UNIVERSIDAD DE SANTIAGO DE CHILE
FACULTAD DE INGENIERÍA
DEPARTAMENTO DE INGENIERÍA INFORMÁTICA

Estructura de Datos y Análisis de Algoritmos

Experiencia 4: Árboles.

Andrés Felipe Muñoz Bravo
19.646.487-5

Profesor:
Mario Inostroza Ponta
Ayudante:
Javiera Torres Muñoz

Santiago - Chile
2016

TABLA DE CONTENIDOS

ÍNDICE DE FIGURAS

No se encuentran elementos de tabla de ilustraciones.

ÍNDICE DE TABLAS

CAPÍTULO 1. INTRODUCCIÓN

Se ha pedido realizar un programa para analizar imágenes según un criterio de uniformidad, para esto es necesaria la implementación del tipo de dato abstracto árbol, en este caso particular se utilizan arboles cuaternarios, los cuales solo tienen 4 hijos. Estos hijos a su vez también son árboles que pueden tener más hijos y así sucesivamente. La función que cumple este tipo de dato, es poder realizar una partición de una imagen, si es que esta no cumple con la condición introducida por el usuario. De esta forma el programa crea grupos de imágenes en los cuales se cumple la condición requerida para cada uno de estos. El funcionamiento del programa se detalla a lo largo de este informe, además de realizar un análisis de lo desarrollado.

Objetivos:

1. Implementar un TDA imagen, con funciones para cargar desde un archivo de texto.
2. Implementar un TDA árbol con funciones para realizar divisiones las imágenes contenidas dentro de este.
3. Agrupar las imágenes divididas que contienen el mismo nivel de uniformidad, esto refiere a la condición con la cual se dividen la imagen en caso de no cumplirse.

Para llevar a cabo este programa se ha trabajado árboles cuaternarios, utilizando el lenguaje de programación C, con el paradigma de programación imperativo procedural.

CAPÍTULO 2. DESCRIPCIÓN DE LA SOLUCIÓN

Para el desarrollo de este programa es utilizada la herramienta sublime text 3, el cual es un editor de texto. La principal ventaja de este editor es que utiliza colores para las diferentes palabras reservadas del lenguaje de programación C, de esta forma ayuda al programador revisar la sintaxis programada. A continuación se explicara detalladamente las estructuras herramientas, técnicas y las principales funciones del programa realizado.

2.1 ESTRUCTURAS

Para el programa se utilizan dos estructuras; Imagen y Arbol. Con estas dos estructuras es posible llevar a cabo el desarrollo del programa. A continuación se detallaran las componentes de estas.

2.1.1 Imagen:

Dentro de la estructura de Imagen se encuentran definido 5 datos:

1. Un doble puntero a entero, con eso se logra una matriz de enteros para guardar los valores de cada pixel de la imagen.
2. Un entero para guardar la dimensión de la imagen, cabe recalcar que solo basta con un valor dado que la imagen debe ser cuadrada.
3. Un entero para guardar la uniformidad de la imagen.
4. Un entero para guardar el valor máximo entre los pixeles guardados en la matriz.
5. Un entero para guardar el valor mínimo entre los pixeles guardados en la matriz.

2.1.2 Árbol:

Dentro de la estructura árbol se encuentran definido 4 datos:

1. Puntero a una estructura imagen.
2. Entero para guardar la fila desde donde comienza la imagen al momento de ser particionada.
3. Entero para guardar la columna desde donde comienza la imagen al momento de ser particionada.
4. Arreglo de 4 punteros a la misma estructura árbol, para guardar las 4 particiones de la imagen.

2.2 FUNCIONES

2.2.1 Partición masiva (Arbol* arbol, int condicion,int* id,Imagen* imagen):

Esta es la función principal del programa, la cual recibe un árbol con la imagen inicial carga en él, un entero con la condición de uniformidad que debe cumplir cada partición de la imagen, un id pasado por referencia para marcar los diferentes grupos, y una imagen pasada por referencia también en la cual se guardaran las marcas realizadas.

Cabe recalcar que dentro de esta función subyacen más funciones para llevar a cabo el proceso de partición total y fusión posterior. A continuación se presenta el algoritmo empleado.

Algoritmo:

- Si la imagen del árbol tiene uniformidad mayor a la condición entregada y la dimensión de la imagen del árbol es mayor a uno:
    - Se realiza una partición al árbol.
    - Se hace una llamada recursiva entregando como árbol el primer hijo de la partición realizada.
    - Se hace una llamada recursiva entregando como árbol el segundo hijo de la partición realizada.
    - Se hace una llamada recursiva entregando como árbol el tercer hijo de la partición realizada.
    - Se hace una llamada recursiva entregando como árbol el cuarto hijo de la partición realizada.
    - Se marcar los grupos respectivos si cumplen la condición necesaria.
- En caso contrario :
    - Se pregunta si el árbol entregado es hoja, en caso afirmativo hacer:
        - Marcar el grupo entero de la imagen entregada, esto quiere decir que se pinta la imagen marca la imagen completa debido a que cumple con la condición de uniformidad.

2.2.2 marcar(Arbol* arbol,Imagen* imagen,int* id):

Esta función marca en una imagen con el id entregado todas las posiciones de la imagen del árbol, pero es necesario saber en qué parte de la imagen principal pertenece esta subimagen del subárbol del árbol principal entregado, esta parte es muy difícil de explicar, por lo que se presenta el algoritmo a continuación:

Algoritmo:

- fil = arbol.fila (asignacion)
- col = arbol.col (asignacion)
- para i=0 hasta arbol.imagen.dimension:
    - para j = 0 hasta arbol.imagen.dimension:
    - imagen.matriz[i][j] = *id; (asignación y desreferenciacion)

2.2.3 marcarGrupos(Arbol* arbol, int condicion,int* id,Imagen* imagen):

Función que verifica la uniformidad de las imágenes adyacentes y marca aquellas imágenes que cumplen con la condición de uniformidad al momento de unirlas, para eso se entrega el mismo id a la función marcar (anteriormente detallada).

2.2.4 particionarArbol(Arbol* arbol):

Esta función realiza la partición de un árbol, creando los cuatro hijos, los cuales son nuevos árboles con sus respectivas imágenes particionadas desde el árbol anterior. A continuación se presenta el algoritmo:

Algoritmo:

- Se inicializan los cuatros arboles dentro del arreglo hijos contenido en el árbol
- Se inicializan 4 imágenes
- Para i = 0 hasta la dimensión de la imagen del árbol principal
    - Para j = 0 hasta la dimensión de la imagen del árbol principal
        - Se ingresan los datos la partición respectiva a cada imagen inicializada anteriormente.
- Se actualiza la información de las imágenes creadas, esto quiere decir que se calcula la uniformidad, el mínimo y máximo para cada una de ellas.
- Se carba a los arboles inicializados (hijos) las imágenes creadas (particiones)
- Se actualiza la fila y columna de cada hijo, esto se refiere a la ubicación en la cual inicia cada imagen particionada dentro de la imagen principal.

CAPÍTULO 3. ANÁLISIS DE LOS RESULTADOS

3.1 TIEMPOS Y ORDEN DE LAS FUNCIONES

A continuación se muestran todas las funciones del programa con su respectivo orden:

Función	T(n)	Orden
existsFile	5	O(n)
rtrim	n	O(n)
inicializarMatriz	n^2+n+4	O(n^2)
showMatriz	n^2+n+2	O(n^2)
inicializarImagen	n^2+n+9	O(n^2)
showImagen	n^2+n+5	O(n^2)
actualizarInfo	4n^2+5	O(n^2)
destruirImagen	n+3	O(n)
createImagen	7n^2+n+21	O(n^2)
inicializarArbol	9	O(1)
cargarImagenArbol	3	O(1)
particionarArbol	24n^2+4n+114	O(n^2)
marcar	n^2+4	O(n^2)
marcarGrupos	20n^2+128	O(n)
uniformidadDosImagenes	5	O(1)
uniformidadTresImagenes	5	O(1)
particionMasiva	4T(n/4)+ O(n^2)	O(n^2)
esHoja	5	O(1)
recorrido	4T(n/4)+ O(n^2)	O(n^2)
esNumero	4	O(1)
titulo	6	O(1)

Tabla 1: Tiempo y orden de funciones para grafos.

3.2 RESULTADOS ENTREGADOS

Como se puede apreciar en el capítulo anterior el orden mayor es n^2, lo cual significa que la implementación del programa ha sido realizada de muy buena forma, lo que conlleva a que sea un programa eficiente.

Dentro de los resultados obtenidos en el programa, se puede mencionar que son los esperados, y funciona en un 100% para todos los casos. De las pruebas realizadas en ningún momento dejo de funcionar el programa.

CAPÍTULO 4. CONCLUSIÓN

En esta cuarta entrega se cumplió el objetivo completamente. Dentro lo los objetivos que se cumplieron están implementar el TDA imagen con las funciones para cargar desde un archivo de texto, como también implementar el TDA árbol con las funciones para realizar divisiones de las imágenes, y fusión de estas mismas que cumplen con una condición previamente ingresada.

La importancia de este laboratorio, es que sirvió para entender de una forma más práctica el cómo funciona la representación de árboles dentro de una computadora, además de cómo operan los algoritmos de búsqueda para estos.

Cabe recalcar que estas representaciones y problemas tan complejos y complicados de resolver para una persona o incluso cientos y miles de personas, pueden ser resueltos de una forma mucho más rápida por un computador, solo basta representar de una buena forma.

CAPÍTULO 5. REFERENCIAS

Departamento de Ingeniería Informática USACH. (2016). Enunciado Laboratorio 4. 2016: USACH.

DCC Universidad de Chile (2007). *Tipos de datos abstractos*. (2016) https://users.dcc.uchile.cl/~bebustos/apuntes/cc30a/TDA/.

Departamento de Ingeniería Informática USACH. (2016). Apuntes de la asignatura, Análisis de algoritmos y estructura de datos. Jacqueline Köhler C. 2016: USACH.