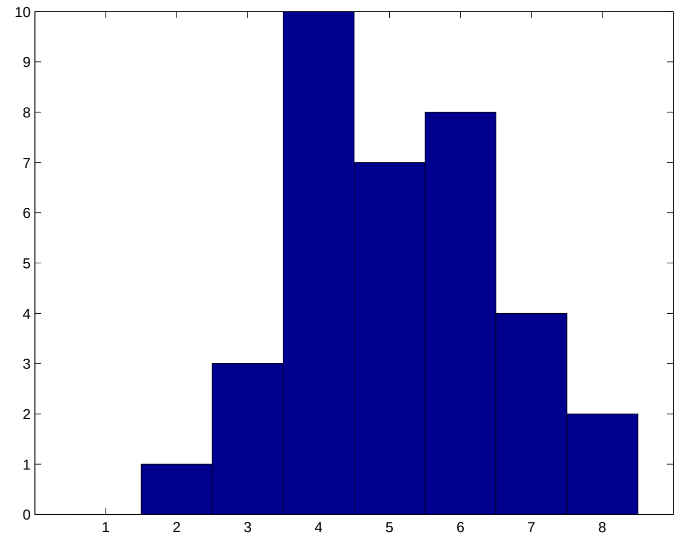
Ranking

- There are 8 questions in Quiz 1 and I figured out 6 of them, what's my ranking in the class?

- A straightforward way to find this out is to draw a histogram. That's the underlying idea for counting sort.

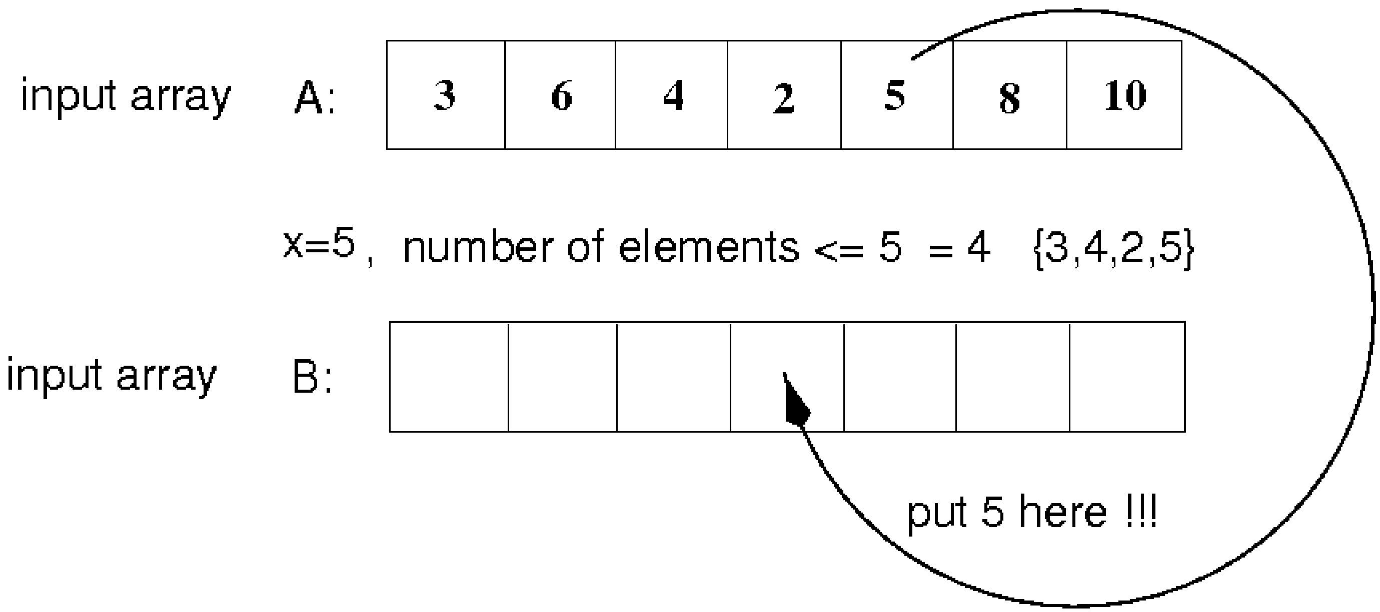


'Beat' the Lower Bound Linear-Time Sorting Algorithms: Counting Sort

- **Assumptions**
  - Each of the n input elements is an integer in the range $[1..r]$ (i.e., $1 \leq A[i] \leq r, i \leq n$)
  - $r = O(n)(r \leq cn)$ (e.g., if $n = 100$, then r can be equal to 100, 200, but not 100^2).

- **Basic idea**
  - For each input element x, find the number of elements $\leq x$ (For each person, find the number of people who scored less).
  - Place x directly in the correct position ("ties" should be taken care of).

Example


input array A:
3
6
4
2
5
8
10
x=5 , number of elements <= 5 = 4 {3,4,2,5}
input array B:
put 5 here !!!

Get the value for each histogram bin

Finding the number of times $A[i]$ appears in $A, 1 \leq i \leq n$:

- Allocate $C[1..r]$ (C is the histogram).

- For each $1 \leq i \leq n$, do $C[A[i]] + +$.

Example

input array A:

3	6	4	1	3	4	1	4

allocate C

1	2	3	4	5	6
0	0	0	0	0	0

i=1, A[1]=3

1	2	3	4	5	6
0	0	1	0	0	0

C[A[1]]=C[3]=1

i=2, A[2]=6

1	2	3	4	5	6
0	0	1	0	0	1

C[A[2]]=C[6]=1

i=3, A[3]=4

1	2	3	4	5	6
0	0	1	1	0	1

C[A[3]]=C[4]=1

.
.
.

i=8, A[8]=4

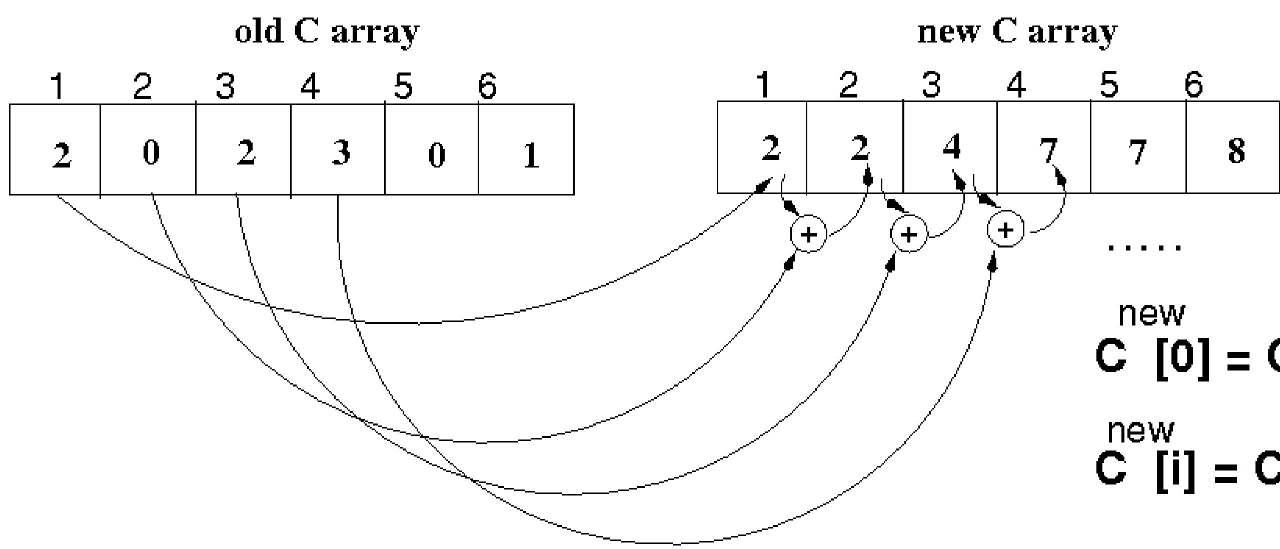
1	2	3	4	5	6
2	0	2	3	0	1

C[A[8]]=C[4]=3

C[i] = number of times element i appears in A

Find the number of people who scored less than you (Find the number of elements $\leq A[i]$)

Compute the cumulative sums (cumulative histogram)



$$C^{new}[0] = C^{old}[0]$$

$$C^{new}[i] = C^{new}[i-1] + C^{old}[i]$$

$$C[i] = \#\text{ elements} <= i$$

Tie breaker: put people in their original order

- Start from the last element of A.
- Decrease C[A[i]] every time A[i] is placed in the correct order.

	1	2	3	4	5	6	7	8
A:	3	6	4	1	3	4	1	4

	1	2	3	4	5	6
C:	2	2	4	7	7	8

	1	2	3	4	5	6
C:	2	2	4	6	7	8

	1	2	3	4	5	6	7	8
B:							4	

	1	2	3	4	5	6
C:	1	2	4	6	7	8

	1	2	3	4	5	6	7	8
B:		1					4	

	1	2	3	4	5	6
C:	1	2	4	5	7	8

	1	2	3	4	5	6	7	8
B:		1				4	4	

⋮

	1	2	3	4	5	6
C:	0	2	1	5	7	7

	1	2	3	4	5	6	7	8
B:	1	1	3	3	4	4	4	6

Algorithm

```
COUNTING-SORT(A, B, r) {
    for i := 0 to r
        C[i] := 0                              Θ(r)
    for j := 1 to length[A]
        C[A[j]] := C[A[j]] + 1;                Θ(n)
    /* Now C is the histogram */
    for i := 1 to r
        C[i] := C[i] + C[i − 1];               Θ(r)
    /* Now C is the cumulative histogram */
    for j := n to 1
        B[C[A[j]]] := A[j];                    Θ(n)
        C[A[j] := C[A[j]] - 1;
}
```

Analysis of Time Complexity

Time Complexity is $\Theta(r) + \Theta(n) = \Theta(r + n)$.
If $r = O(n)$, then $\Theta(r + n) = \Theta(n)$.

Where did we beat the $\Omega(nlgn)$ lower bound?

- **Comments:**
  - Counting sort is not an in place sort.
  - Counting sort is stable (elements with the same value appear in the output array in the same order they do in the input array).

Radix Sort

A **radix** is the number of unique digits used to represent a number in a positional numeral system.

- In the decimal system, the radix is 10. For example the number "42" has two digits, which are 4 and 2.
- In hexadecimal, the radix is 16, and each digit is 4 bits wide. For example the hexadecimal number 0xAB has two digits, A and B.

The Radix Sort first sorts the input values according to their least significant digit, then according to the second lest significant digit, and so on. The Radix Sort is then a multipass sort, and the number of passes equals the number of digits in the input values.

Least Significant Digit First

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

Algorithm

RADIX-SORT(A, d) $\{$

for i := 1 to d

use a **stable** sort to sort array A on digit i

$\}$

Complexity: Given n d-digit numbers in which each digit can take up to r possible values, Radix-Sort correctly sorts these numbers in $\Theta(d(n+r))$.

Can we sort on the "most significant digit first"?

Why must it be “a stable sort”?