

Algoritmos y Estructuras de Datos

Mario Storti, Jorge D'Elía, Rodrigo Paz, Lisandro Dalcín, y Martín Pucheta

<{jdelia,mpucheta}@intec.unl.edu.ar>,
<{mario.storti,dalcinl,rodrigo.r.paz}@gmail.com>

www: <http://www.cimec.org.ar/aed>

Facultad de Ingeniería y Ciencias Hídricas

Universidad Nacional del Litoral <http://fich.unl.edu.ar>

Centro Internacional de Métodos Computacionales en Ingeniería

<http://www.cimec.org.ar>

(Document version: aed-2.0.4-102-g184c1cd 'clean)

(Date: Sat Feb 25 15:25:54 2012 -0300)

Indice

1. Diseño y análisis de algoritmos	9
1.1. Conceptos básicos de algoritmos	9
1.1.1. Ejemplo: Sincronización de acceso a objetos en cálculo distribuido	10
1.1.2. Introducción básica a grafos	11
1.1.3. Planteo del problema mediante grafos	11
1.1.4. Algoritmo de búsqueda exhaustiva	12
1.1.5. Generación de las coloraciones	13
1.1.6. Crecimiento del tiempo de ejecución	15
1.1.7. Búsqueda exhaustiva mejorada	16
1.1.8. Algoritmo heurístico ávido	19
1.1.9. Descripción del algoritmo heurístico en pseudo-código	21
1.1.10. Crecimiento del tiempo de ejecución para el algoritmo ávido	26
1.1.11. Conclusión del ejemplo	27
1.2. Tipos abstractos de datos	27
1.2.1. Operaciones abstractas y características del TAD CONJUNTO	28
1.2.2. Interfase del TAD CONJUNTO	28
1.2.3. Implementación del TAD CONJUNTO	30
1.3. Tiempo de ejecución de un programa	30
1.3.1. Notación asintótica	32
1.3.2. Invariancia ante constantes multiplicativas	33
1.3.3. Invariancia de la tasa de crecimiento ante valores en un conjunto finito de puntos	33
1.3.4. Transitividad	34
1.3.5. Regla de la suma	34
1.3.6. Regla del producto	34
1.3.7. Funciones típicas utilizadas en la notación asintótica	34
1.3.8. Equivalencia	36
1.3.9. La función factorial	36
1.3.10. Determinación experimental de la tasa de crecimiento	37
1.3.11. Otros recursos computacionales	38
1.3.12. Tiempos de ejecución no-polinomiales	39
1.3.13. Problemas P y NP	39
1.3.14. Varios parámetros en el problema	40
1.4. Conteo de operaciones para el cálculo del tiempo de ejecución	40

1.4.1. Bloques if	40
1.4.2. Lazos	41
1.4.3. Suma de potencias	45
1.4.4. Llamadas a rutinas	45
1.4.5. Llamadas recursivas	46
2. Tipos de datos abstractos fundamentales	48
2.1. El TAD Lista	48
2.1.1. Descripción matemática de las listas	49
2.1.2. Operaciones abstractas sobre listas	49
2.1.3. Una interfase simple para listas	50
2.1.4. Funciones que retornan referencias	52
2.1.5. Ejemplos de uso de la interfase básica	54
2.1.6. Implementación de listas por arreglos	58
2.1.6.1. Eficiencia de la implementación por arreglos	63
2.1.7. Implementación mediante celdas enlazadas por punteros	64
2.1.7.1. El tipo posición	65
2.1.7.2. Celda de encabezamiento	66
2.1.7.3. Las posiciones <code>begin()</code> y <code>end()</code>	68
2.1.7.4. Detalles de implementación	69
2.1.8. Implementación mediante celdas enlazadas por cursores	70
2.1.8.1. Cómo conviven varias celdas en un mismo espacio	72
2.1.8.2. Gestión de celdas	73
2.1.8.3. Analogía entre punteros y cursores	73
2.1.9. Tiempos de ejecución de los métodos en las diferentes implementaciones.	76
2.1.10. Interfase STL	77
2.1.10.1. Ventajas de la interfase STL	77
2.1.10.2. Ejemplo de uso	78
2.1.10.2.1. Uso de plantillas y clases anidadas	78
2.1.10.2.2. Operadores de incremento prefijo y postfijo:	79
2.1.10.3. Detalles de implementación	79
2.1.10.4. Listas doblemente enlazadas	82
2.2. El TAD pila	82
2.2.1. Una calculadora RPN con una pila	83
2.2.2. Operaciones abstractas sobre pilas	84
2.2.3. Interfase para pila	84
2.2.4. Implementación de una calculadora RPN	85
2.2.5. Implementación de pilas mediante listas	88
2.2.6. La pila como un adaptador	89
2.2.7. Interfase STL	90
2.3. El TAD cola	90
2.3.1. Intercalación de vectores ordenados	91
2.3.1.1. Ordenamiento por inserción	91
2.3.1.2. Tiempo de ejecución	93

2.3.1.3. Particularidades al estar las secuencias pares e impares ordenadas	93
2.3.1.4. Algoritmo de intercalación con una cola auxiliar	94
2.3.2. Operaciones abstractas sobre colas	95
2.3.3. Interfase para cola	95
2.3.4. Implementación del algoritmo de intercalación de vectores	96
2.3.4.1. Tiempo de ejecución	97
2.4. El TAD correspondencia	97
2.4.1. Interfase simple para correspondencias	100
2.4.2. Implementación de correspondencias mediante contenedores lineales	102
2.4.3. Implementación mediante contenedores lineales ordenados	103
2.4.3.1. Implementación mediante listas ordenadas	105
2.4.3.2. Interfase compatible con STL	106
2.4.3.3. Tiempos de ejecución para listas ordenadas	109
2.4.3.4. Implementación mediante vectores ordenados	110
2.4.3.5. Tiempos de ejecución para vectores ordenados	112
2.4.4. Definición de una relación de orden	113
3. Árboles	114
3.1. Nomenclatura básica de árboles	114
3.1.0.0.1. Altura de un nodo.	116
3.1.0.0.2. Profundidad de un nodo. Nivel.	116
3.1.0.0.3. Nodos hermanos	116
3.2. Orden de los nodos	116
3.2.1. Particionamiento del conjunto de nodos	117
3.2.2. Listado de los nodos de un árbol	119
3.2.2.1. Orden previo	119
3.2.2.2. Orden posterior	119
3.2.2.3. Orden posterior y la notación polaca invertida	120
3.2.3. Notación Lisp para árboles	121
3.2.4. Reconstrucción del árbol a partir de sus órdenes	122
3.3. Operaciones con árboles	124
3.3.1. Algoritmos para listar nodos	124
3.3.2. Inserción en árboles	125
3.3.2.1. Algoritmo para copiar árboles	126
3.3.3. Supresión en árboles	128
3.3.4. Operaciones básicas sobre el tipo árbol	129
3.4. Interfase básica para árboles	129
3.4.1. Listados en orden previo y posterior y notación Lisp	132
3.4.2. Funciones auxiliares para recursión y sobrecarga de funciones	133
3.4.3. Algoritmos de copia	133
3.4.4. Algoritmo de poda	133
3.5. Implementación de la interfase básica por punteros	134
3.5.1. El tipo iterator	134
3.5.2. Las clases cell e iterator_t	136

3.5.3. La clase tree	139
3.6. Interfase avanzada	141
3.6.1. Ejemplo de uso de la interfase avanzada	145
3.7. Tiempos de ejecución	148
3.8. Árboles binarios	148
3.8.1. Listados en orden simétrico	149
3.8.2. Notación Lisp	149
3.8.3. Árbol binario lleno	149
3.8.4. Operaciones básicas sobre árboles binarios	150
3.8.5. Interfases e implementaciones	151
3.8.5.1. Interfase básica	151
3.8.5.2. Ejemplo de uso. Predicados de igualdad y espejo	151
3.8.5.3. Ejemplo de uso. Hacer espejo "in place"	153
3.8.5.4. Implementación con celdas enlazadas por punteros	154
3.8.5.5. Interfase avanzada	159
3.8.5.6. Ejemplo de uso. El algoritmo apply y principios de programación funcional.	160
3.8.5.7. Implementación de la interfase avanzada	161
3.8.6. Árboles de Huffman	164
3.8.6.1. Condición de prefijos	166
3.8.6.2. Representación de códigos como árboles de Huffman	166
3.8.6.3. Códigos redundantes	167
3.8.6.4. Tabla de códigos óptima. Algoritmo de búsqueda exhaustiva	168
3.8.6.4.1. Generación de los árboles	169
3.8.6.4.2. Agregando un condimento de programación funcional	171
3.8.6.4.3. El algoritmo de combinación	172
3.8.6.4.4. Función auxiliar que calcula la longitud media	174
3.8.6.4.5. Uso de comb y codelen	175
3.8.6.5. El algoritmo de Huffman	176
3.8.6.6. Implementación del algoritmo	178
3.8.6.7. Un programa de compresión de archivos	181
4. Conjuntos	190
4.1. Introducción a los conjuntos	190
4.1.1. Notación de conjuntos	190
4.1.2. Interfase básica para conjuntos	191
4.1.3. Análisis de flujo de datos	192
4.2. Implementación por vectores de bits	197
4.2.1. Conjuntos universales que no son rangos contiguos de enteros	198
4.2.2. Descripción del código	198
4.3. Implementación con listas	200
4.3.0.1. Similitud entre los TAD conjunto y correspondencia	200
4.3.0.2. Algoritmo lineal para las operaciones binarias	201
4.3.0.3. Descripción de la implementación	203
4.3.0.4. Tiempos de ejecución	206

Esta es una muestra, haga clic en el enlace de descarga para obtener el tutorial completo

