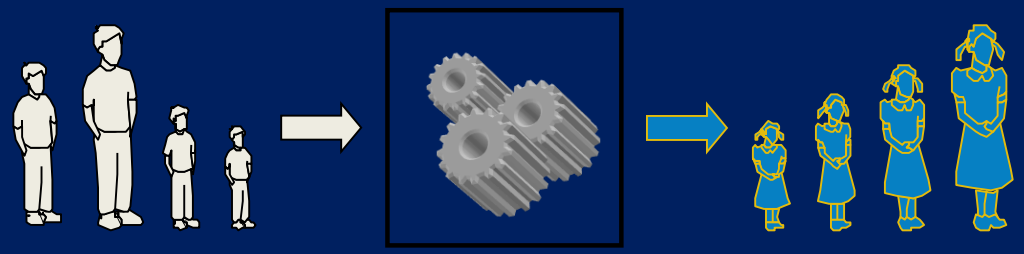
GLA UNIVERSITY MATHURA
Recognised by UGC Under Section 2(f)
Accredited with A Grade by NAAC

DESIGN & ANALYSIS OF ALGORITHM (BCSC0012)

Chapter 4: Sorting

Bubble Sort

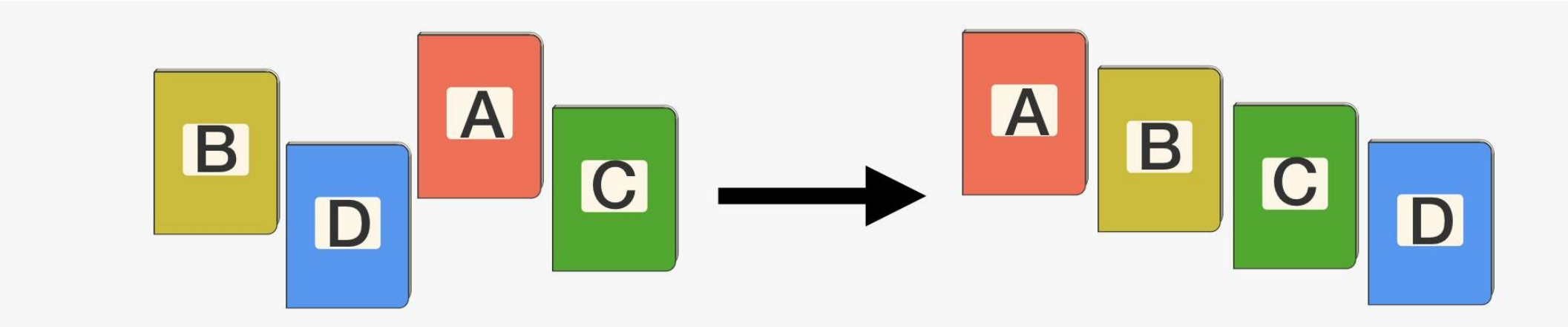


Prof. Anand Singh Jalal

Department of Computer Engineering & Applications

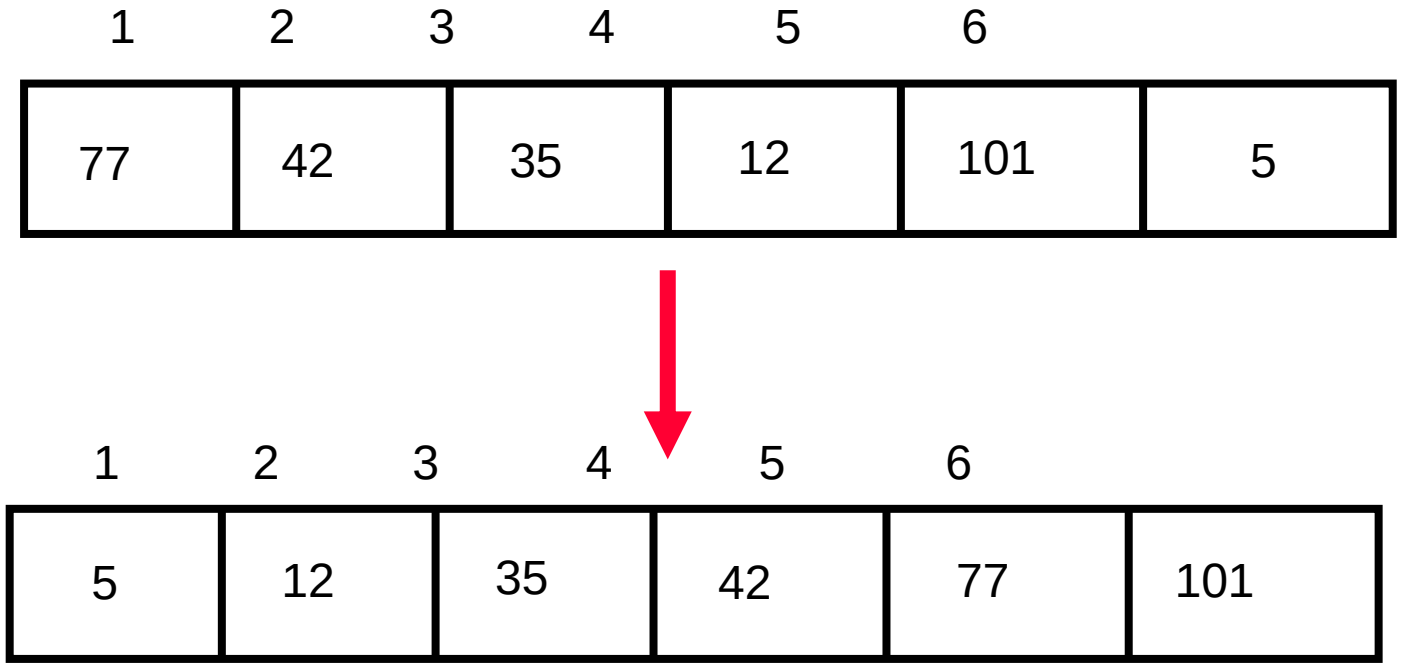
The Sorting Problem

- **Input:**
  - A sequence of n numbers $a_1, a_2, \ldots, a_n$
- **Output:**
  - A permutation (reordering) $a_1', a_2', \ldots, a_n'$ of the input sequence such that $a_1' \leq a_2' \leq \cdots \leq a_n'$



The Sorting Problem

- **Sorting takes an unordered collection and makes it an ordered one.**



The Sorting Problem: Some Definitions

- **Internal Sort**
  - The data to be sorted is all stored in the computer's main memory.
- **External Sort**
  - Some of the data to be sorted might be stored in some external, slower, device.
- **In Place Sort**
  - The amount of extra space required to sort the data is constant with the input size.

The Sorting Problem: Some Definitions

Stable sort

A STABLE sort preserves relative order of records with equal keys

Sorted on first key:

Aaron	4	A	664-480-0023	097 Little
Andrews	3	A	874-088-1212	121 Whitman
Battle	4	C	991-878-4944	308 Blair
Chen	2	A	884-232-5341	11 Dickinson
Fox	1	A	243-456-9091	101 Brown
Furia	3	A	766-093-9873	22 Brown
Gazsi	4	B	665-303-0266	113 Walker
Kanaga	3	B	898-122-9643	343 Forbes
Rohde	3	A	232-343-5555	115 Holder
Quilici	1	C	343-987-5642	32 McCosh

Sort file on second key:

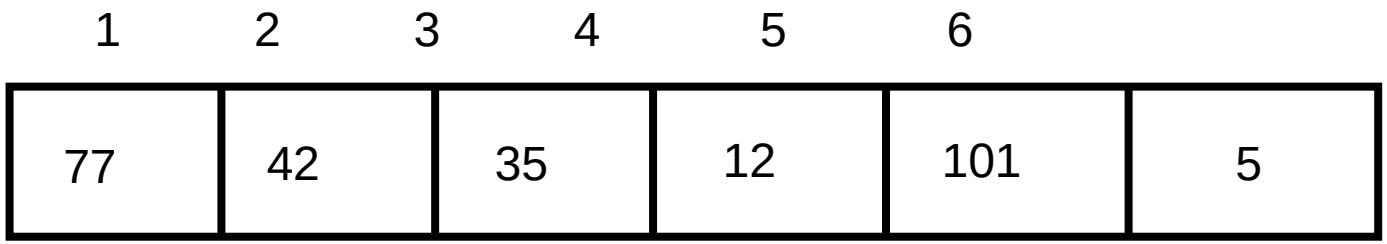
Fox	1	A	243-456-9091	101 Brown
Quilici	1	C	343-987-5642	32 McCosh
Chen	2	A	884-232-5341	11 Dickinson
Kanaga	3	B	898-122-9643	343 Forbes
Andrews	3	A	874-088-1212	121 Whitman
Furia	3	A	766-093-9873	22 Brown
Rohde	3	A	232-343-5555	115 Holder
Battle	4	C	991-878-4944	308 Blair
Gazsi	4	B	665-303-0266	113 Walker
Aaron	4	A	664-480-0023	097 Little

Records with key value 3 are not in order on first key!!

Bubble Sort

"Bubbling Up" the Largest Element

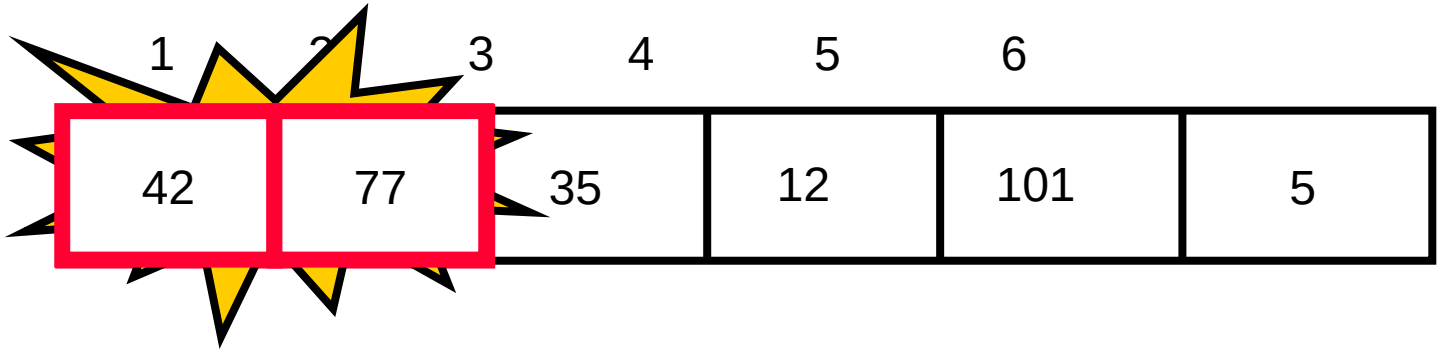
- **Traverse a collection of elements**
  - **Move from the front to the end**
  - **“Bubble” the largest value to the end using pair-wise comparisons and swapping**



Bubble Sort ...

"Bubbling Up" the Largest Element

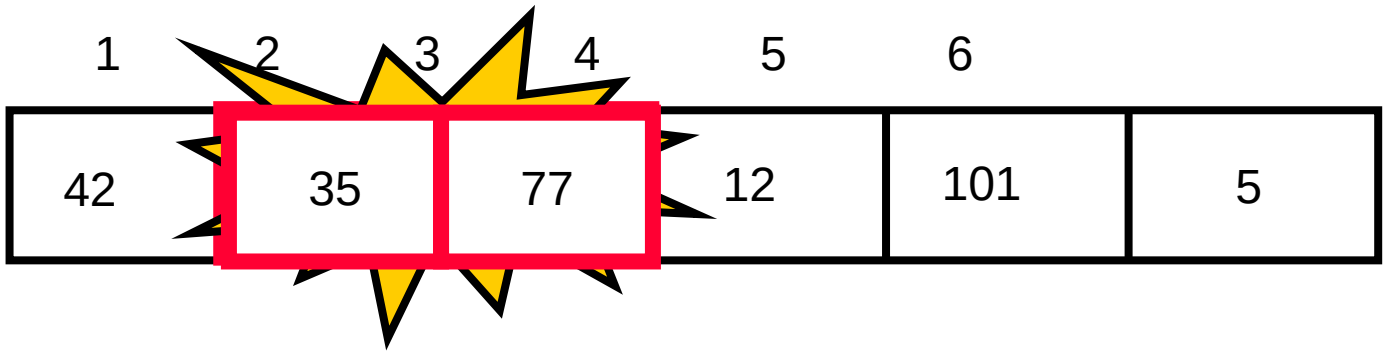
- Traverse a collection of elements
  - Move from the front to the end
  - “Bubble” the largest value to the end using pair-wise comparisons and swapping



Bubble Sort ...

"Bubbling Up" the Largest Element

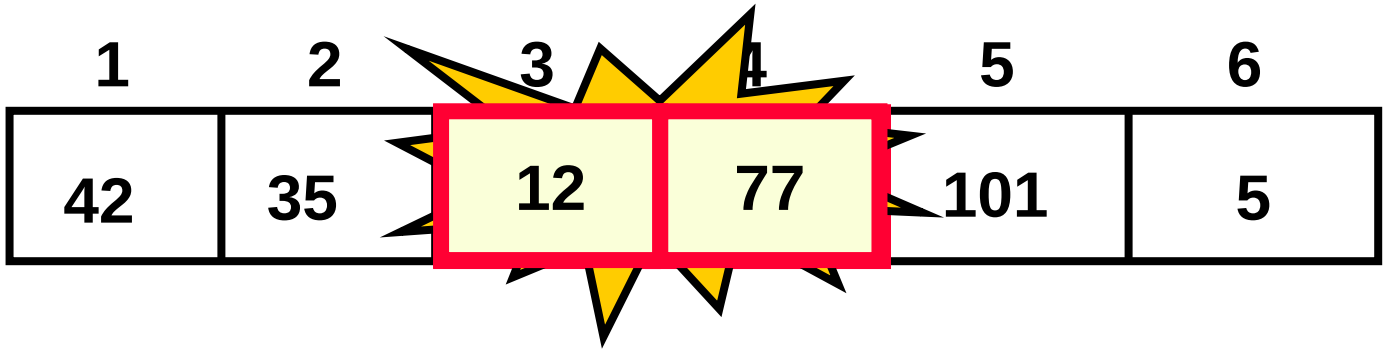
- **Traverse a collection of elements**
  - **Move from the front to the end**
  - **“Bubble” the largest value to the end using pair-wise comparisons and swapping**



Bubble Sort ...

"Bubbling Up" the Largest Element

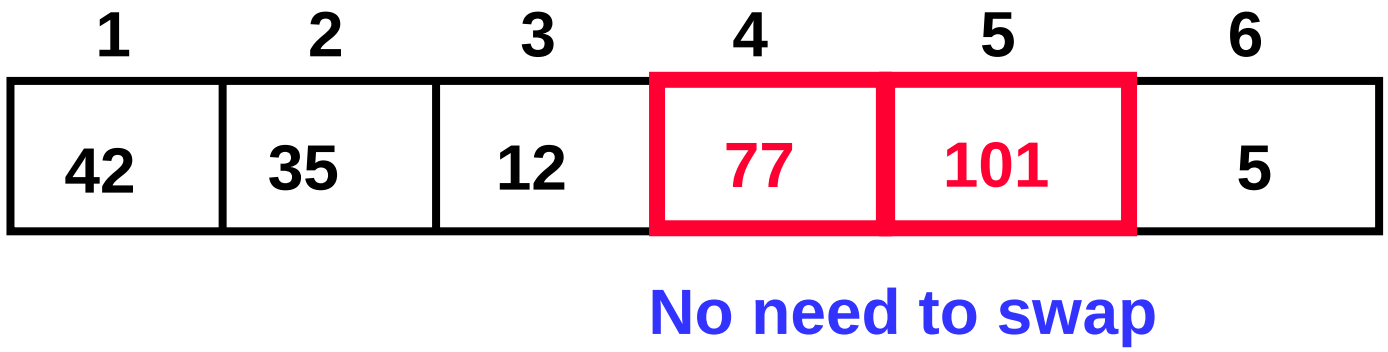
- Traverse a collection of elements
  - Move from the front to the end
  - “Bubble” the largest value to the end using pair-wise comparisons and swapping



Bubble Sort ...

"Bubbling Up" the Largest Element

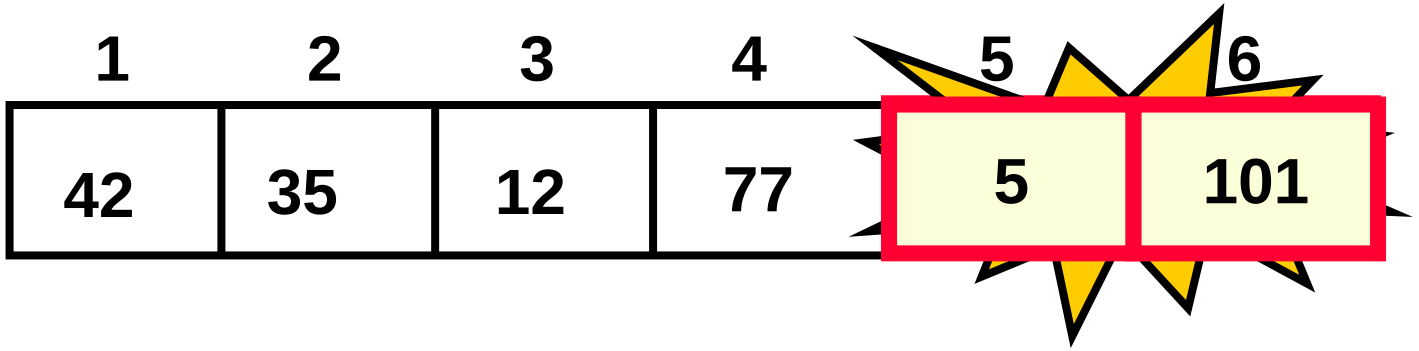
- Traverse a collection of elements
  - Move from the front to the end
  - “Bubble” the largest value to the end using pair-wise comparisons and swapping



Bubble Sort ...

"Bubbling Up" the Largest Element

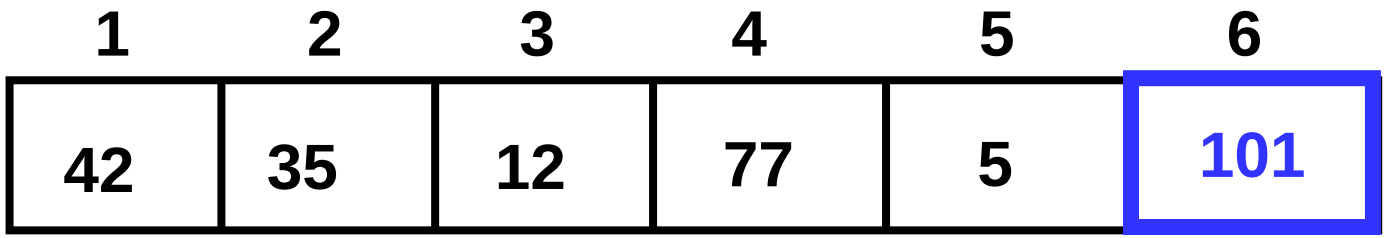
- Traverse a collection of elements
  - Move from the front to the end
  - “Bubble” the largest value to the end using pair-wise comparisons and swapping



Bubble Sort ...

"Bubbling Up" the Largest Element

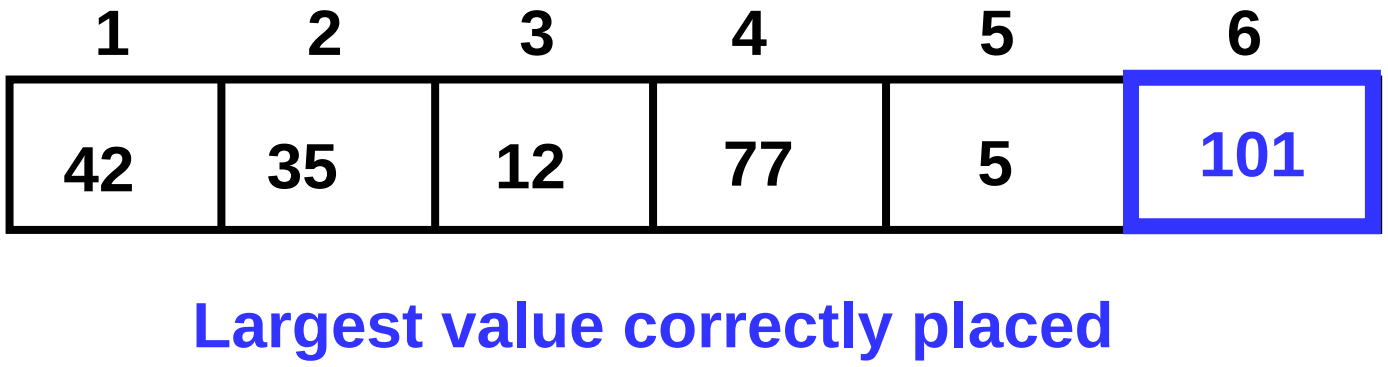
- Traverse a collection of elements
  - Move from the front to the end
  - “Bubble” the largest value to the end using pair-wise comparisons and swapping



Largest value correctly placed

Bubble Sort ...

- Notice that only the largest value is correctly placed
- All other values are still out of order
- So we need to repeat this process



Largest value correctly placed

Bubble Sort …

Repeat "Bubble Up" How Many Times?

- If we have N elements…
- And if each time we bubble an element, we place it in its correct location…
- Then we repeat the "bubble up" process N – 1 times.
- This guarantees we'll correctly place all N elements.

Bubble Sort ...

"Bubbling" All the Elements

N - 1

1	2	3	4	5	6
42	35	12	77	5	101

1	2	3	4	5	6
35	12	42	5	77	101

1	2	3	4	5	6
12	35	5	42	77	101

1	2	3	4	5	6
12	5	35	42	77	101

1	2	3	4	5	6
5	12	35	42	77	101

Bubble Sort ...

Reducing the Number of Comparisons

1	2	3	4	5	6
77	42	35	12	101	5

1	2	3	4	5	6
42	35	12	77	5	101

1	2	3	4	5	6
35	12	42	5	77	101

1	2	3	4	5	6
12	35	5	42	77	101

1	2	3	4	5	6
12	5	35	42	77	101

Bubble Sort: Algorithm

```
void bubble_sort (int a [ ], int n)
{
int i,j, temp;
for (i=0;i<n-1;i++)    //Making passes through array
   {
    for (j=0;j<n-1-i;j++)
       if (a[j]>a[j+1])  //If adjacent elements are in wrong order
         {                    // Swap them
           temp=a[j+1];
           a[j+1]=a[j];
           a[j]=temp;
         }
    }
}
```

Bubble Sort: Already Sorted Collections?

- What if the collection was already sorted?
- What if only a few elements were out of place and after a couple of “bubble ups,” the collection was sorted?

- We want to be able to detect this and “stop early”!

1	2	3	4	5	6
5	12	35	42	77	101

Bubble Sort: Already Sorted Collections?...

Using a Boolean "Flag"

- We can use a boolean variable to determine if any swapping occurred during the "bubble up."
- If no swapping occurred, then we know that the collection is already sorted!
- This boolean "flag" needs to be reset after each "bubble up."

Bubble Sort: Already Sorted Collections?...

Example

N	8
to_do	7
index	

did_swap	true

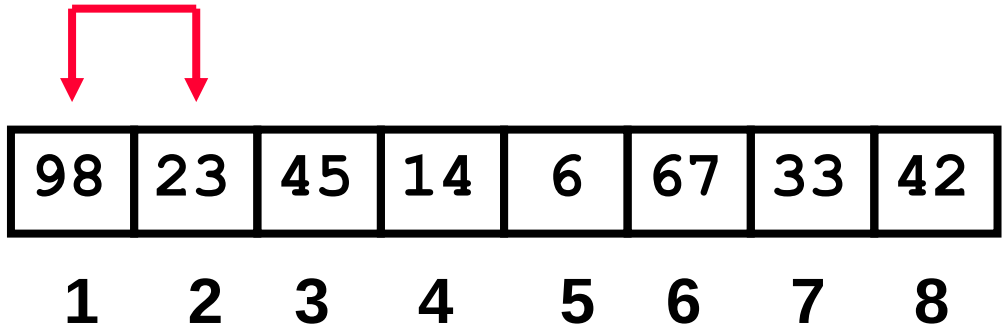
98	23	45	14	6	67	33	42
1	2	3	4	5	6	7	8

Bubble Sort: Already Sorted Collections?...

Example

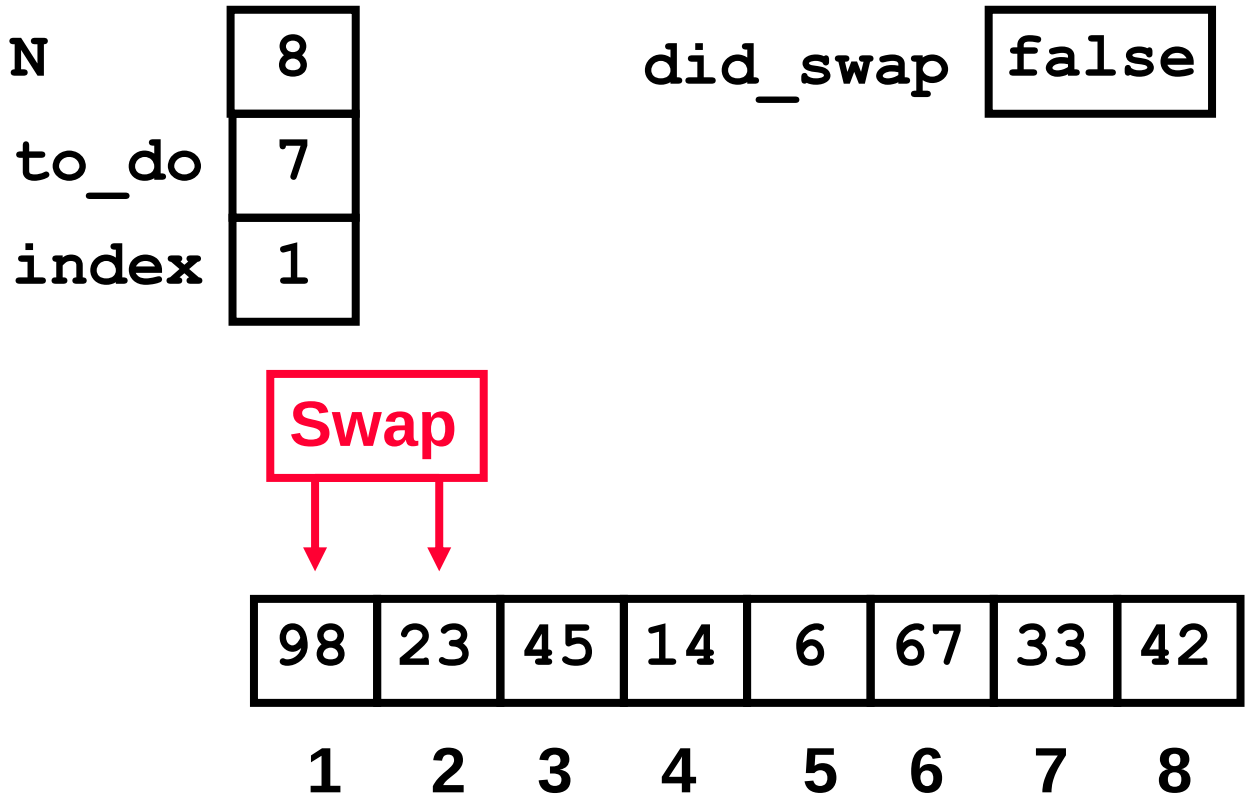
N	8
to_do	7
index	1

did_swap	false



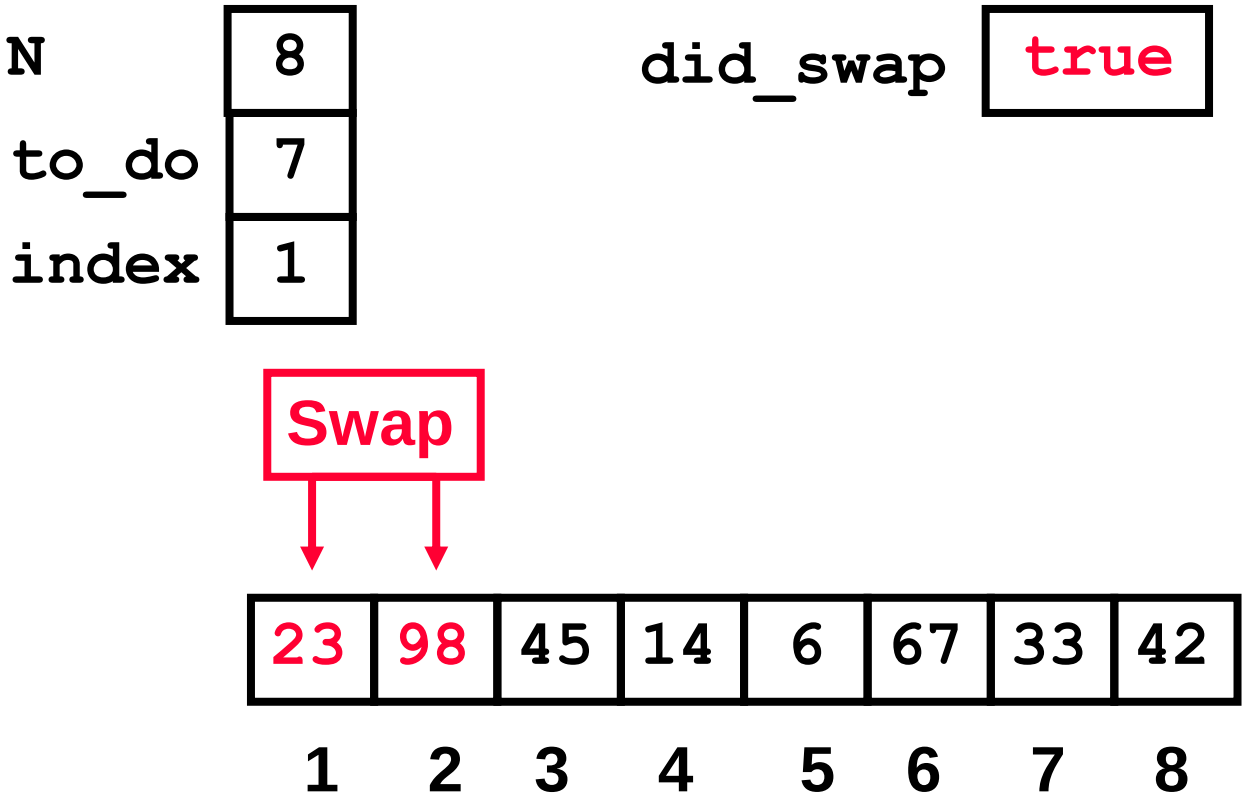
Bubble Sort: Already Sorted Collections?...

Example



Bubble Sort: Already Sorted Collections?...

Example

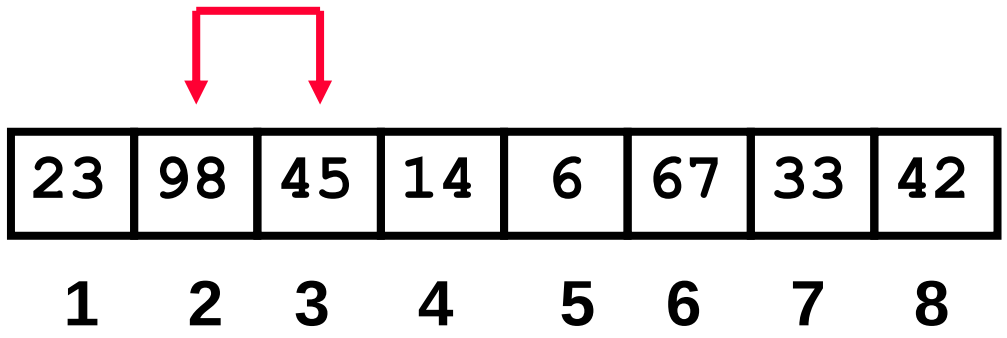


Bubble Sort: Already Sorted Collections?...

Example

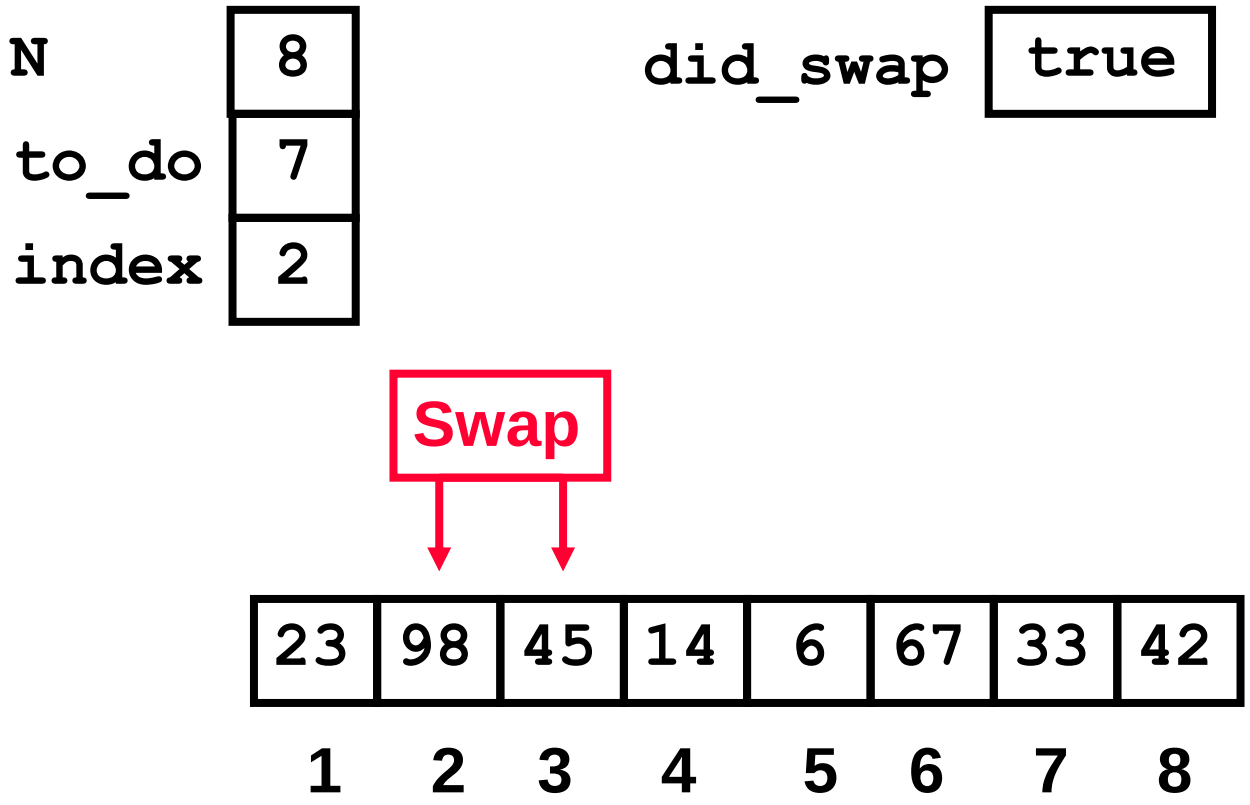
N	8
to_do	7
index	2

did_swap	true



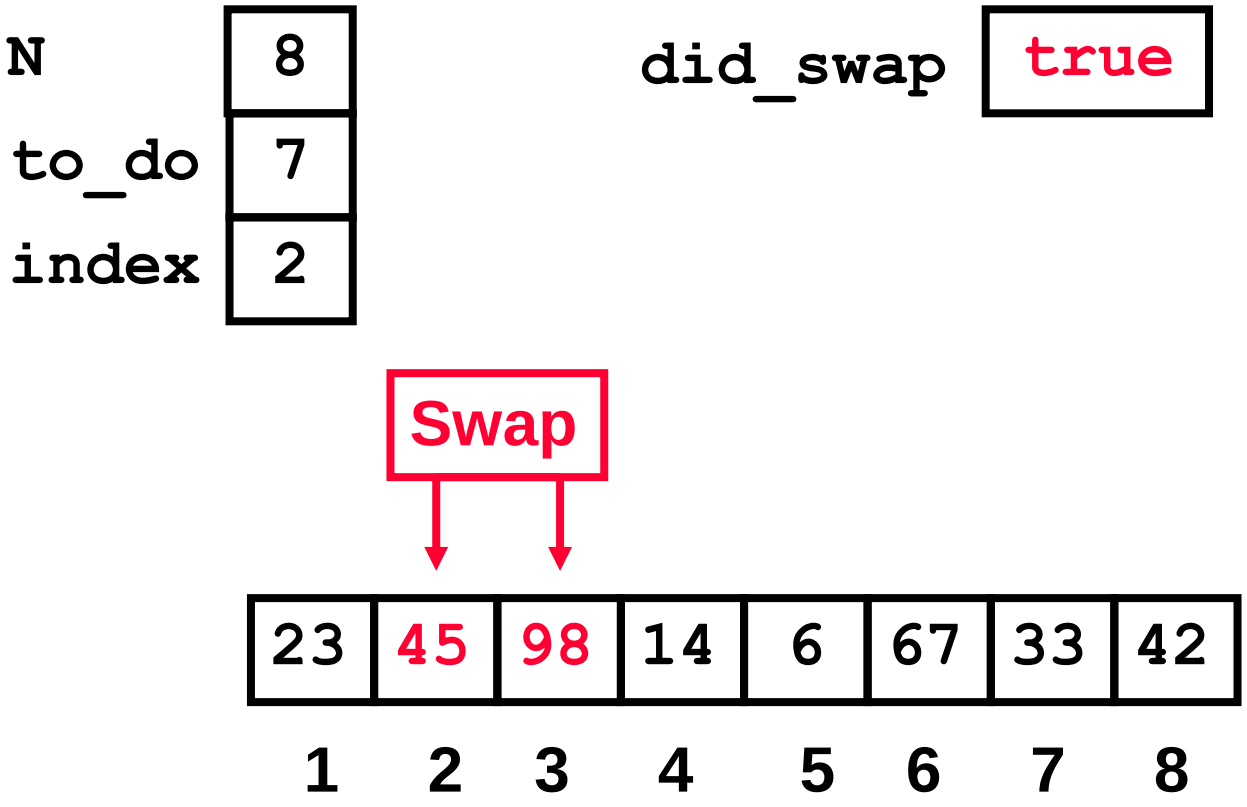
Bubble Sort: Already Sorted Collections?...

Example



Bubble Sort: Already Sorted Collections?...

Example

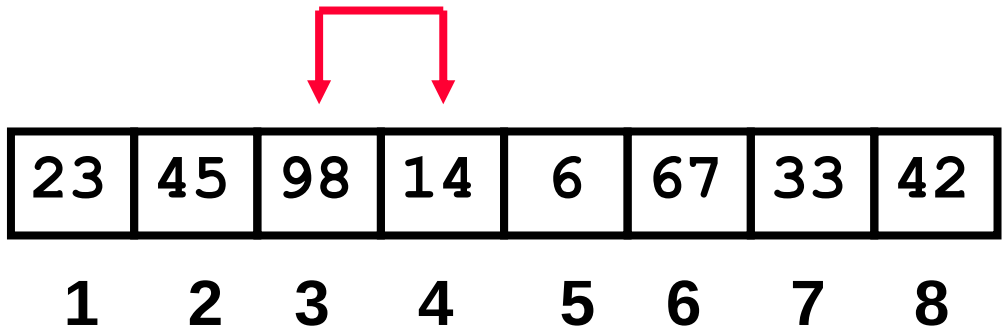


Bubble Sort: Already Sorted Collections?...

Example

N	8
to_do	7
index	3

did_swap	true

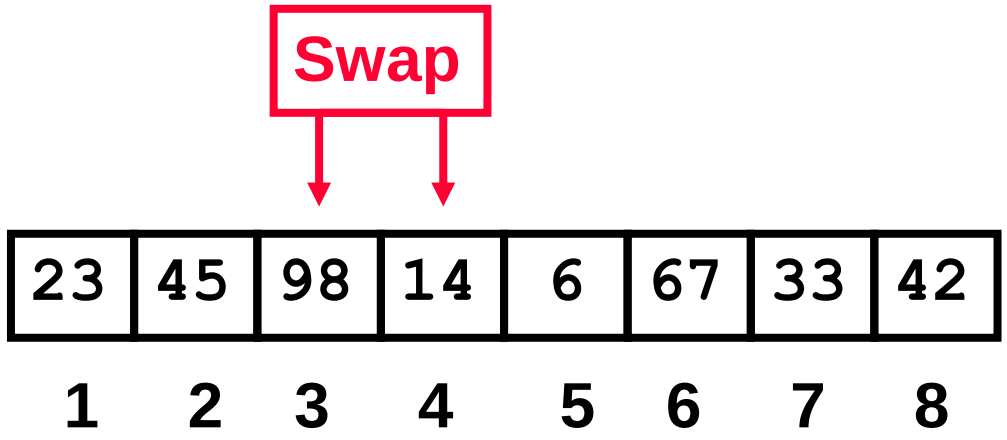


Bubble Sort: Already Sorted Collections?...

Example

N	8
to_do	7
index	3

did_swap	true

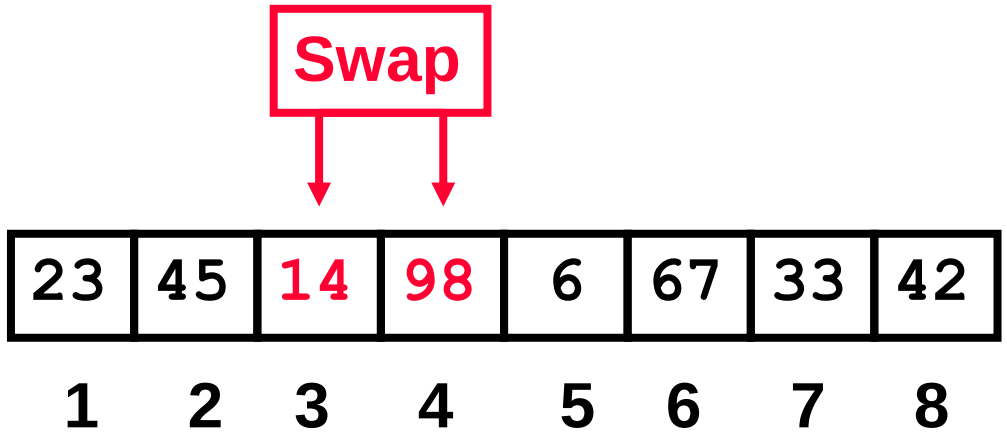


Bubble Sort: Already Sorted Collections?...

Example

N	8
to_do	7
index	3

did_swap	true

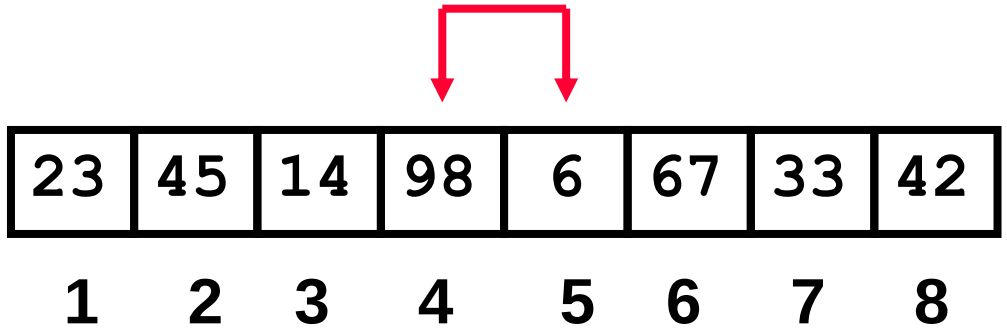


Bubble Sort: Already Sorted Collections?...

Example

N	8
to_do	7
index	4

did_swap	true

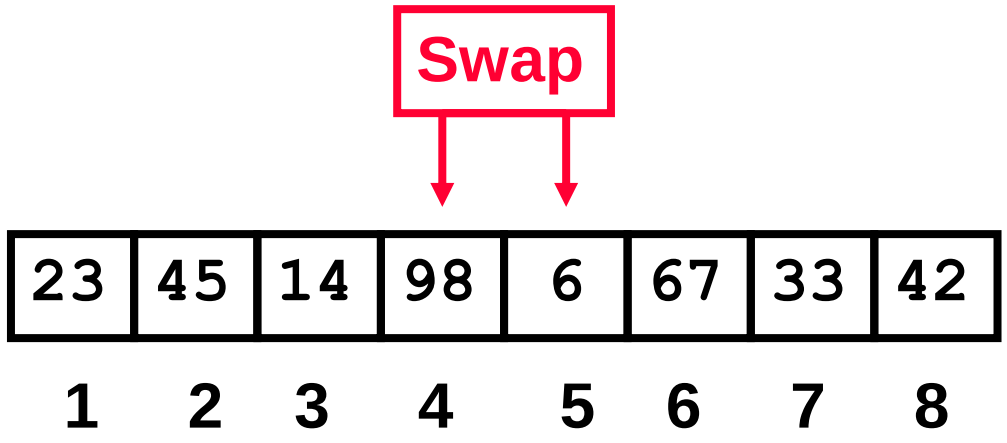


Bubble Sort: Already Sorted Collections?...

Example

N	8
to_do	7
index	4

did_swap	true

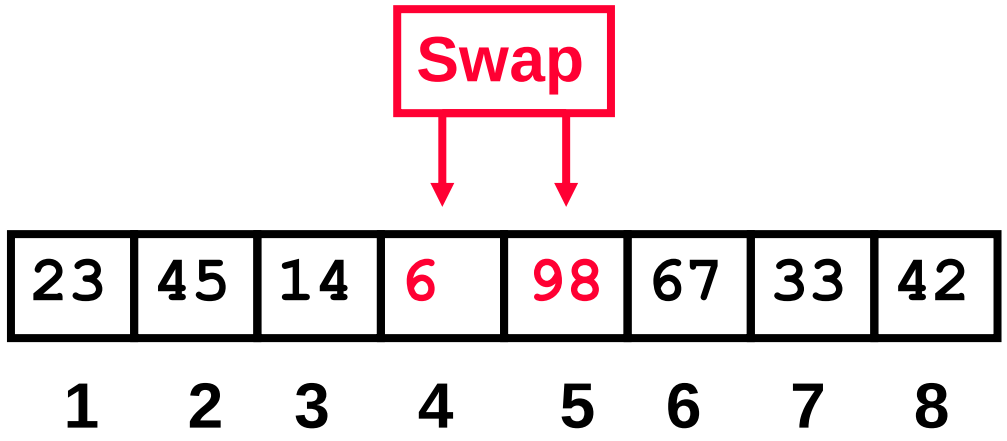


Bubble Sort: Already Sorted Collections?...

Example

N	8
to_do	7
index	4

did_swap	true

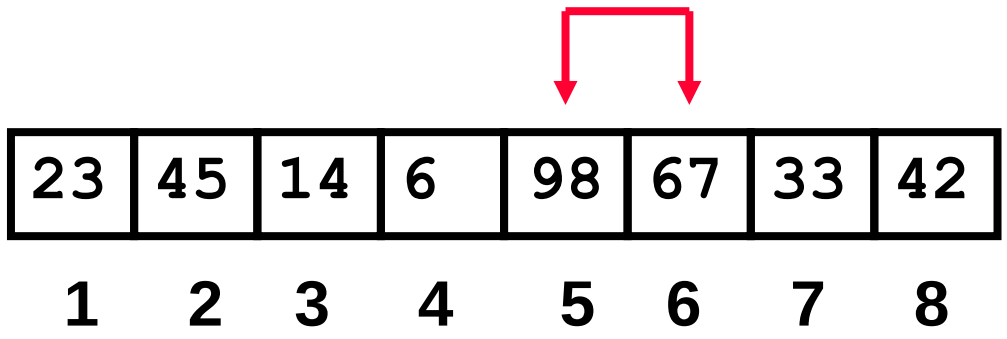


Bubble Sort: Already Sorted Collections?...

Example

N	8
to_do	7
index	5

did_swap	true

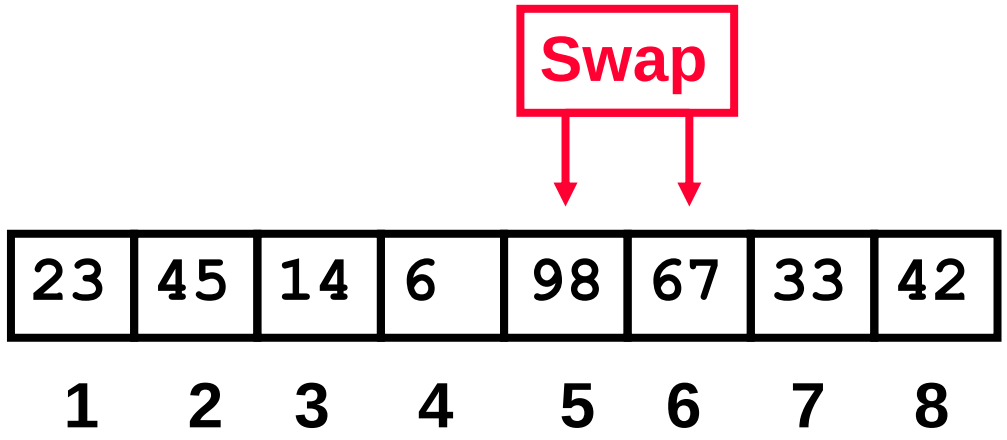


Bubble Sort: Already Sorted Collections?...

Example

N	8
to_do	7
index	5

did_swap	true

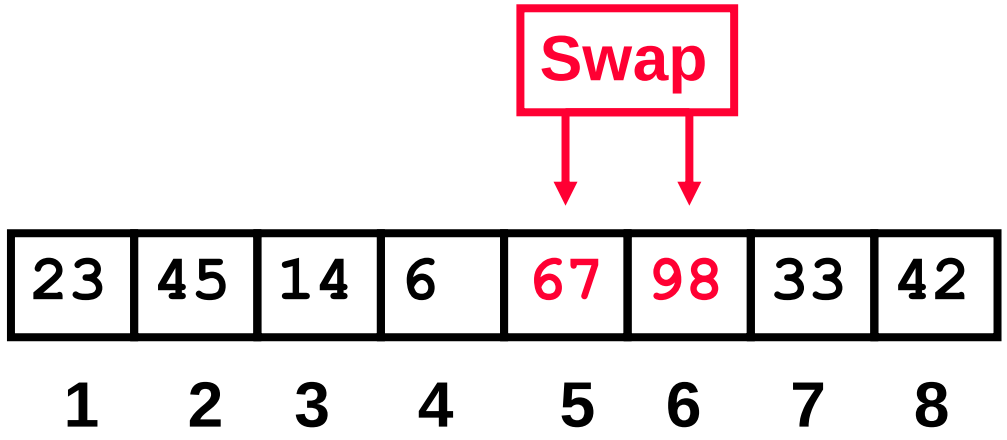


Bubble Sort: Already Sorted Collections?...

Example

N	8
to_do	7
index	5

did_swap	true

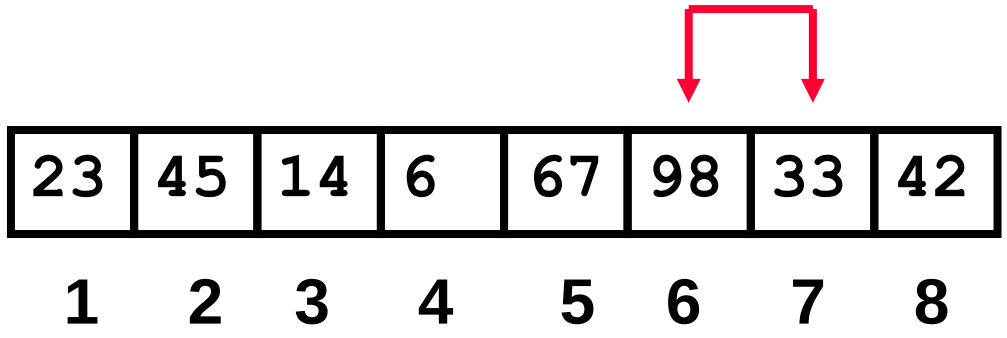


Bubble Sort: Already Sorted Collections?...

Example

N	8
to_do	7
index	6

did_swap	true

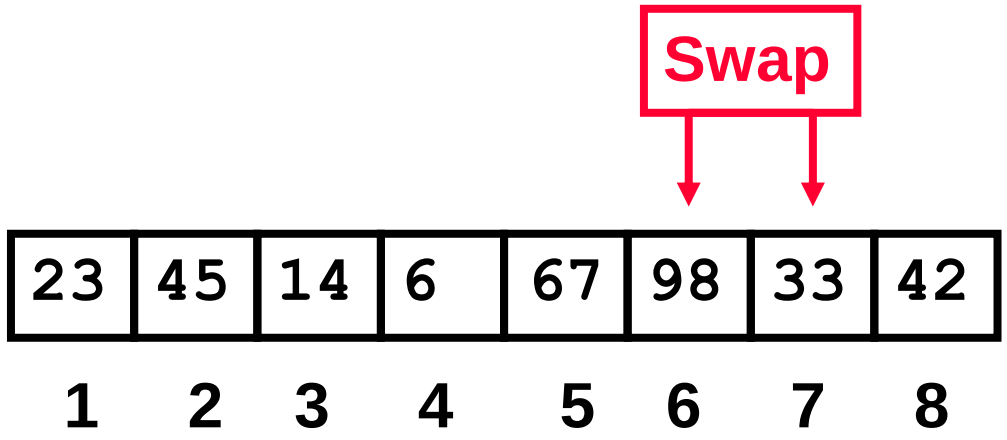


Bubble Sort: Already Sorted Collections?...

Example

N	8
to_do	7
index	6

did_swap	true

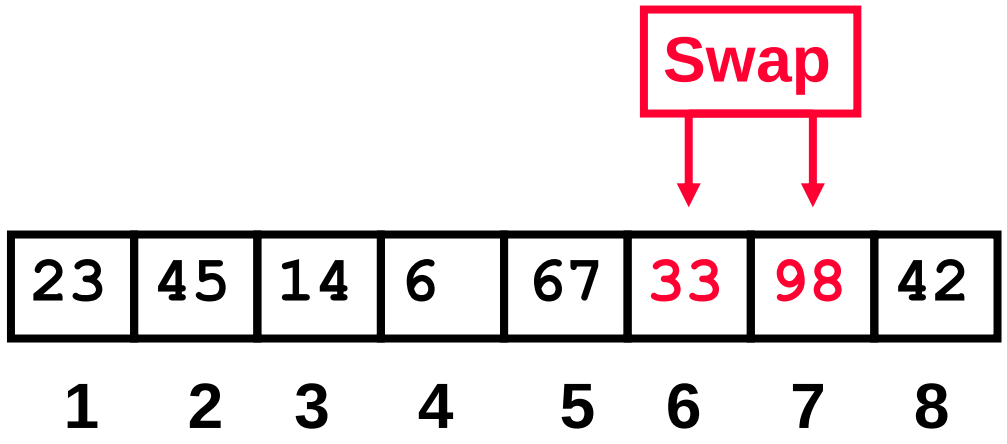


Bubble Sort: Already Sorted Collections?...

Example

N	8
to_do	7
index	6

did_swap	true

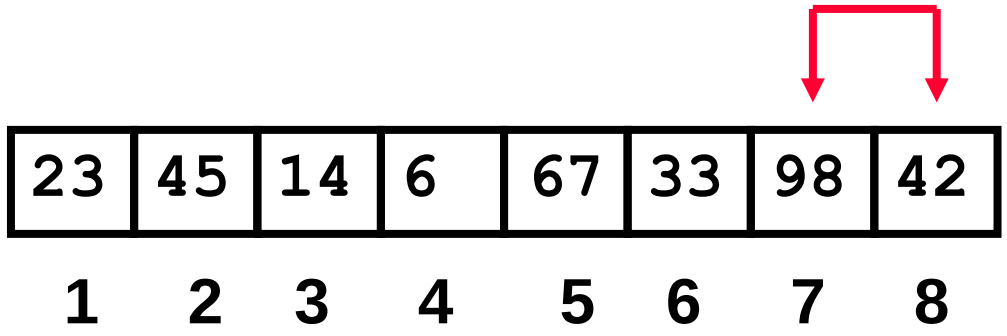


Bubble Sort: Already Sorted Collections?...

Example

N	8
to_do	7
index	7

did_swap	true

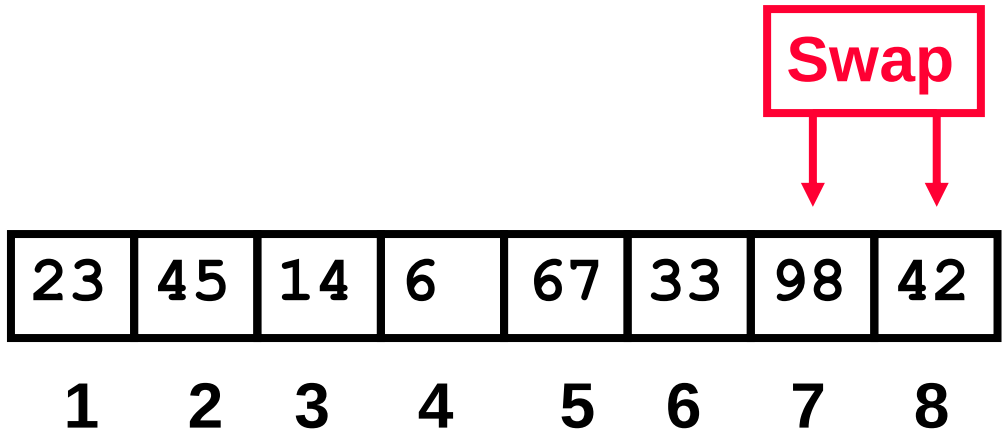


Bubble Sort: Already Sorted Collections?...

Example

N	8
to_do	7
index	7

did_swap	true

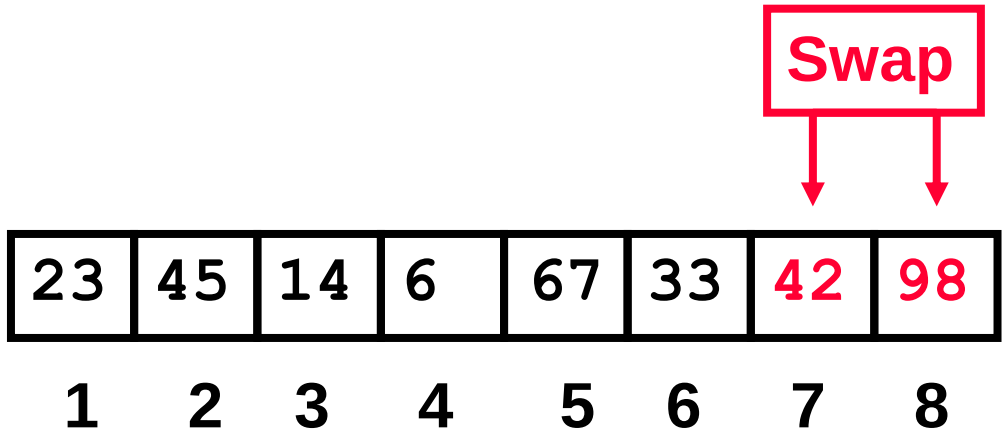


Bubble Sort: Already Sorted Collections?...

Example

N	8
to_do	7
index	7

did_swap	true



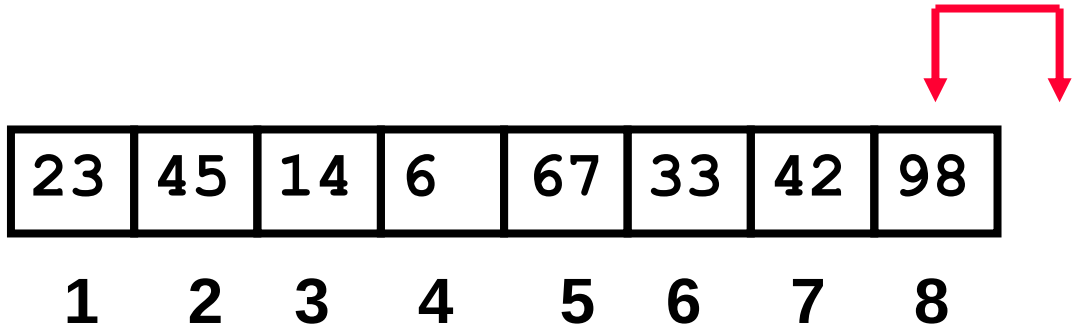
Bubble Sort: Already Sorted Collections?...

After First Pass of Outer Loop

N	8
to_do	7
index	8

did_swap	true

Finished first “Bubble Up”

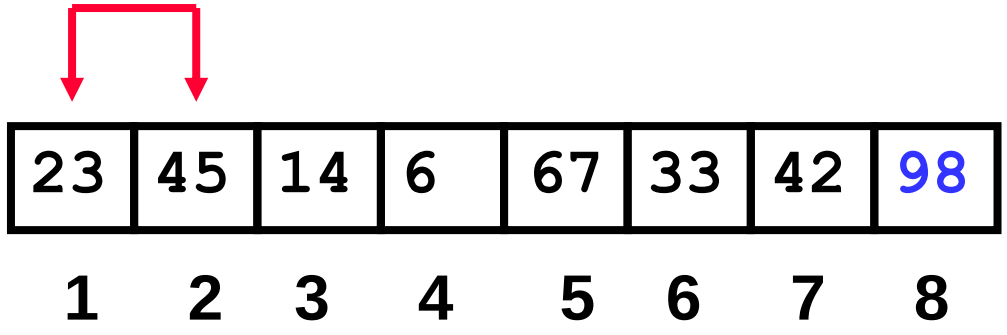


Bubble Sort: Already Sorted Collections?...

The Second "Bubble Up"

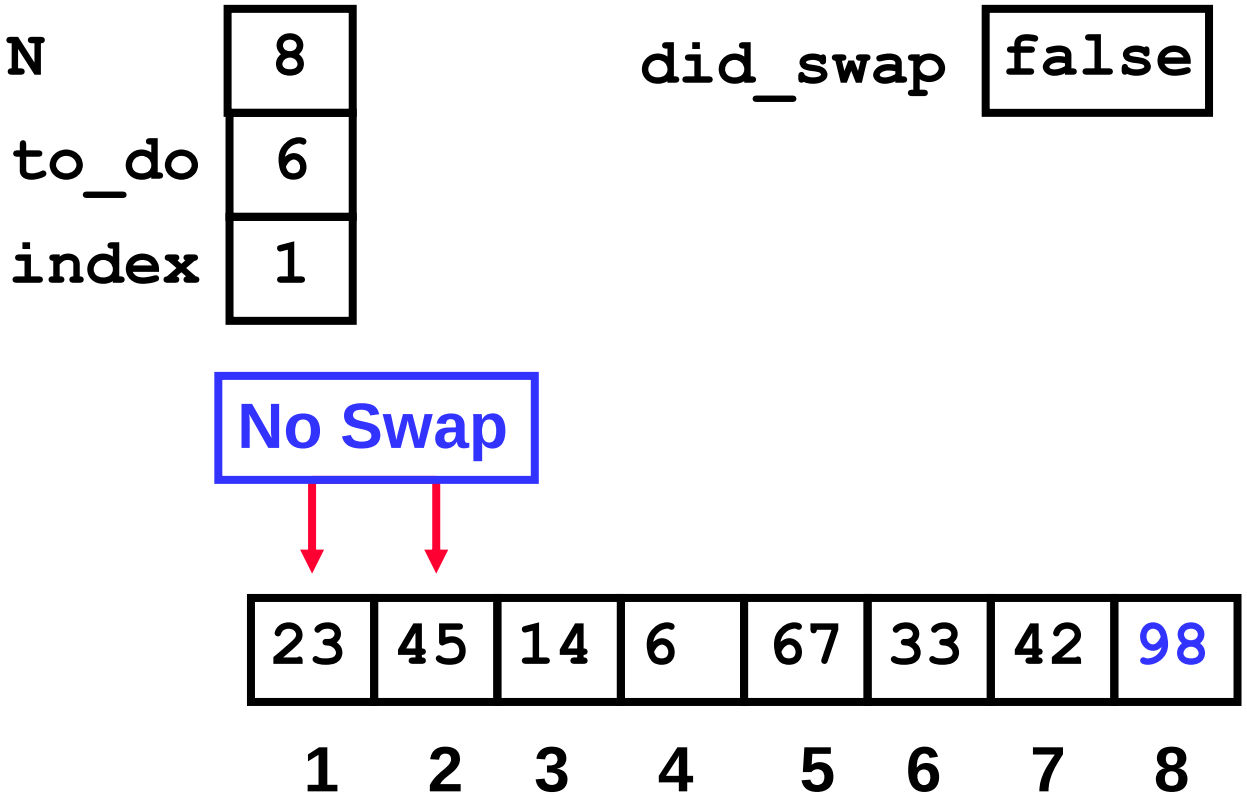
N	8
to_do	6
index	1

did_swap | false



Bubble Sort: Already Sorted Collections?...

The Second "Bubble Up"

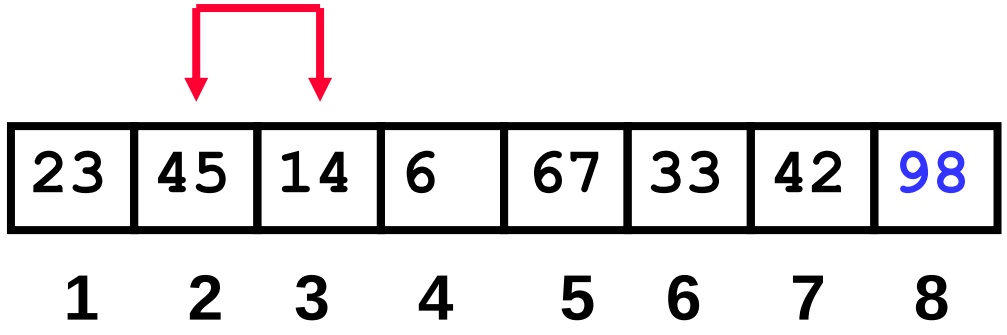


Bubble Sort: Already Sorted Collections?...

The Second "Bubble Up"

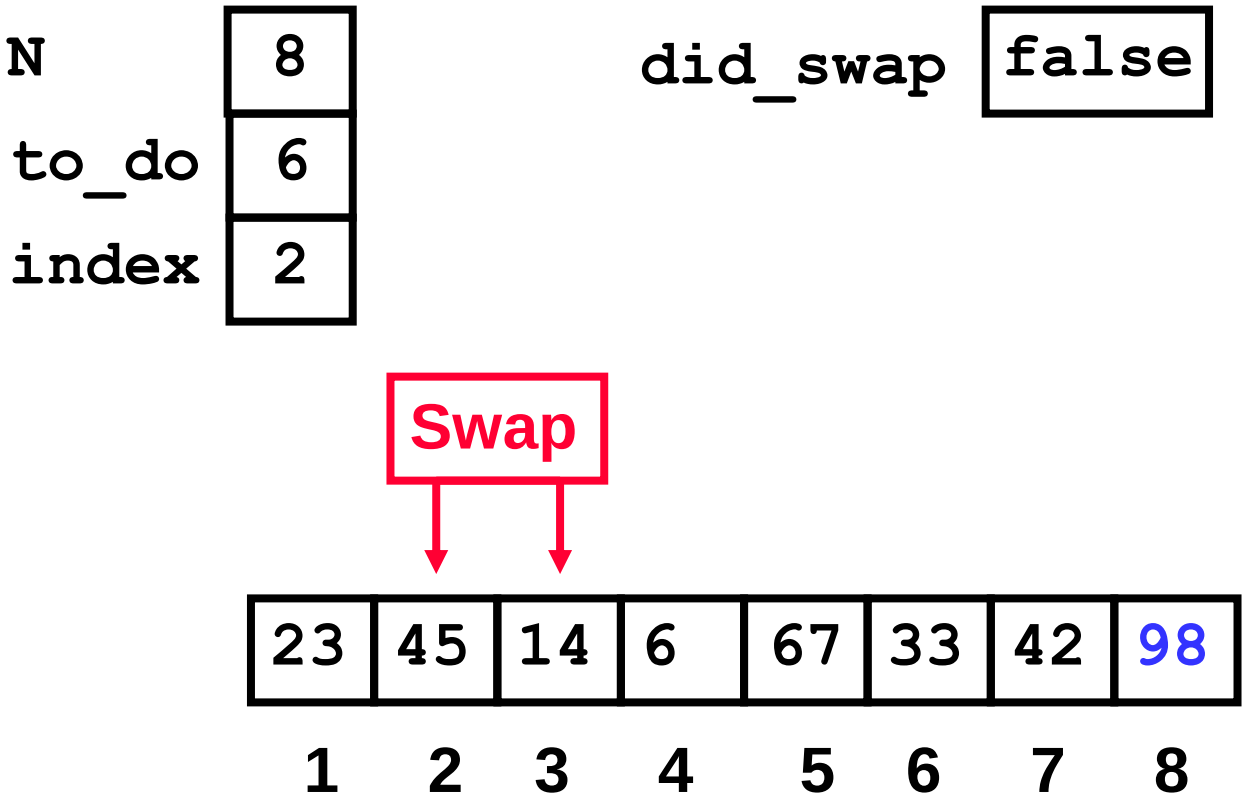
N	8
to_do	6
index	2

did_swap	false



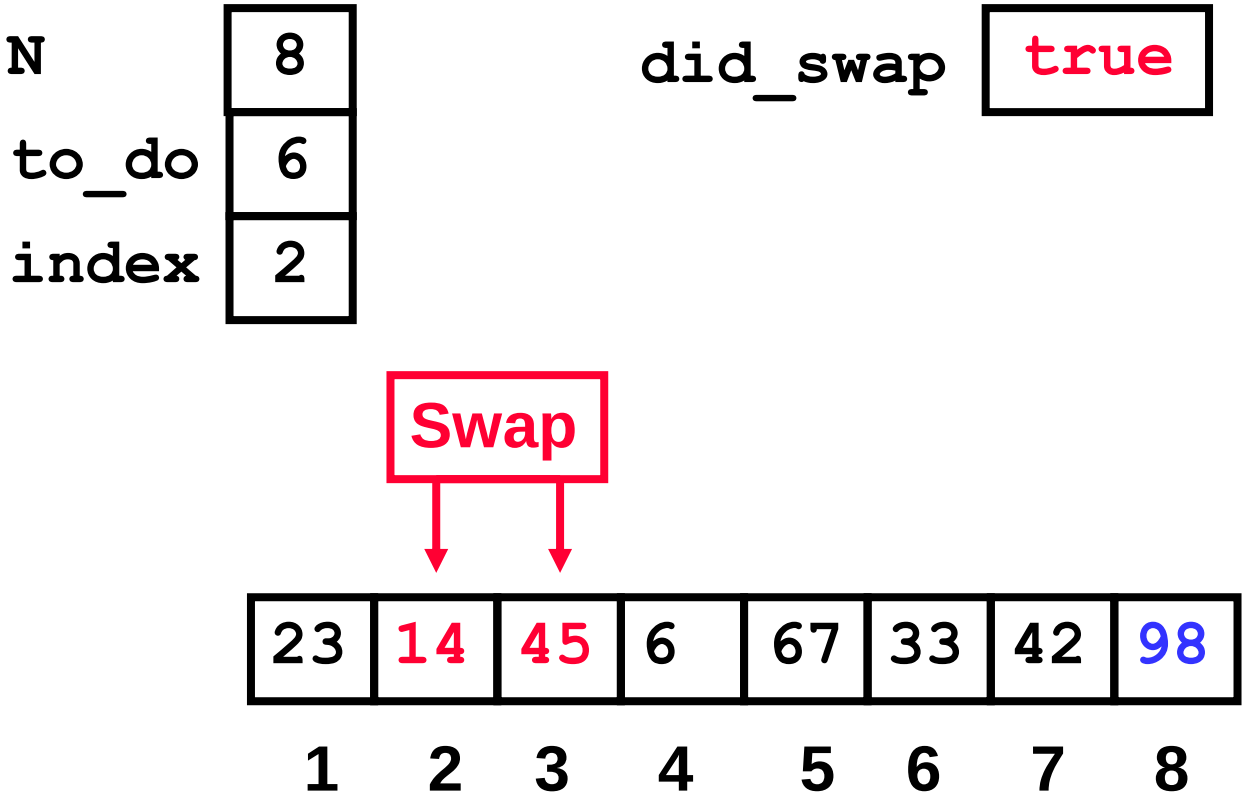
Bubble Sort: Already Sorted Collections?...

The Second "Bubble Up"



Bubble Sort: Already Sorted Collections?...

The Second "Bubble Up"

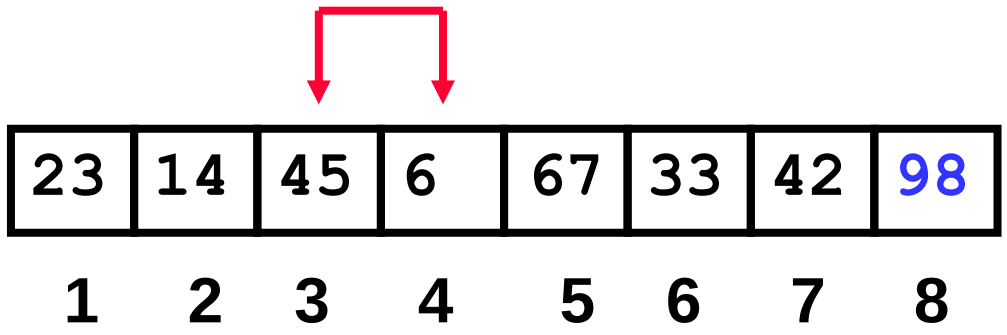


Bubble Sort: Already Sorted Collections?...

The Second "Bubble Up"

N	8
to_do	6
index	3

did_swap	true

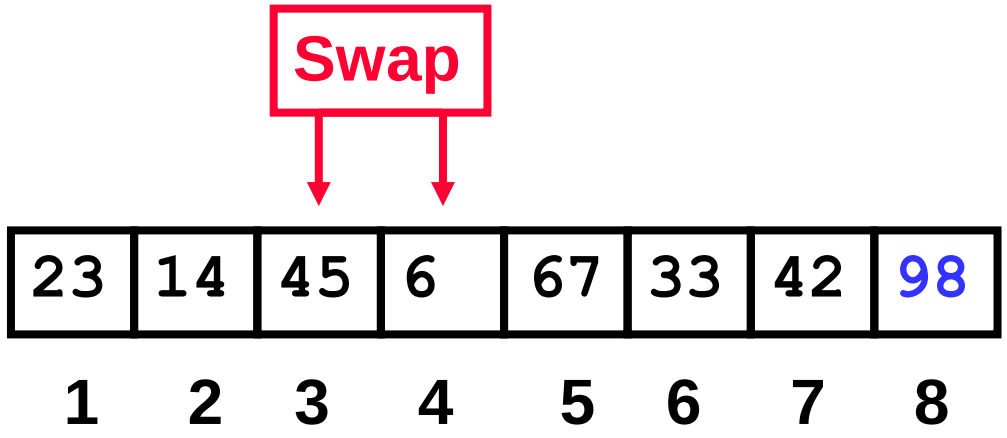


Bubble Sort: Already Sorted Collections?...

The Second "Bubble Up"

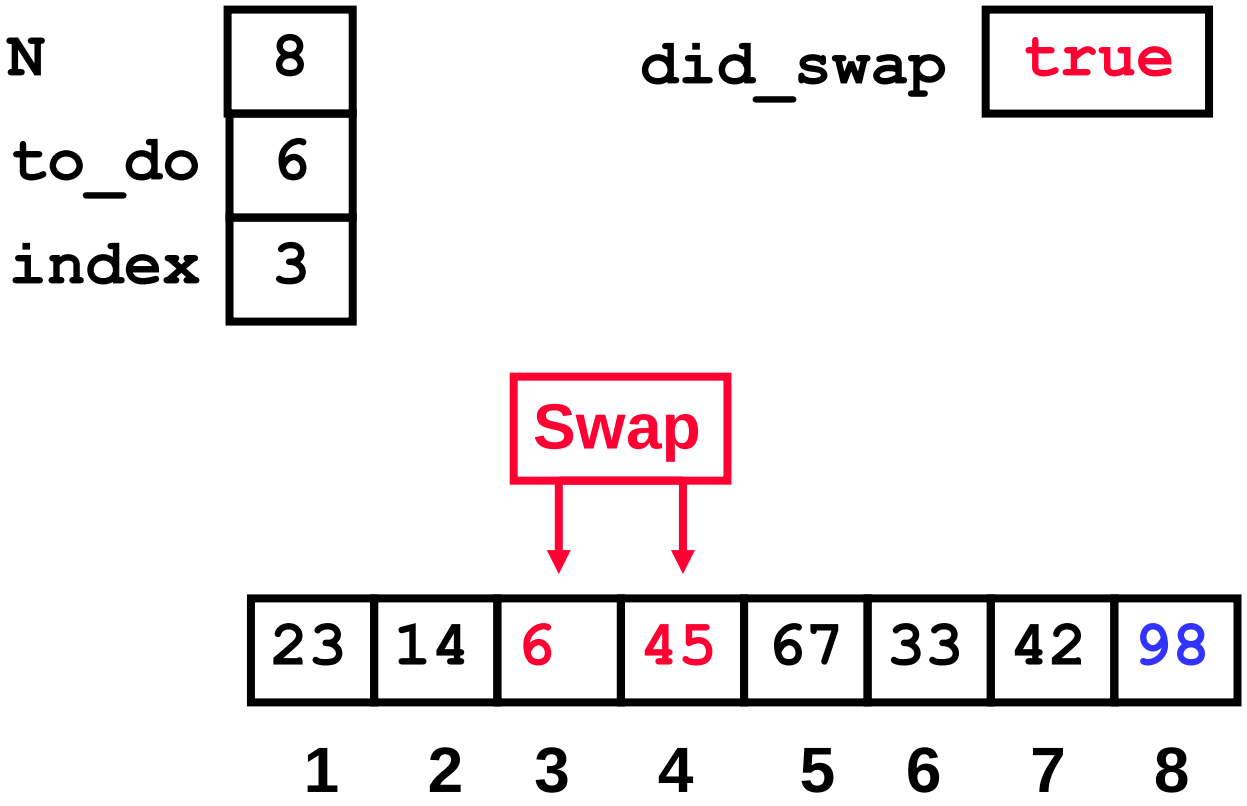
N	8
to_do	6
index	3

did_swap	true



Bubble Sort: Already Sorted Collections?...

The Second "Bubble Up"

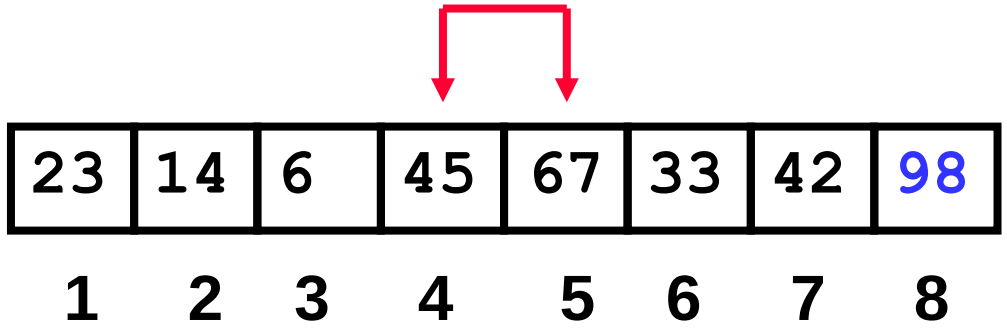


Bubble Sort: Already Sorted Collections?...

The Second "Bubble Up"

N	8
to_do	6
index	4

did_swap	true

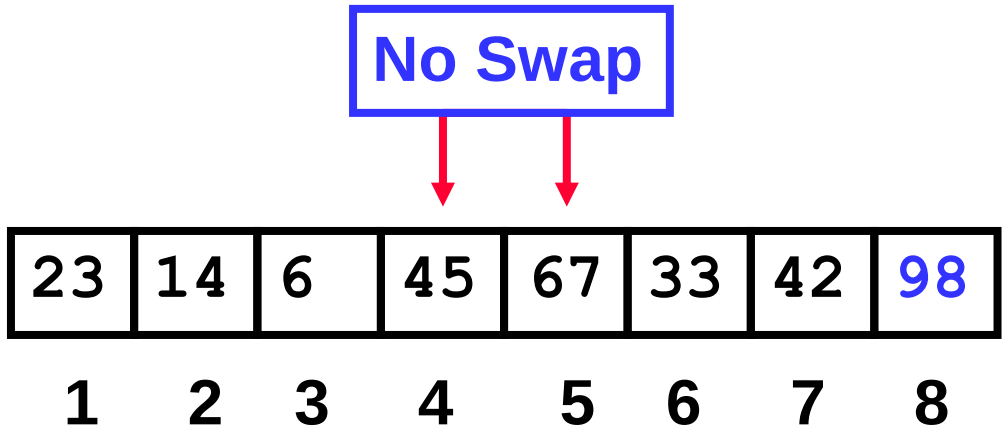


Bubble Sort: Already Sorted Collections?...

The Second "Bubble Up"

N	8
to_do	6
index	4

did_swap	true

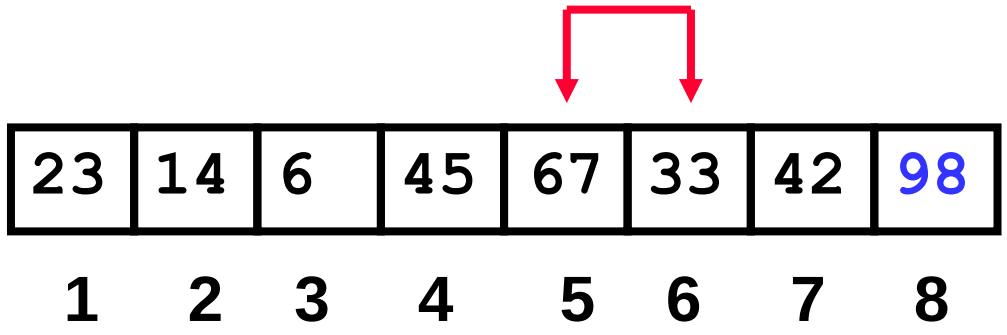


Bubble Sort: Already Sorted Collections?...

The Second "Bubble Up"

N	8
to_do	6
index	5

did_swap	true

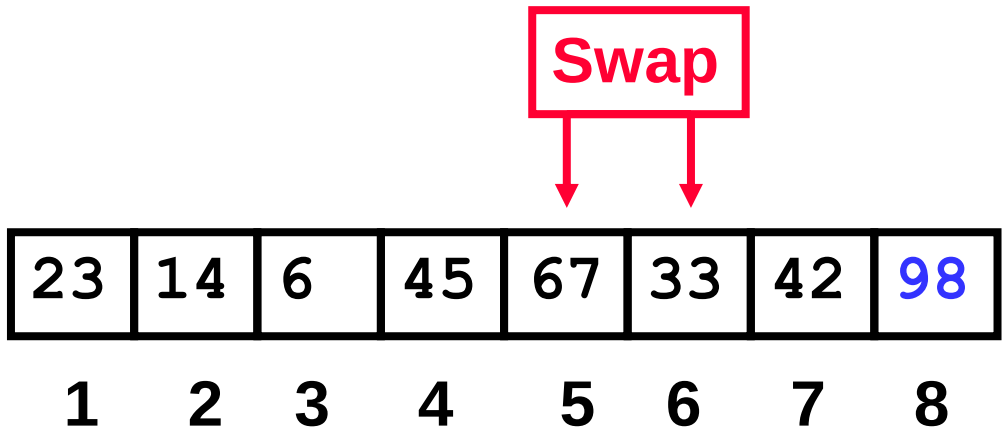


Bubble Sort: Already Sorted Collections?...

The Second "Bubble Up"

N	8
to_do	6
index	5

did_swap	true

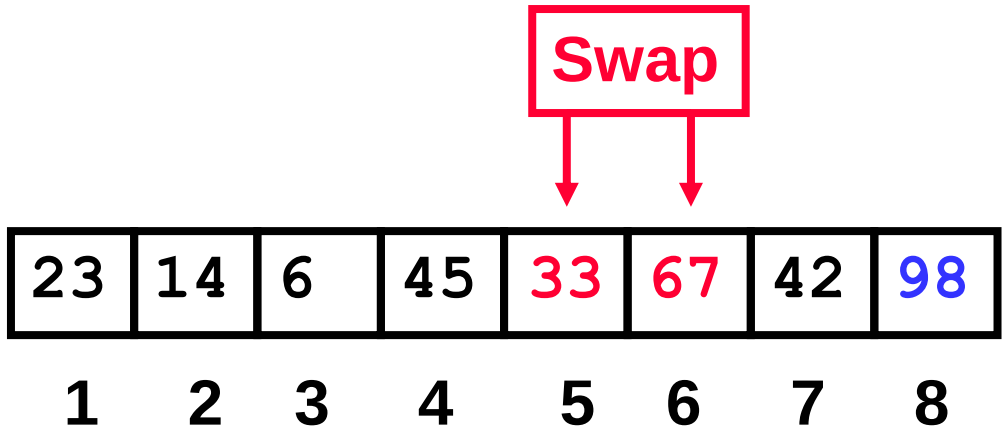


Bubble Sort: Already Sorted Collections?...

The Second "Bubble Up"

N	8
to_do	6
index	5

did_swap | true

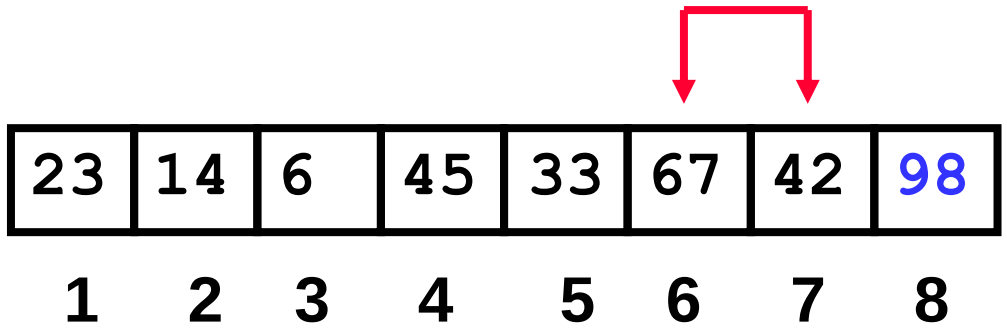


Bubble Sort: Already Sorted Collections?...

The Second "Bubble Up"

N	8
to_do	6
index	6

did_swap	true

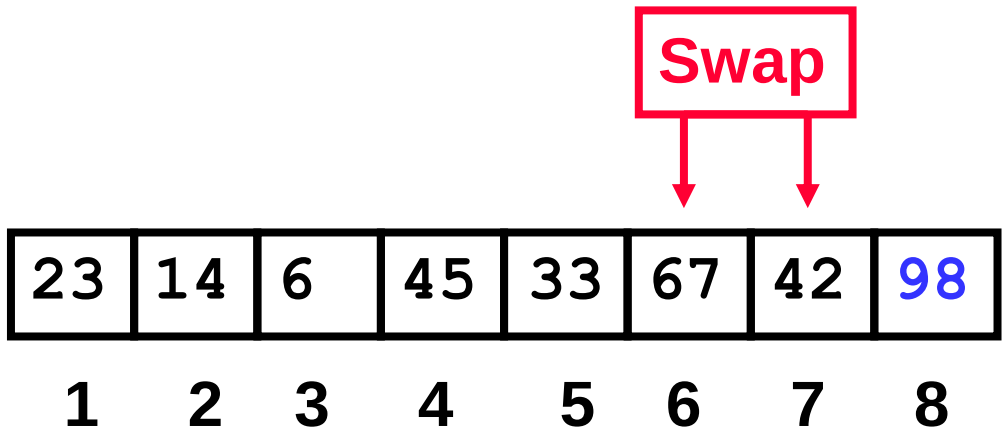


Bubble Sort: Already Sorted Collections?...

The Second "Bubble Up"

N	8
to_do	6
index	6

did_swap	true

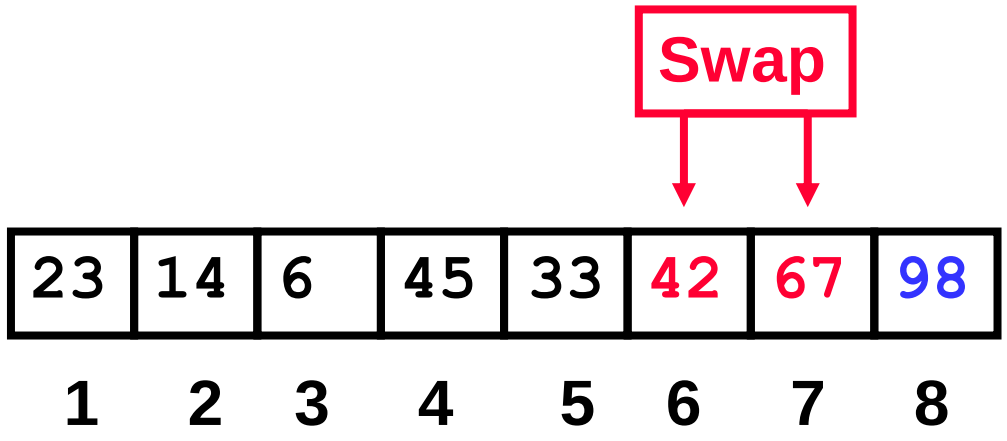


Bubble Sort: Already Sorted Collections?...

The Second "Bubble Up"

N	8
to_do	6
index	6

did_swap	true



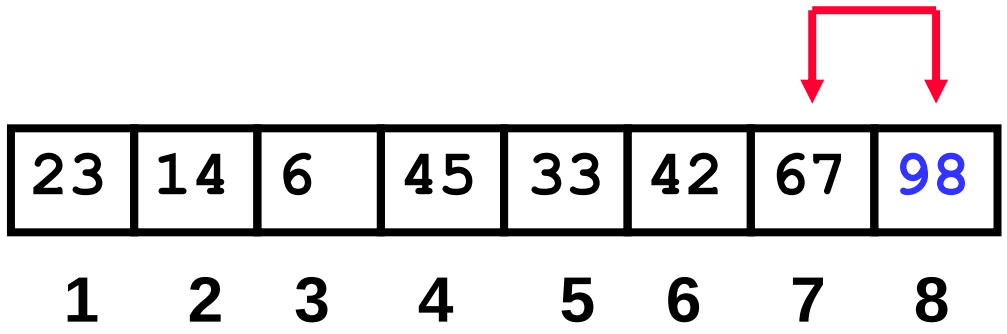
Bubble Sort: Already Sorted Collections?...

After Second Pass of Outer Loop

N	8
to_do	6
index	7

did_swap	true

Finished second "Bubble Up"

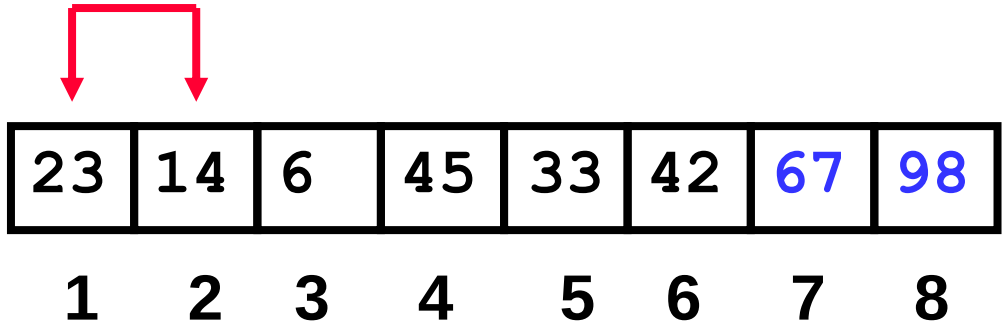


Bubble Sort: Already Sorted Collections?...

The Third "Bubble Up"

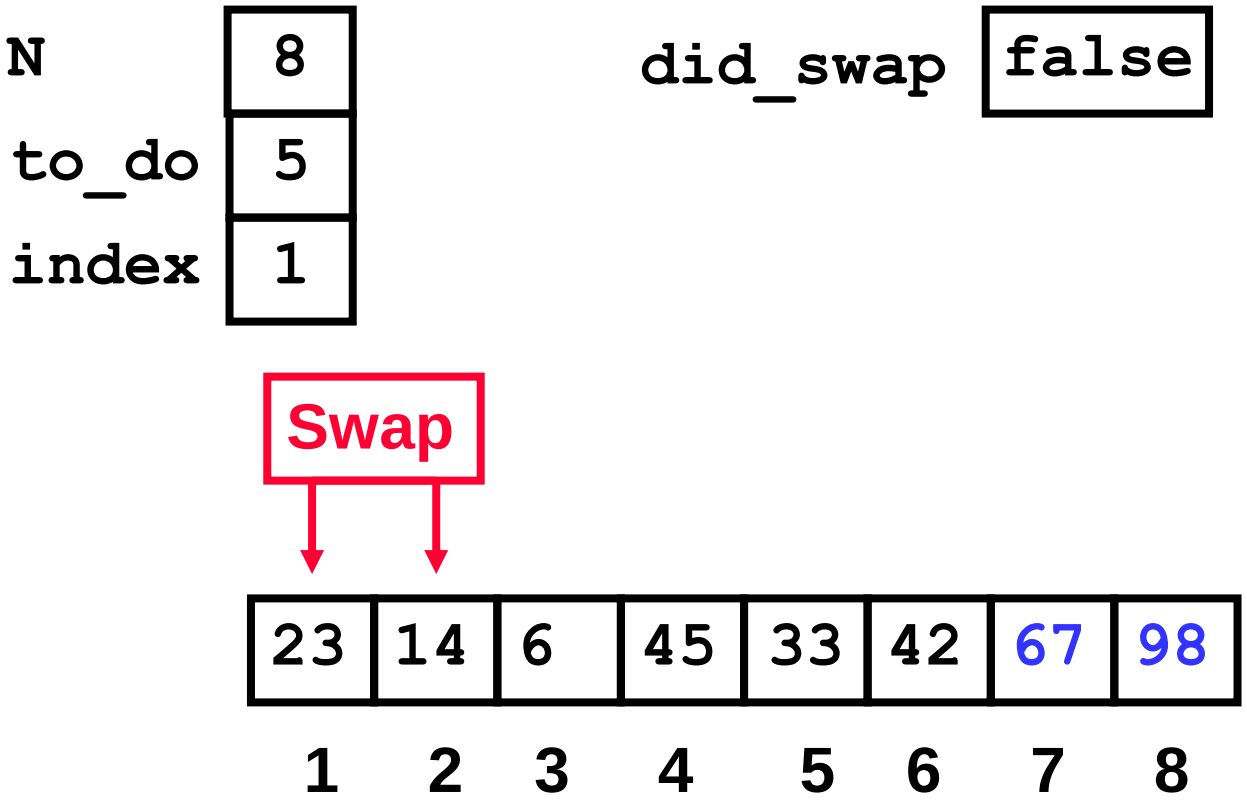
N	8
to_do	5
index	1

did_swap | false



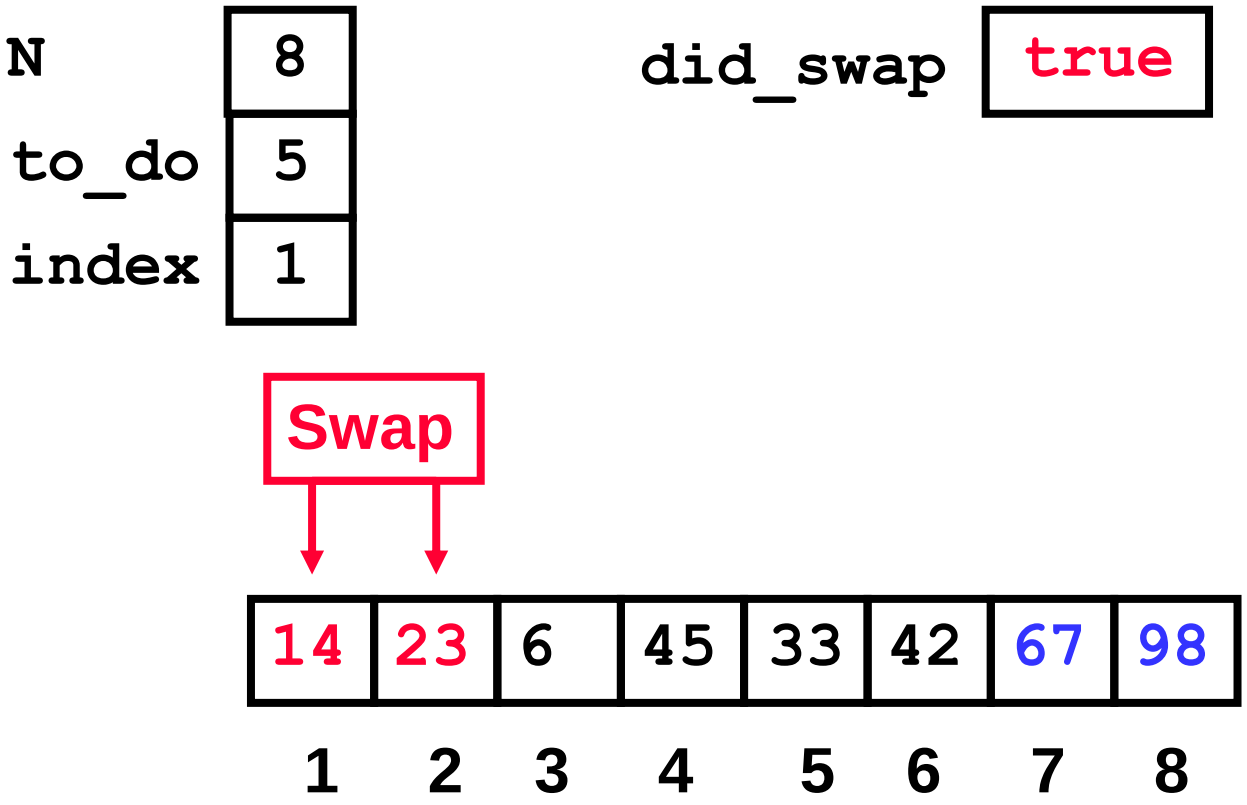
Bubble Sort: Already Sorted Collections?...

The Third "Bubble Up"



Bubble Sort: Already Sorted Collections?...

The Third "Bubble Up"

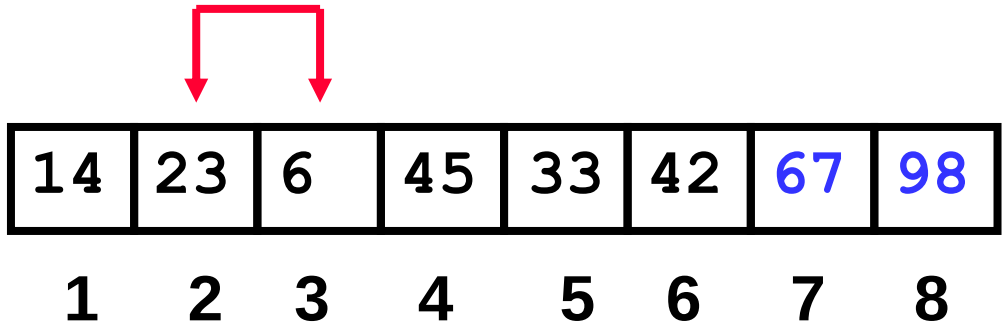


Bubble Sort: Already Sorted Collections?...

The Third "Bubble Up"

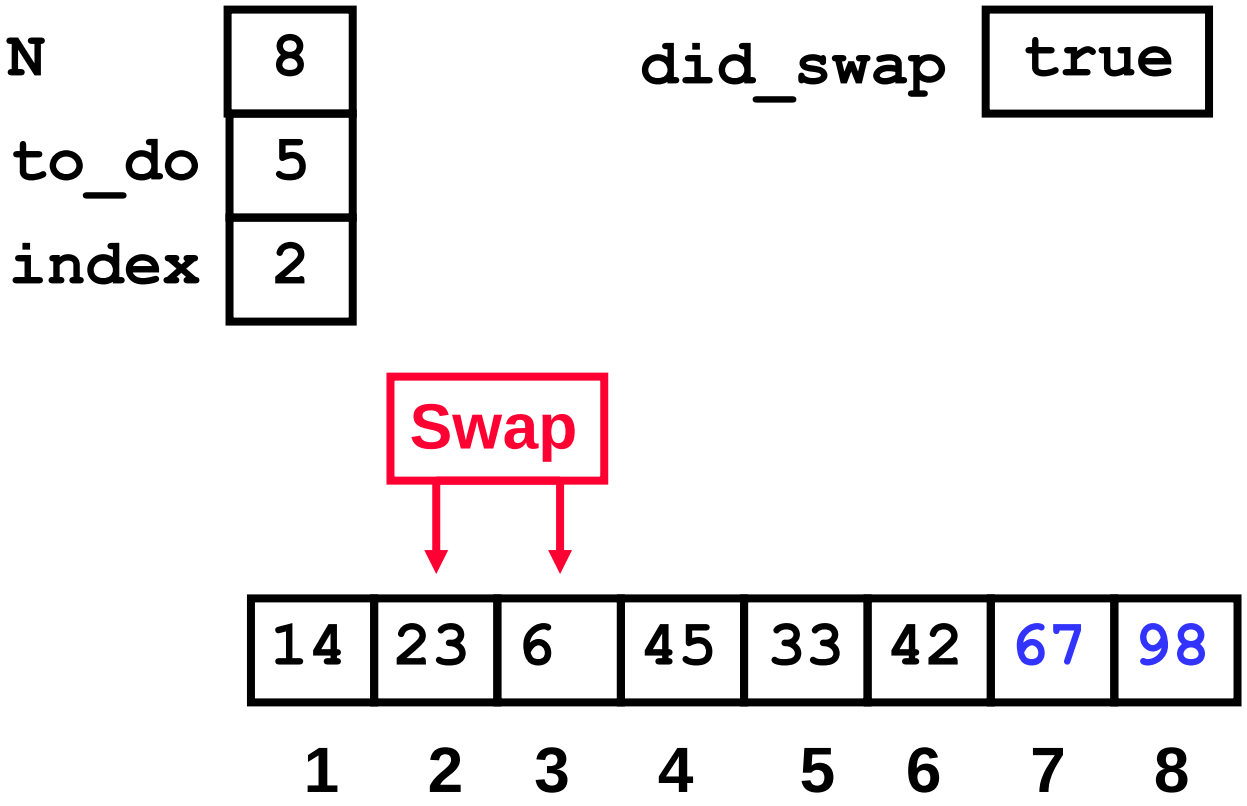
N	8
to_do	5
index	2

did_swap	true



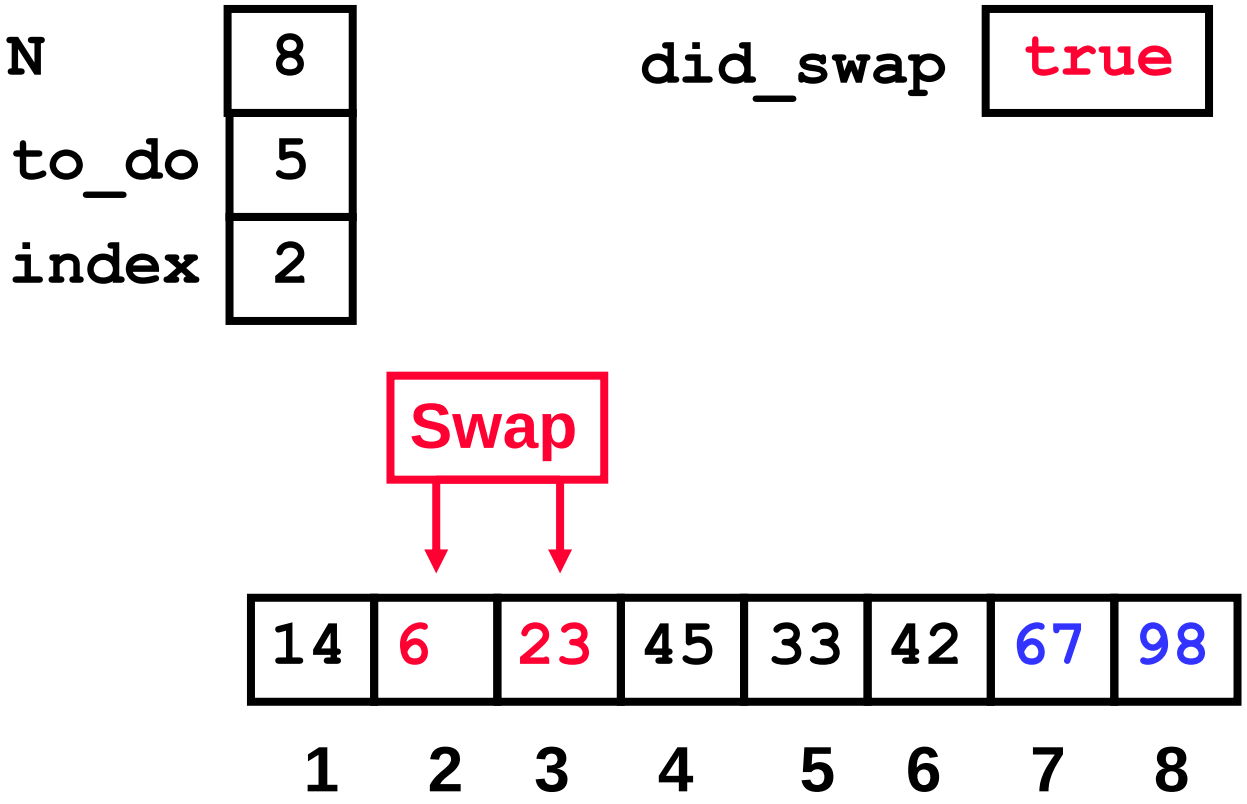
Bubble Sort: Already Sorted Collections?...

The Third "Bubble Up"



Bubble Sort: Already Sorted Collections?...

The Third "Bubble Up"

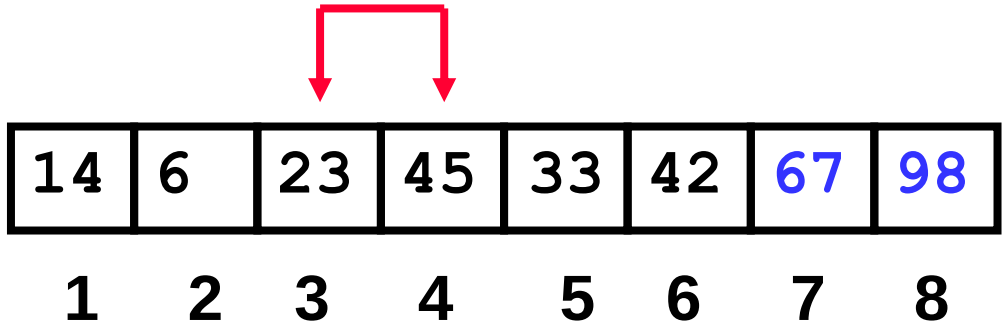


Bubble Sort: Already Sorted Collections?...

The Third "Bubble Up"

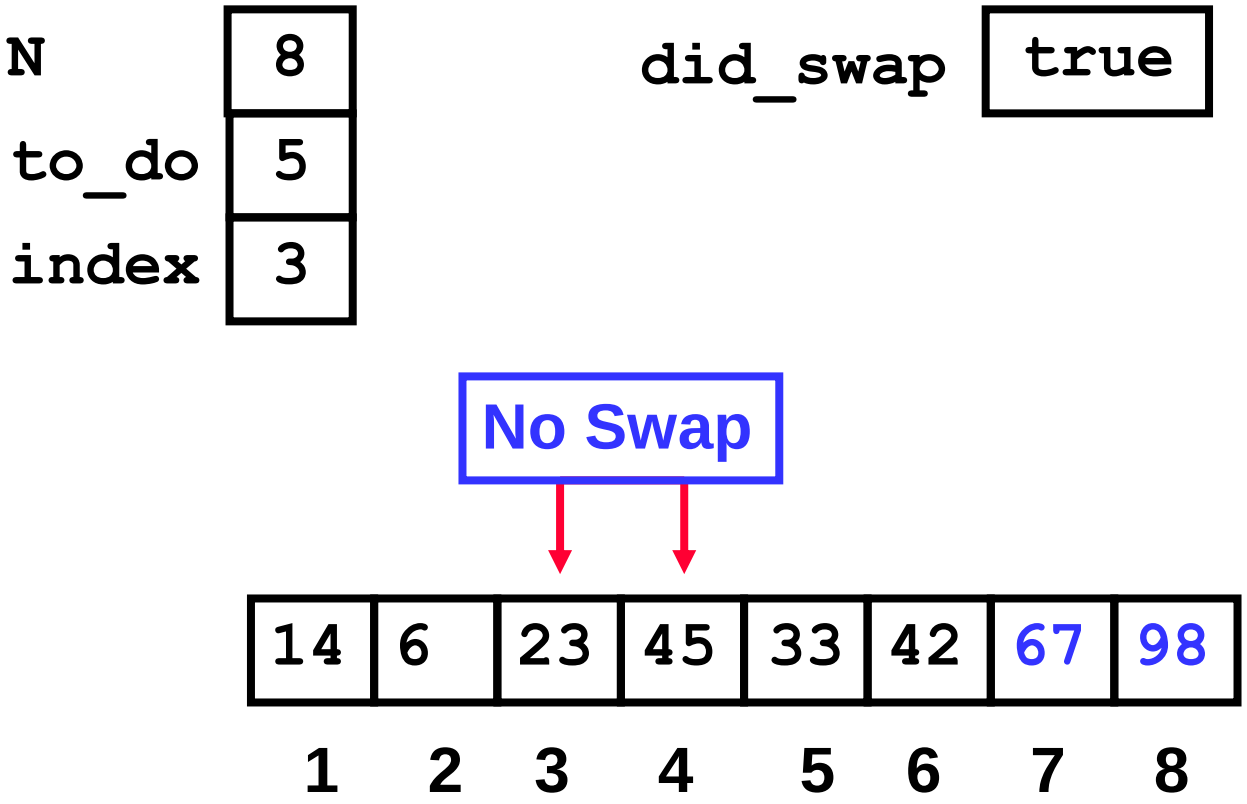
N	8
to_do	5
index	3

did_swap	true



Bubble Sort: Already Sorted Collections?...

The Third "Bubble Up"

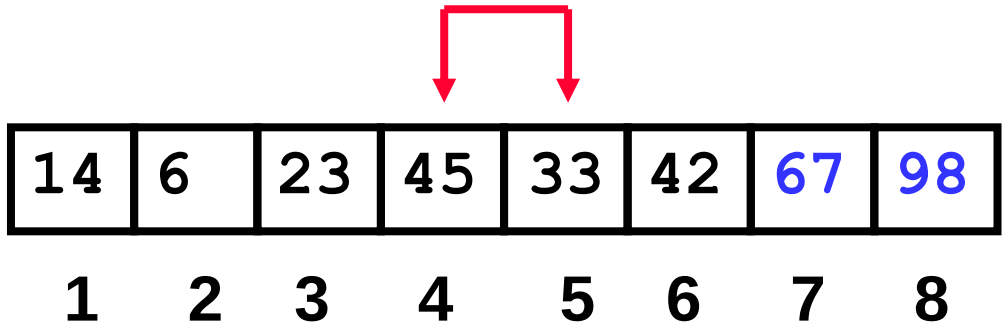


Bubble Sort: Already Sorted Collections?...

The Third "Bubble Up"

N	8
to_do	5
index	4

did_swap	true

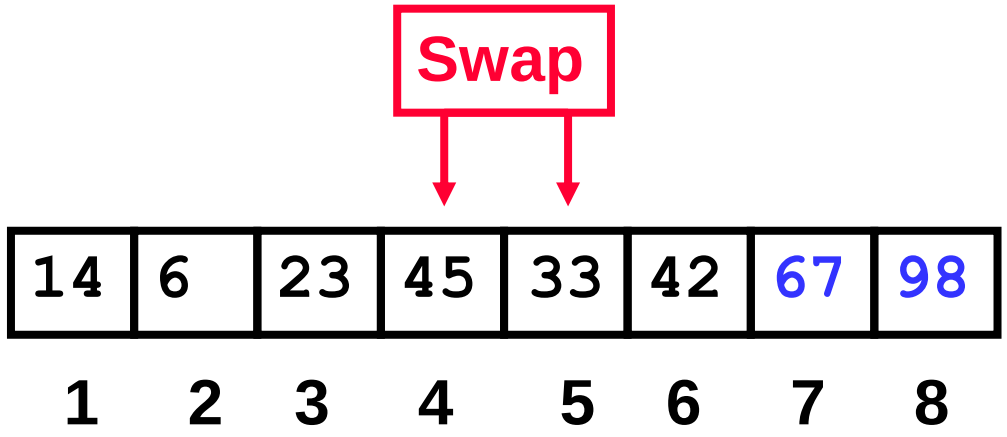


Bubble Sort: Already Sorted Collections?...

The Third "Bubble Up"

N	8
to_do	5
index	4

did_swap	true

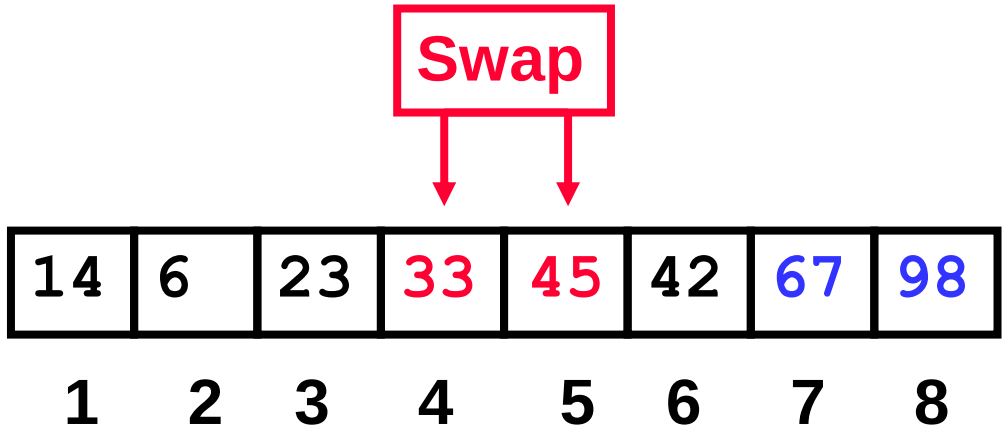


Bubble Sort: Already Sorted Collections?...

The Third "Bubble Up"

N	8
to_do	5
index	4

did_swap	true

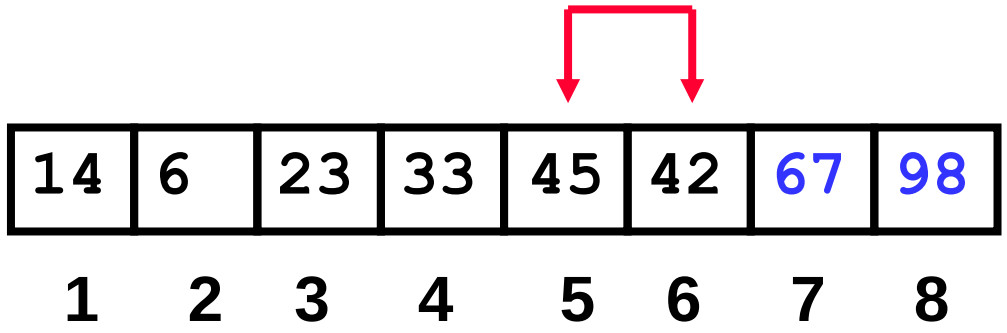


Bubble Sort: Already Sorted Collections?...

The Third "Bubble Up"

N	8
to_do	5
index	5

did_swap	true

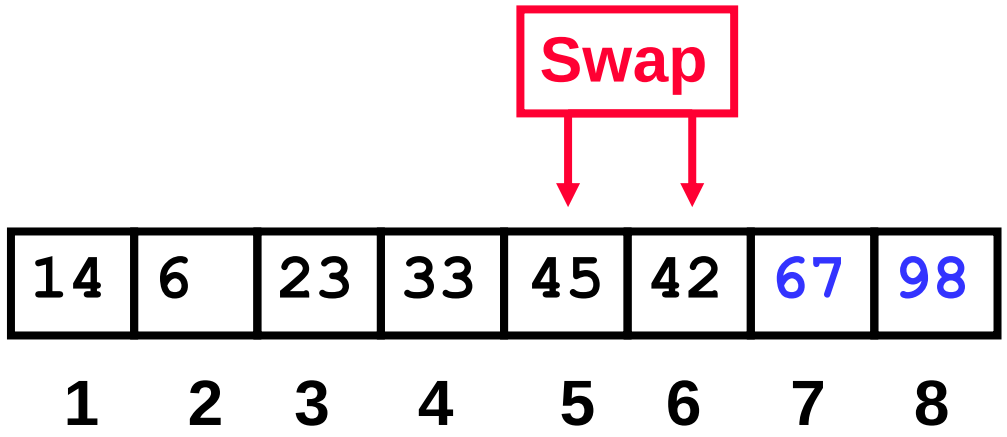


Bubble Sort: Already Sorted Collections?...

The Third "Bubble Up"

N	8
to_do	5
index	5

did_swap	true

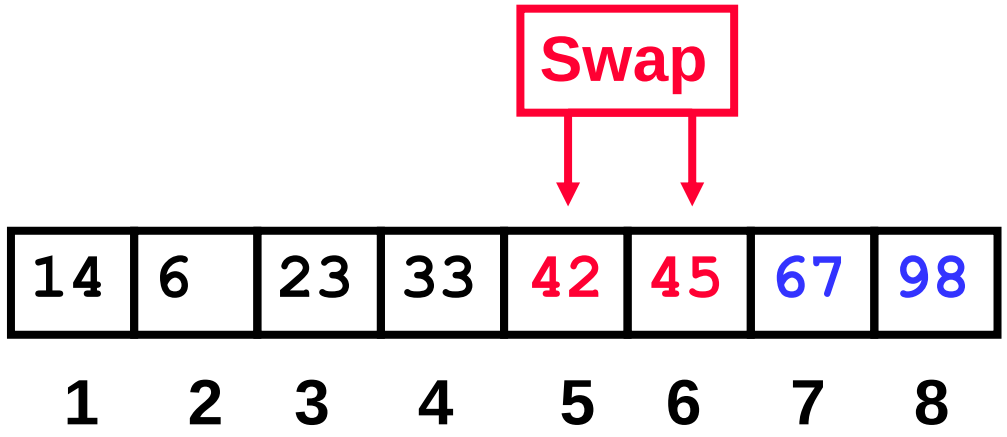


Bubble Sort: Already Sorted Collections?...

The Third "Bubble Up"

N	8
to_do	5
index	5

did_swap | true



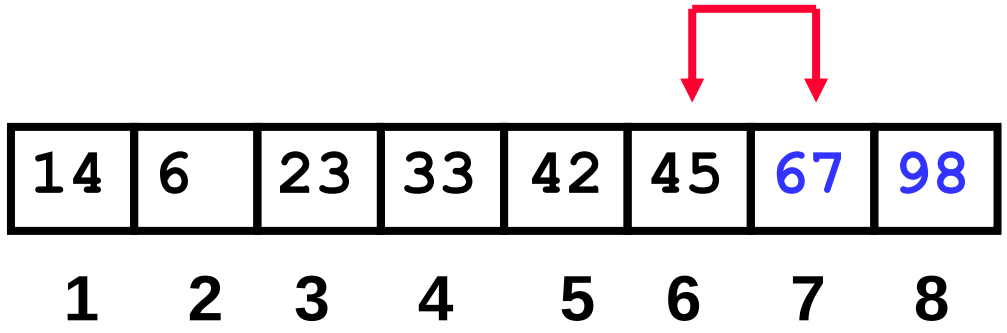
Bubble Sort: Already Sorted Collections?...

After Third Pass of Outer Loop

N	8
to_do	5
index	6

did_swap	true

Finished third "Bubble Up"

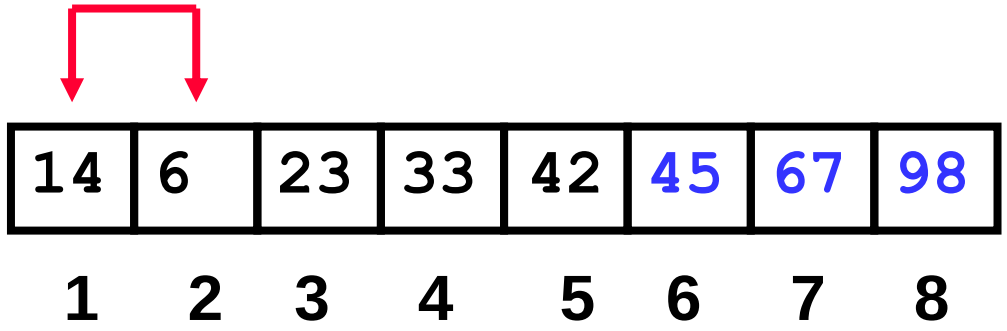


Bubble Sort: Already Sorted Collections?...

The Fourth "Bubble Up"

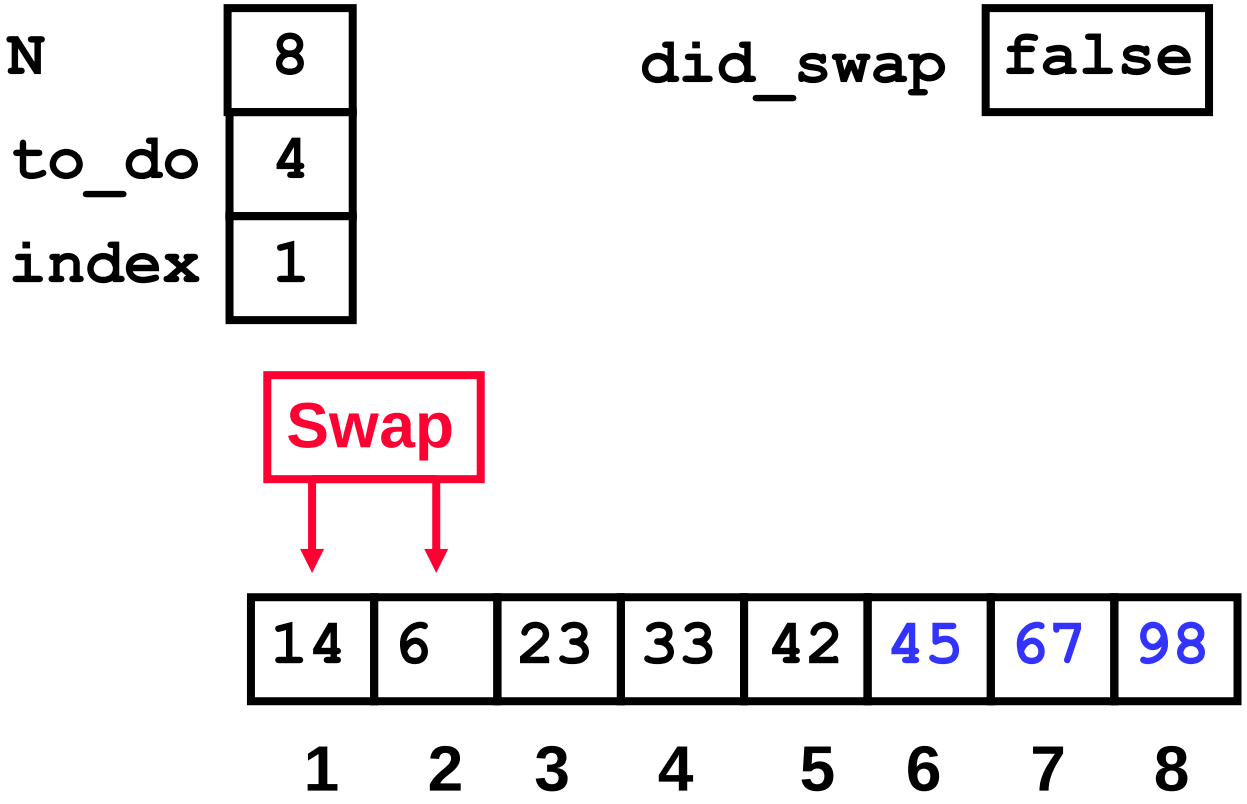
N	8
to_do	4
index	1

did_swap | false



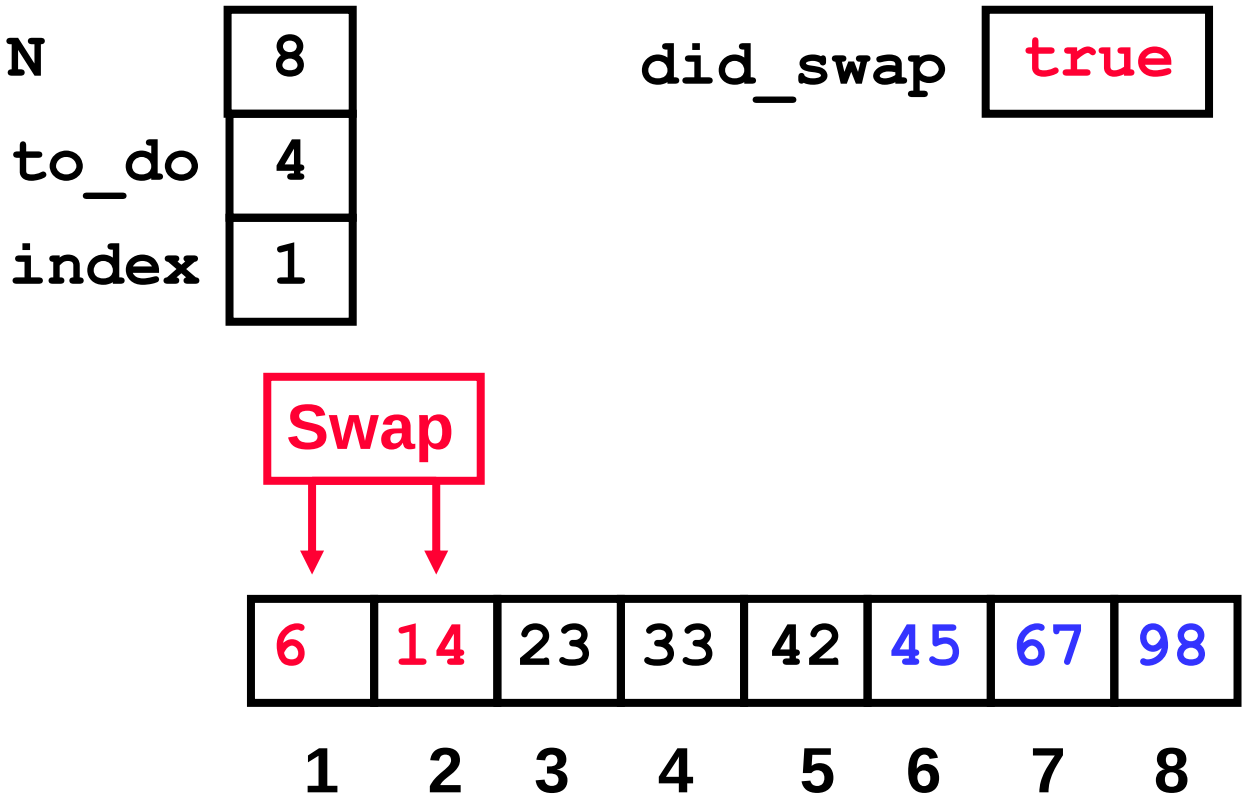
Bubble Sort: Already Sorted Collections?...

The Fourth "Bubble Up"



Bubble Sort: Already Sorted Collections?...

The Fourth "Bubble Up"

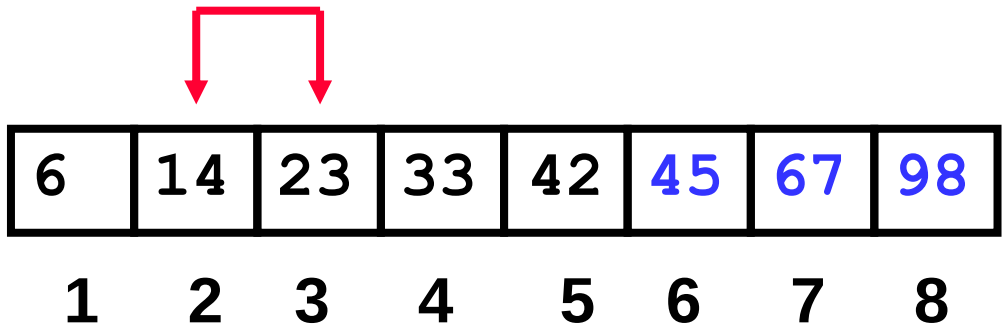


Bubble Sort: Already Sorted Collections?...

The Fourth "Bubble Up"

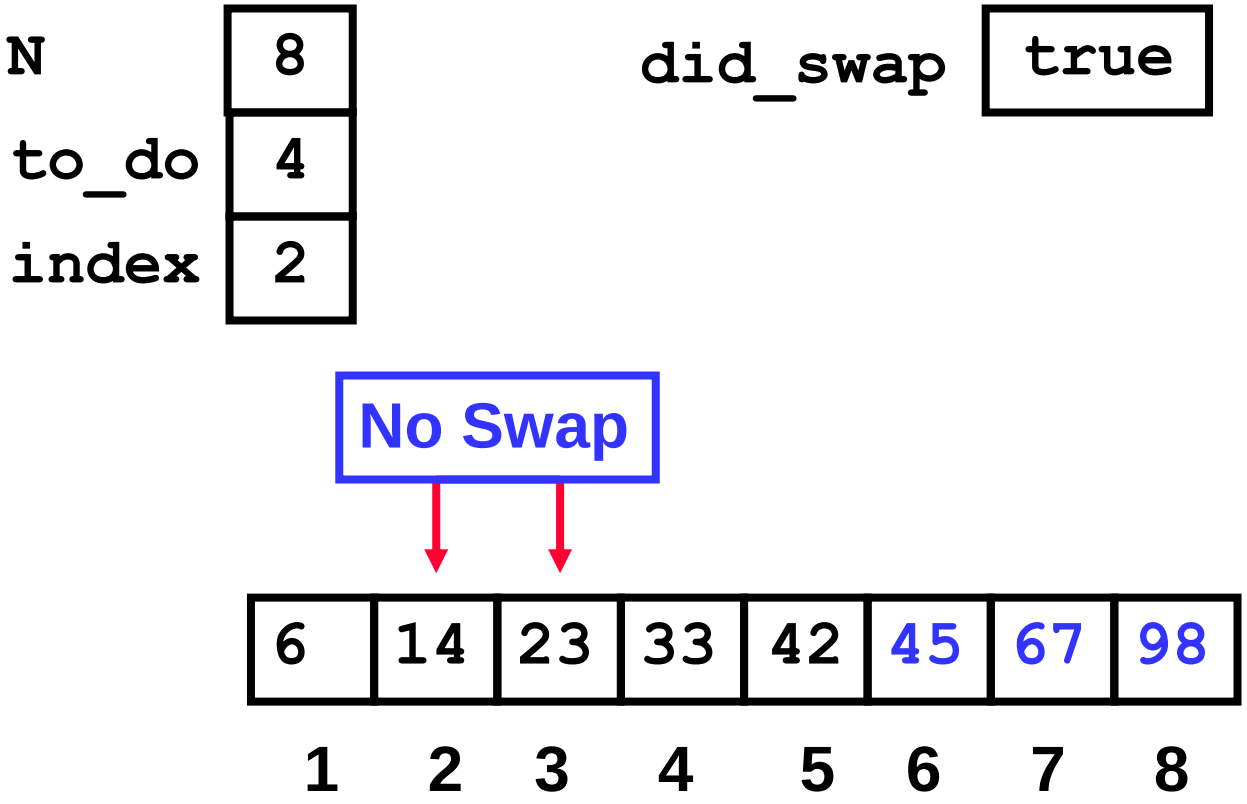
N	8
to_do	4
index	2

did_swap	true



Bubble Sort: Already Sorted Collections?...

The Fourth "Bubble Up"

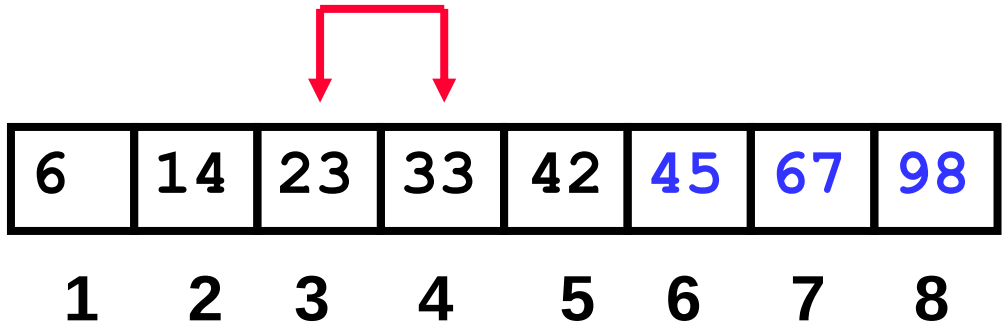


Bubble Sort: Already Sorted Collections?...

The Fourth "Bubble Up"

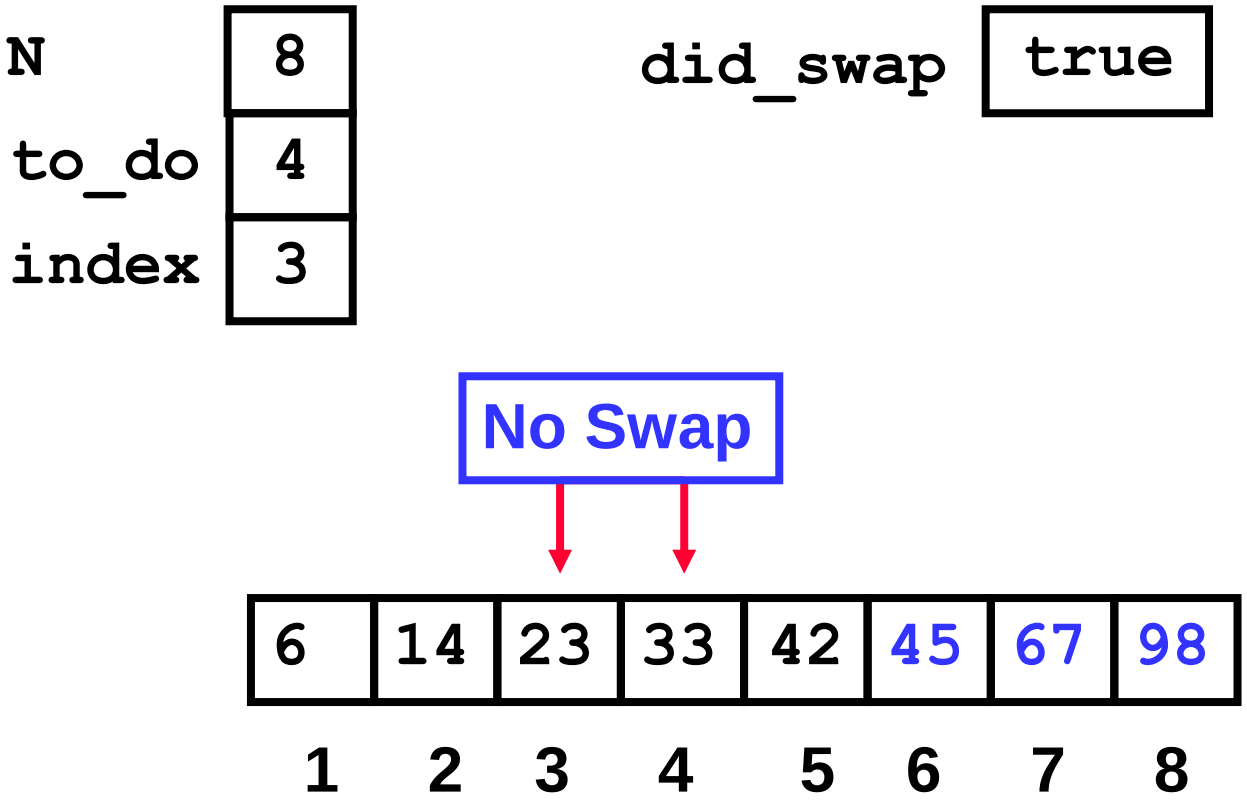
N	8
to_do	4
index	3

did_swap	true



Bubble Sort: Already Sorted Collections?...

The Fourth "Bubble Up"

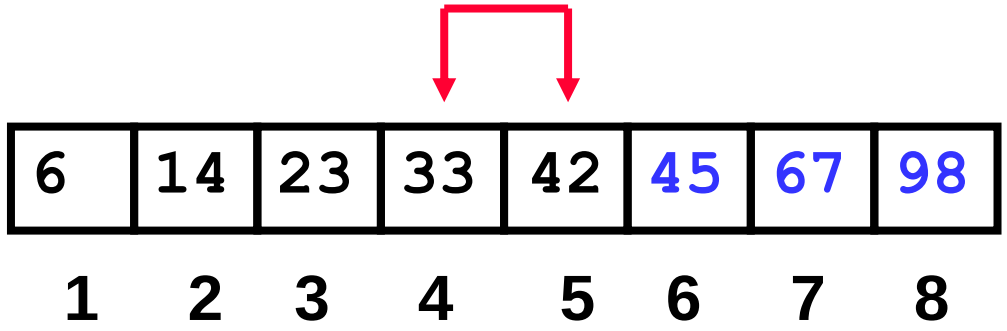


Bubble Sort: Already Sorted Collections?...

The Fourth "Bubble Up"

N	8
to_do	4
index	4

did_swap	true

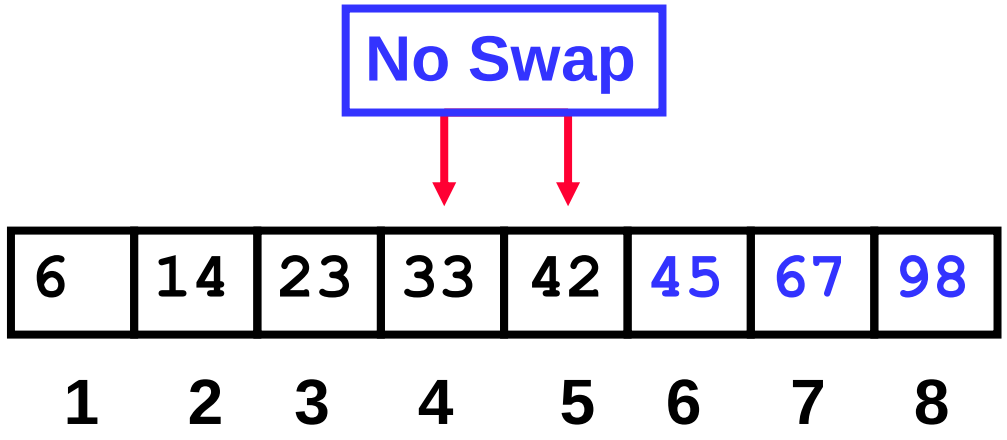


Bubble Sort: Already Sorted Collections?...

The Fourth "Bubble Up"

N	8
to_do	4
index	4

did_swap	true



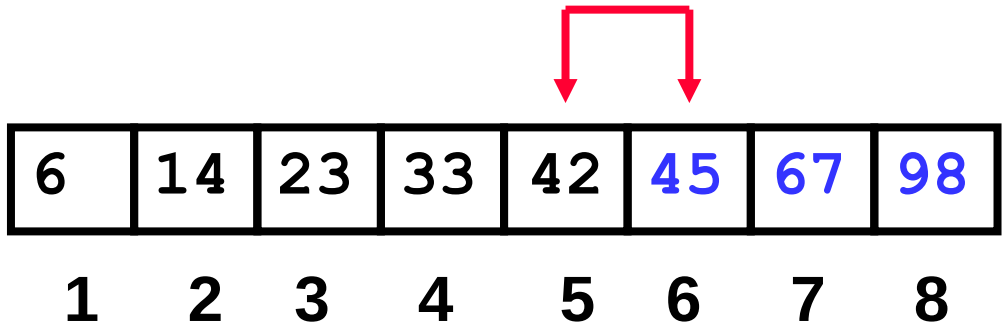
Bubble Sort: Already Sorted Collections?...

After Fourth Pass of Outer Loop

N	8
to_do	4
index	5

did_swap	true

Finished fourth “Bubble Up”

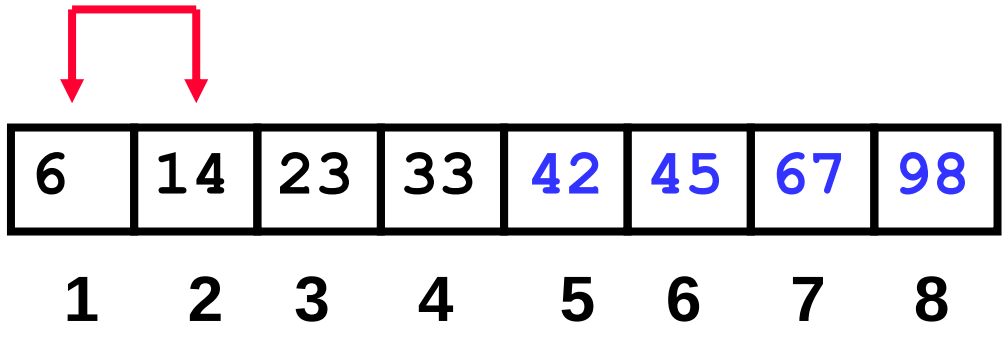


Bubble Sort: Already Sorted Collections?...

The Fifth "Bubble Up"

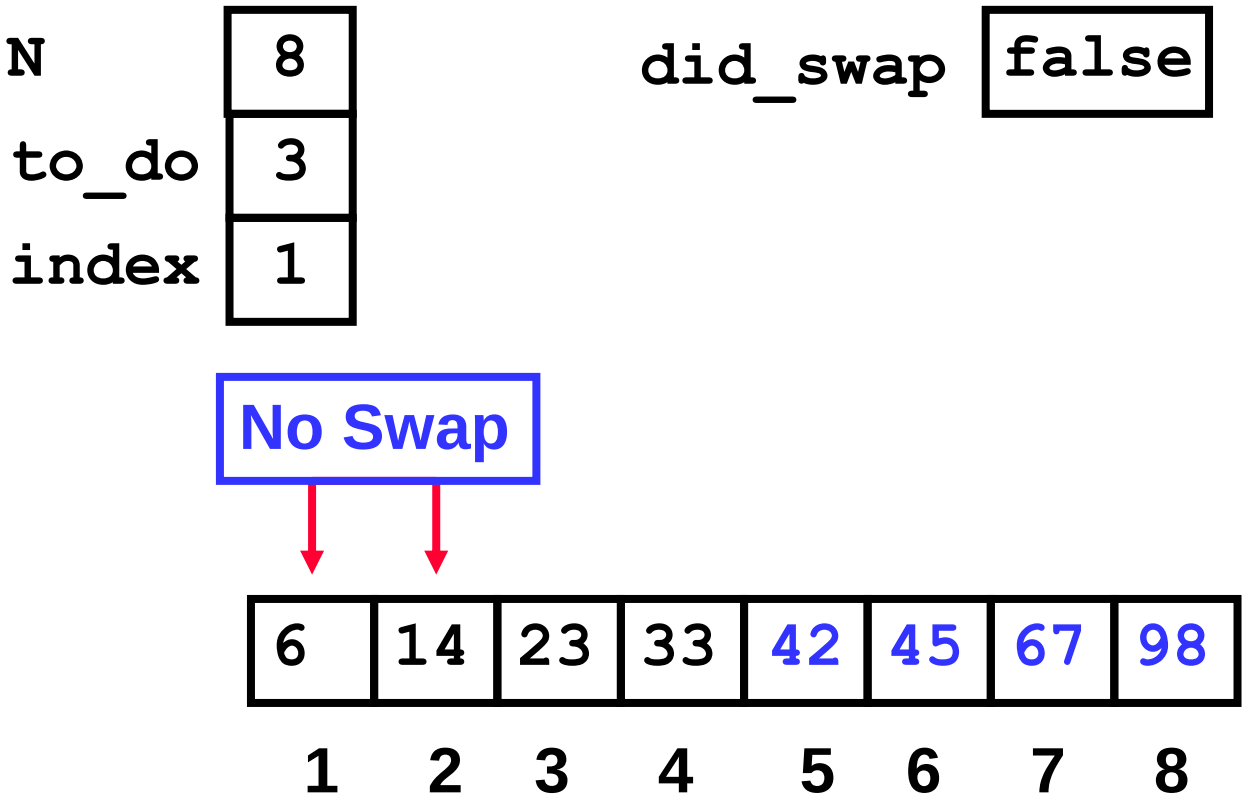
N	8
to_do	3
index	1

did_swap | false



Bubble Sort: Already Sorted Collections?...

The Fifth "Bubble Up"

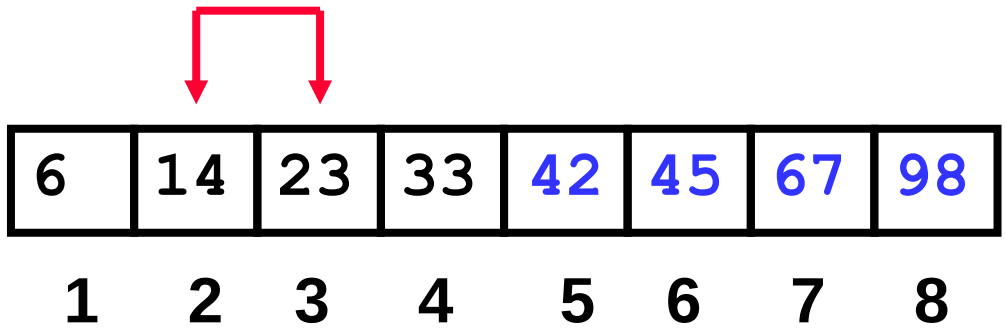


Bubble Sort: Already Sorted Collections?...

The Fifth "Bubble Up"

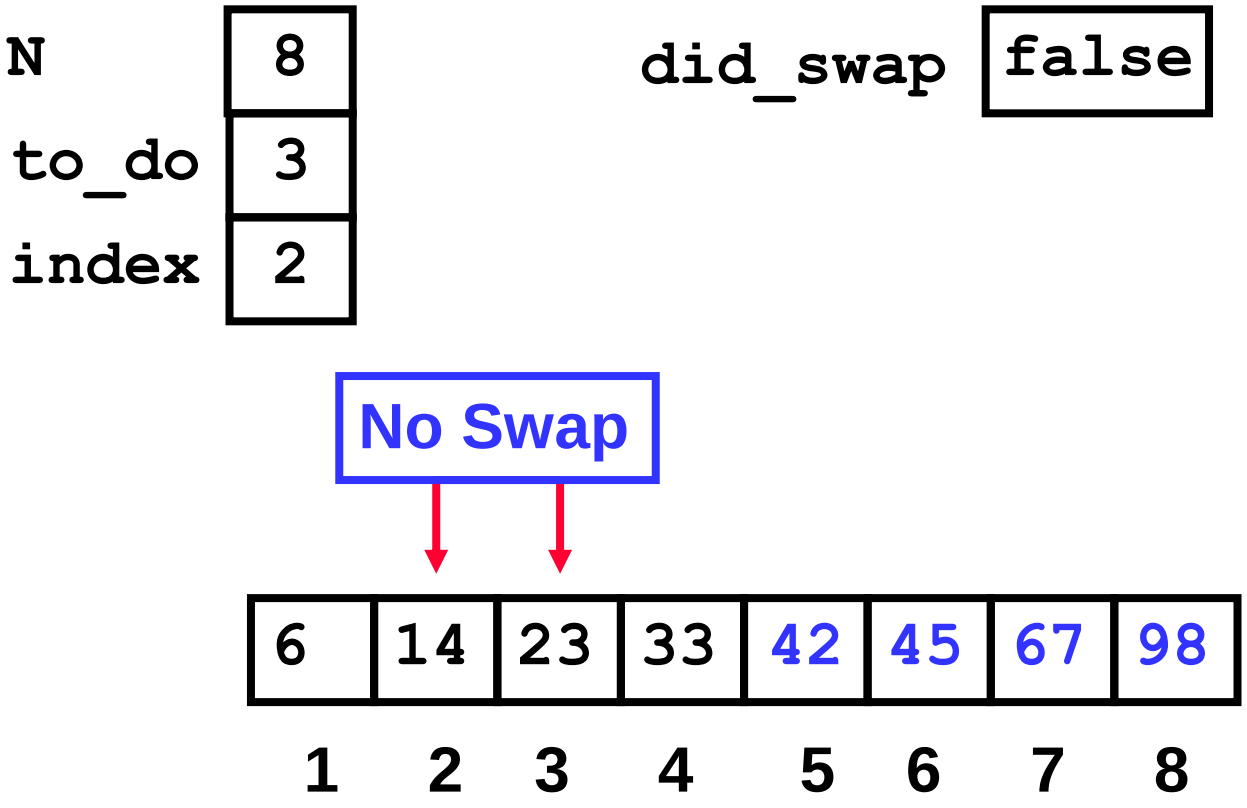
N	8
to_do	3
index	2

did_swap	false



Bubble Sort: Already Sorted Collections?...

The Fifth "Bubble Up"

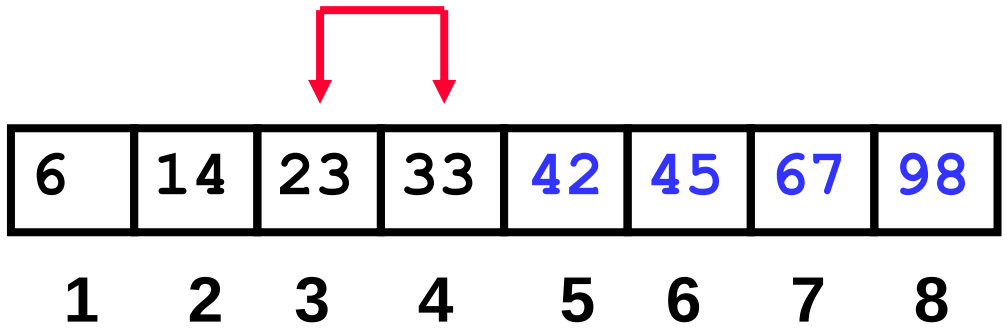


Bubble Sort: Already Sorted Collections?...

The Fifth "Bubble Up"

N	8
to_do	3
index	3

did_swap	false

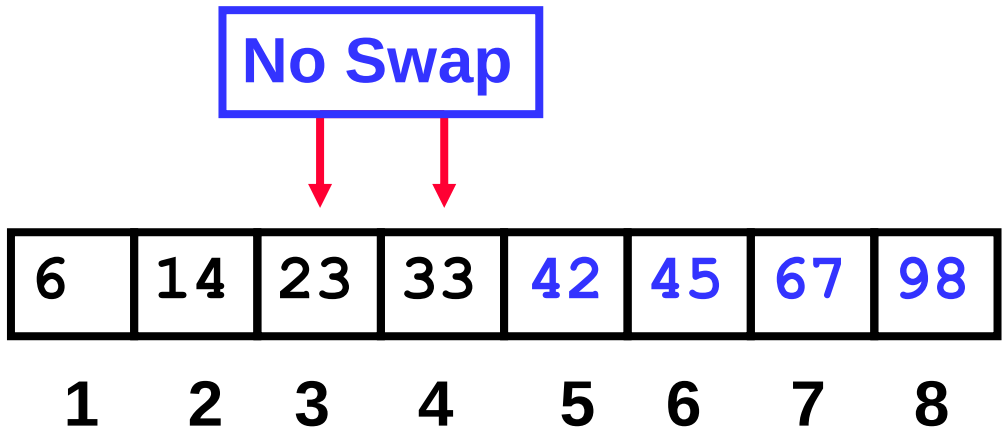


Bubble Sort: Already Sorted Collections?...

The Fifth "Bubble Up"

N	8
to_do	3
index	3

did_swap | false



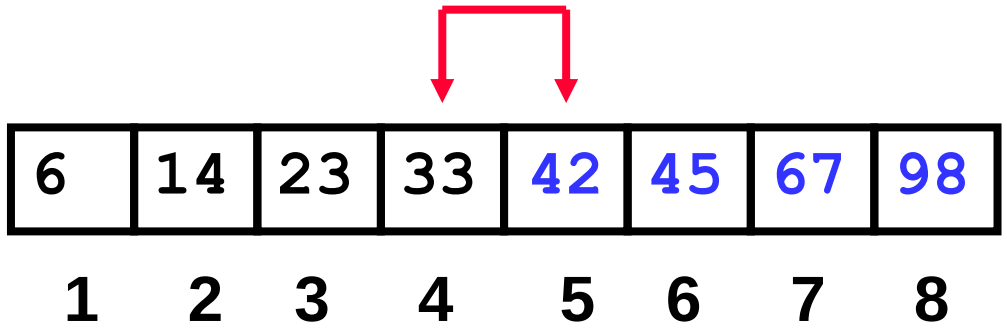
Bubble Sort: Already Sorted Collections?...

After Fifth Pass of Outer Loop

N	8
to_do	3
index	4

did_swap | false

Finished fifth “Bubble Up”



Bubble Sort: Already Sorted Collections?...

Finished "Early"

N	8
to_do	3
index	4

did_swap: false

We didn't do any swapping, so all of the other elements must be correctly placed.

We can "skip" the last two passes of the outer loop.

6	14	23	33	42	45	67	98
1	2	3	4	5	6	7	8

- "Bubble Up" algorithm will move largest value to its correct location (to the right)
- Repeat "Bubble Up" until all elements are correctly placed:
  - Maximum of N-1 times
  - Can finish early if no swapping occurs
- We reduce the number of elements we compare each time one is correctly placed

Bubble Sort: Time Complexity Analysis

- Bubble sort uses two loops- inner loop and outer loop.
- The inner loop deterministically performs O(n) comparisons.

- **Best-Case Time Complexity**
  - **Array is already sorted**
  - **Need 1 iteration with (N-1) comparisons**

Called Linear Time
O(N)
Order-of-N

- **Worst-Case Time Complexity**
  - **Need N-1 iterations**
  - **(N-1) + (N-2) + (N-3) + …. + (1) = (N-1)* N / 2**

Called Quadratic Time
$O(N^2)$
Order-of-N-square

Bubble Sort: Time Complexity Analysis …

Average Case:

- In average case, bubble sort may require (n/2) passes and O(n) comparisons for each pass.
- Hence, the average case time complexity of bubble sort is O(n/2 x n) = $\Theta(n^2)$.

Time Complexity	
Best Case	$\Omega(n)$
Average Case	$\Theta(n^2)$
Worst Case	$O(n^2)$

Bubble Sort: Properties

Some of the important properties of bubble sort algorithm are-

- Bubble sort is a **stable sorting** algorithm.
- Bubble sort is an **in-place sorting** algorithm.
- The worst case time complexity of bubble sort algorithm is $\mathbf{O(n^2)}$.
- The space complexity of bubble sort algorithm is $\mathbf{O(1)}$.
- Number of swaps in bubble sort = Number of inversion pairs present in the given array.
- Bubble sort is beneficial when array **elements are less** and the array is **nearly sorted**. Also used in computer graphics

Any Questions ?

Dr. Anand Singh Jalal
Professor
Email: asjalal@gla.ac.in