Universidad de Costa Rica
Facultad de Ingeniería
Escuela de Ciencias de la Computación e Informática

Estructuras de Datos y Análisis de Algoritmos
CI-1221
Grupo 001

I Tarea programada

Profesora:
Sandra Kikut

Elaborado por:
Andreína Alvarado González | B40259
Otto Mena Kikut | B23753

25 de Septiembre del 2015

Índice.

Introducción.

Una de las partes fundamentales, a la hora de aprender sobre estructuras de datos y análisis de algoritmos, es entender bien como están constituidas las distintas partes que se deben tener en cuenta a la hora de hablar sobre modelos matemáticos, estructuras de datos, algoritmos, ... Esta tarea, consta de cuatro etapas cuyo fin es precisamente, tratar de comprender todas estas partes y la manera correcta de concebirlas.

Una primer etapa, se enfoca en la especificación de los modelos matemáticos. Es decir, su definición formal y el establecimiento de operadores básicos que permitan la implementación de cualquier algoritmo que se desee a partir de estos operadores básicos. Así mismo, a estos operadores se le definiran sus clásulas de efecto, requiere y modifica.

En una segunda etapa, se pretende implementar mediante estructuras de datos específicas, estos modelos con sus operadores básicos específicos a nivel computacional.

La tercer etapa, se basa en la implentación, de igual manera, a nivel computacional de algoritmos para árboles n-arios, tomando en cuenta, que estos algoritmos deben funcionar independientemente de qué estructura de datos se esté usando.

Finalmente, durante las lecciones, se han establecido ciertos órdenes de duración para algunos algoritmos y operadores básicos. En la cuarta etapa, se pretende realizar una serie de análisis teórico y de tiempo real de ejecución, con el fin de conocer, establecer y esperimentar respecto a la realidad entre la teoría y la práctica de la duración de estos algoritmos.

Objetivos.

• Definir, especificar, implementar y usar los modelos lógicos Cola, Pila, Árbol n-ario tal que sí importa el orden entre los hijos de un nodo.

• Realizar un análisis teórico y un análisis real del tiempo de ejecución de las diferentes estructuras de datos y algoritmos utilizados.

Enunciado.

Para la primer etapa:

Especificar de manera lógica, formal y completa los operadores básicos de la Cola, Pila y Árbol n-ario. Para cada operador debe incluir: nombre, parámetros con sus tipos y las cláusulas Efecto (claro, completo y conciso), Requiere y Modifica.

Para la segunda etapa:

- Implementar el modelo Cola utilizando la estructura de datos: arreglo circular.
- Implementar el modelo Pila utilizando la estructura de datos: lista simplemente enlazada.
- Implementar el modelo Árbol n-ario tal que sí importa el orden entre los hijos de un nodo utilizando las estructuras de datos: arreglo con señalador al padre; lista de hijos por lista implemente enlazada (lista principal) y lista simplemente enlazada (sublistas); hijo más izquierdo-hermano derecho por punteros; e hijo más izquierdo-hermano derecho por punteros, con puntero al padre y al hermano izquierdo.

Para la tercera etapa:

Especificar distintos algoritmos para el modelo árbol tal que sí importa el orden entre los hijos de un nodo, utilizando sus operadores básicos. Para cada algoritmo se debe especificar nombre, parámetros con sus tipos y las cláusulas efecto (claro, completo y conciso), requiere y modifica.

Además, hacer un programa de prueba de los algoritmos implementados. Este programa, deberá permitir usar los operadores básicos del árbol.

Para la cuarta etapa:

Hacer un análisis empírico (tiempo y espacio real) de la complejidad computacional de las estructuras de datos, operadores básicos y algoritmos implementados en esta tarea. Para ello, se deberá hacer cálculos de tiempo real de ejecución de los diferentes operadores y algoritmos, para diferentes tamaños de n (n muy grandes) y para diferentes tipos de árboles (diferentes alturas y anchuras). Dichos cálculos deberán ser mostrados en tablas y gráficos.

Además, se deberá comparar los cálculos reales con los teóricos e incluir una sección de conclusiones sobre la eficiencia de cada estructura de datos.

Desarrollo.

Modelos

Modelo Cola. Definición del Modelo Cola.

Según el libro *Estructura de datos y algoritmos* de Alfred V. Aho, Jeffrey D. Ullman y John E. Hopcroft: una *cola* es un tipo especial de lista en el cual los elementos se insertan en un extremo *el posterior* y se suprimen en el otro *el anterior o frente*. Las colas a menudo se les conoce también como "FIFO" o lista "primero en entrar, primero en salir".

Modelo Cola. Definición y especificación de operadores básicos del Modelo Cola.

Crear(cola C):

Cláusulas del operador básico Crear(cola C)

Efecto	Inicializa la cola C como vacía.
Requiere	No aplica.
Modifica	Cola C.

Destruir(cola C):

Cláusulas del operador básico Destruir(cola C)

Efecto	Destruye la cola C, dejándola inutilizable.
Requiere	Cola C inicializada.
Modifica	Cola C.

Vaciar(cola C):

Cláusulas del operador básico Vaciar(cola C)

Efecto	Vacía cola C, dejándola con 0 elementos.
Requiere	Cola C inicializada.
Modifica	Cola C.

Vacía(cola C), devuelve algo de tipo booleano:

Cláusulas del operador básico Vacía(cola C)

Efecto	Devuelve verdadero si cola C está vacía y falso si no.
Requiere	Cola C inicializada.
Modifica	No aplica.

Agregar(elemento e, cola C):

Cláusulas del operador básico Agregar(elemento e, cola C)

Efecto	Agrega un nuevo elemento en la cola C, de manera que este elemento quede en la parte de atrás de la cola.
Requiere	Cola C inicializada.
Modifica	Cola C.

Sacar(cola C):

Cláusulas del operador básico Sacar(cola C)

Efecto	Borra el primer elemento en cola C.
Requiere	Cola C inicializada y con al menos un elemento.
Modifica	Cola C.

Frente(cola C), devuelve algo de tipo elemento:

Cláusulas del operador básico Frente(cola C)

Efecto	Devuelve el primer elemento en cola C.
Requiere	Cola C inicializada y con al menos un elemento.
Modifica	No aplica.

Modelo Pila. Definición del Modelo Pila.

Según el libro *Estructura de datos y algoritmos* de Alfred V. Aho, Jeffrey D. Ullman y John E. Hopcroft: una pila es un tipo especial de lista en la que todas las inserciones y supresiones tienen lugar en un extremo denominado *tope*. A menudo a las pilas también se les conoce como "LIFO" o listas "último en entrar, primero en salir".

Modelo Pila. Definición y especificación de operadores básicos del Modelo Pila.

Iniciar(pila P):

Cláusulas del operador básico Iniciar(pila P)

Efecto	Inicializa la pila P como vacía.
Requiere	No aplica.
Modifica	Pila P.

Destruir(pila P):

Cláusulas del operador básico Destruir(pila P)

Efecto	Destruye la pila P, dejándola inutilizable.
Requiere	Pila P inicializada.
Modifica	Pila P.

Vaciar(pila P):

Cláusulas del operador básico Vaciar(pila P)

Efecto	Vacía pila P, dejándola con 0 elementos.
Requiere	Pila P inicializada.
Modifica	Pila P.

Vacía(pila P), devuelve algo de tipo booleano:

Cláusulas del operador básico Vacía(pila P)

Efecto	Devuelve verdadero si pila P está vacía y falso si no.
Requiere	Pila P inicializada.
Modifica	No aplica.

Poner(elemento e, pila P):

Cláusulas del operador básico Poner(elemento e, pila P)

Efecto	Agrega un nuevo elemento en la pila P.
Requiere	Pila P inicializada.
Modifica	Pila P.

Quitar(pila P):

Cláusulas del operador básico Quitar(pila P)

Efecto	Borra el último elemento que se puso en pila P.
Requiere	Pila P inicializada y con al menos un elemento.
Modifica	Pila P.

Tope(pila P), devuelve algo de tipo elemento:

Cláusulas del operador básico Tope(pila P)

Efecto	Devuelve el último elemento que se puso en pila P.
Requiere	Pila P inicializada y con al menos un elemento.
Modifica	No aplica.

Modelo Árbol n-ario. Definición del Modelo Árbol n-ario.

Según el libro *Estructura de datos y algoritmos* de Alfred V. Aho, Jeffrey D. Ullman y John E. Hopcroft: en general, un árbol impone una estructura jerárquica sobre una colección de objetos. En específico, un árbol es una colección de elementos llamados *nodos*, uno de los cuales se distingue como *raíz*, junto con una relación de "paternidad" que impone una estructura jerárquica sobre los nodos. Un nodo, como un elemento de una lista, puede ser del tipo que se desee. A menudo se representa un nodo por medio de una letra, una cadena de caracteres o un círculo con un número en su interior. Formalmente, un árbol se puede definir de manera recursiva como sigue:

(1) Un solo nodo es, por sí mismo, un árbol. Ese nodo es también la raíz de dicho árbol.
(2) Supóngase que n es un nodo y que $A_1, A_2, ..., A_k$ son árboles con raíces $n_1, n_2, ..., n_k$, respectivamente. Se puede construir un nuevo árbol haciendo que n se constituya en el padre de los nodos $n_1, n_2, ..., n_k$. En dicho árbol, n es la raíz y $A_1, A_2, ..., A_k$ son los *subárboles* de la raíz. Los nodos $n_1, n_2, ..., n_k$ reciben el nombre de *hijos* del nodo n.

Además, dos árboles A_1 y $A2$, son iguales sí y solo sí todos los hijos de cada nodo $n_{1,1}, n_{1,2}, ..., n_{1,m}$ del árbol A_1, se encuentra en el mismo orden que los hijos de los nodos $n_{2,1}, n_{2,2}, ..., n_{2,m}$ del árbol A_2. Es decir, importa el orden entre los hijos.

Modelo Árbol n-ario. Definición y especificación de operadores básicos del Modelo Árbol n-ario.

Crear(árbol A)

Cláusulas del operador básico Crear(árbol A)

Efecto	Inicializa el árbol A como vacío.
Requiere	No aplica.
Modifica	Árbol A.

Destruir(árbol A):

Cláusulas del operador básico Destruir(árbol A)

Efecto	Destruye el árbol A, dejándolo inutilizable.
Requiere	Árbol A inicializado.
Modifica	Árbol A.

Vaciar(árbol A):

Cláusulas del operador básico Vaciar(árbol A)

Efecto	Vacía árbol A, dejándolo sin etiquetas.
Requiere	Árbol A inicializado.
Modifica	Árbol A.

Vacío(árbol A), devuelve algo de tipo booleano:

Cláusulas del operador básico Vacío(árbol A)

Efecto	Devuelve verdadero si árbol A está vacío y falso si no.
Requiere	Árbol A inicializado.
Modifica	No aplica.

PonerRaíz(elemento e, árbol A):

Cláusulas del operador básico PonerRaíz(elemento e, árbol A)

Efecto	Agrega la raíz de árbol A.
Requiere	Árbol A inicializado y con 0 elementos.
Modifica	Árbol A.

AgregarHijo(elemento e, nodo n, árbol A):

Cláusulas del operador básico AgregarHijo(etiqueta e, nodo n, árbol A)

Efecto	Le agrega un nuevo hijo al nodo n con etiqueta e en árbol A.
Requiere	Árbol A inicializado y n válido en A.
Modifica	Árbol A.

BorrarHoja(nodo n, árbol A):

Cláusulas del operador básico BorrarHoja(nodo n, árbol A)

Efecto	Elimina la hoja n en árbol A.
Requiere	Árbol A inicializado, n válido en A y n hoja.
Modifica	Nodo n en árbol A.

ModificarEtiqueta(etiqueta e, nodo n, árbol A):

Cláusulas del operador básico ModificarEtiqueta(etiqueta e, nodo n, árbol A)

Efecto	Modifica la etiqueta del nodo n por e en árbol A.
Requiere	Árbol A inicializado y nodo n válido en A.
Modifica	Nodo n en árbol A.

Raíz(árbol A), devuelve algo de tipo nodo:

Cláusulas del operador básico Raíz(árbol A)

Efecto	Devuelve el nodo raíz en árbol A.
Requiere	Árbol A inicializado y con al menos 1 elemento.
Modifica	No aplica.

Padre(nodo n, árbol A), devuelve algo de tipo nodo:

Cláusulas del operador básico Padre(nodo n, árbol A)

Efecto	Devuelve el nodo padre del nodo n en árbol A.
Requiere	Árbol A inicializado y nodo n válido en A.
Modifica	No aplica.

HijoMásIzquierdo(nodo n, árbol A), devuelve algo de tipo nodo:

Cláusulas del operador básico HijoMásIzquierdo(nodo n, árbol A)

Efecto	Devuelve el hijo más izquierdo del nodo n en árbol A.
Requiere	Árbol A inicializado y n válido en A. Si el nodo n no tiene hijos, este método devuelve $nodoNulo$.
Modifica	No aplica.

HermanoDerecho(nodo n, árbol A), devuelve algo de tipo nodo:

Cláusulas del operador básico HermanoDerecho(nodo n, árbol A)

Efecto	Devuelve el hermano derecho del nodo n en árbol A.
Requiere	Árbol A inicializado y n válido en A. Si el nodo n no tiene hermanos derechos, este método devuelve $nodoNulo$.
Modifica	No aplica.

Etiqueta(nodo n, árbol A), devuelve algo de tipo etiqueta:

Cláusulas del operador básico Etiqueta(nodo n, árbol A)

Efecto	Devuelve la etiqueta del nodo n en árbol A.
Requiere	Árbol A inicializado y n válido en A.
Modifica	No aplica.

Bibliografía.

[1] Alfred V. Aho, Jeffrey D. Ullman, John E. Hopcroft. *Estructura de datos y algoritmos.* 1988.