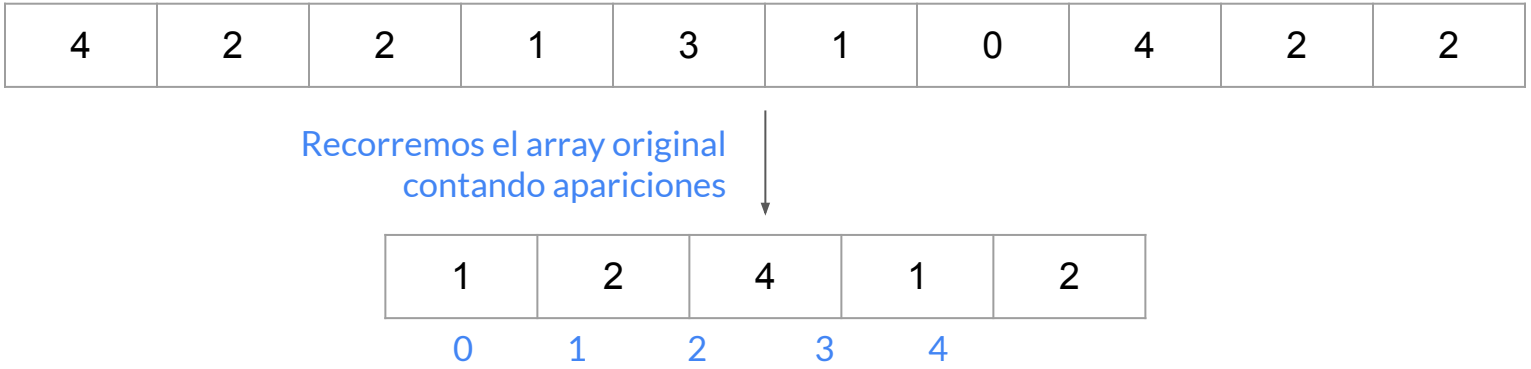
Otros algoritmos de ordenamiento

- Hasta ahora vimos algoritmos que no tienen requiere
- Sin requieres, vimos que lo mejor que podemos llegar es O(n logn)
- ¿Y si podemos alguna restricción sobre el array de ingreso?

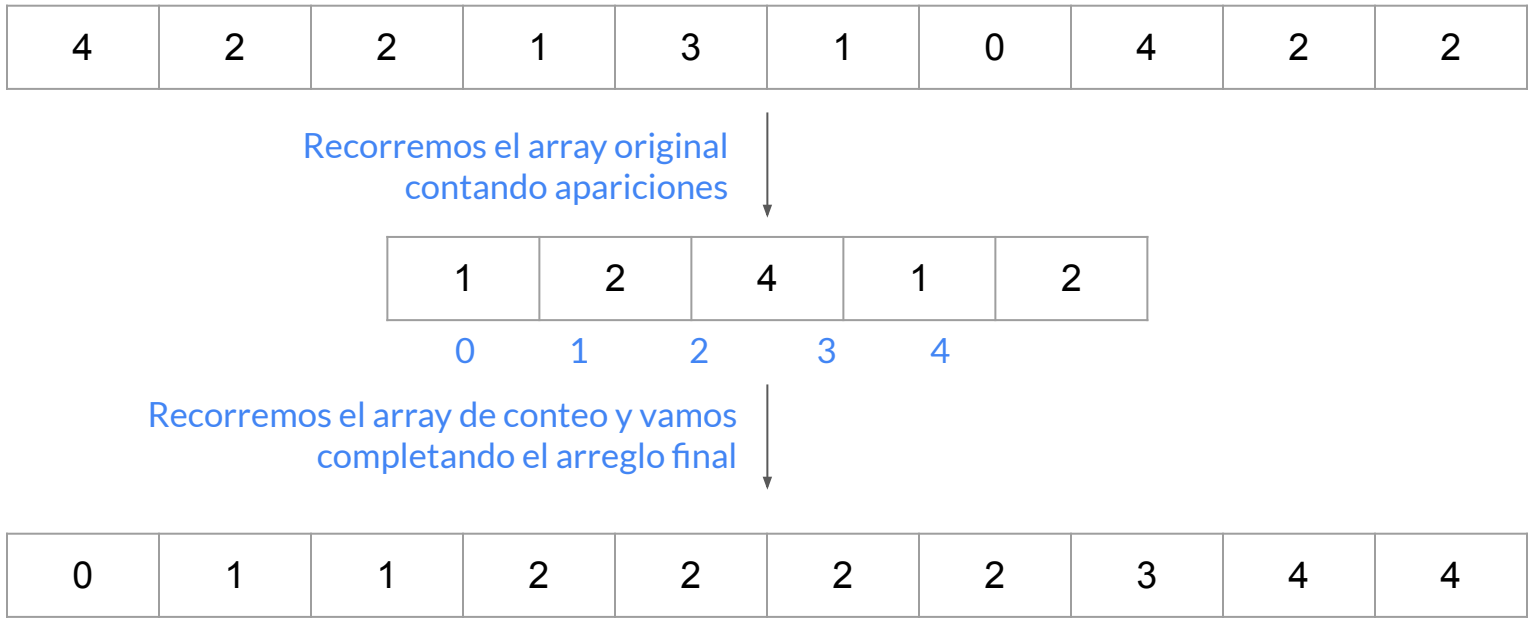
¿Como podríamos ordenar este arreglo sabiendo que solo va a tener números del 0 al 4?

4	2	2	1	3	1	0	4	2	2

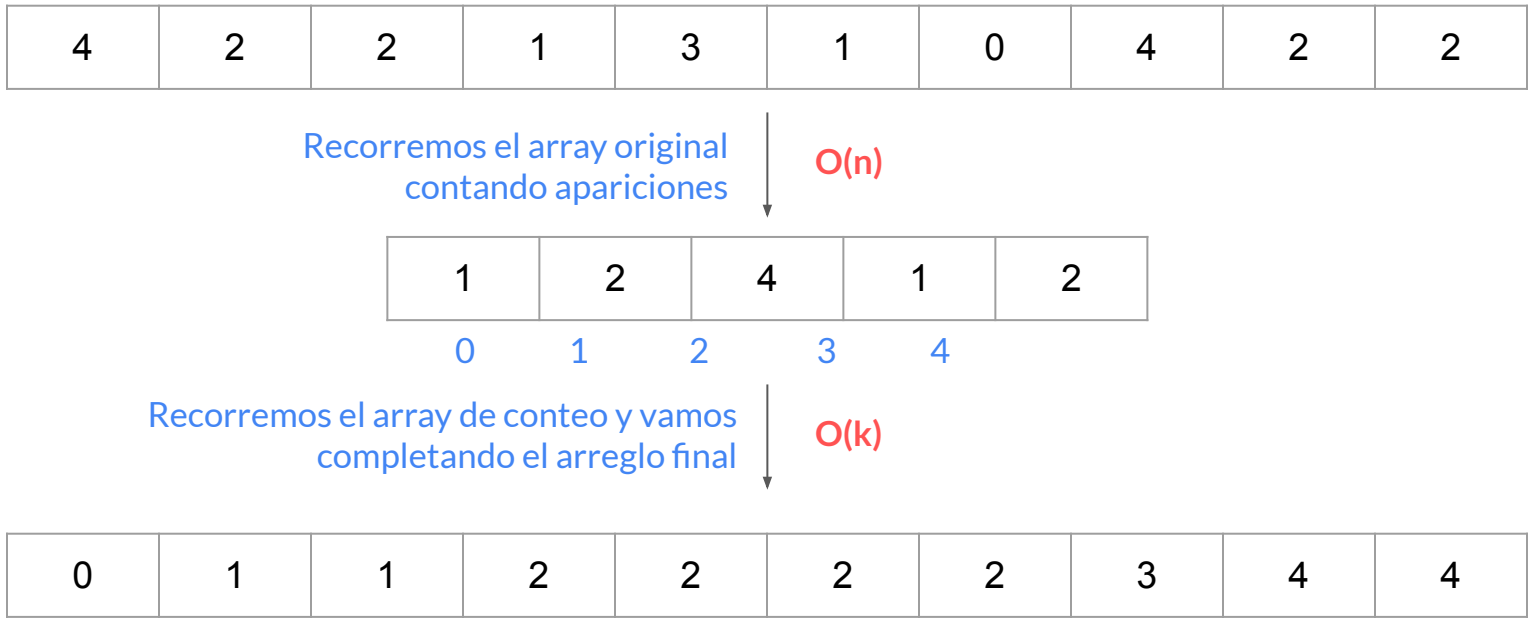
Counting sort



Counting sort



Counting sort



Counting sort

- Si el rango de valores va de 0 a k, el Array de conteo es obvio
  - cada posición representa al elemento correspondiente a su índice
- Si el rango de valores está entre otros dos enteros, se adapta muy sencillamente
  - Almacenamos en dos variables extras lo lḿites del rango y convertimos sin perder en complejidad
- Si $(k_2 - k_1) \sim n$, la complejidad es **O(n)!!**
- Con el algoritmo que vimos no es estable
  - Pero se puede hacer
  - Haciendo que el arreglo de conteo tenga la suma acumulada
- Así como lo vimos solo sirve para ordenar enteros

Radix sort

¿Cómo hacemos para ordenar enteros con muchos dígitos?

329	457	657	839	436	720	355

Radix sort

¿Cómo hacemos para ordenar enteros con muchos dígitos?

329	457	657	839	436	720	355

Ordenamos por la cifra menos significativa

720	355	436	457	657	329	839

Radix sort

¿Cómo hacemos para ordenar enteros con muchos dígitos?

329	457	657	839	436	720	355

Ordenamos por la cifra menos significativa

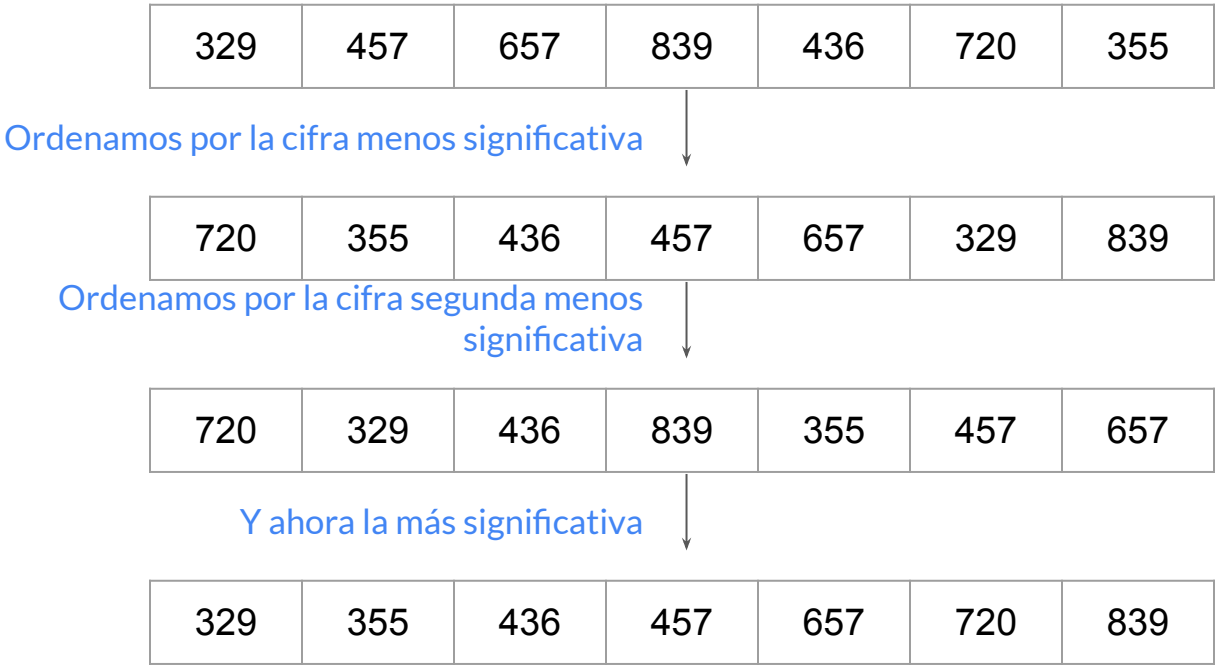
720	355	436	457	657	329	839

Ordenamos por la cifra segunda menos significativa

720	329	436	839	355	457	657

Radix sort

¿Cómo hacemos para ordenar enteros con muchos dígitos?



Radix sort

¿Cómo hacemos para ordenar enteros con muchos dígitos?

329	457	657	839	436	720	355

Ordenamos por la cifra menos significativa ↓ O(n)

720	355	436	457	657	329	839

Ordenamos por la cifra segunda menos significativa ↓ O(n)

720	329	436	839	355	457	657

Y ahora la más significativa ↓ O(n)

329	355	436	457	657	720	839

USAR UN ALGORITMO ESTABLE

Radix sort

- Es importante en cada paso ordenar con un algoritmo estable
- Si podemos usar counting sort, es lo mejor
- La complejidad es O(d(n+k))
  - d: cantidad de dígitos (longitud de cada elemento)
  - k: posible valores de cada dígito
  - n: cantidad de elementos a oredenar
- Si d y k son fijos y chicos, la complejidad es O(n)
- Se puede generalizar a elementos de otro tipo, como tuplas

Bucket sort

0.78	0.17	0.39	0.26	0.72	0.94	0.21	0.12	0.23	0.68

Bucket sort

0.78	0.17	0.39	0.26	0.72	0.94	0.21	0.12	0.23	0.68

Agrupamos en Listas Enlazadas
por primer decimal

-	0.17 0.12	0.26 0.21 0.23	0.39	-	-	0.68	0.78 0.72	-	0.94

Bucket sort

0.78	0.17	0.39	0.26	0.72	0.94	0.21	0.12	0.23	0.68

Agrupamos en Listas Enlazadas
por primer decimal

-	0.17 0.12	0.26 0.21 0.23	0.39	-	-	0.68	0.78 0.72	-	0.94

Ordenamos con insertion sort
cada lista enlazada

-	0.12 0.17	0.21 0.23 0.26	0.39	-	-	0.68	0.72 0.78	-	0.94

Bucket sort

0.78	0.17	0.39	0.26	0.72	0.94	0.21	0.12	0.23	0.68

Agrupamos en Listas Enlazadas por primer decimal ↓ O(n)

-	0.17 0.12	0.26 0.21 0.23	0.39	-	-	0.68	0.78 0.72	-	0.94

Ordenamos con insertion sort cada lista enlazada ↓

-	0.12 0.17	0.21 0.23 0.26	0.39	-	-	0.68	0.72 0.78	-	0.94

Bucket sort

- Armar el arreglo de listas enlazadas es siempre O(n)
- Ordenar las listas enlazadas es O(n) si elegimos bien los buckets
  - Lo óptimo es que en promedio cada bucket tenga la misma cantidad de elementos
- La complejidad entonces es O(n)