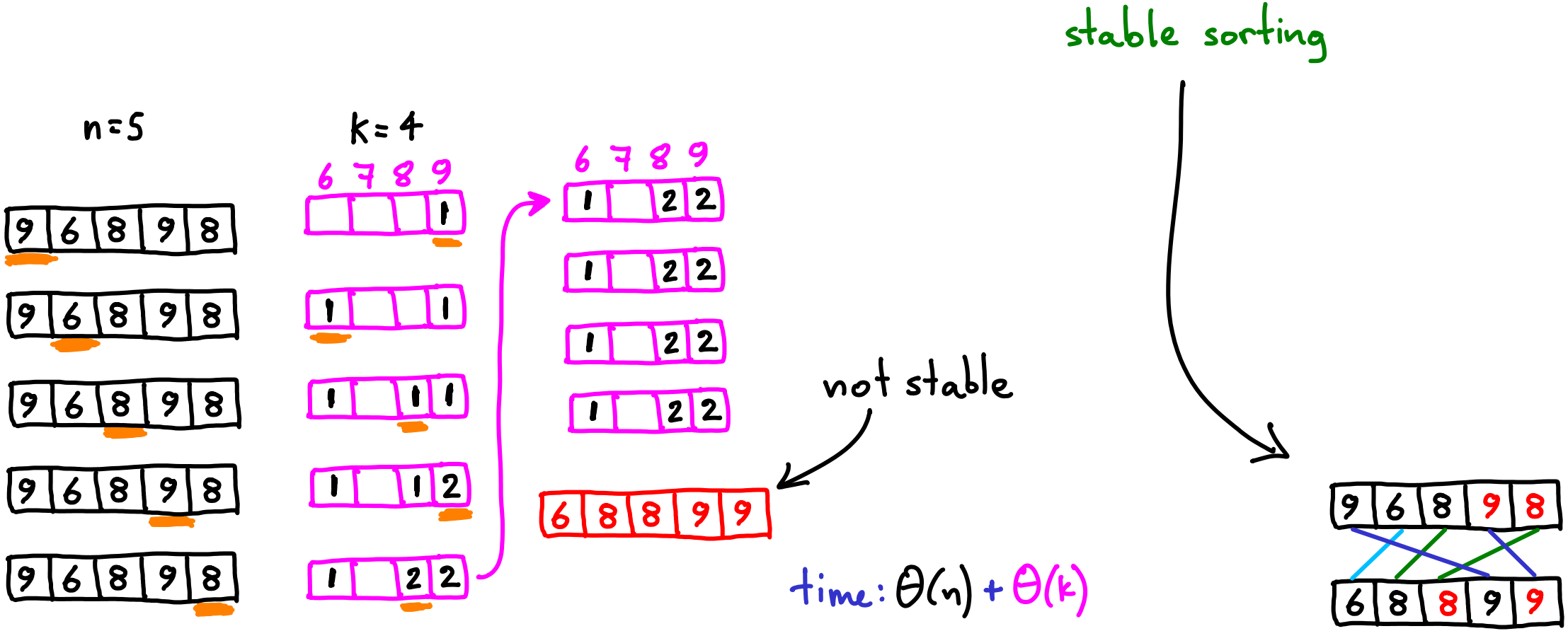
COUNTING SORT: assume input: n integers within range of size k

Use an array with k indices matching range

↳ keep track of how many integers have each value



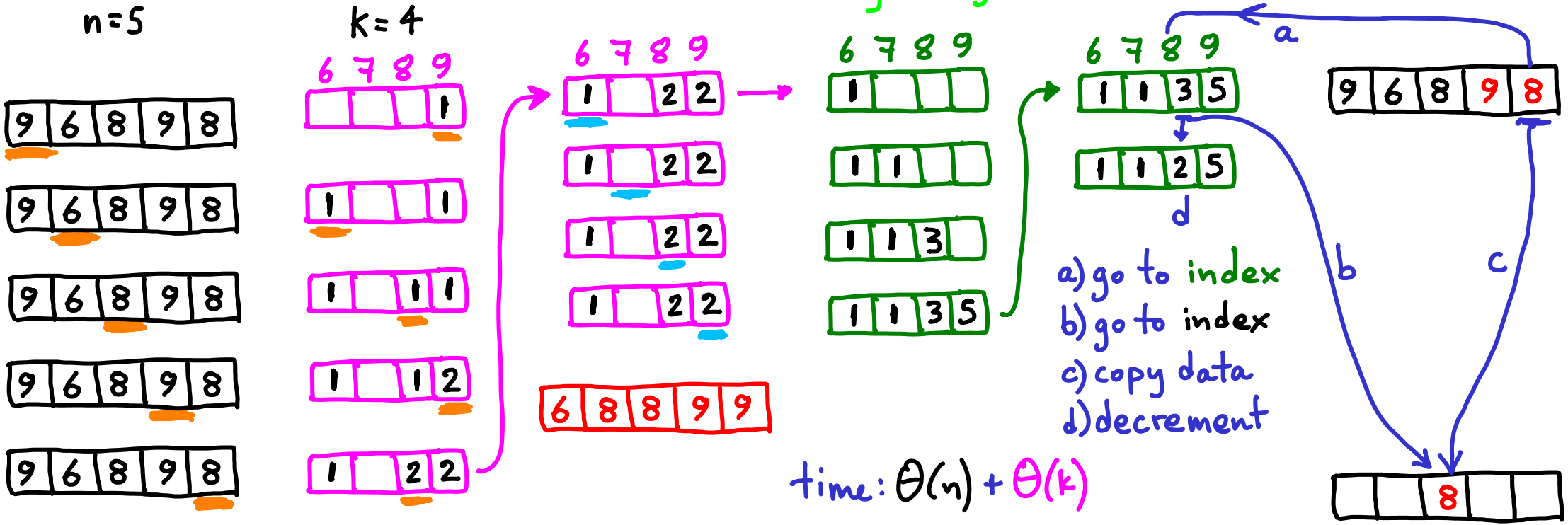
COUNTING SORT: assume input: n integers within range of size k

Use an array with k indices matching range

↳ keep track of how many integers have each value

Use a second array of size k to get stable sorting

↳ how many integers are ≤ each value



time: $\Theta(n) + \Theta(k)$

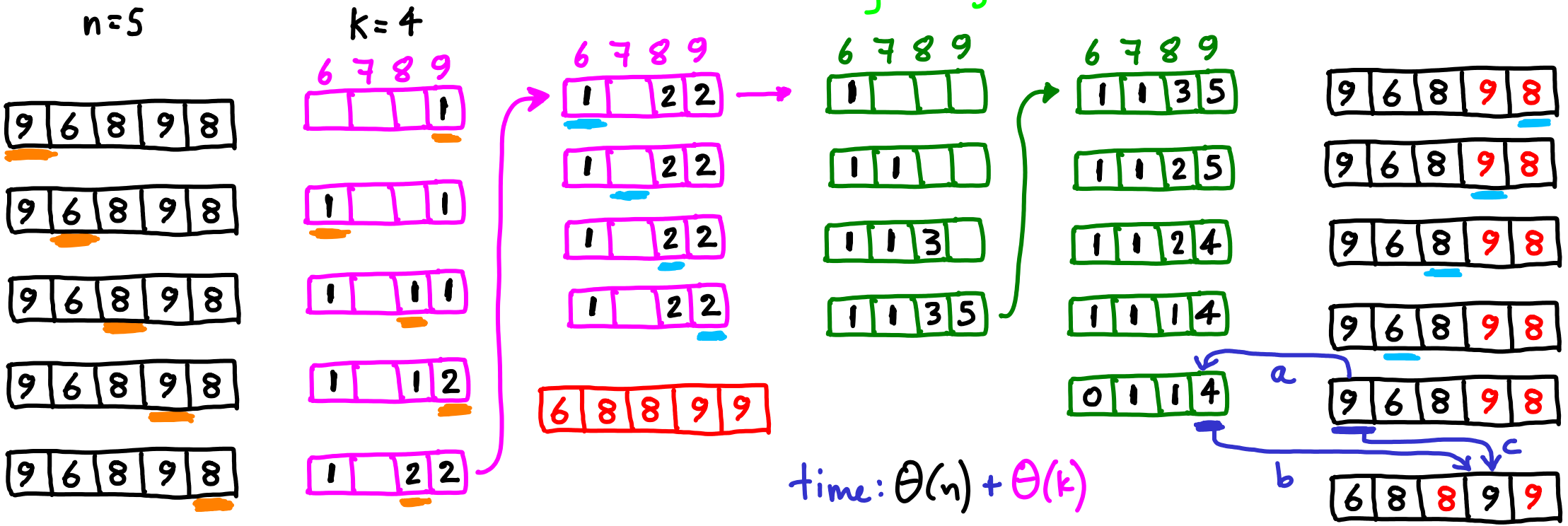
COUNTING SORT: assume input: n integers within range of size k

Use an array with k indices matching range

↳ keep track of how many integers have each value

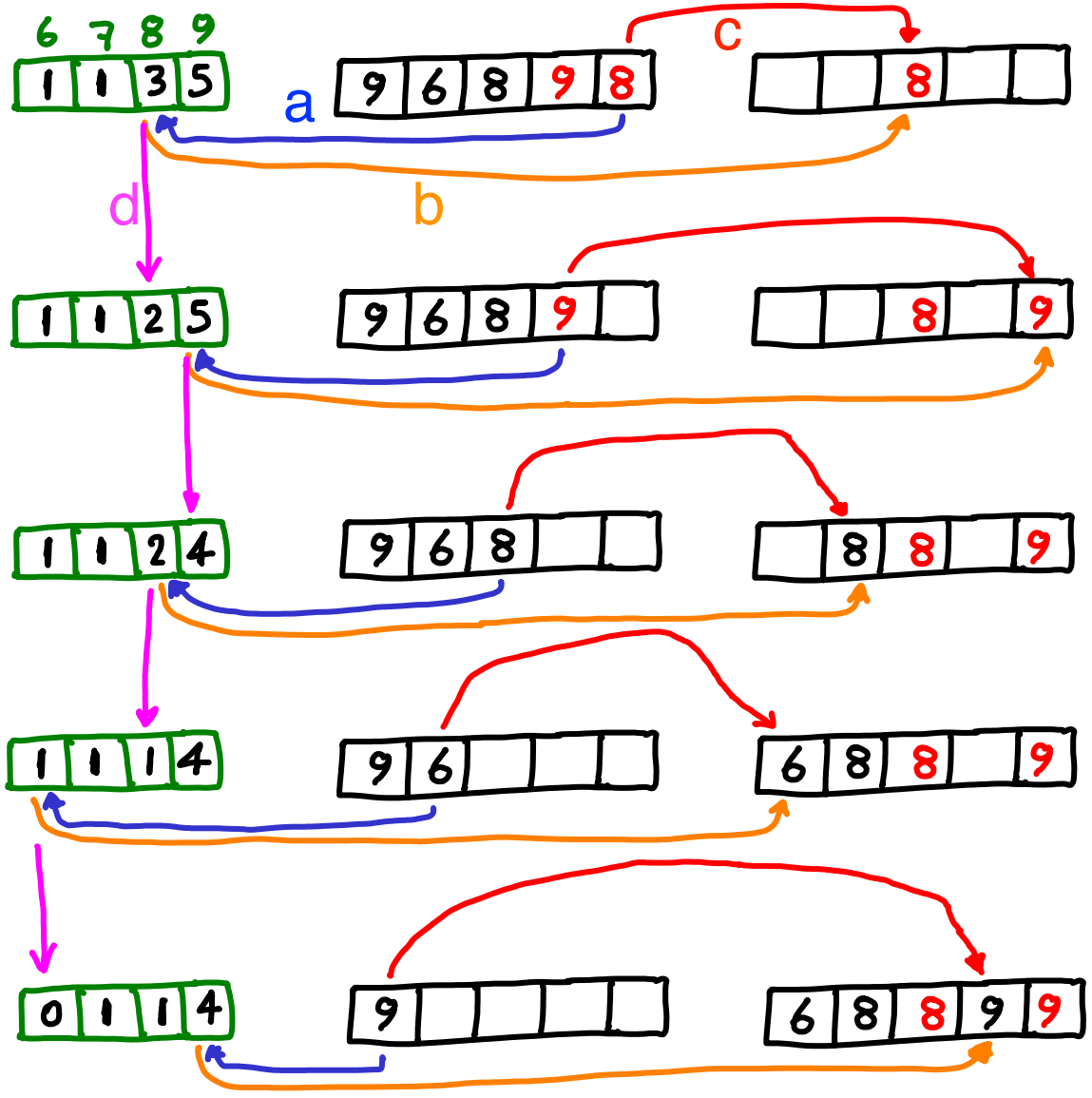
Use a second array of size k to get stable sorting

↳ how many integers are ≤ each value



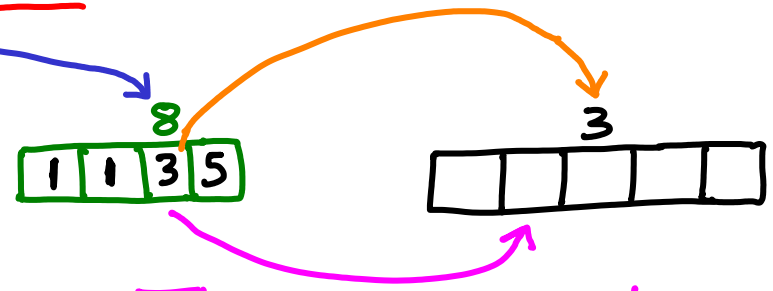
time: $\Theta(n) + \Theta(k)$

STABLE COUNTING SORT recap



the first time we point to the count array at index i...

... it tells us where the last element with value i must go



Then we decrement so the next element with value i (to the left in input array) will have its correct empty place.

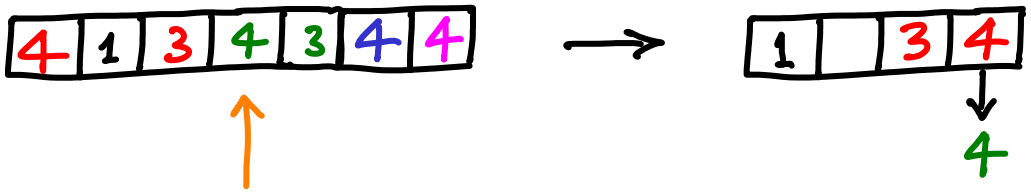
We will only decrement once per element with value i, so we maintain

in index i > max # in index i-1

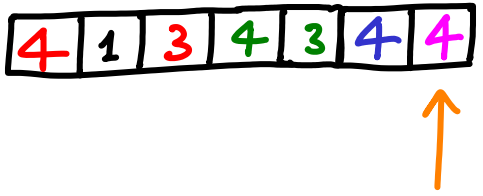
no conflicts

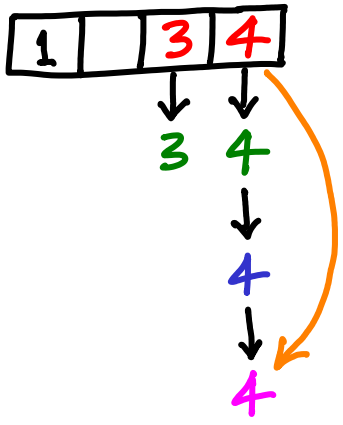
There is also a similar variant in which we use a helper array to record the leftmost target position for each value. (Omitted here)

A different version follows.



linked list





need a pointer to last element in linked list

otherwise $O(n^2)$

trivial →

$O(n)$ for all linked lists

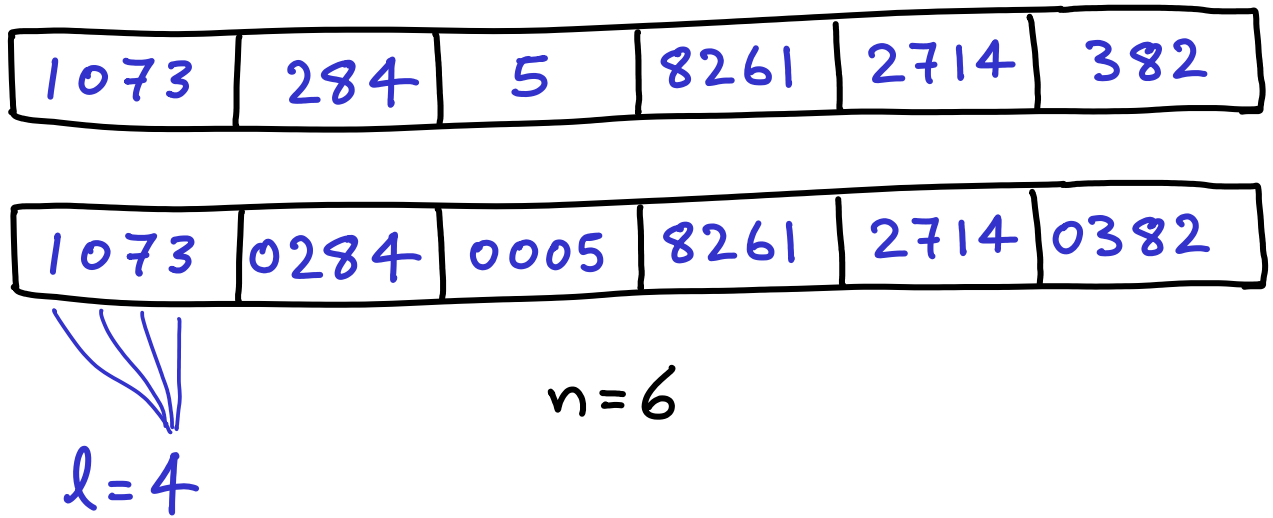
$+O(k)$ just to create the counter array

RADIX SORT

n: number of elements

l: (max) length of each element

r: radix (#symbols available at each digit) e.g., binary, decimal, hex



r = 9

(0...8)

There is a version of Radix sort
that does what most of us find intuitive:

Group all elements by most significant digit
& recurse within each group.

There is another way that is easier to implement.

RADIX SORT

3 2 9

4 5 7

6 5 7

8 3 9

4 3 6

7 2 0

3 5 5

$n = 7$

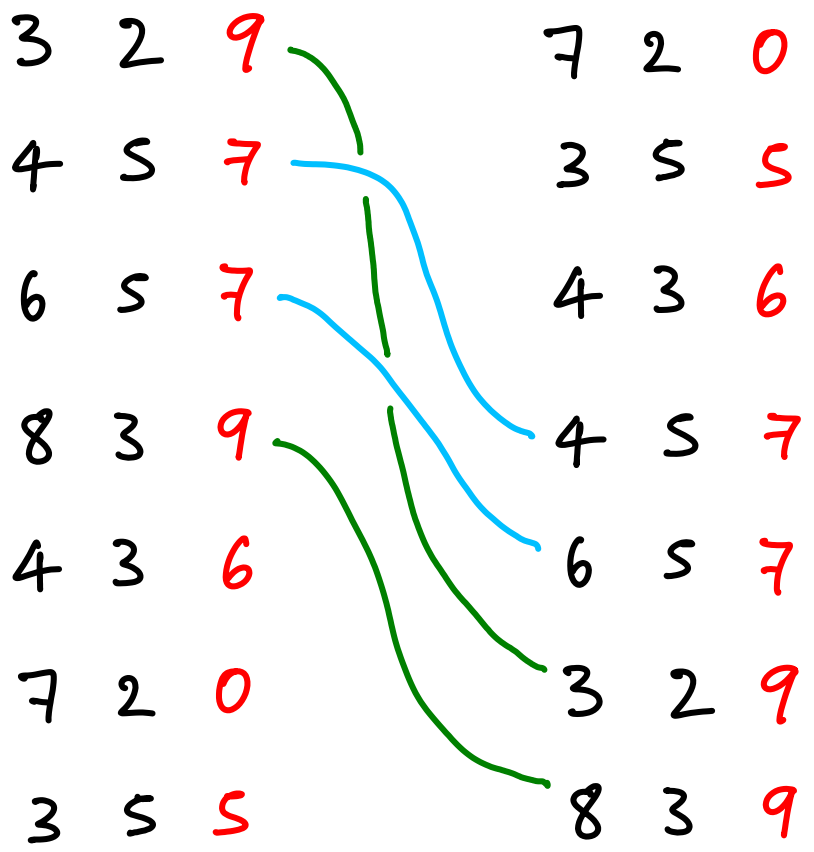
$l = 3$

$r = 10$

RADIX SORT

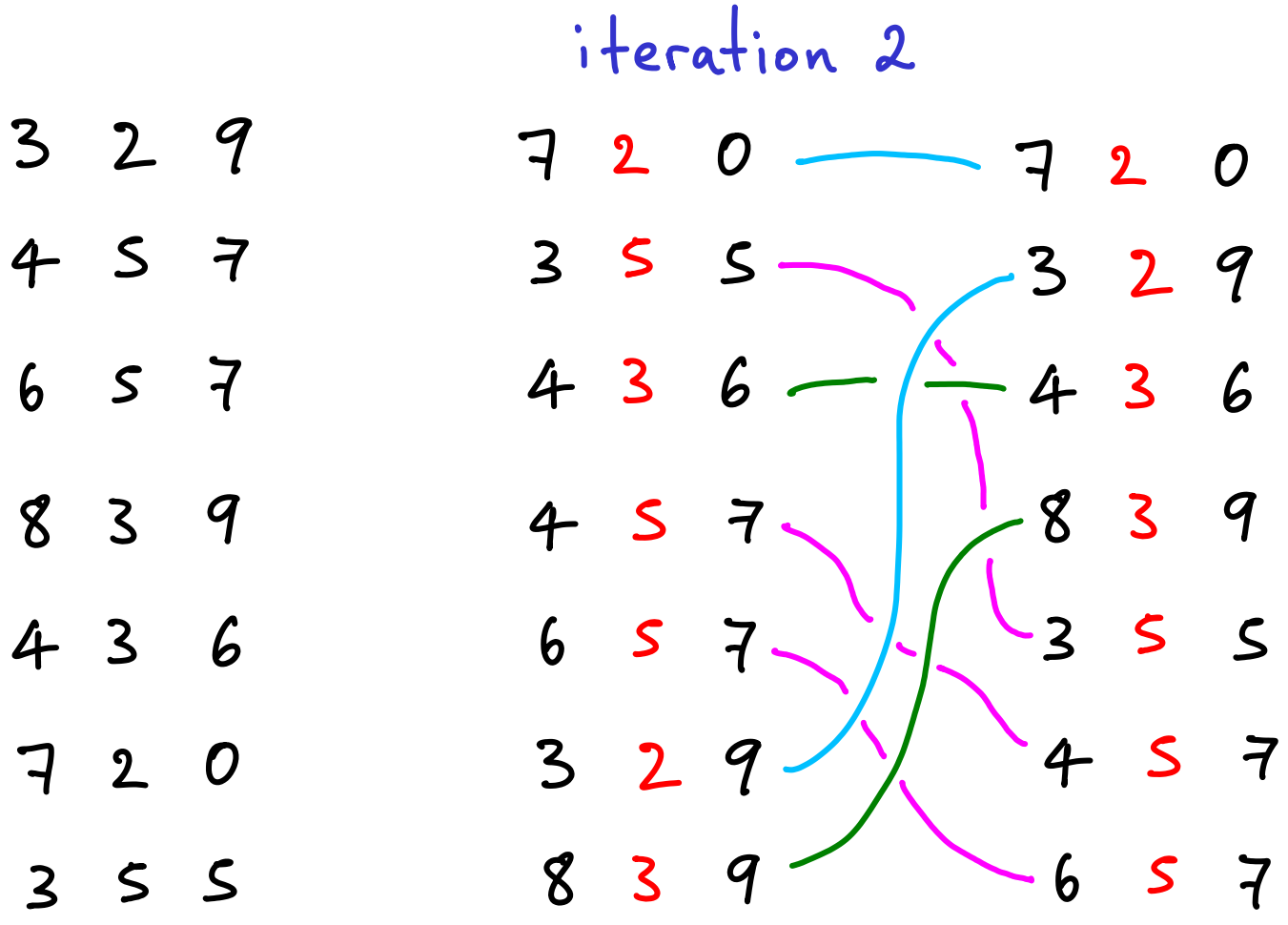
uses the least significant digit.

use stable counting sort → $\Theta(n+r)$



RADIX SORT

uses the least significant digit.



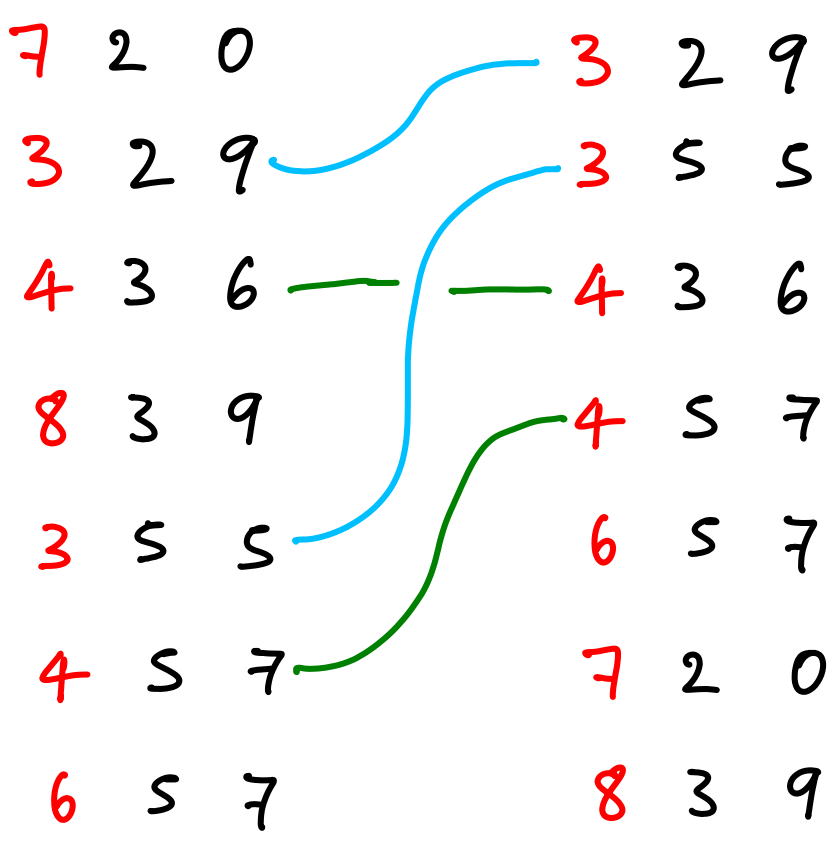
RADIX SORT

uses the least significant digit.

3	2	9
4	5	7
6	5	7
8	3	9
4	3	6
7	2	0
3	5	5

7	2	0
3	5	5
4	3	6
4	5	7
6	5	7
3	2	9
8	3	9

iteration 3



RADIX SORT uses the least significant digit.

$\Theta(l \cdot (n+r))$

3 2 9	7 2 0	7 2 0	3 2 9
4 5 7	3 5 5	3 2 9	3 5 5
6 5 7	4 3 6	4 3 6	4 3 6
8 3 9	4 5 7	8 3 9	4 5 7
4 3 6	6 5 7	3 5 5	6 5 7
7 2 0	3 2 9	4 5 7	7 2 0
3 5 5	8 3 9	6 5 7	8 3 9

Assume by induction:

after iteration i you have sorted all elements by the last i digits.

Then stable sort preserves correct order if there are ties at digit i+1