Sorting Data

Ananda Gunawardena

The problem of sorting

$Set = \{ A_i \mid i = 0, 1, \ldots, n-1 \}$

Find a permutation $(0, 1, \ldots n-1)$ such that

$A[i] \leq A[i+1]$ for all $i = 0, \ldots, n-2$

?

CompareTo

Complexity – To determine a set is sorted — $O(n)$

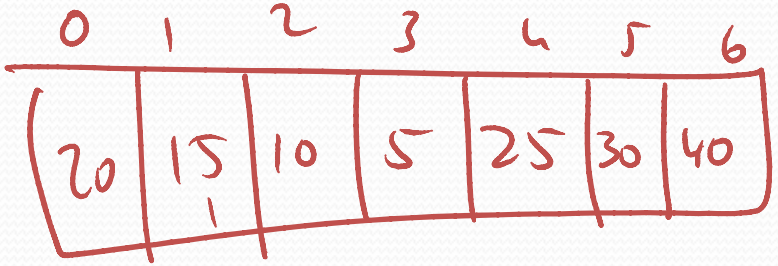
To sort a set $\longrightarrow O(n^2)$, $O(n \log n)$

Flips = 0 ⟺ A set is sorted

Flips and inversions

Flip is $A[i] > A[i+1]$

Inversion is $A[i] > A[j]$ $\quad i < j$



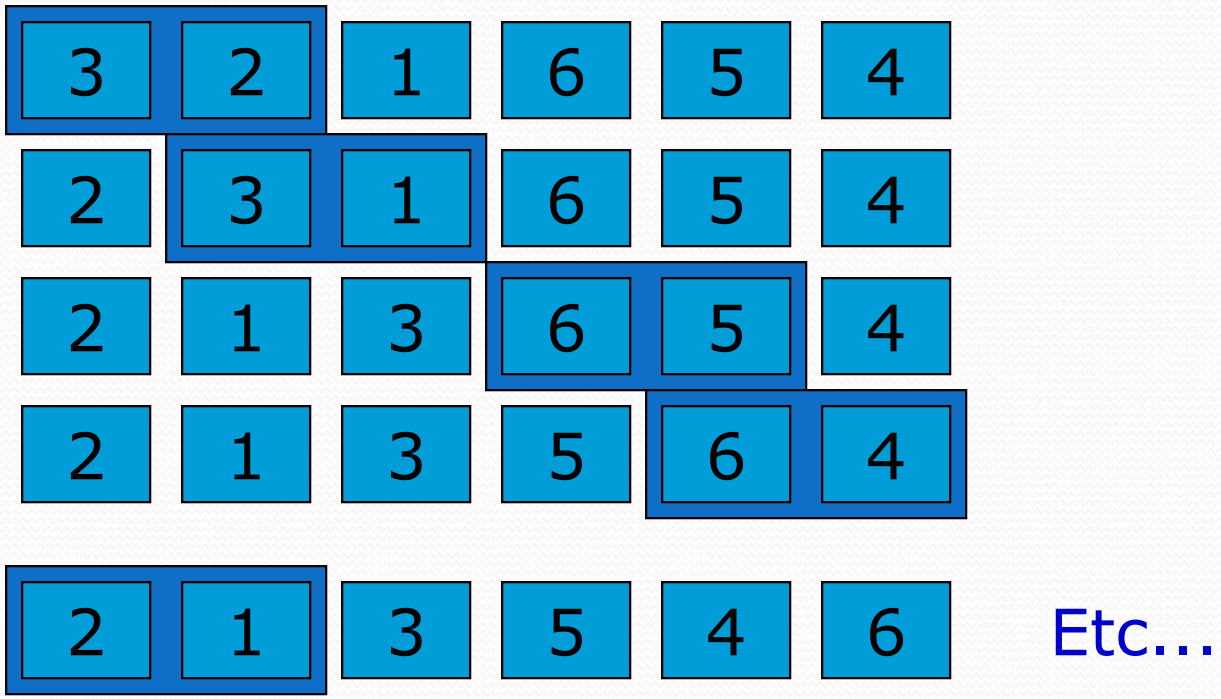
Flips = ? = 3

Inversions = 3 + 2 + 1 = 6

A Set is Sorted when inversions = 0

Naïve sorting algorithms

- Bubble sort: *scan for flips, until all are fixed*



Naïve Sorting

```
for i=1 to n-1
  { for j=0 to n-i-1
      if (A[j].compareTo(A[j+1])>0)
          swap(A[j], A[j+1]);
    if (no swaps) break;
  }
```

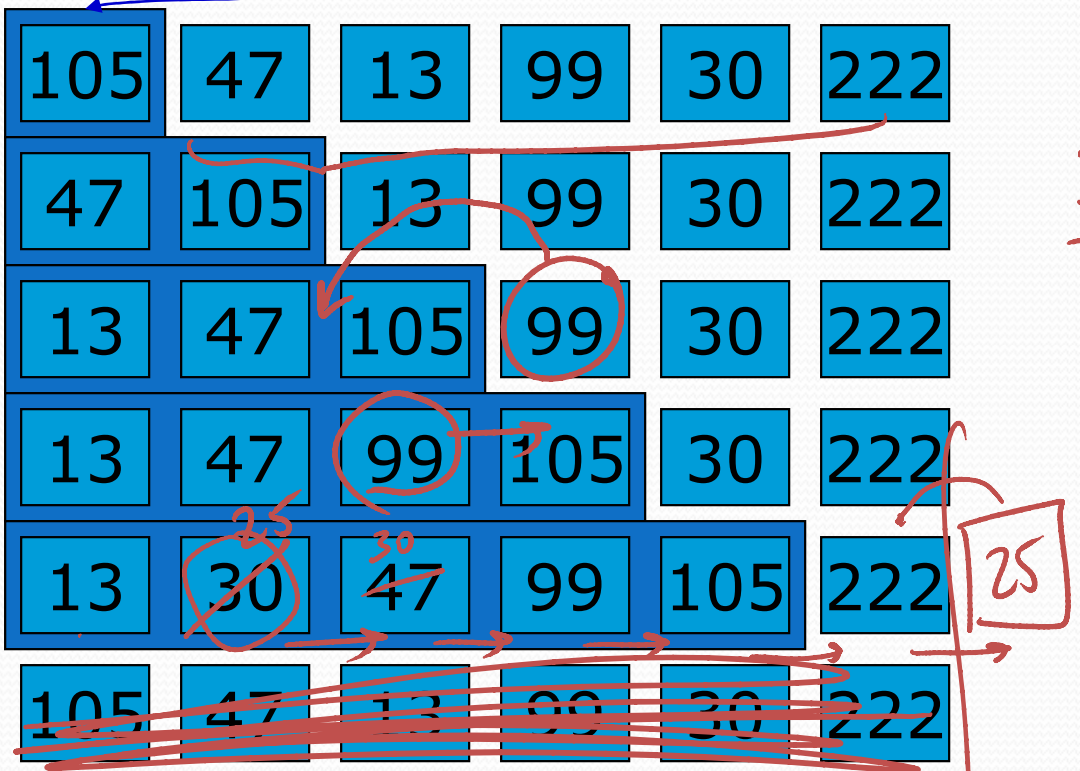
n-1
+ n-2
n-3
⋮
$O(n^2)$

- What happens if
  - All keys are equal? $O(n)$
  - Keys are sorted in reverse order? $O(n^2)$
  - Keys are sorted? $O(n)$
  - keys are randomly distributed? $O(n^2)$

- Exercise: Count the number of operations in bubble sort and find a Big O analysis for bubble sort

Insertion sort

Sorted subarray



bubble

Swaps are expensive

Insertion

2 3 4 5 6 7 8

Insertion sort

- **Algorithm**

```
for   i = 1   to   n-1   do
      insert a[i] in the proper place
           in   a[0:i-1]
```

n-1
+ n-2
+ ...
O(n²)

- **Correctness**
  - Note: after i steps, the sub-array A[0:i] is sorted

How fast is insertion sort?

To insert a[i] into a[0:i-1], **slide** all elements larger than a[i] to the right.

```
tmp = a[i];
for (j = i; j>0 && a[j-1]>tmp; j--)
      a[j] = a[j-1];
a[j] = tmp;
```

# of slides = O(#inversions)

very fast if array is nearly sorted to begin with

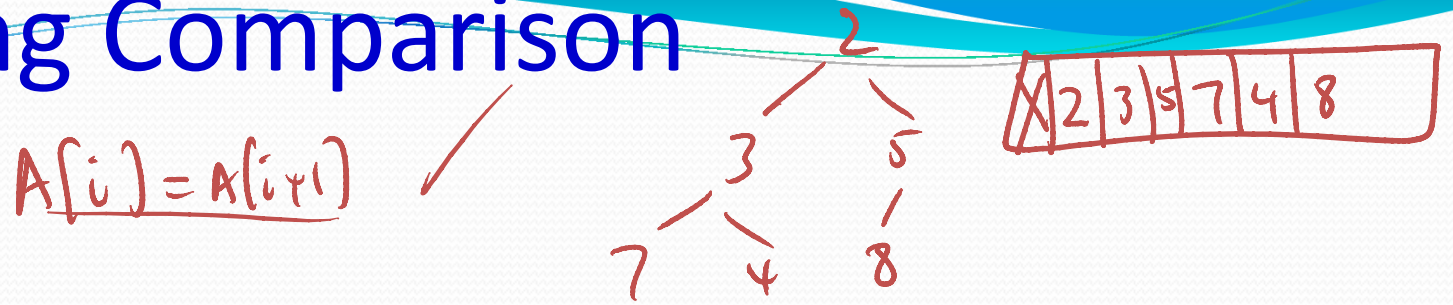
Selection sort

- **Algorithm**

```
for  i = n-1  to  1  do
     Find the largest entry in the
          in  the subarray A[0:i]
     Swap with A[i]
```

What is the runtime complexity of selection sort?

Sorting Comparison



- Discuss the pros and cons of each of the naïve sorting algorithms

	Comparisons	Swaps	Shift
bubble →	$O(n^2)$	$O(n^2)$	—
✓ insertion →	$O(n^2)$	—	$O(n^2)$
Selection →	$O(n^2)$	$O(n)$	—

Usually faster than Swaps

$O(n)$

mem copy

Sorting, in a different way

Bucket Sort

Bucket sort

- In addition to comparing pairs of elements, we require these additional restrictions:
  - all elements are non-negative integers
  - all elements are less than a predetermined maximum value
- Elements are usually keys paired with other data

Bucket sort

1 3 3 1 2

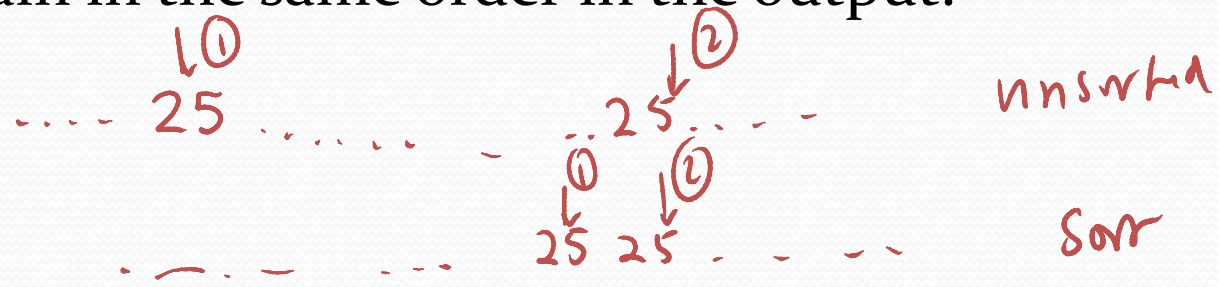
1

2

3

Bucket sort characteristics

- Runs in $O(N)$ time.
- Easy to implement each bucket as a linked list.
- Is *stable*:
  - If two elements (A,B) are equal with respect to sorting, and they appear in the input in order (A,B), then they remain in the same order in the output.



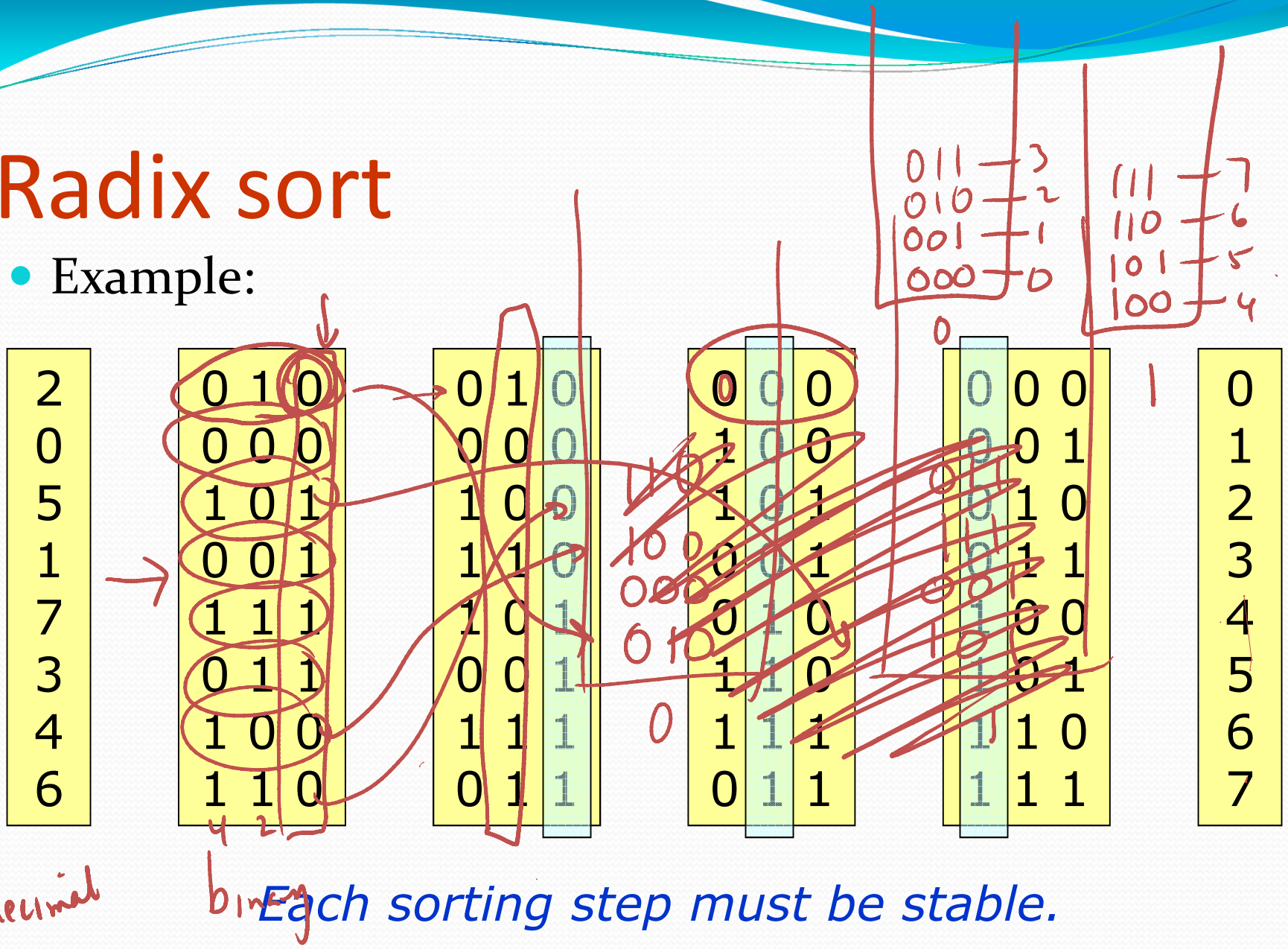
Radix Sort

Radix sort

- If your integers are in a larger range then do bucket sort on each digit
- Start by sorting with the low-order digit using a STABLE bucket sort.
- Then, do the next-lowest,and so on

Radix sort

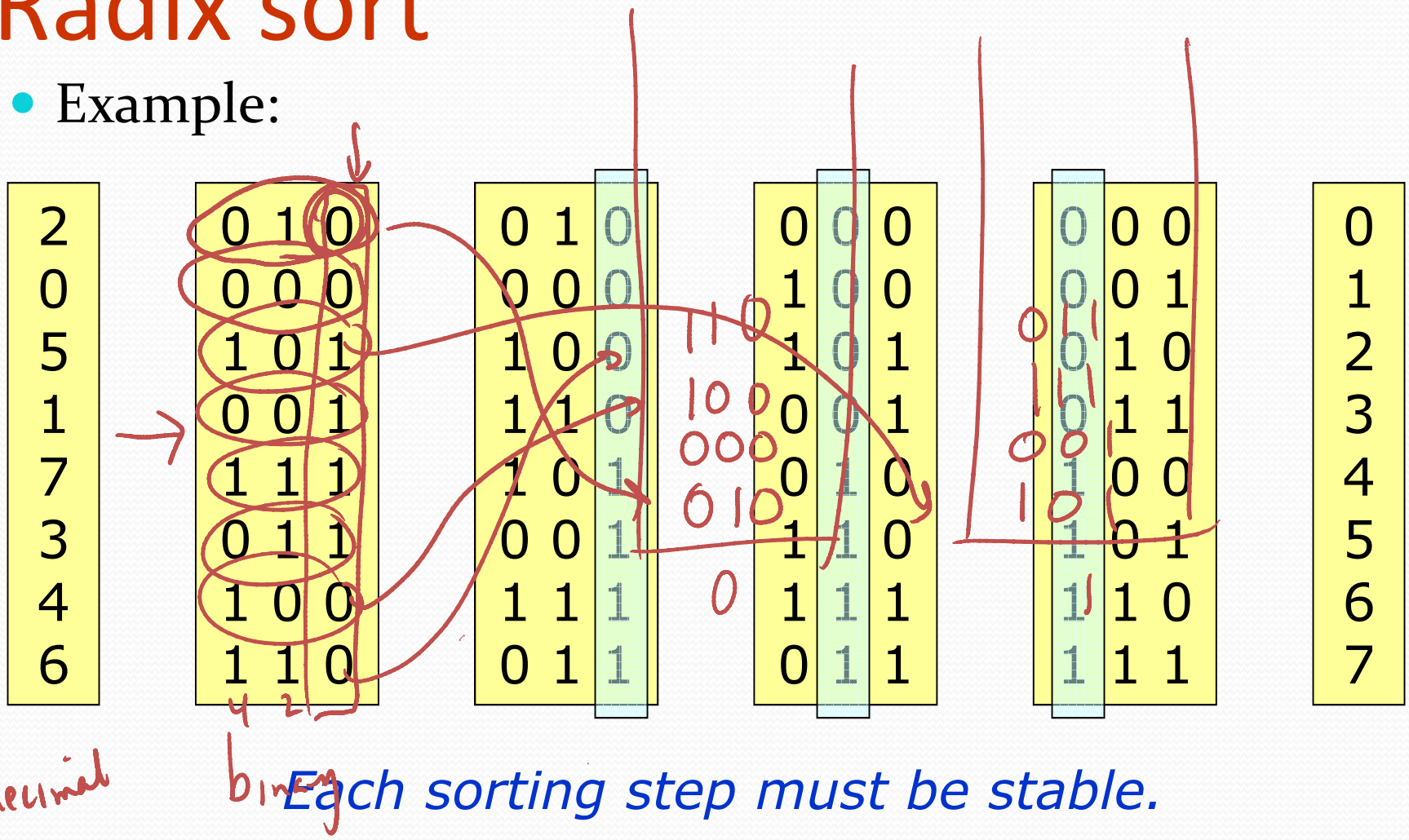
- Example:



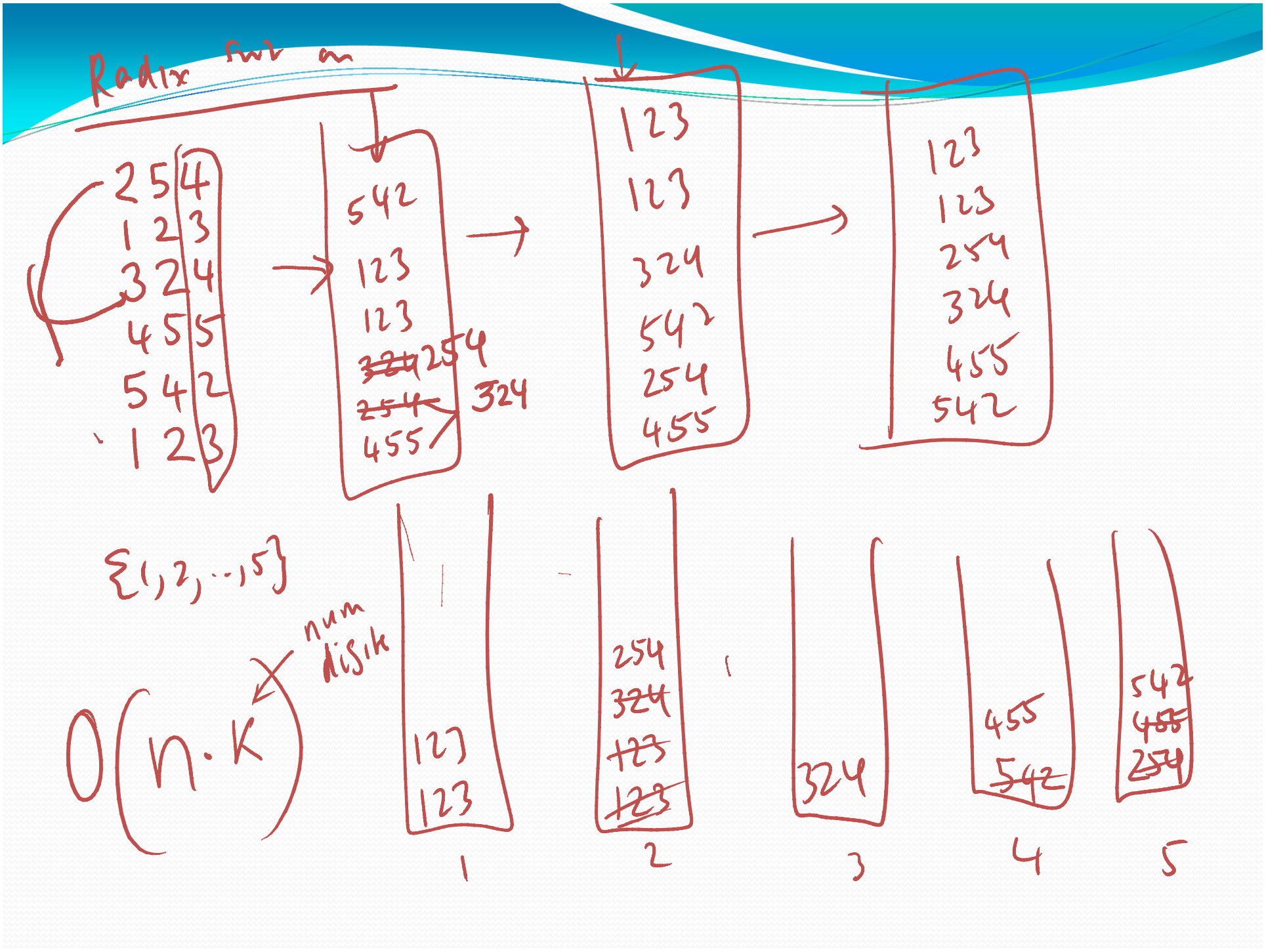
Each sorting step must be stable.

Radix sort

- Example:



Each sorting step must be stable.


Radix sort on
254
123
324
455
542
123
542
123
123
324
254
254
455
123
123
324
542
254
455
123
123
254
324
455
542
{1, 2, .., 5}
num digit
O(n · K)
123
123
254
324
324
455
542
1
2
3
4
5

Radix sort characteristics

- Each sorting step can be performed via bucket sort, and is thus $O(N)$.
- If the numbers are all b bits long, then there are b sorting steps.
- Hence, radix sort is $O(bN)$.

What about non-binary?

- Radix sort can be used for decimal numbers and alphanumeric strings.