

PROYECTO 2022

Sebastian Orjuela Sanchez

Sergio Herrera Tellez

FACULTAD DE INGENIERÍA

INGENIERÍA DE SISTEMAS

2022

**DESCRIPCION**

**Grafo:**

Entrada: Imagen y secuencia de semilla

Salidas:El costo de cada arista está dado por la diferencia (o distancia) entre los colores

Condiciones:la imagen se considera un grafo

**HuffmanTree:**

entradas:imagen

salidas:codificar o descodificar la imagen

condiciones:

**Cargar\_Imagen:**

entradas: Nombre de la imagen

Salidas:mostrar los píxeles de la imagen

Condiciones:El nombre de la imagen se debe encontrar los archivos

**Cargar\_Volumen:**

entradas: nombre de la imagen a seleccionar

Salidas: tamaño de ancho y alto de píxeles ordenados

Condiciones: el nombre de la imagen debe coincidir para que retorne el número de píxeles

**info\_imagen**:

entradas:nombre de la imagen a seleccionar y píxeles de la misma imagen seleccionada

Salidas: muestra los píxeles de la imagen mostrando el ancho de píxeles y el alto de píxeles

Condiciones: la imagen debe estar cargada al principio para la lectura de esta misma

**info\_volumen**:

entradas:nombre de la imagen , cantidad de pixeles y tamaño de los pixeles

Salidas:mostrar el nombre base , cantidad de imágenes , ancho de píxeles y alto de píxeles

Condiciones:El volumen debe estar cargado en la memoria para que se pueda ejecutar de manera correcta

**proyeccion\_2D**:

entradas: entrada de píxeles ordenados

Salidas:serie de pixeles en la cual se enfoque en una parte o en una posición

Condiciones:los píxeles deben de estar organizados para que esta pueda hacer su proyección de manera correcta

**TAD UTILIZADOS**

**TAD Grafo :**

Datos minimos:

vertices, vector

aristas,vector

operaciones:

insertarVertice(T vertice);

insertarArista(T,T,U);

cantVertices();

cantAristas();

buscarVertice(T vertice);

buscarArista(T,T);

eliminarVertice(T Vertice);

eliminarArista(T,T);

llenarGrafo(T cantVertices, U cantAristas);

dijkstra(T,T);

**TAD HUFFMANTREE**

Datos minimos:

m\_Root

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

operaciones:

HuffmanTree();

~HuffmanTree();

crear(std::priority\_queue<HuffmanTreeNode\*, std::vector<HuffmanTreeNode\*>, compare>&arb);

Isvalid() const;

ordenar(std::map<char,int>&bases);

decode(const std::string& code\_queue) const;

encode( char character) const;

clear();

loadfromfile(const std::string& filename);

addcharacter(const char& c, const std::string& code);

PrintAsPNG(const std::string& filename) const;

encode(const std::map<char, int>& bases,int secuencias, std::map<std::string,std::pair<int,int> >&sSecu)const;

int numeroDeNodos();

encode(std::map<char,int>& m);

**Caegar\_imagen**

datos minimos:

nombre\_imagen

codigo,entero

ancho,entero

alto,entero

opercaciones:

obtenernombre()

obtenercodigo()

obtenerancho()

obteneralto()

fijarnombre()

fijarcodigo()

fijarancho()

fijaralto()

**Cargar\_volumen:**

datos minimos:

nombre\_imagen

operaciones:

nombreVolumen();

Tamano();

**info\_v**

datos minimos:

nombre base,cadena de caracteres

tamaño,entero

ancho,entero

alto,entero

operaciones:

obtnernombre()

obtnertamaño()

obtnerancho()

obtneralto()

fijarnombre()

fijartamaño()

fijarancho()

fijaralto()

**proyección2d**

datos minimos:

nombre\_archivo

operaciones:

obtnernombre()

fijarnombre()

**info\_imagen**

datos minimos:

nombre\_imagen,cadena de caracteres

ancho,entero

alto,entero

operaciones

obtnernombre()

obtnerancho()

obtneralto()

fijarnombre()

fijarancho()

fijaralto()

