cual se guardan los códigos de Huffman y el texto comprimido de Huffman pero en un bitset para así poder guardarlo en un archivo binario.

g) Edge

Aspectos sintácticos:

-nombre: Edge.

-Modelo: connect, lengh.

Para cada operación:

-Nombre: Edge.

-Parámetros de entrada:

- data (Cm).

-Resultado devuelto: No retorna nada.

Semánticos: Es una estructura en la cual se representan las aristas.

h) Vertex

Aspectos sintácticos:

-nombre: Vertex.

-Modelo: data, Edge, distance, prev, visited.

Semánticos: Es una estructura en la cual se guardan los vértices del grafo usado.

i) Graph

Aspectos sintácticos:

-nombre: Graph.

-Modelo: size, vertexes.

Para cada operación:

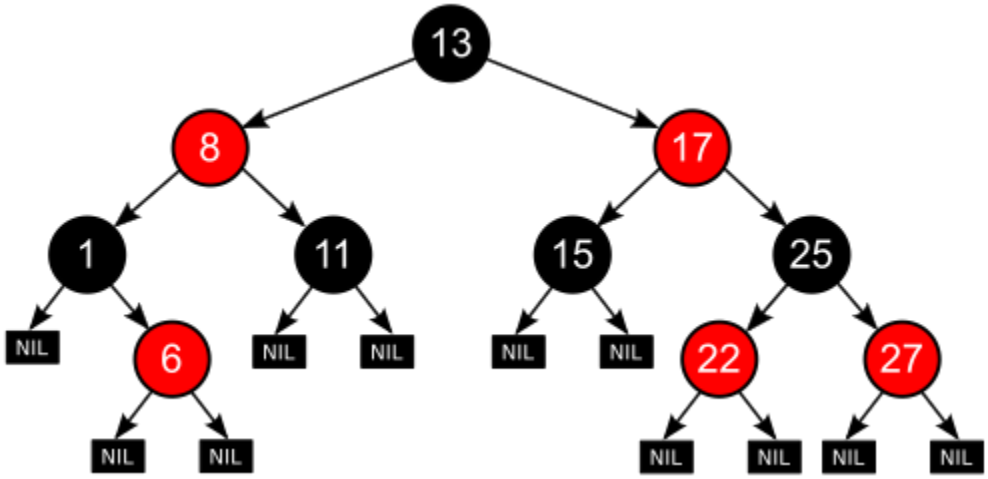
-Nombre: insert_vertex.

-Parámetros de entrada:

-data (Cm).

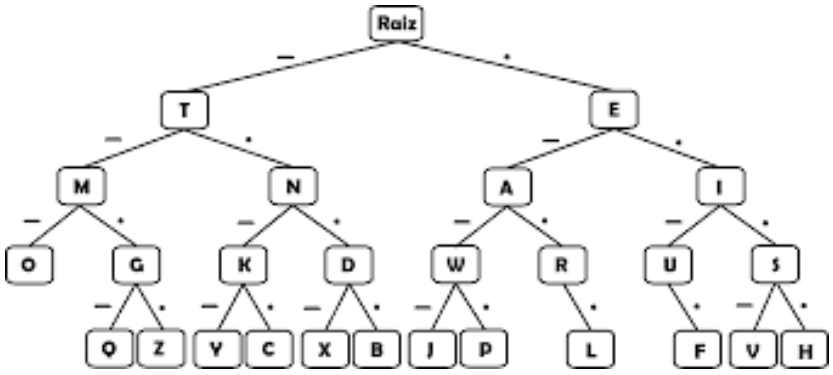
-Resultado devuelto: No retorna nada.

Semánticos: Es una estructura en la cual se guarda el grafo el cual se maneja en el programa.



Ejemplo de árbol rojinegro usado implementado en el proyecto.

Tomado de: https://es.wikipedia.org/wiki/Árbol_rojo-negro



Ejemplo del árbol de Huffman con código morse.

Tomado de: http://sedici.unlp.edu.ar/bitstream/handle/10915/71682/Documento_completo.pdf-PDFA.pdf?sequen