Resumen estudio UT6 (1).

http://www.cs.oswego.edu/~mohammad/classes/csc241/samples/sort/Sort2-E.html

Selection Sort - O(n2)

Selection sort is one of the **O(n2)** sorting algorithms, which makes it quite inefficient for sorting large data volumes. Selection sort is notable for its programming simplicity and it can over perform other sorts in certain situations (see complexity analysis for more details).

Heapsort - O (n\log n)

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

Worst-case performance O(n\log n)

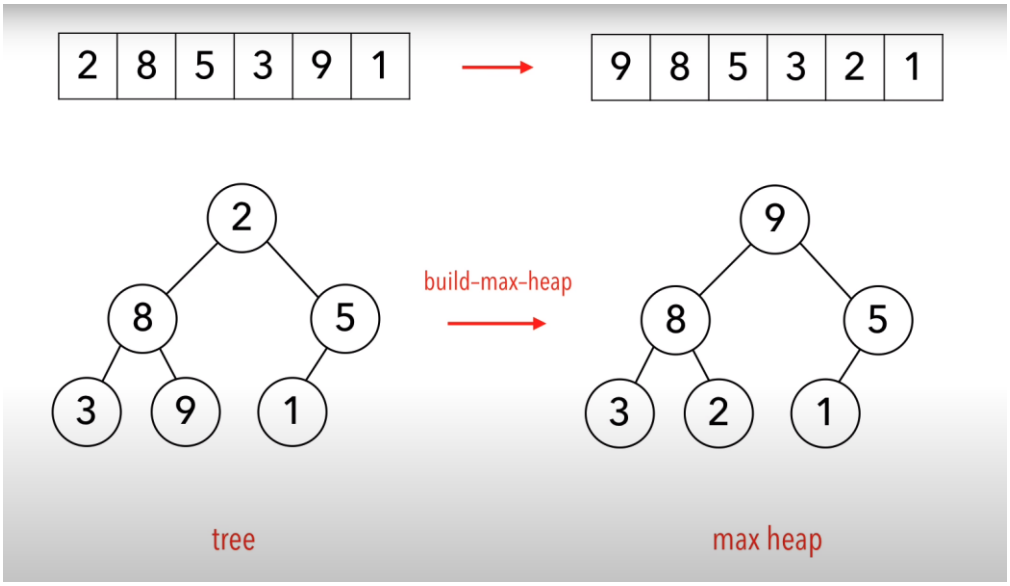
Best-case performance O(n\log n) or O(n)

Average performance O(n\log n)

Worst-case space complexity O(n)

Heap: Ordered binary tree

Max heap: parent > child



- Se forma un arbol con el arreglo

- Se invierte la raiz con el mayor

- Se invierte el mayor (raiz) con el ultimo

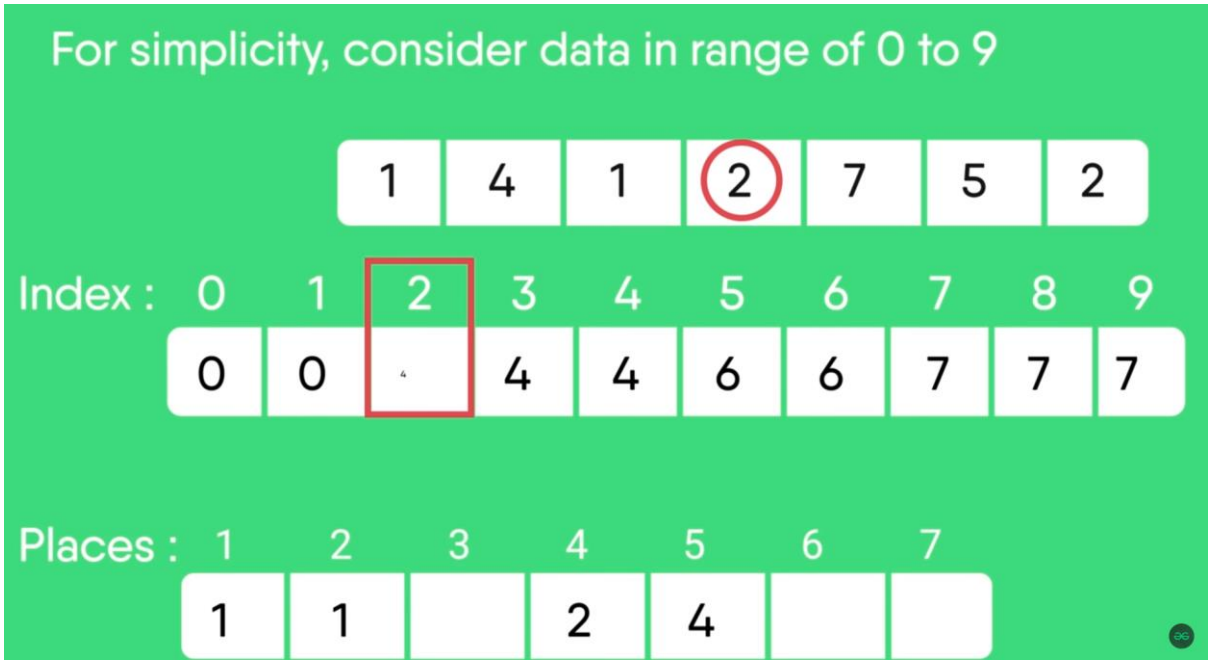
- Finalmente se elimina el último (mayor) y se considera ordenado

- Volver a repetir

Counting Sort - O (n)

Se cuenta el número de elementos de cada clase para luego ordenarlos. Solo puede ser utilizado por tanto para ordenar elementos que sean contables (como los números enteros en un determinado intervalo, pero no los números reales, por ejemplo).

El primer paso consiste en averiguar cuál es el intervalo dentro del que están los datos a ordenar (valores mínimo y máximo). Después se crea un vector de números enteros con tantos elementos como valores haya en el intervalo [mínimo, máximo], y a cada elemento se le da el valor 0 (0 apariciones). Tras esto se recorren todos los elementos a ordenar y se cuenta el número de apariciones de cada elemento (usando el vector que hemos creado). Por último, basta con recorrer este vector para tener todos los elementos ordenados.

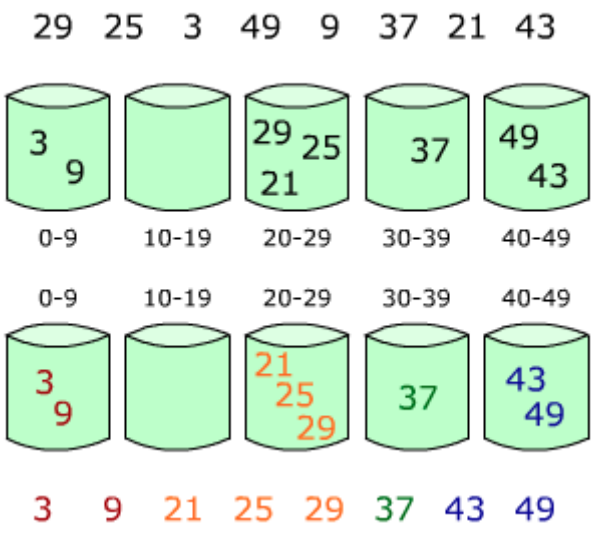


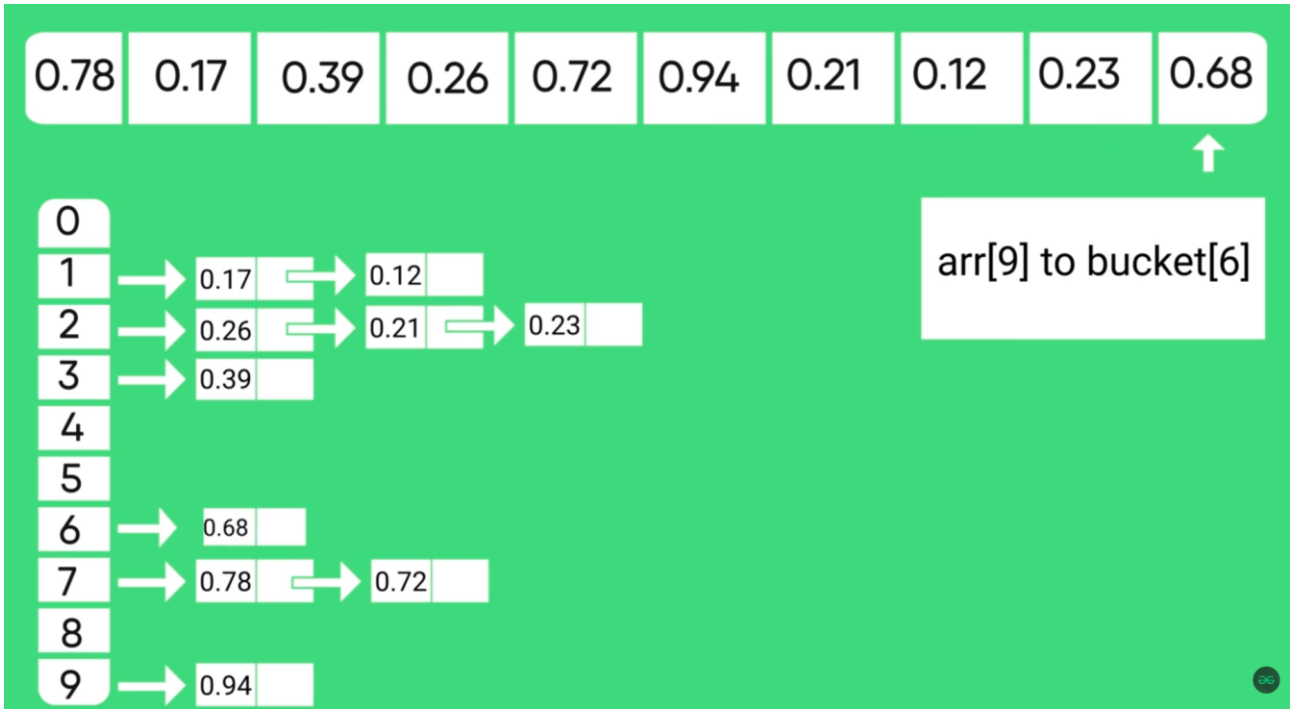
Bucket Sort – O(n2)

O(n2) – Caso todo al mismo bucket

Distribuye todos los elementos a ordenar entre un número finito de casilleros. Cada casillero solo 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.

Cuando los elementos a ordenar están uniformemente distribuidos la complejidad computacional de este algoritmo es de O(n).





- Se subdivide en buckets por valor y se mete todo lo que “coincida” ahí

- Se ordena internamente cada bucket con insertar ordenado

- Se recorre cada bucket insertando al final de un nuevo arreglo (ordenado)

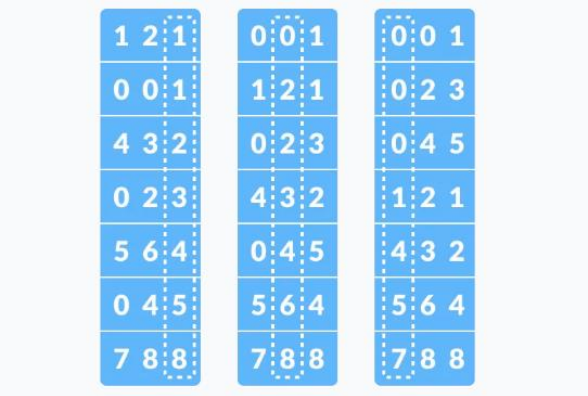
Radix Sort

Is a non-comparative sorting algorithm. It avoids comparison by creating and distributing elements into buckets according to their radix. For elements with more than one significant digit, this bucketing process is

repeated for each digit, while preserving the ordering of the prior step, until all digits have been considered. For this reason, radix sort has also been called bucket sort and digital sort.

Radix sort can be applied to data that can be sorted lexicographically, be they integers, words, punch cards, playing cards, or the mail.

Ordena desde el carácter menos significante al mas significante (?



sorting the integers according to units, tens and hundreds place digits