Sorting in linear time

Counting sort: No comparisons between elements.

- ***Input***: $A[1 \,.\,.\, n]$, where $A[j] \in \{1, 2, \ldots, k\}$.
- ***Output***: $B[1 \,.\,.\, n]$, sorted.
- ***Auxiliary storage***: $C[1 \,.\,.\, k]$.

Counting sort

for $i \leftarrow 1$ **to** k

  do $C[i] \leftarrow 0$

for $j \leftarrow 1$ **to** n

  do $C[A[j]] \leftarrow C[A[j]] + 1$ $\rhd$ $C[i] = |\{\text{key} = i\}|$

for $i \leftarrow 2$ **to** k

  do $C[i] \leftarrow C[i] + C[i-1]$ $\rhd$ $C[i] = |\{\text{key} \le i\}|$

for $j \leftarrow n$ **downto** 1

do $B[C[A[j]]] \leftarrow \mathrm{A}[j]$

  $C[A[j]] \leftarrow C[A[j]] - 1$

Counting-sort example

	1	2	3	4	5
A:	4	1	3	4	3
B:					

	1	2	3	4
C:				

Loop 1

	1	2	3	4	5
A:	4	1	3	4	3
B:					

	1	2	3	4
C:	0	0	0	0

for $i \leftarrow 1$ **to** k

  do $C[i] \leftarrow 0$

Loop 2

	1	2	3	4	5
A:	4	1	3	4	3
B:					

	1	2	3	4
C:	0	0	0	1

for $j \leftarrow 1$ **to** n

  do $C[A[j]] \leftarrow C[A[j]] + 1 \quad \triangleright\ C[i] = |\{\text{key} = i\}|$

Loop 2

	1	2	3	4	5
A:	4	1	3	4	3

	1	2	3	4
C:	1	0	0	1

B:					

for $j \leftarrow 1$ **to** n

  do $C[A[j]] \leftarrow C[A[j]] + 1$ $\triangleright$ $C[i] = |\{\text{key} = i\}|$

Loop 2

A:

1	2	3	4	5
4	1	3	4	3

C:

1	2	3	4
1	0	1	1

B:

for $j \leftarrow 1$ **to** n

  do $C[A[j]] \leftarrow C[A[j]] + 1$ $\triangleright$ $C[i] = |\{\text{key} = i\}|$

Loop 2

	1	2	3	4	5
A:	4	1	3	4	3

	1	2	3	4
C:	1	0	1	2

B:					

for $j \leftarrow 1$ **to** n

  do $C[A[j]] \leftarrow C[A[j]] + 1$ $\rhd$ $C[i] = |\{\text{key} = i\}|$

Loop 2

	1	2	3	4	5
A:	4	1	3	4	3

	1	2	3	4
C:	1	0	2	2

B:					

for $j \leftarrow 1$ **to** n

  do $C[A[j]] \leftarrow C[A[j]] + 1$ $\triangleright$ $C[i] = |\{\text{key} = i\}|$

Loop 3

	1	2	3	4	5
A:	4	1	3	4	3
B:					

	1	2	3	4
C:	1	0	2	2
C':	1	1	2	2

for $i \leftarrow 2$ **to** k

  do $C[i] \leftarrow C[i] + C[i-1]$ $\rhd$ $C[i] = |\{\text{key} \le i\}|$

Loop 3

	1	2	3	4	5
A:	4	1	3	4	3
B:					

	1	2	3	4
C:	1	0	2	2
C':	1	1	3	2

for $i \leftarrow 2$ **to** k

  do $C[i] \leftarrow C[i] + C[i-1]$ $\rhd$ $C[i] = |\{\text{key} \leq i\}|$

Loop 3

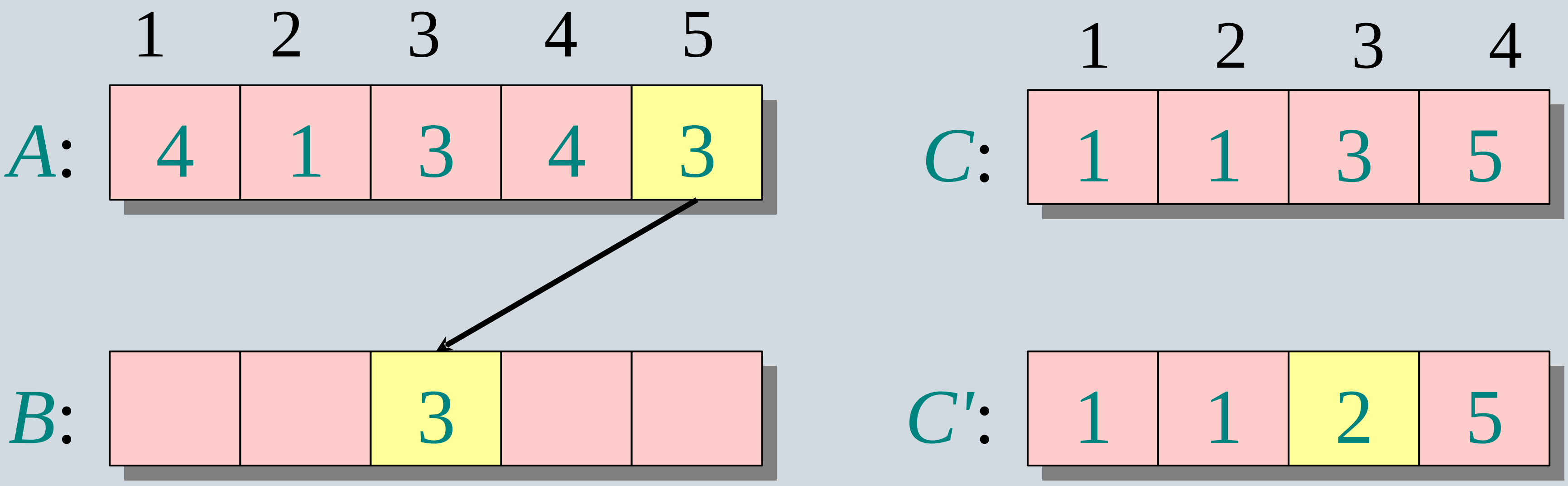
	1	2	3	4	5
A:	4	1	3	4	3
B:					

	1	2	3	4
C:	1	0	2	2
C':	1	1	3	5

for $i \leftarrow 2$ **to** k

  do $C[i] \leftarrow C[i] + C[i-1]$ $\triangleright$ $C[i] = |\{\text{key} \leq i\}|$

Loop 4

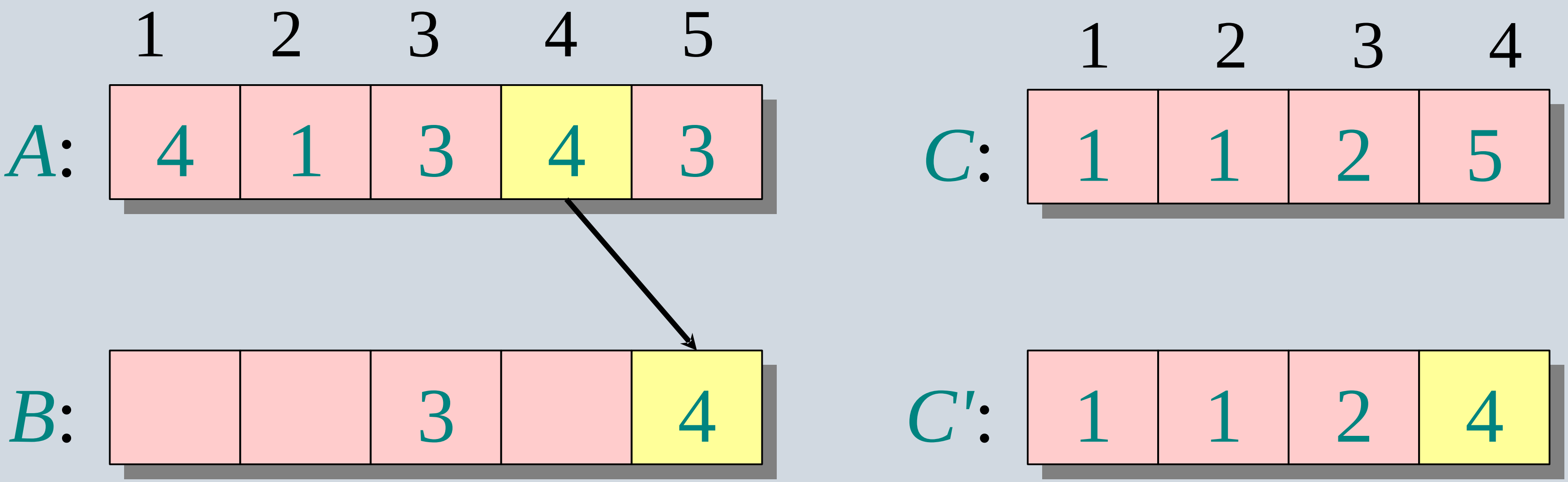


for $j \leftarrow n$ **downto** 1

do $B[C[A[j]]] \leftarrow \mathrm{A}[j]$

$\quad C[A[j]] \leftarrow C[A[j]] - 1$

Loop 4

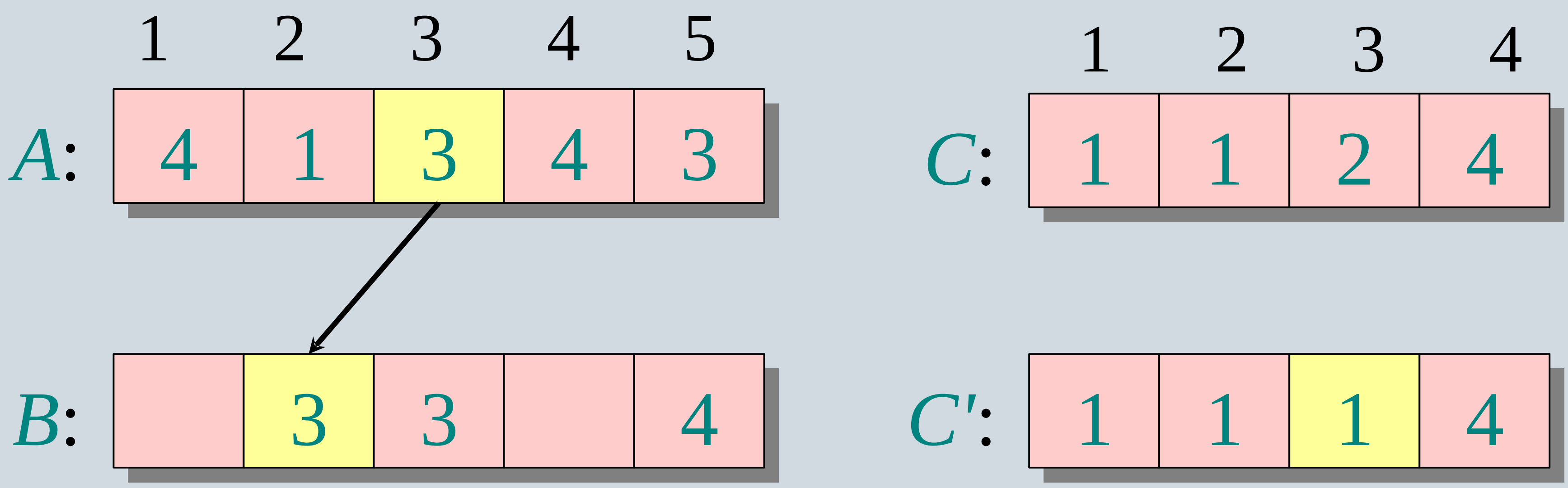


for $j \leftarrow n$ **downto** 1

do $B[C[A[j]]] \leftarrow \mathrm{A}[j]$

$\quad C[A[j]] \leftarrow C[A[j]] - 1$

Loop 4

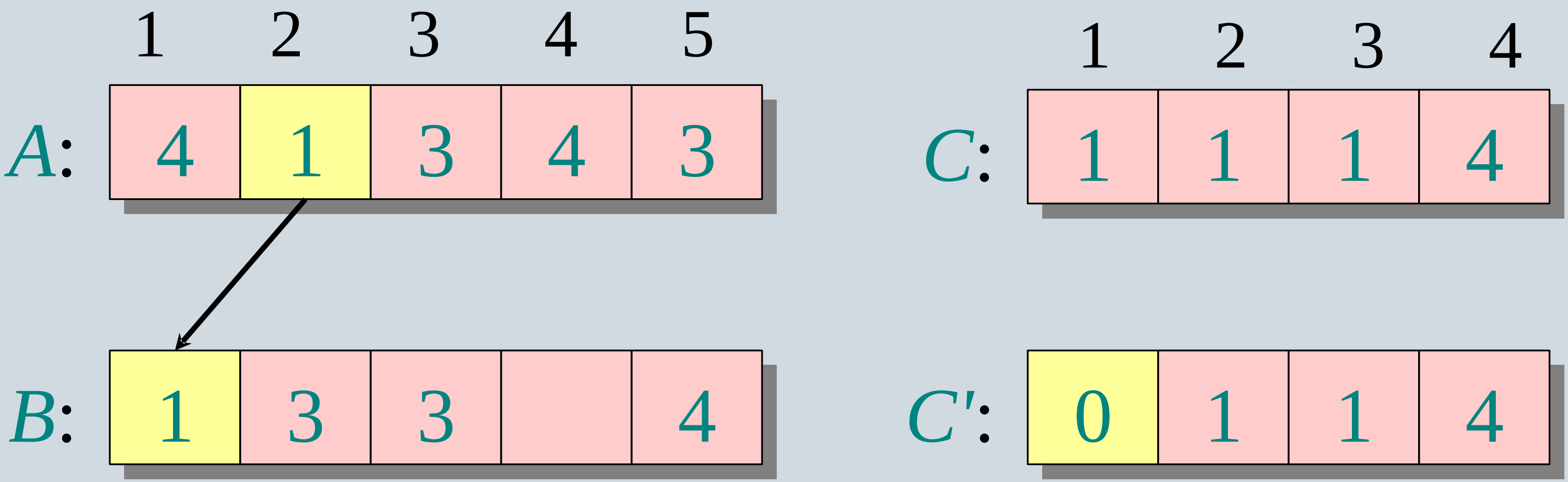


for $j \leftarrow n$ **downto** 1

do $B[C[A[j]]] \leftarrow A[j]$

$C[A[j]] \leftarrow C[A[j]] - 1$

Loop 4

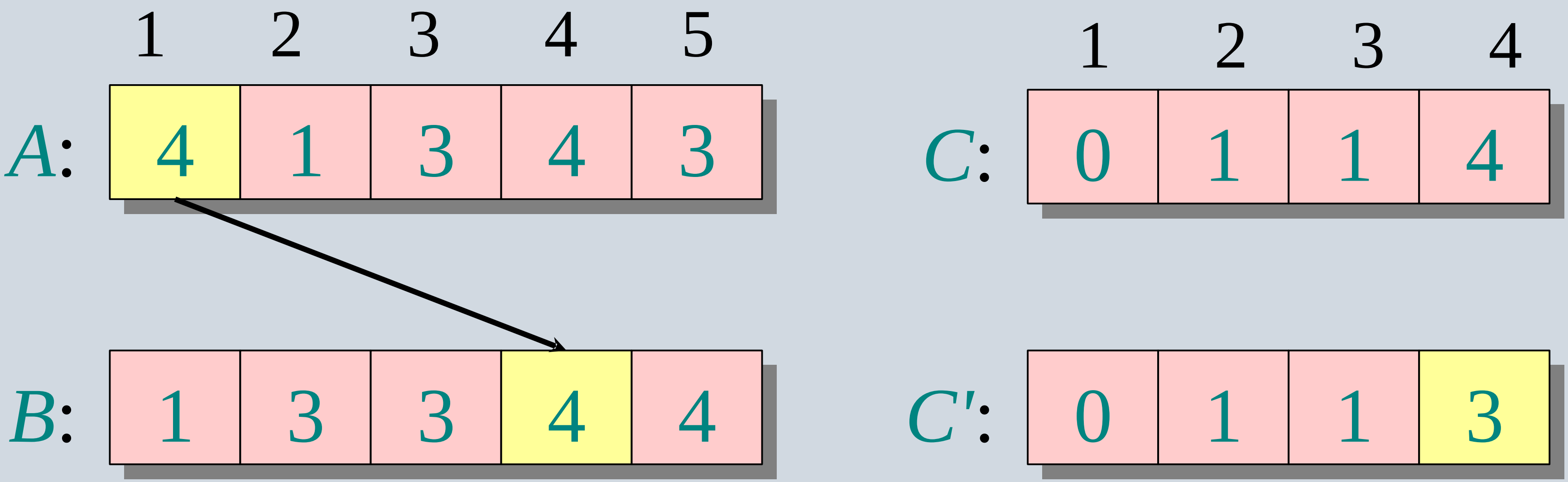


for $j \leftarrow n$ **downto** 1

do $B[C[A[j]]] \leftarrow \mathrm{A}[j]$

$\quad C[A[j]] \leftarrow C[A[j]] - 1$

Loop 4



for $j \leftarrow n$ **downto** 1

do $B[C[A[j]]] \leftarrow A[j]$

$C[A[j]] \leftarrow C[A[j]] - 1$

Analysis

$$\Theta(k) \quad \left\{ \begin{array}{l} \textbf{for } i \leftarrow 1 \textbf{ to } k \\ \qquad \textbf{do } C[i] \leftarrow 0 \end{array} \right.$$

$$\Theta(n) \quad \left\{ \begin{array}{l} \textbf{for } j \leftarrow 1 \textbf{ to } n \\ \qquad \textbf{do } C[A[\,j]] \leftarrow C[A[\,j]] + 1 \end{array} \right.$$

$$\Theta(k) \quad \left\{ \begin{array}{l} \textbf{for } i \leftarrow 2 \textbf{ to } k \\ \qquad \textbf{do } C[i] \leftarrow C[i] + C[i-1] \end{array} \right.$$

$$\Theta(n) \quad \left\{ \begin{array}{l} \textbf{for } j \leftarrow n \textbf{ downto } 1 \\ \textbf{do } B[C[A[\,j]]] \leftarrow \mathrm{A}[\,j] \\ \qquad C[A[\,j]] \leftarrow C[A[\,j]] - 1 \end{array} \right.$$

$$\Theta(n + k)$$

Running time

If $k = O(n)$, then counting sort takes $\Theta(n)$ time.

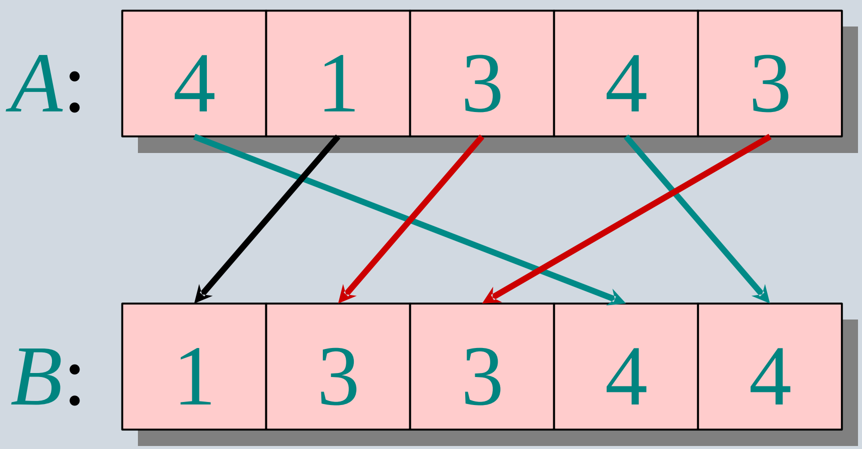
- But, sorting takes $\Omega(n \lg n)$ time!
- Where's the fallacy?

Answer:

- ***Comparison sorting*** takes $\Omega(n \lg n)$ time.
- Counting sort is not a ***comparison sort***.
- In fact, not a single comparison between elements occurs!

Stable sorting

Counting sort is a ***stable*** sort: it preserves the input order among equal elements.



Radix sort

- ***Origin***: Herman Hollerith's card-sorting machine for the 1890 U.S. Census.

- Digit-by-digit sort.
- Hollerith's original (bad) idea: sort on most-significant digit first.
- Good idea: Sort on ***least-significant digit first*** with auxiliary ***stable*** sort.

Operation of radix sort

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

Correctness of radix sort

Induction on digit position

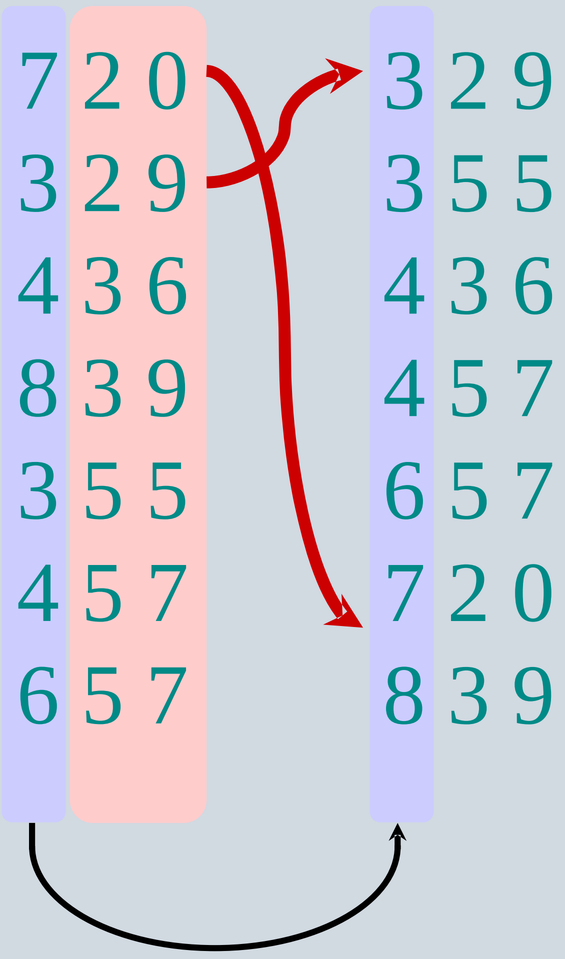
- Assume that the numbers are sorted by their low-order $t-1$ digits.
- Sort on digit t

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

Correctness of radix sort

Induction on digit position

- Assume that the numbers are sorted by their low-order $t-1$ digits.
- Sort on digit t
  - ③ Two numbers that differ in digit t are correctly sorted.



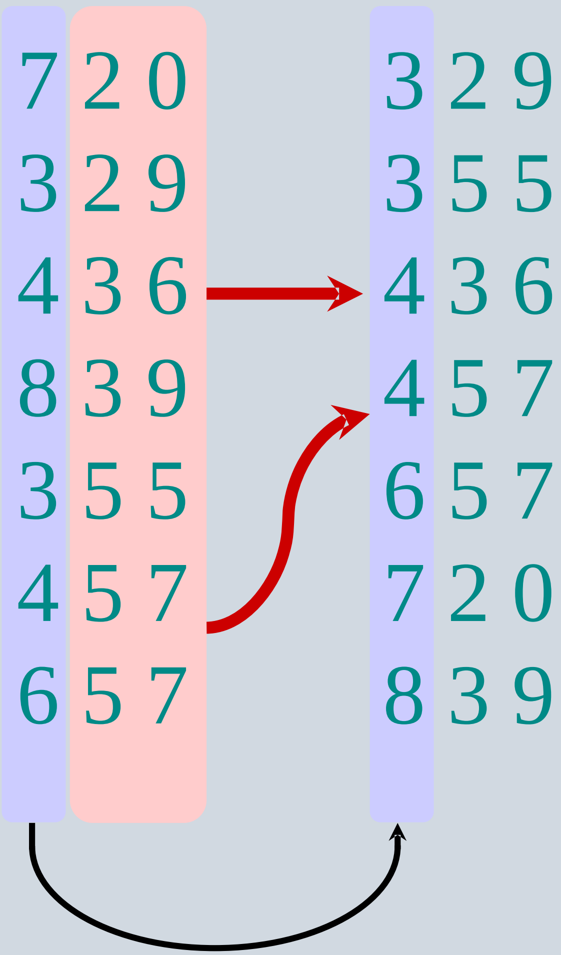
Correctness of radix sort

Induction on digit position

- Assume that the numbers are sorted by their low-order $t-1$ digits.
- Sort on digit t

  ③ Two numbers that differ in digit t are correctly sorted.

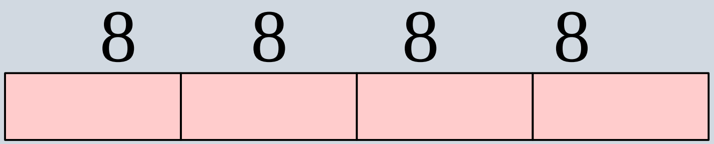
  ③ Two numbers equal in digit t are put in the same order as the input ⇒ correct order.



Analysis of radix sort

- Assume counting sort is the auxiliary stable sort.
- Sort n computer words of b bits each.
- Each word can be viewed as having b/r base-2^r digits.

Example: 32-bit word



$r = 8 \Rightarrow b/r = 4$ passes of counting sort on base-2^8 digits; or $r = 16 \Rightarrow b/r = 2$ passes of counting sort on base-2^{16} digits.

How many passes should we make?