# Método de Ordenamiento "Bucket Sort"

Diego C. Navia Fuentes

31 de Julio 2013

# Índice general

# Capítulo 1

# Introducción

Es un algoritmo de ordenación, que esta basado en la partición del vector o matriz de entrada en varias partes mas pequeñas llamados Bucket y el uso de algún otro algoritmo o recursivamente el mismo para ordenar los buckets creados.

# Capítulo 2

# Desarrollo

## 2.1. Historia

El creador de este algoritmo fue Herman Hollerith que nació el 29 febrero 1860 hasta (1929) . Era hijo de inmigrantes alemanes, nació en Buffalo, Nueva York y fue un Estadístico del Censo. Él desarrolló una perforadora de tarjetas Tabulating Machine. máquina de Hollerith incluyó tabulador y clasificador, y se utilizó para generar el censo de población oficial de 1890. El censo tomó seis meses, y en otros dos años, todos los datos del censo se completó y se define. Hollerith formó la empresa Tabulating Machine en 1896. La compañía se fusionó con International Time Recording Company y Computan Scale Company para formar equipo Tabulating Recording Company (CTR) en 1911. CTR fue el predecesor de IBM. CTR cambió su nombre a International Business Machines Corporation en 1924. Hollerith se desempeñó como ingeniero de consultoría con el CTR hasta su retiro en 1921. Hay referencias a Harold H. Seward, un científico de la computación, como el desarrollador de Radix sort en 1954 en el MIT. También desarrolló el tipo de recuentoy es el primero conocido por haber generado un algoritmo similar a la Base de ordenación

## 2.2. Descripción del algoritmo

El algoritmo divide la vector de entrada en Bucket. Cada Bucket contiene una cierta gama de elementos de entrada (los elementos deben ser distribuidos de acuerdo a un patrón o regla de ordenación).En la segunda fase Bucket tiene que ser ordenado con otro algoritmo de ordenación o el mismo recursivamente. Por último, el algoritmo combina todos los buckets ordenados. Porque cada buckets contiene diferentes gamas de valores de los elementos. Bucket short se limita a copiar los elementos de cada Bucket en el vector de salida (concatena los Bucket). La complejidad asintótica de Bucket short ,es de tipo O(n) en el mejor de los casos.

### 2.2.1. Implementación del Algoritmo

#### 2.2.1.1. Algoritmo en Java

```
public static int[] bucketSort(int[] arr,int mayordelArray) {
int i, j;
int[] count = new int[mayordelArray];
Arrays.fill(count, 0);
for(i = 0; i < arr.length; i++ ) {
count[arr[i]]++;
}
for(i=0,j=0; i <mayordelArray; i++) {
for(; count[i]>0; (count[i])–) {
arr[j++] = i; }
}
return arr;
}
```

#### 2.2.1.2. Algoritmo en C/C++

```
void bucketSort(int A[],int largo,int mayordelArray){
int cuenta[mayordelArray];
llenarConCero(cuenta,mayordelArray);
for (int i=0; i<largo; i++) {
cuenta[A[i]]++;
}
for (int i=0,j=0; i<mayordelArray; i++) {
for (; cuenta[i]>0; (cuenta[i])–) {
A[j++]=i;
}
}
cout<<endl;
}
```

#### 2.2.1.3. Explicación y uso

El algoritmo consta de que tiene un arreglo de entrada con N casillas,donde se va contando en la referencia(cuantas veces se repite el numero ) y se guarda en otro vector o arreglo en la casilla que es igual al numero del casillero de entrada.


## 2.3. El mejor de los casos

Para este algoritmo de ordenación, el mejor de los casos es que los datos de entrada sea uniforme,es decir, que no sean repetidos la complejidad seria sercana a O(n+m) (n largo del arreglo de entrada,m mayor del arreglo o vector).

## 2.4. El peor de los casos

El peor de los casos de este algoritmo,es que los datos del vector de entrada esten muy repetidos.

## 2.5. Experimento practico

El experimento constara de hacer corres el algoritmo Bucket short en el lenguaje de programación Java, con cantidades de datos .Para probar su comportamiento en el tiempo de ejecución y su tendencia a grandes valores

Características de la maquina :

-Core I5 intel

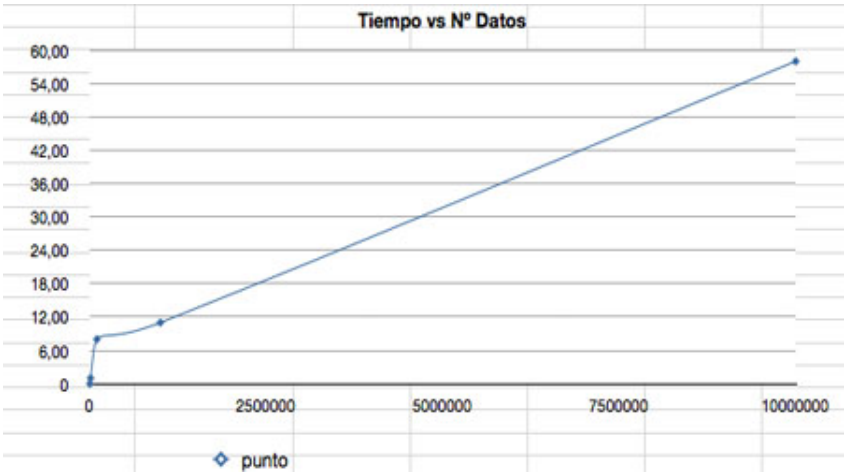
-4GB en RAM

-Sistema Operativo OSX

### 2.5.1. Tabla de valores del experimento

| Nº Datos | Tiempo(milisegundos) |
|---|---|
| 10 | 0 |
| 100 | 0 |
| 1000 | 0 |
| 10000 | 1 |
| 100000 | 8 |
| 1000000 | 11 |
| 10000000 | 56 |

### 2.5.2. Gráfico




Como se ve en el gráfico tiempo a un comportamiento lineal ,en algunos de los casos que mencionamos anteriormente

## 2.6. Comparación con otros Algoritmos

En comparación con otros algoritmos conocidos, como el método de la Burbuja o el Inserción .Este método alcanza muchos mejores tiempos.El único detalle desfavorable es que este algoritmo tiene que contar y pasar por todos los casilleros,donde se pierde la optimización del tiempo.

| Algoritmo | Complejidad |
|---|---|
| Bucket Sort | O(n+m) |
| Burbuja | $O(n^2)$ |
| Inserción | $O(n^2)$ |
| Arbol Binario | O(nlogn) |

# Capítulo 3

# Conclusión

En el informe se mostró la características del algoritmo como la estrategia que utilizaba de crear Bucket o cubos mas pequeños ordenamos por alguna regla ,para solucionar el problema de ordenar muchos datos alcanzando mejores tiempos.

Se mostró que existes casos ideales donde el algoritmo de comporta de diferentes formas que en otras,como el mejor de los casos donde los datos entrantes tenían que ser uniformes(que no se repitan a lo largo del vector),donde tenia una complejidad mucho menor ,que cuando estaba en el caso de que los datos fueran muy repetidos .

También se implemento el algoritmo en algoritmo en forma practica ,para dar una base mas tangible al método de ordenamiento.

# Capítulo 4

# Bibliografía


1. http://www.algorridin.com/bucket-Sort/algoritmin
2. http://www.exforsys.com/tutorials/c-algorithms/bucket-sort.html
3. http://estructuras-de-datos.wikispaces.com/BUCKET+SORT
4. http://www.personal.kent.edu/~rmuhamma/Algorithms/MyAlgorithms/Sorting/bucketSort.htm