Analisis De Algoritmos

Algoritmo de ordenamiento Radix-Sort

Universidad Tecnologica Metropolitana

Julio 2013

Introducción

- Para recuperar información de manera eficiente es deseable que aquella esté ordenada. Ordenar es simplemente colocar información de una manera especial basándonos en algún criterio a escoger.

Introducción

- Para recuperar información de manera eficiente es deseable que aquella esté ordenada. Ordenar es simplemente colocar información de una manera especial basándonos en algún criterio a escoger.
- Se han desarrollado muchas técnicas en este ámbito, cada una con características específicas, y con ventajas y desventajas sobre las demás. La principal característica de un método de ordenamiento debe ser la eficiencia.

Introducción

- Para recuperar información de manera eficiente es deseable que aquella esté ordenada. Ordenar es simplemente colocar información de una manera especial basándonos en algún criterio a escoger.
- Se han desarrollado muchas técnicas en este ámbito, cada una con características específicas, y con ventajas y desventajas sobre las demás. La principal característica de un método de ordenamiento debe ser la eficiencia.
- El método de ordenamiento en el que se basa esta presentación es RADIX SORT. La idea básica de este algoritmo es considerar que las claves están formadas por dígitos. Así, para ordenar las claves, el método las ordena por cada uno de sus dígitos, del más significativo al menos significativo, o viceversa. Se puede considerar como un algoritmo de ordenación estable y tiene una complejidad algorítmica lineal con el número de elementos a ordenar.

Historia

- En la década de los 80 en Estados Unidos no se puede terminar el censo (en concreto, no se llega a contar el número de habitantes solteros) Herman Hollerith (empleado de la oficina del censo, de 20 años de edad) inventa una máquina tabuladora eléctrica para resolver el problema; en esencia es una implementación física del Radix sort.

Funcionamiento Del Método

- El ordenamiento Radix es un algoritmo que ordena enteros procesando sus dígitos de forma individual. Como los enteros pueden representar cadenas de caracteres (por ejemplo, nombres o fechas).

Funcionamiento Del Método

- El ordenamiento Radix es un algoritmo que ordena enteros procesando sus dígitos de forma individual. Como los enteros pueden representar cadenas de caracteres (por ejemplo, nombres o fechas).
- Existen dos clasificaciones de Radix sort: el de dígito menos significativo (LSD) se usa típicamente para ordenar secuencias de enteros como "1, 2, 3, 4, 5, 6, 7, 8, 9, 10z el de dígito más significativo (MSD) que trabaja en sentido contrario, se usa frecuentemente para ordenar cadenas de caracteres, como las palabras o representaciones de enteros de longitud fija. Una secuencia como "b, c, d, e, f, g, h, i, j, ba"será ordenada léxicamente como "b, ba, c, d, e, f, g, h, i, j"

1. Se tienen los siguientes números en un vector que deseamos ordenar

Pos	0	1	2	3	4	5	6	7
valores	125	7	58	17	5	328	168	218

Teorema

Nota: La cantidad máxima de dígitos es 3, por lo tanto los números que contengan 1 o 2 dígitos se les agrega ceros a la izquierda, solo para efectos de explicacion, es decir:

valores	125	007	058	017	005	328	168	218

1. Se chequea el último dígito de cada número y se coloca en la columna correspondiente(cola) a su valor.

valores	125	007	058	017	005	328	168	218

Dígitos	d0	d1	d2	d3	d4	d5	d6	d7	d8	d9
						125		007	058	
						005		017	328	
									168	
									218	

Radix-Sort
Ejemplo

1. Se chequea el último dígito de cada número y se coloca en la columna correspondiente(cola) a su valor.

valores	125	007	058	017	005	328	168	218

Dígitos	d0	d1	d2	d3	d4	d5	d6	d7	d8	d9
						125		007	058	
						005		017	328	
									168	
									218	

2. Se sacan cada uno de los valores de las colas y se almacenan en el vector a partir de la cola cero.

Pos	0	1	2	3	4	5	6	7
valores	125	005	007	017	058	328	168	218

1. Luego se chequea el segundo dígito más a la izquierda de los números y se almacenan en su cola respectiva a su valor

Dígitos	d0	d1	d2	d3	d4	d5	d6	d7	d8	d9
	005	017	125			058	168			
	007	218	328							

Radix-Sort
Ejemplo

1. Luego se chequea el segundo dígito más a la izquierda de los números y se almacenan en su cola respectiva a su valor

Dígitos	d0	d1	d2	d3	d4	d5	d6	d7	d8	d9
	005	017	125			058	168			
	007	218	328							

2. Se sacan cada uno de los valores de las colas y se almacenan en el vector a partir de la cola cero.

Pos	0	1	2	3	4	5	6	7
valores	005	007	017	218	125	328	058	168

1. Y finalmente se chequea el último dígito más a la izquierda de los números y se almacenan en su cola respectiva a su valor

Dígitos	d0	d1	d2	d3	d4	d5	d6	d7	d8	d9
	005	125	218	328						
	007	168								
	017									
	058									

1. Y finalmente se chequea el último dígito más a la izquierda de los números y se almacenan en su cola respectiva a su valor

Dígitos	d0	d1	d2	d3	d4	d5	d6	d7	d8	d9
	005	125	218	328						
	007	168								
	017									
	058									

2. Se sacan cada uno de los valores de las colas y se almacenan en el vector a partir de la cola cero.

Pos	0	1	2	3	4	5	6	7
valores	005	007	017	058	125	168	218	328

1. Como ya no se tienen más dígitos a la izquierda se termina con el proceso y ya quedan ordenados.

Pos	0	1	2	3	4	5	6	7
valores	5	7	17	58	125	168	218	328

Complejidad

- Los requerimientos del tiempo para el método dependen del número de dígitos (d) y el número de elementos del archivo (n). Por lo tanto su complejidad es:

$$\Theta(n*d)$$

Ejemplos de utilización

1. Ordenar Fechas (Año/mes/día)

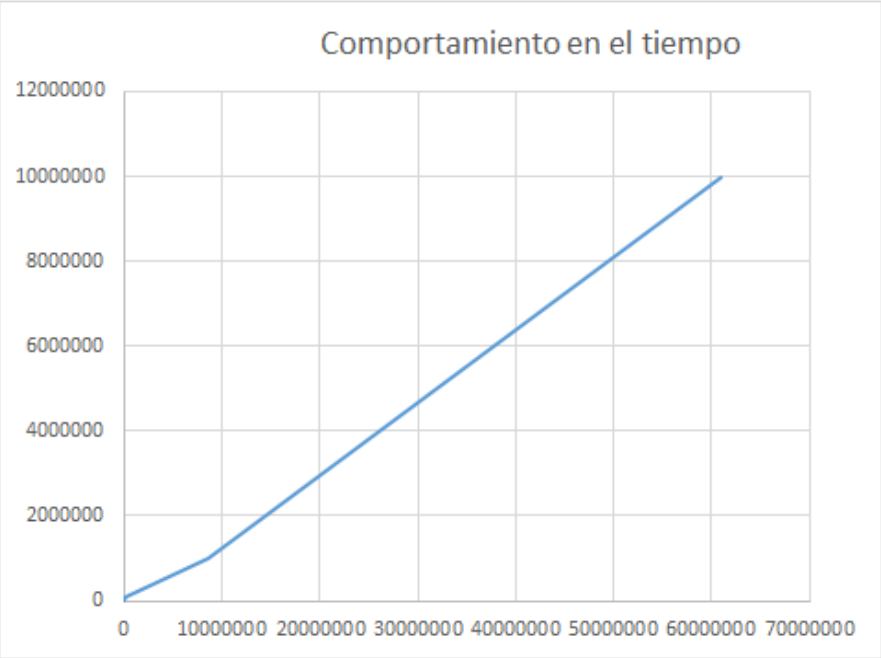
Ejemplos de utilización

1. Ordenar Fechas (Año/mes/día)
2. Ordenar mazo de cartas

Ejemplos de utilización

1. Ordenar Fechas (Año/mes/día)
2. Ordenar mazo de cartas
3. Ordenar alfabéticamente la lista de un curso

Experimento



Tiempo	Tamaño N
0	10
1	100
39	1000
458	10000
49148	100000
8611079	1000000
60960000	10000000

Ventajas

1. Tiene complejidad lineal y es de ordenación estable

Ventajas

1. Tiene complejidad lineal y es de ordenación estable
2. Es un método constante. Su complejidad siempre es O(m*n).

Ventajas

1. Tiene complejidad lineal y es de ordenación estable
2. Es un método constante. Su complejidad siempre es O(m*n).
3. Es razonablemente eficiente cuando la cantidad de datos no es demasiado grande.

Desventajas

1. Problemas potenciales de espacio de almacenamiento y llaves de tamaño variable y/o aleatorias.

Desventajas

1. Problemas potenciales de espacio de almacenamiento y llaves de tamaño variable y/o aleatorias.
2. El espacio extra

Desventajas

1. Problemas potenciales de espacio de almacenamiento y llaves de tamaño variable y/o aleatorias.
2. El espacio extra
3. Utilizando arreglos significa que ocuparemos 10*N espacio adicional para datos numéricos y 26*N para datos alfabéticos, y mucho más espacio para alfanuméricos.

Conclusiones

1. El algoritmo debe ser capaz de adaptarse.

Conclusiones

1. El algoritmo debe ser capaz de adaptarse.
2. No se basa en comparaciones y realiza una ordenación de datos estable.

Conclusiones

1. El algoritmo debe ser capaz de adaptarse.
2. No se basa en comparaciones y realiza una ordenación de datos estable.
3. Presenta un buen rendimiento si el conjunto de datos a ordenar cabe en algún nivel de memoria caché.

Conclusiones

1. El algoritmo debe ser capaz de adaptarse.
2. No se basa en comparaciones y realiza una ordenación de datos estable.
3. Presenta un buen rendimiento si el conjunto de datos a ordenar cabe en algún nivel de memoria caché.
4. El mejor rendimiento se obtiene con datos ni muy grandes, ni muy pequeños.