BucketSort

E.Aguila M. Marabolí P. Rain B. Vásquez

26 de Julio del 2013

Índice

1. Introducción

El estudio de algoritmos de ordenamiento tiene una gran importancia dentro de la Ciencia de la Computación, pues una buena cantidad de los procesos realizados por medios computacionales requieren que sus datos estén ordenados. Además, el hecho de almacenar los datos de manera ordenada permite implementar algoritmos de búsqueda muy rápidos (por ejemplo: búsqueda binaria). Esta y muchas otras razones de fin practico impulsaron el estudio y la búsqueda de algoritmos de ordenamiento recientes.

2. Descripcion

El ordenamiento por casilleros (bucket 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.

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

3. Caracteristicas

- Corre en tiempo lineal cuando la entrada se toma de una distribución uniforme.
- Rápido porque asume algo sobre la entrada (como counting sort, asume números en un rango pequeño).
    - Bucket Sort asume números generados aleatoriamente y distribuidos uniformemente en un rango de [0,1).
- Divide el intervalo de [0,1) en n sub-intervalos de igual tamaño.
    - Se ordenan los números en cada partición.
    - Se recorren las particiones en orden listando los elementos.

4. Complejidad

Para un arreglo de largo n y k Buckets.

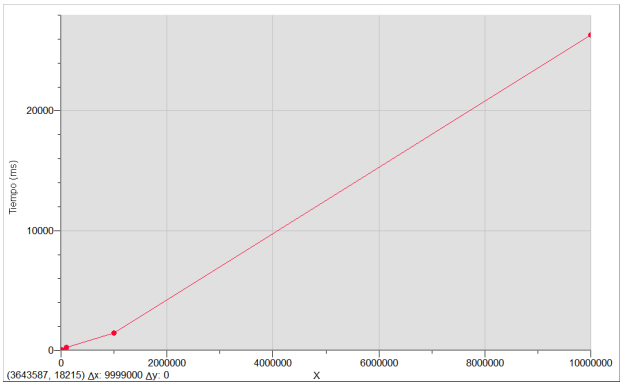
- Peor Caso: O(n^2)
- Caso Promedio: O(n + k)
- Complejidad Espacial: O(nk)

5. Problemas

- Cuanto más reducido sea el rango de valores que pueden tomar los casilleros y mayor sean los datos del conjunto, más desequilibrio existirá. Si hay desequilibrio, ordenara particiones de tamaños diferentes.
- El rendimiento depende del tamaño y sobre todo la distribución que posean los datos que se estén ordenando.
- Esto puede usarse a favor cuando se están ordenando elementos de estructuras lineales, como las listas.

6. Resultados

Nř de Elementos	Tiempo(ms)
1000	12
10000	50
100000	245
1000000	1440
10000000	26345

Figura 6.1: 

7. Conclusiones

- Una ventaja de Bucket sort con respecto a todos los demás es que, si se puede implementar, entonces es el método más rápido que existe para ordenar listas muy grandes.
- Si los datos están muy desequilibrados puede desencadenar en un mal rendimiento del algoritmo.

Referencias

[http://www.teachingtree.co/watch/bucket-sort]

[http://scanftree.com/Data_Structure/bucket-Sort]