Sorting in linear time

Counting Sort

Sort in linear time an array A of non-negative integers if $\max(A) = O(|A|)$, i.e. $\max(A) = k$, $|A| = n$

Example

10 10000 5

$|A| = 3$

$k = 10000$

Target time complexity is $O(n + k)$ time

Any idea ?

Pos	1	2	3	4	5	6
A	1	3	5	(4)	6	5

C	0	1	0	1	[1]	2	1
	0	1	2	3	(4)	5	6 = k

A_S 1 3 4 5 5 6

$C[A[i]] += 1$

Counting Sort* (A, k)

let $C[0 \ldots k]$ 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
```

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

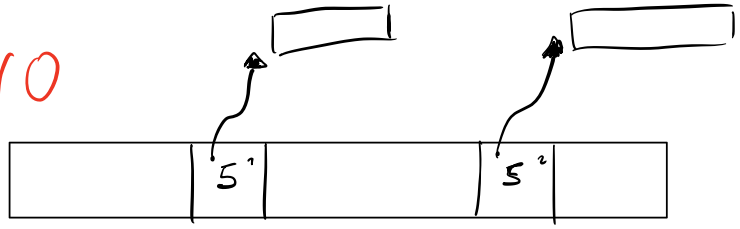
$$= \Theta(k + n)$$

Overall running time is $\Theta(k+n)$ time

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

In place? NO

stable? NO



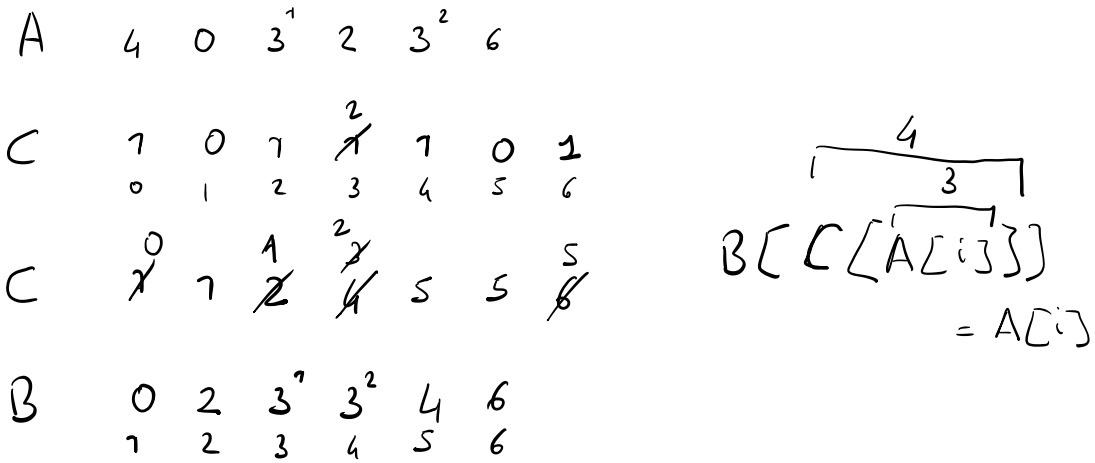
Stable Counting Sort

```
Counting Sort (A, k)
1  let C[0 ... k] be a new array
2  for   i = 0 to k                          // Θ(k)
3        C[i] = 0
4  for   i = 1 to A.length                   // Θ(n)
5        C[A[i]] += 1
6  for   i = 1 to k                          C[i] = Σ_{j=0}^{i} C[j]
7        C[i] = C[i] + C[i-1]                // prefix sums
                                             // Θ(k)
   // C[i] is the number of elements ≤ i in A
   // => values i goes in A_S[C[i-1]+1 ... C[i]]
8  for   i = A.lenght downto 1
9        B[C[A[i]]] = A[i]                   // Θ(n)
10       C[A[i]] -= 1
```

$\Theta(n+k)$ time

A	0	1	0	1	2	2	1	1	3	4	5	2
	0		1		2		3		4		5	
C	2		4		3		1		1		1	
C'	2		6		9		10		11		12	

A_S	0	0	1	1	1	2	2	2	3	4	5	
	1	2	3	4	5	6	7	8	9	10	11	12


A 4 0 3 2 3 6
C 1 0 1 1 1 0 1
0 1 2 3 4 5 6
C 0 1 1 2 2 5
B 0 2 3 3 4 6
1 2 3 4 5 6
4
3
B[C[A[i]]]
= A[i]

Radix Sort

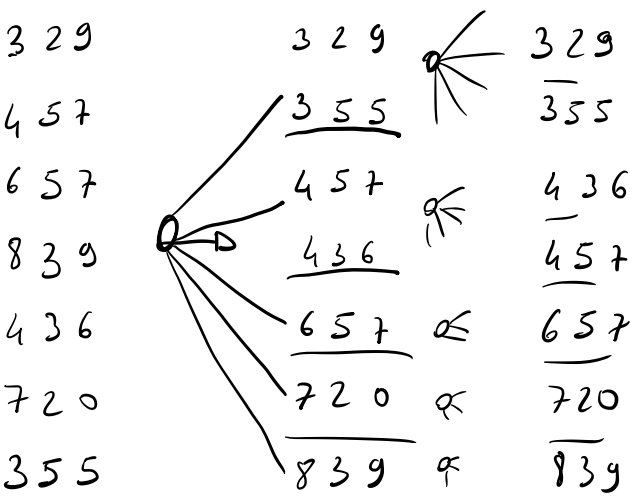
CS sorts in $\Theta(n)$ if $k = O(n)$

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

for some constant c

	CS	RS
n	1000	1000
k	3000	$(1000)^4 = 1.000.000.000.000$

Process digits from most significant one

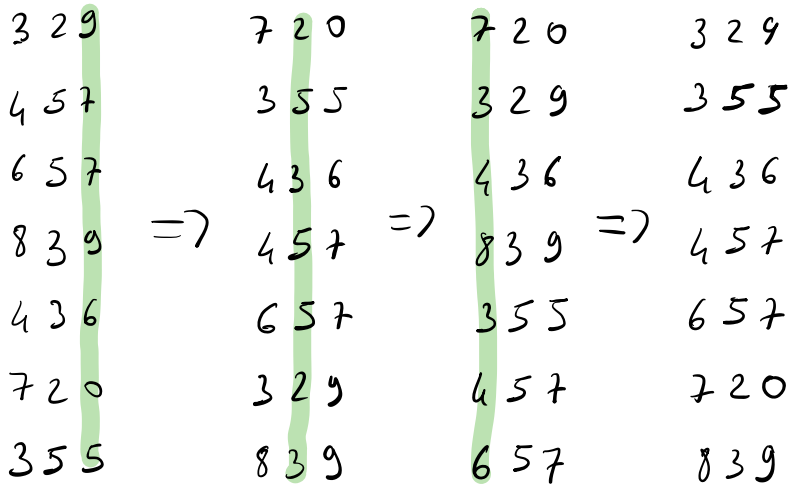


Alot of groups

$10 + 10^2 + 10^3 + \dots + 10^d$

process numbers from the least significant digit.

- no groups
- big surprise: it works!



correctness: proof by induction on d

RadixSort (A, d)

for i = 1 to d (i.e. counting sort)

use a stable sorting algorithm

to sort A on digit i

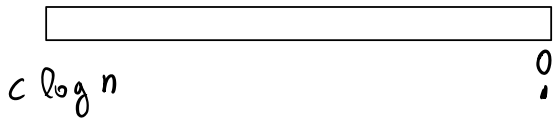
RS (with counting sort) $\Theta(d(n+k))$ time

where k is the largest value for a digit

$k = O(n^c) \leq n^c$

$\log n^c = c \log n$ bits to represent any number up to n^c

a number in A in binary



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

- if digits are bits,

  $d = c \log n \qquad k = 2$

  RS runs $\Theta(c \log n (n+2))$

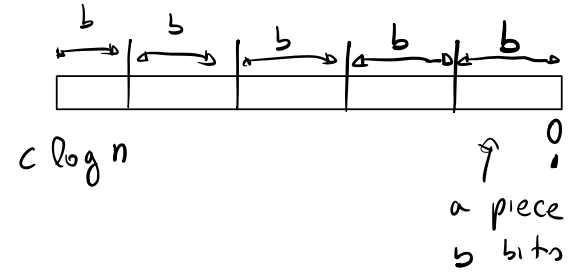
  $= \Theta(n \log n)$ time

- if $d = 1$,

  $k = n^c$

  RS runs $\Theta(1 \cdot (n + n^c)) = \Theta(n^c)$

- Let's optimize choices of d and k



$$d = \frac{c \log n}{b}$$

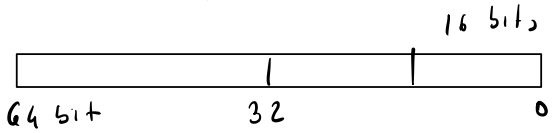
$$k = 2^b$$

RS runs $\Theta\left(\frac{c \log n}{b}\left(n + 2^b\right)\right)$ time

if $b = \log n$, $\Theta\left(\frac{c \log n}{\log n}\left(n + 2^{\log n}\right)\right)$

$= \Theta(2cn) = \Theta(n)$ time

Real life



- 4 rounds of CS
- array C has size $2^{16} = 65536$

RS runs in $\Theta(n + 65536)$