DEFINICIONES 2

FABIO DIAZ RAIGOSO

561 210 210

PRESENTADO AL DOCENTE

ING. JORGE RUBIANO

UNIVERSIDAD DE CUNDINAMARCA

INTELIGENCIA ARTIFICIAL

CHIA, CUNDINAMARCA

2014

En computación y matemáticas un algoritmo de ordenamiento es un algoritmo que pone elementos de una lista o un vector en una secuencia dada por una relación de orden, es decir, el resultado de salida ha de ser una permutación —o reordenamiento— de la entrada que satisfaga la relación de orden dada. Las relaciones de orden más usadas son el orden numérico y el orden lexicográfico. Ordenamientos eficientes son importantes para optimizar el uso de otros algoritmos (como los de búsqueda y fusión) que requieren listas ordenadas para una ejecución rápida. También es útil para poner datos en forma canónica y para generar resultados legibles por humanos.

Los algoritmos de ordenamiento se pueden clasificar de las siguientes maneras:

La más común es clasificar según el lugar donde se realice la ordenación

**Algoritmos de ordenamiento interno**: en la memoria del ordenador.

**Algoritmos de ordenamiento externo**: en un lugar externo como un disco duro.

Por el tiempo que tardan en realizar la ordenación, dadas entradas ya ordenadas o inversamente ordenadas:

**Algoritmos de ordenación natural**: Tarda lo mínimo posible cuando la entrada está ordenada.

**Algoritmos de ordenación no natural**: Tarda lo mínimo posible cuando la entrada está inversamente ordenada.

La **Ordenación de burbuja** (Bubble Sort en inglés) es un sencillo algoritmo de ordenamiento. Funciona revisando cada elemento de la lista que va a ser ordenada con el siguiente, intercambiándolos de posición si están en el orden equivocado. Es necesario revisar varias veces toda la lista hasta que no se necesiten más intercambios, lo cual significa que la lista está ordenada. Este algoritmo obtiene su nombre de la forma con la que suben por la lista los elementos durante los intercambios, como si fueran pequeñas "burbujas". También es conocido como el método del intercambio directo. Dado que solo usa comparaciones para operar elementos, se lo considera un algoritmo de comparación, siendo el más sencillo de implementar.

Éste algoritmo es esencialmente un algoritmo de fuerza bruta

El **ordenamiento rápido** (quicksort en inglés) es un algoritmo creado por el científico británico en computación C. A. R. Hoare basado en la técnica de divide y vencerás, que permite, en promedio, ordenar n elementos en un tiempo proporcional a n log n.

El **ordenamiento por montículos** (heapsort en inglés) es un algoritmo de ordenamiento no recursivo, no estable, con complejidad computacional \Theta(n\log n)

Este algoritmo consiste en almacenar todos los elementos del vector a ordenar en un montículo (heap), y luego extraer el nodo que queda como nodo raíz del montículo (cima) en sucesivas iteraciones obteniendo el conjunto ordenado. Basa su funcionamiento en una propiedad de los montículos, por la cual, la cima contiene siempre el menor elemento (o el mayor, según se haya definido el montículo) de todos los almacenados en él. El algoritmo, después de cada extracción, recoloca en el nodo raíz o cima, la última hoja por la derecha del último nivel. Lo cual destruye la propiedad heap del árbol. Pero, a continuación realiza un proceso de "descenso" del número insertado de forma que se elige a cada movimiento el mayor de sus dos hijos, con el que se intercambia. Este intercambio, realizado sucesivamente "hunde" el nodo en el árbol restaurando la propiedad montículo del árbol y dejando paso a la siguiente extracción del nodo raíz.

El algoritmo, en su implementación habitual, tiene dos fases. Primero una fase de construcción de un montículo a partir del conjunto de elementos de entrada, y después, una fase de extracción sucesiva de la cima del montículo. La implementación del almacén de datos en el heap, pese a ser conceptualmente un árbol, puede realizarse en un vector de forma fácil. Cada nodo tiene dos hijos y por tanto, un nodo situado en la posición i del vector, tendrá a sus hijos en las posiciones 2 x i, y 2 x i +1 suponiendo que el primer elemento del vector tiene un índice = 1. Es decir, la cima ocupa la posición inicial del vector y sus dos hijos la posición segunda y tercera, y así, sucesivamente. Por tanto, en la fase de ordenación, el intercambio ocurre entre el primer elemento del vector (la raíz o cima del árbol, que es el mayor elemento del mismo) y el último elemento del vector que es la hoja más a la derecha en el último nivel. El árbol pierde una hoja y por tanto reduce su tamaño en un elemento. El vector definitivo y ordenado, empieza a construirse por el final y termina por el principio.

El **ordenamiento por casilleros** (bucket sort o bin sort, en inglés) es un algoritmo de ordenamiento que distribuye todos los elementos a ordenar entre un número finito de casilleros. Cada casillero sólo puede contener los elementos que cumplan unas determinadas condiciones. En el ejemplo esas condiciones son intervalos de números. Las condiciones deben ser excluyentes entre sí, para evitar que un elemento pueda ser clasificado en dos casilleros distintos. Después cada uno de esos casilleros se ordena individualmente con otro algoritmo de ordenación (que podría ser distinto según el casillero), o se aplica recursivamente este algoritmo para obtener casilleros con menos elementos. Se trata de una generalización del algoritmo Pigeonhole sort. Cuando los elementos a ordenar están uniformemente distribuidos la complejidad computacional de este algoritmo es de O(n).

El algoritmo contiene los siguientes pasos:

1. Crear una colección de casilleros vacíos

2. Colocar cada elemento a ordenar en un único casillero

3. Ordenar individualmente cada casillero

4. devolver los elementos de cada casillero concatenados por orden {1}

**Teorema Minimax**

John von Neumann es el creador del teorema minimax, quien dio la siguiente noción de lo que era un juego:

"Un juego es una situación conflictiva en la que uno debe tomar una decisión sabiendo que los demás también toman decisiones, y que el resultado del conflicto se determina, de algún modo, a partir de todas las decisiones realizadas."

También afirmó que:

"Siempre existe una forma racional de actuar en juegos de dos participantes, si los intereses que los gobiernan son completamente opuestos."

La demostración a esa afirmación se llama Teoría Minimax y surge en 1926.

Este teorema establece que en los juegos bipersonales de suma cero, donde cada jugador conoce de antemano la estrategia de su oponente y sus consecuencias, existe una estrategia que permite a ambos jugadores minimizar la pérdida máxima esperada. En particular, cuando se examina cada posible estrategia, un jugador debe considerar todas las respuestas posibles del jugador adversario y la pérdida máxima que puede acarrear. El jugador juega, entonces, con la estrategia que resulta en la minimización de su máxima pérdida. Tal estrategia es llamada óptima para ambos jugadores sólo en caso de que sus minimaxes sean iguales (en valor absoluto) y contrarios (en signo). Si el valor común es cero el juego se convierte en un sinsentido.

En los juegos de suma no nula, existe tanto la estrategia Minimax como la Maximin. La primera intenta minimizar la ganancia el rival, o sea busca que el rival tenga el peor resultado. La segunda intenta maximizar la ganancia propia, o sea busca que el jugador obtenga el mejor resultado.(2)

**Clausura**

En Informática, una clausura es una función que es evaluada en un entorno conteniendo una o más variables dependientes de otro entorno. Cuando es llamada, la función puede acceder a estas variables. El uso explícito de clausuras se asocia con programación funcional y con lenguajes como el ML y el Lisp. Construcciones como los objetos en otros lenguajes pueden ser también modelados con clausuras.

En algunos lenguajes, una clausura puede aparecer cuando una función está definida dentro de otra función, y la función más interna refiere a las variables locales de la función externa. En tiempo de ejecución, cuando la función externa se ejecuta, se forma una clausura, consistiendo en el código de la función interna y referencias a todas las variables de la función externa que son requeridas por la clausura.

Una clausura puede ser usada para asociar una función con un conjunto de variables "privadas", que persisten en las invocaciones de la función. El ámbito de la variable abarca sólo al de la función definida en la clausura, por lo que no puede ser accedida por otro código del programa. No obstante, la variable mantiene su valor de forma indefinida, por lo que un valor establecido en una invocación permanece disponible para la siguiente. Como consecuencia, las clausuras pueden ser usadas para esconder estados e implementar programación orientada a objetos.

El concepto de clausura fue desarrollado en los años 60 e implementado de forma completa, por primera vez, como una característica más del lenguaje de programación Scheme. Desde entonces, muchos otros lenguajes han sido diseñados para soportar clausuras.(3 )

**Las tablas de verdad.**

Las tablas de verdad nos ayudan a establecer el valor de verdad de diferentes razonamientos lógicos construidos a base de la combinación de dos o más enunciados.

Los enunciados se identifican con las letras del alfabeto, usualmente las de la segunda mitad del alfabeto:  p, q, r, s, t, etc.

Puede usarse cualquier símbolo para identificar a los enunciados.

La tabla de verdad más simple es la que corresponde a los valores de verdad de un solo enunciado nuclear.

|  |
| --- |
| **P** |
| **V** |
| **F** |

Cuando hay dos enunciados, p y q, las tablas de verdad para los cuatro conectivos básicos (conjunción, disyunción, implicación y doble implicación), tienen cuatro niveles (2 elevado al número de enunciados). Se pretende en la tabla que se puedan establecer todas las combinaciones de valores de verdad asumidos por los enunciados.

Los conectivos lógicos son Y (para la conjunción), O (para la disyunción inclusiva), SI… ENTONCES (para la implicación o condicional), SI Y SOLO SI (para la doble implicación o bicondicional).

Las tablas de verdad son las siguientes (4)

|  |  |  |  |
| --- | --- | --- | --- |
| **Conjunción** | **Disyunción** | **Implicación** | **Doble implicación** |
| |  |  |  | | --- | --- | --- | | **P** | **Q** | **P Y Q** | | V | V | V | | V | F | F | | F | V | F | | F | F | f | | |  |  |  | | --- | --- | --- | | **P** | **Q** | **P O Q** | | V | V | V | | V | F | V | | F | V | V | | F | F | F | | |  |  |  | | --- | --- | --- | | **P** | **Q** | **P ENTONCES Q** | | V | V | V | | V | F | F | | F | V | V | | F | F | V | | |  |  |  | | --- | --- | --- | | **P** | **Q** | **P SI Y SOLO SI Q** | | V | V | V | | V | F | F | | F | V | F | | F | F | V | |

1 <http://es.wikipedia.org/wiki/Algoritmo_de_ordenamiento> consultado 10/03/2014

2 <http://es.wikipedia.org/wiki/Minimax> consultado 10/03/2014

3 <http://es.wikipedia.org/wiki/Clausura_(inform%C3%A1tica)> consultado 10/03/2014

4 <http://es.wikipedia.org/wiki/Tabla_de_verdad> consultado 10/03/2014