Sorting in linear time

Counting Sort

Sort in linear time an array A of ~~positive~~ non-negative integers if the maximum value k in A is $k = O(n)$

How? any idea?

$k = 6$

A	1	3	5	4	6	5
	1	2	3	4	5	6 = n

C	0	$\cancel{0}$ 1	0	$\cancel{0}$ 1	$\cancel{0}$ 1	$\cancel{0}$ $\cancel{1}$ 2	$\cancel{0}$ 1
	0	1	2	3	4	5	6 = k

How many 2s in A?

A_s 1 3 4 5 5 6

terrible approach (A)

for i = 0 to k
  scan A to count i

$\Theta(k \cdot n)$ time

```
CountingSort*(A, k)
    let C[0 ... k] be a new array
    for i = 0 to k                        // Θ(k)
        C[i] = 0
    for i = 1 to A.length                 // Θ(n)
        C[A[i]] += 1
    j = 1
    for i = 0 to k
        for p = 1 to C[i]                 // Θ(C[i])
            A[j] = i
            j += 1
```

$$\Theta\left(\sum_{i=0}^{k} \left(\underline{1} + \underline{C[i]}\right)\right)$$

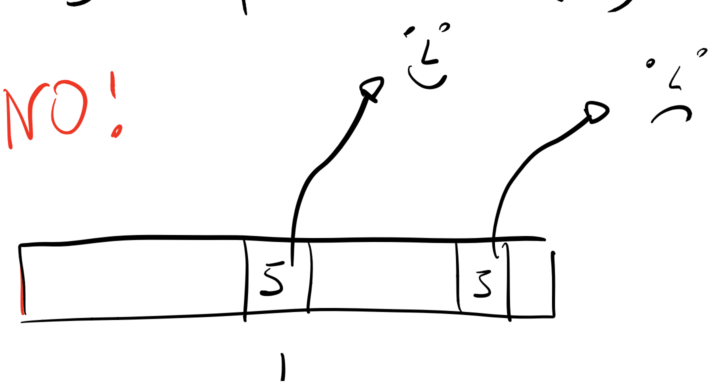
$$= \Theta(n + k)$$

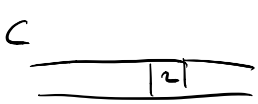
$\Theta(n + k)$ time

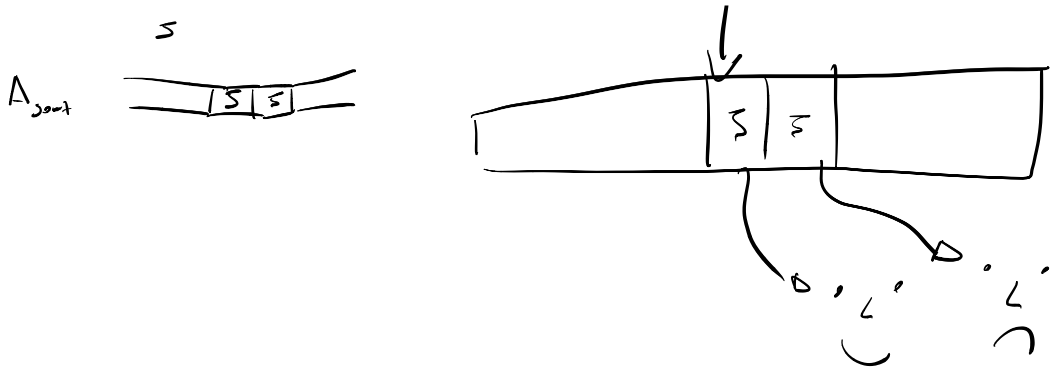
this is $\Theta(n)$ if $k = O(n)$

inplace? NO!

stable? NO!

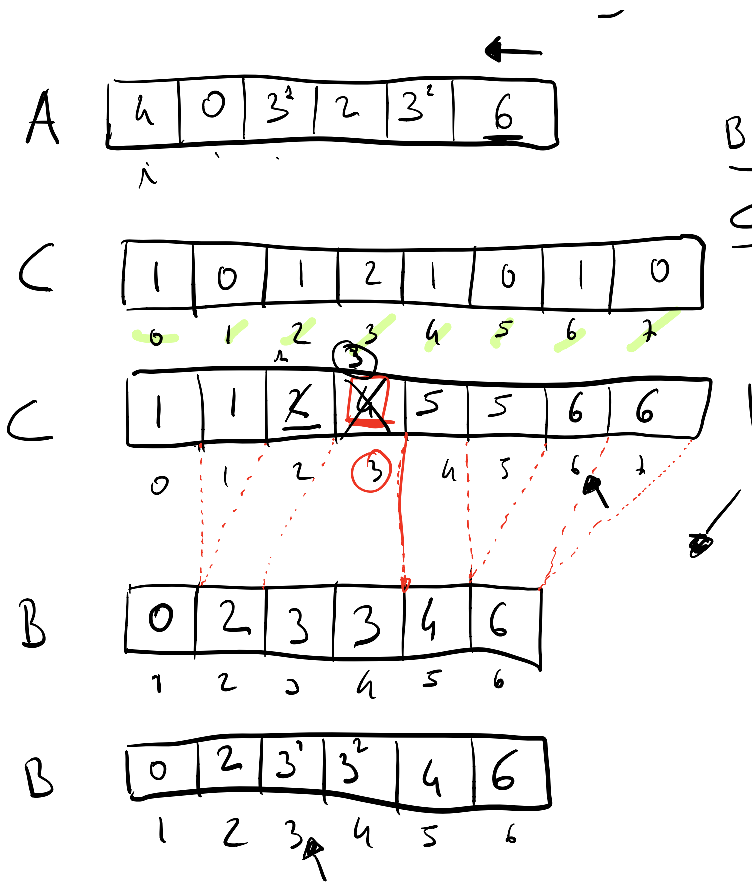






Stable Counting

```
Counting Sort (A, B, k)
  let C[0, .. k] be a new array
  for i = 0 to k
      C[i] = 0
  for i = 1 to A.length
      C[A[i]] += 1
  for i = 1 to k
      C[i] = C[i] + C[i-1]  // prefix sums
  // C[i] is the number of elements
  //  ≤ i in A  => last i in A sorted
  //               is in position C[i]
  for i = A.length down to 1
      B[C[A[i]]] = A[i]
      C[A[i]] -= 1
```



$B[C[A[i]]] = A[i]$

$C[A[i]] \mathrel{-}= 1$

Radix Sort

CS sorts in $\Theta(n)$ if $k = O(n)$

RS sorts in $\Theta(n)$ if $k = O(n^c)$
for some constant c

	CS	RS
n	1000	1000
k	$\boxed{3000}$	$\boxed{(1000)^4}$

1 000 000 000 000

process digits from most to least significant
intuitive but 10^d different group to deal with

329	329	329
457	355	355
657	457	436
839	436	457
436	657	657
720	720	720
355	839	839

process the digits from least to most significant

- no groups !
- big surprise it's correct

329	720	720	329
457	355	329	355
657	436	436	436
839	457	839	457
436	657	355	657
720	329	457	720
355	839	657	839

formal proof is by induction on the number of digits

number	3	4	2	9	7
i =	5	4	3	2	1

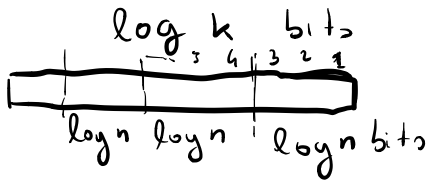
RadixSort (A, d)

for i = 1 to d
use a stable sorting algorithm
to sort A on digit i

RS take $\Theta(d(n+k))$ time
where k is maximum value for a digit

$\boxed{\text{Theory}}$ RS runs in $\Theta(n)$ if $k = O(n^c)$

for some constant c



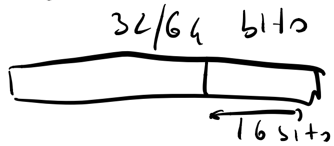
since $k = O(n^c)$, $\log k \approx c \log n$

thus, we have $\boxed{c}$ digits

we $\Theta(d(n + 2^{\log n}))$

$= \Theta(cn) = \Theta(n)$

REAL LIFE



64 bits number

16 bits each digit

4 rounds of counting sort

array C has size $2^{16} = 65536$

$\Theta(n + 65536)$ time for each counting sort

$= \Theta(n)$