

Sorting Algorithms

Abdul Ghafoor

Popular Sorting Algorithms

- Bubble Sort
- Selection Sort
- Insertion Sort
- Merge Sort
- Quick Sort
- Heap Sort

Bubble Sort

Sorting

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

1	2	3	4	5	
677	42	35	12	101	5



1	2	3	4	5	
56	12	35	42	77	101

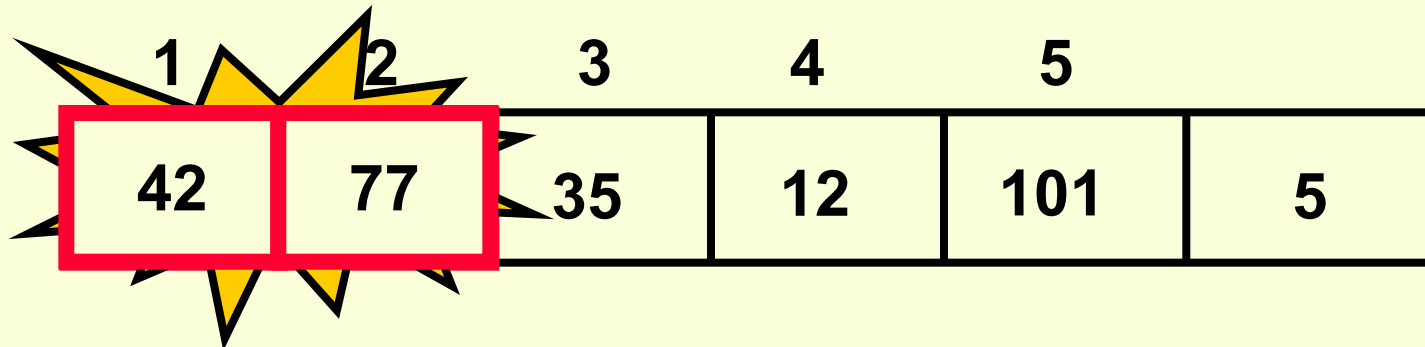
"Bubbling Up" the Largest Element

- 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**

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

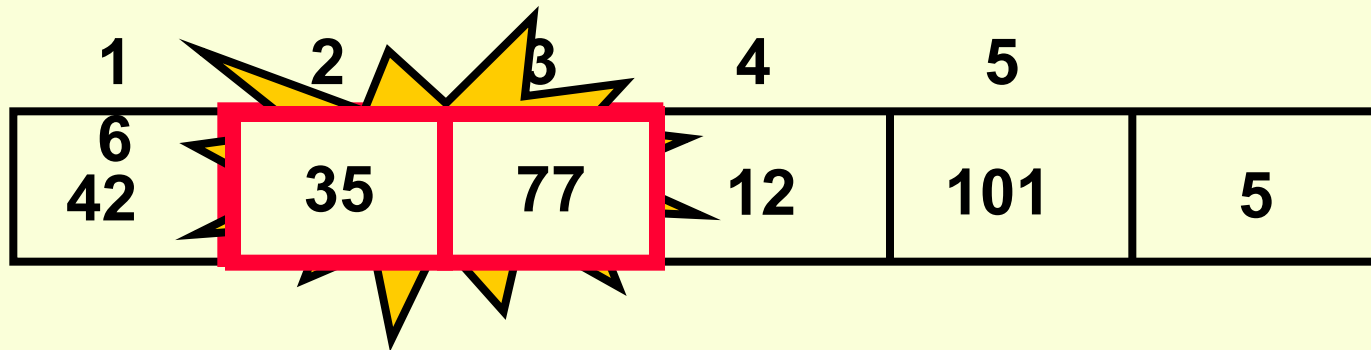
"Bubbling Up" the Largest Element

- 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



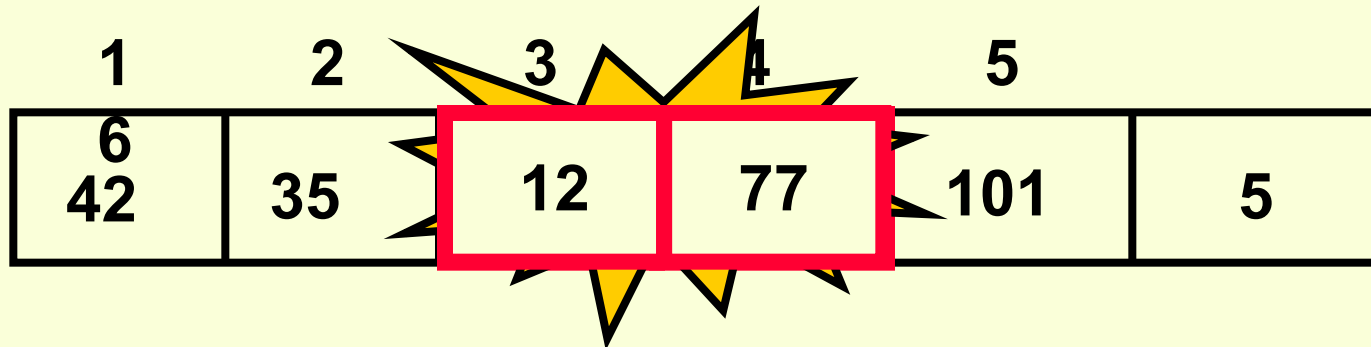
"Bubbling Up" the Largest Element

- 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



"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



"Bubbling Up" the Largest Element

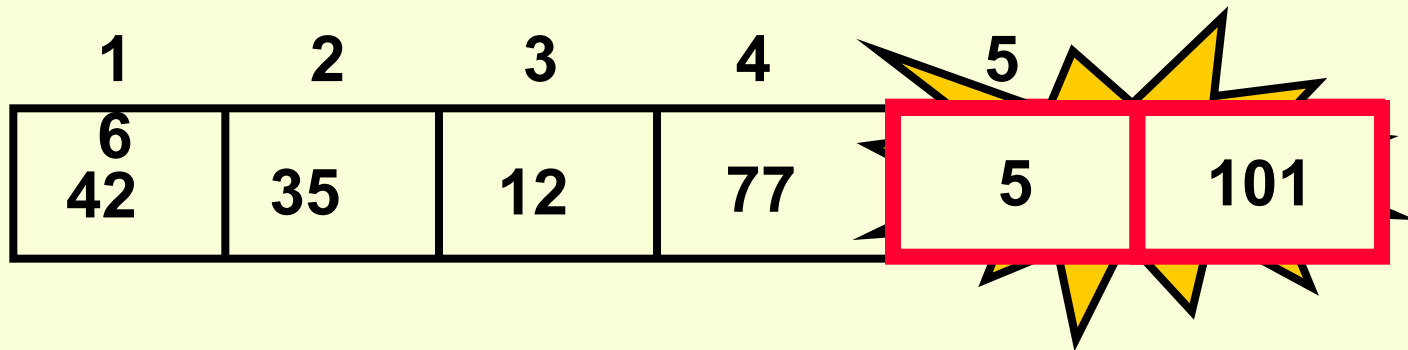
- Traverse a collection of elements
 - Move from the front to the end
 - “Bubble” the largest value to the end using pair-wise comparisons and swapping

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

No need to swap

"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



"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

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

Largest value correctly placed

Items of Interest

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

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

Largest value correctly placed

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.

“Bubbling” All the Elements

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

1
2

Reducing the Number of Comparisons

1	2	3	4	5	
67	42	35	12	101	5

1	2	3	4	5	
42	35	12	77	5	101

1	2	3	4	5	
35	12	42	5	77	101

1	2	3	4	5	
12	35	5	42	77	101

1	2	3	4	5	
12	5	35	42	77	101

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	
5	12	35	42	77	101

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.”

An Animated Example

N

8

 did_swap

true

to_do

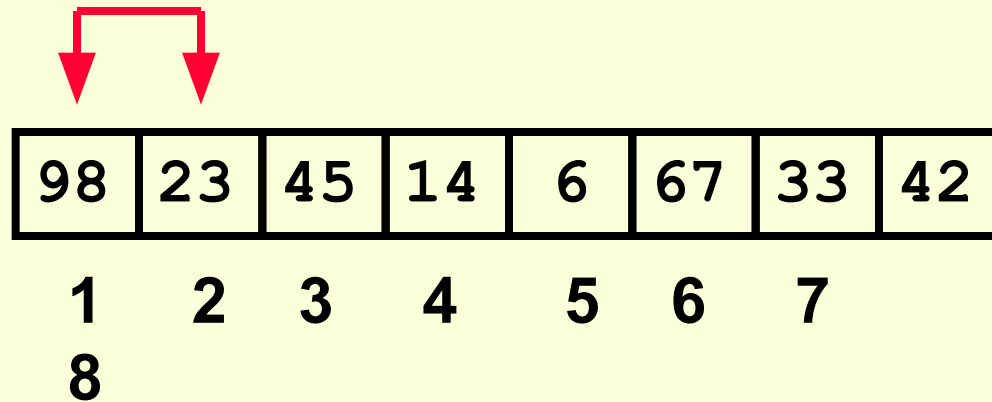
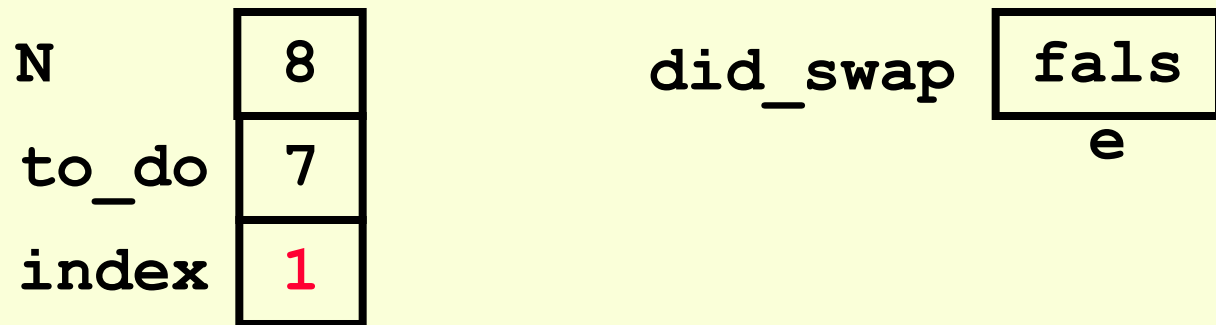
7

index

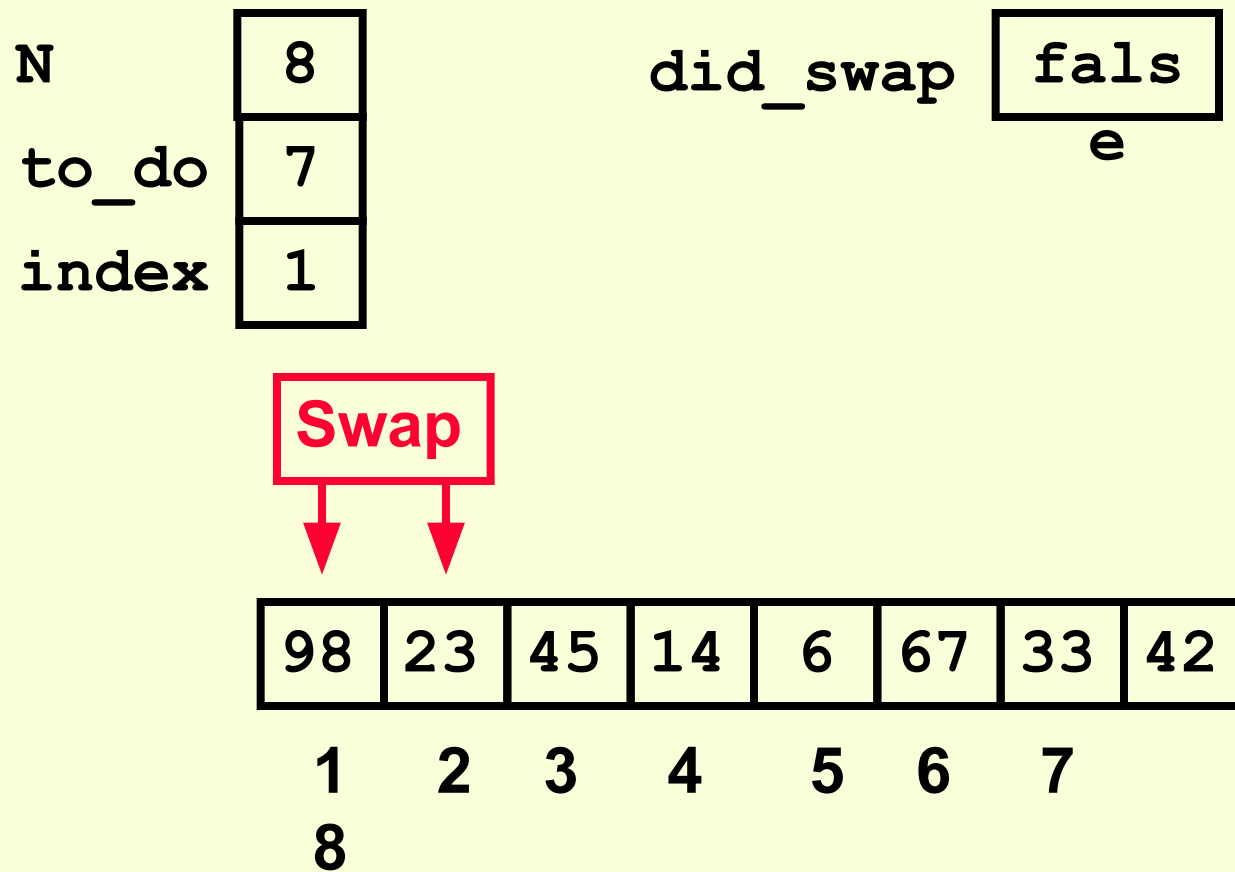
--

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

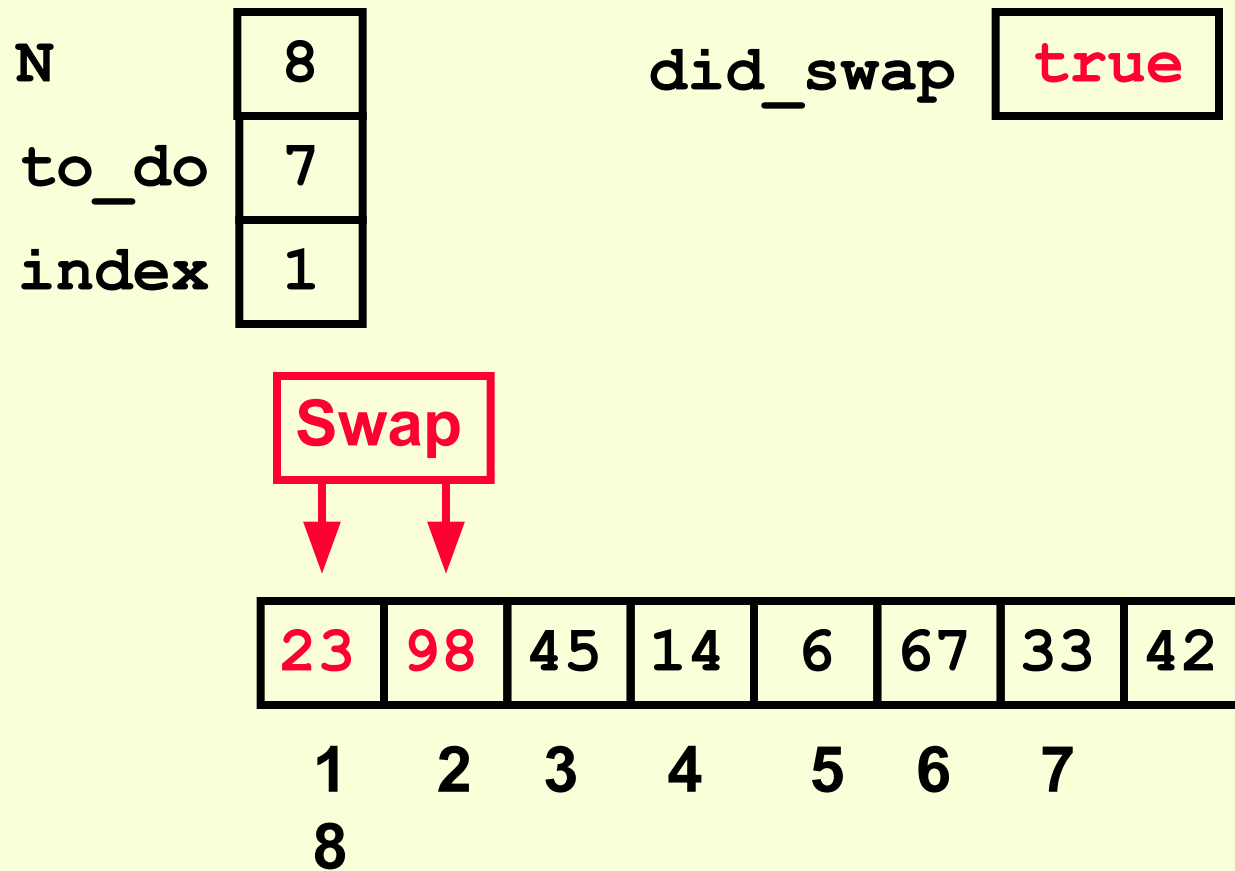
An Animated Example



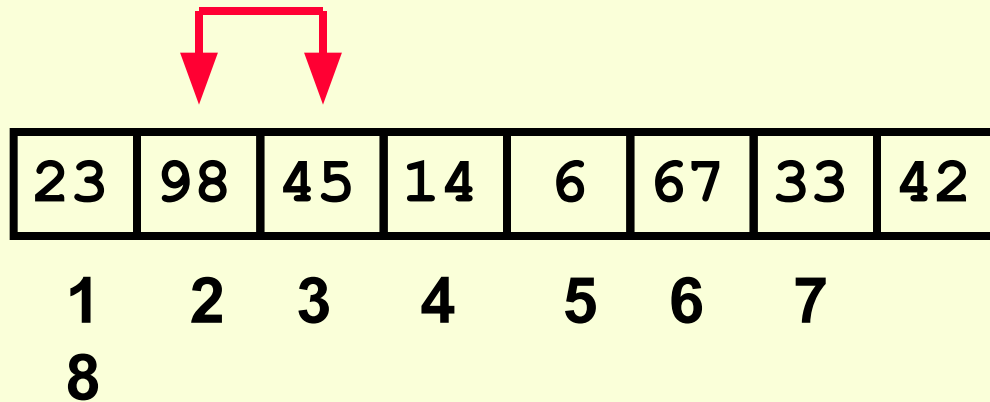
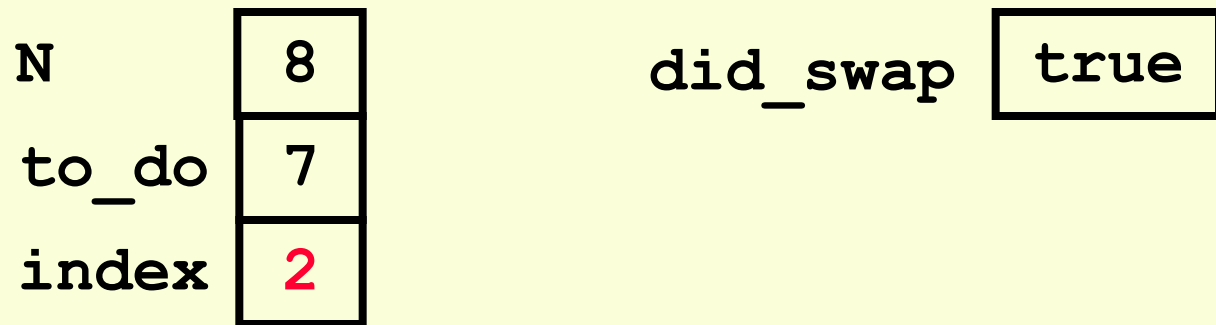
An Animated Example



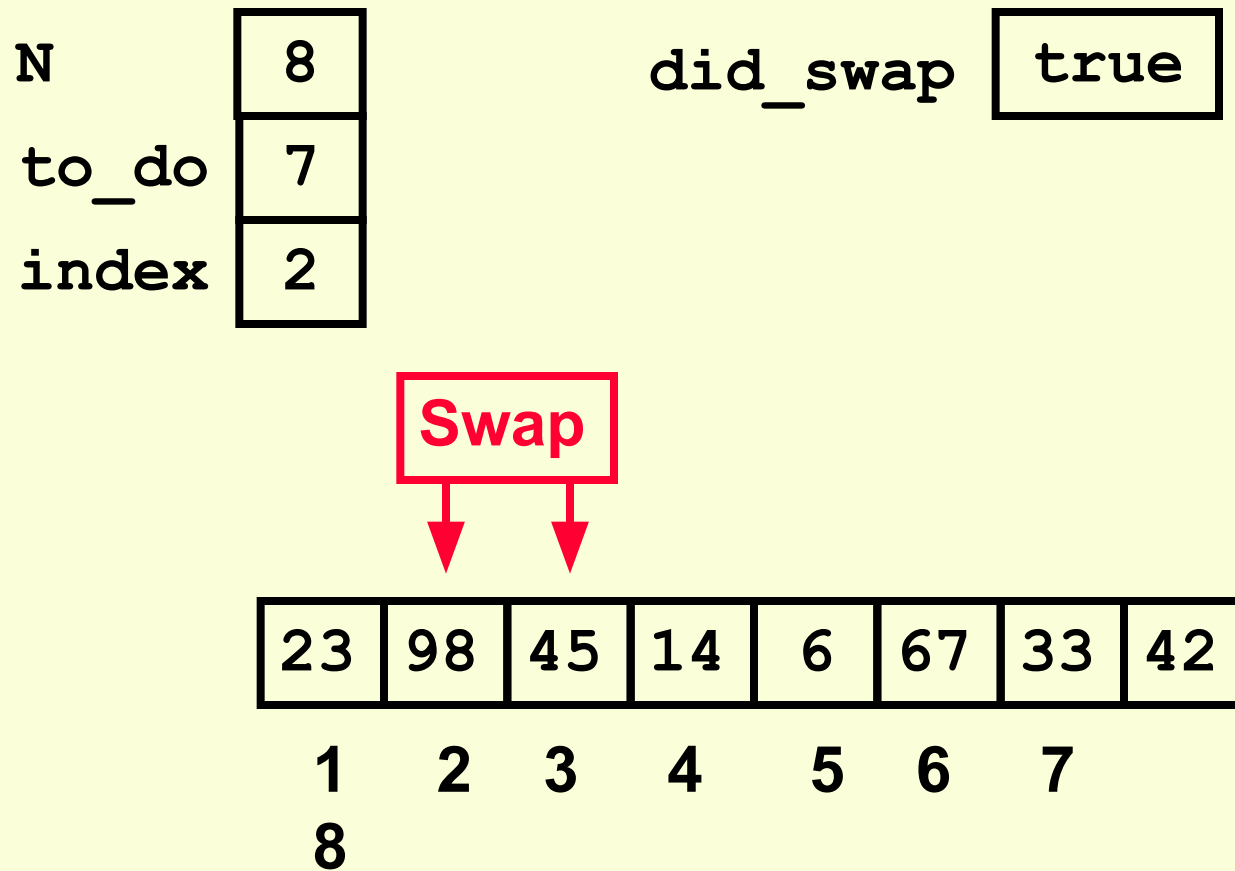
An Animated Example



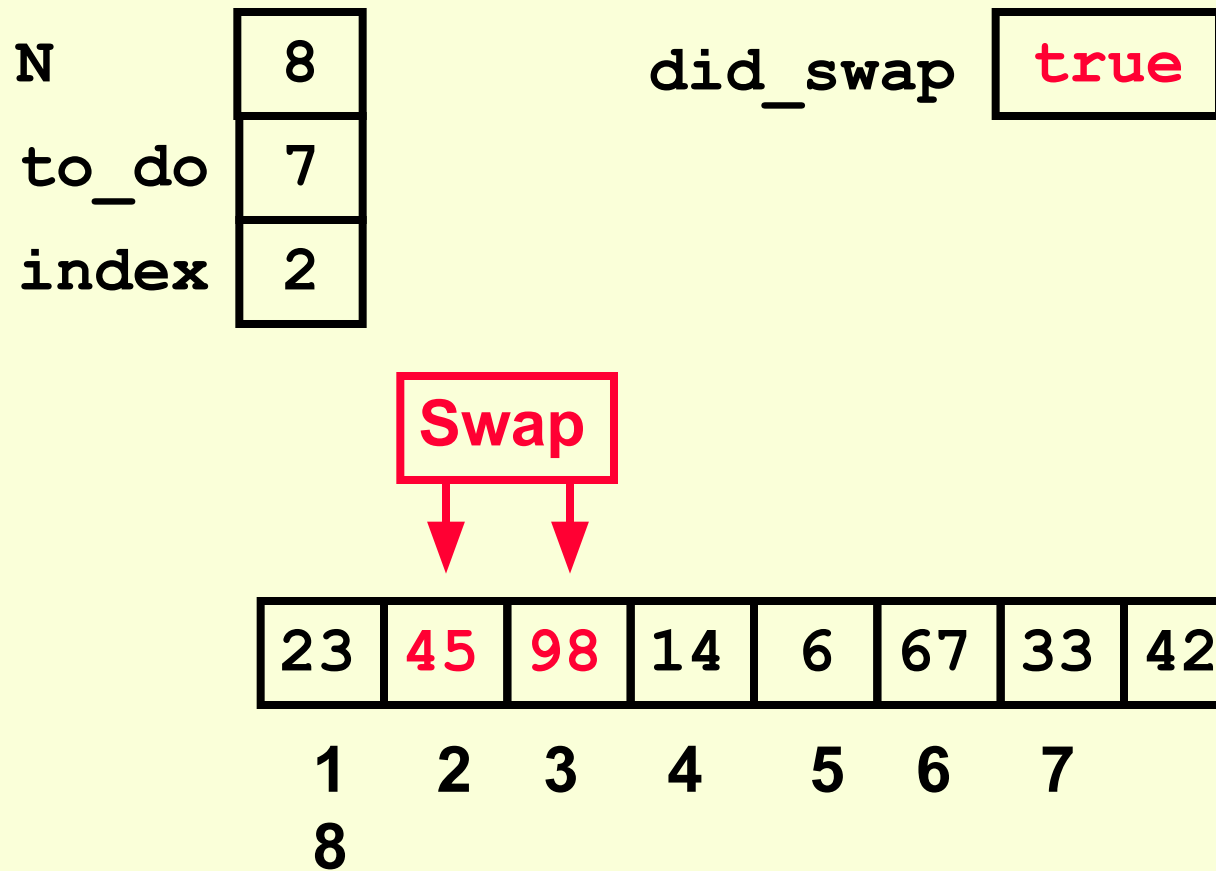
An Animated Example



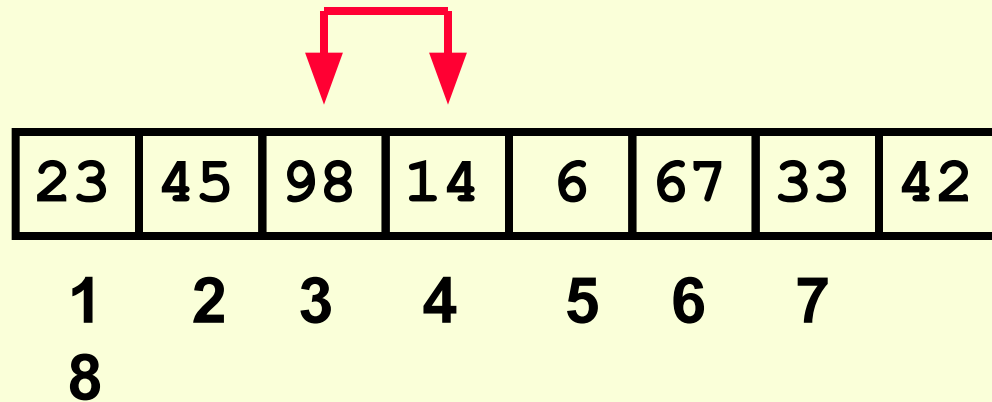
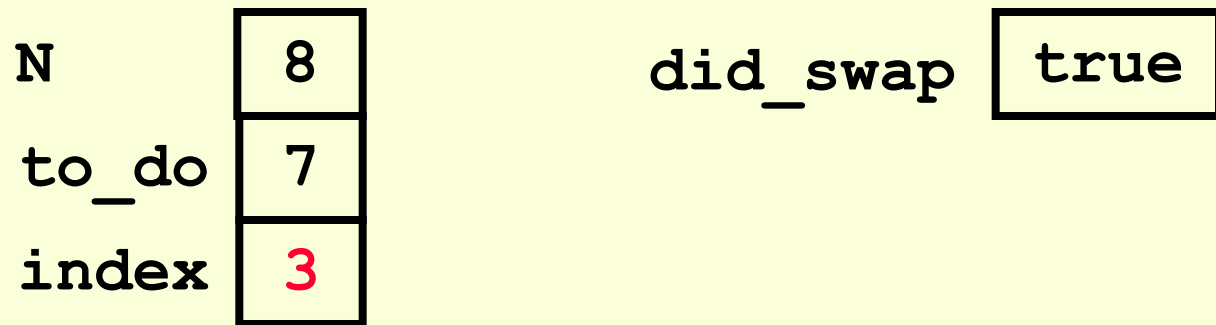
An Animated Example



An Animated Example



An Animated Example

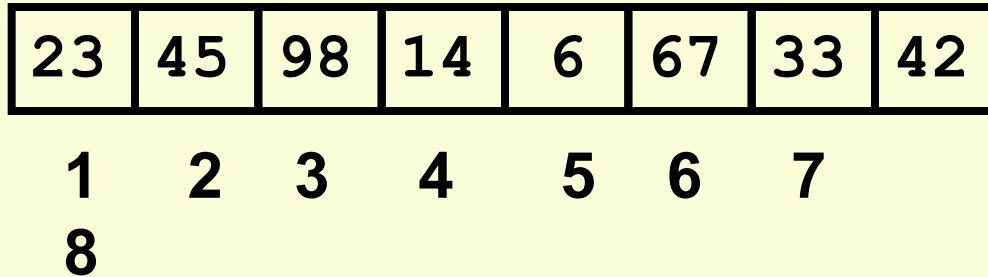


An Animated Example

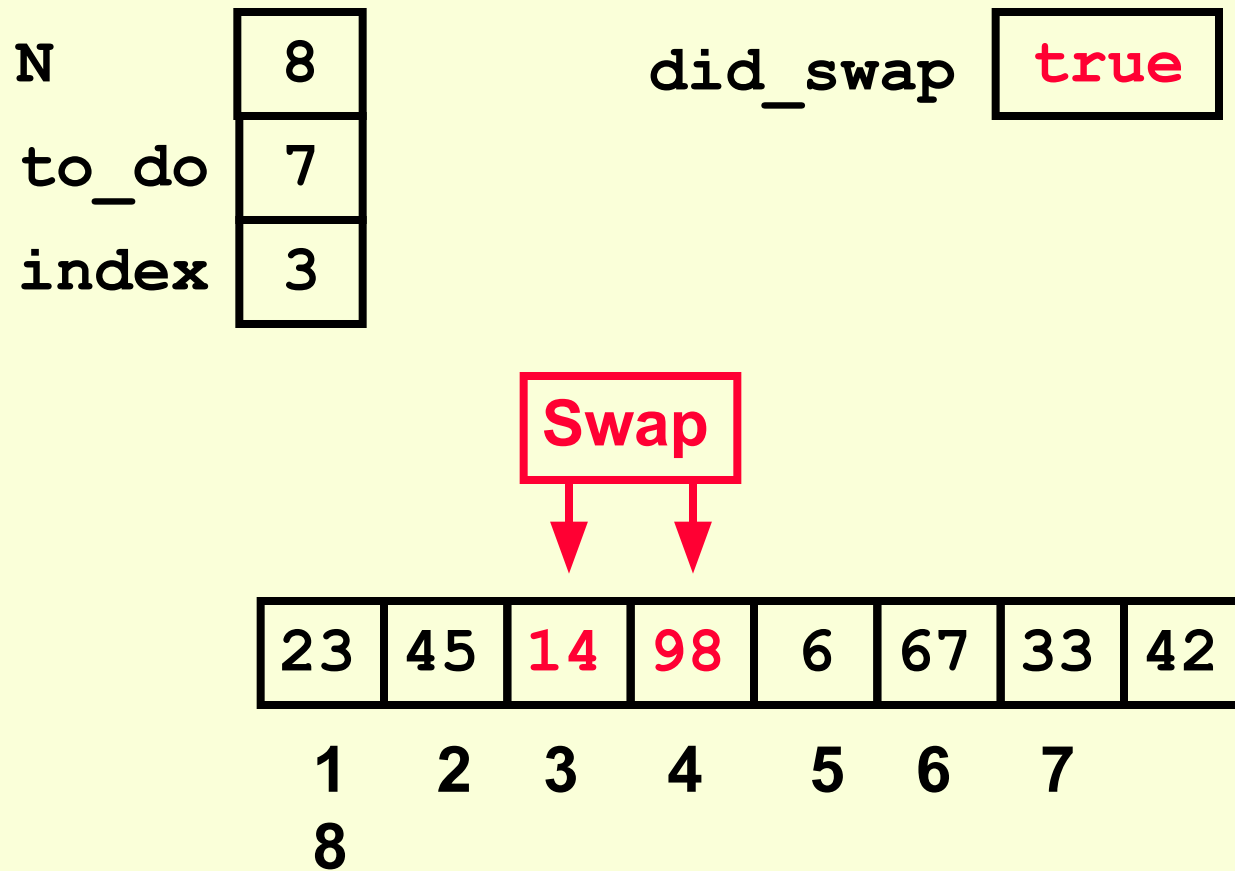
N	8
to_do	7
index	3

did_swap **true**

Swap



An Animated Example



An Animated Example

N

8

to_do

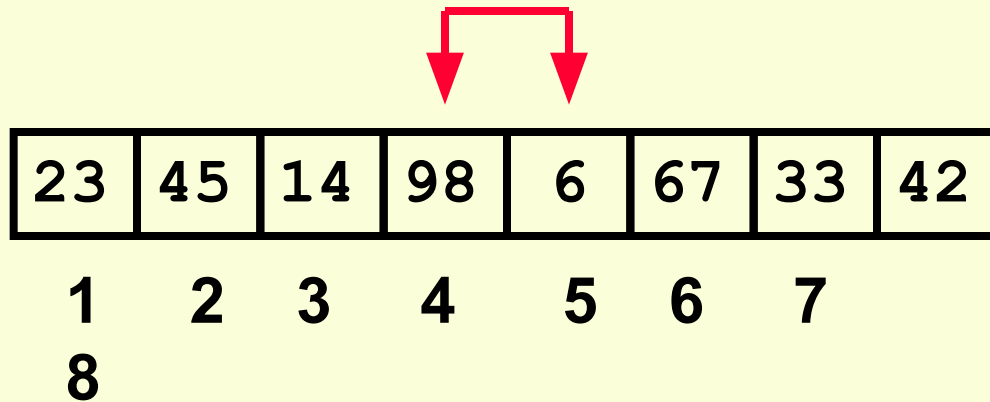
7

index

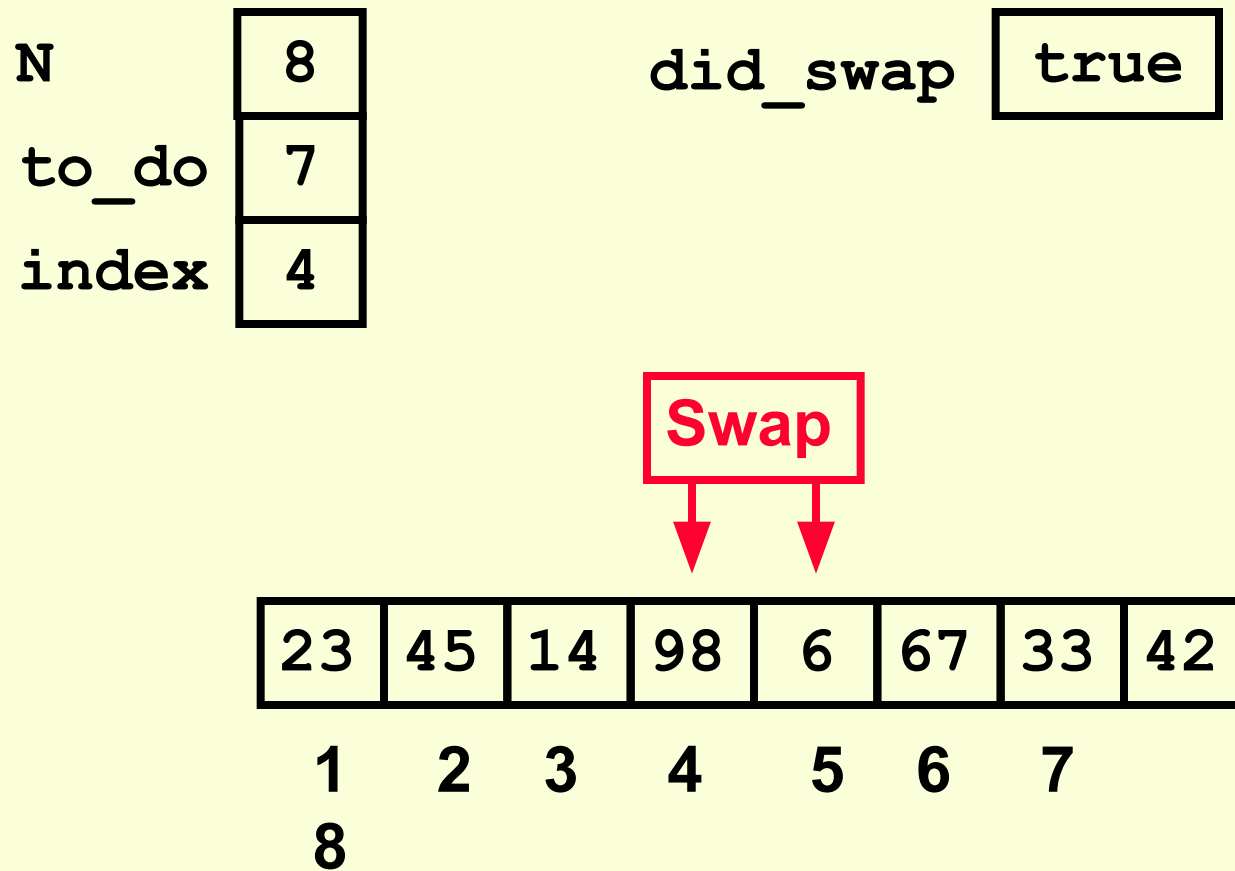
4

did_swap

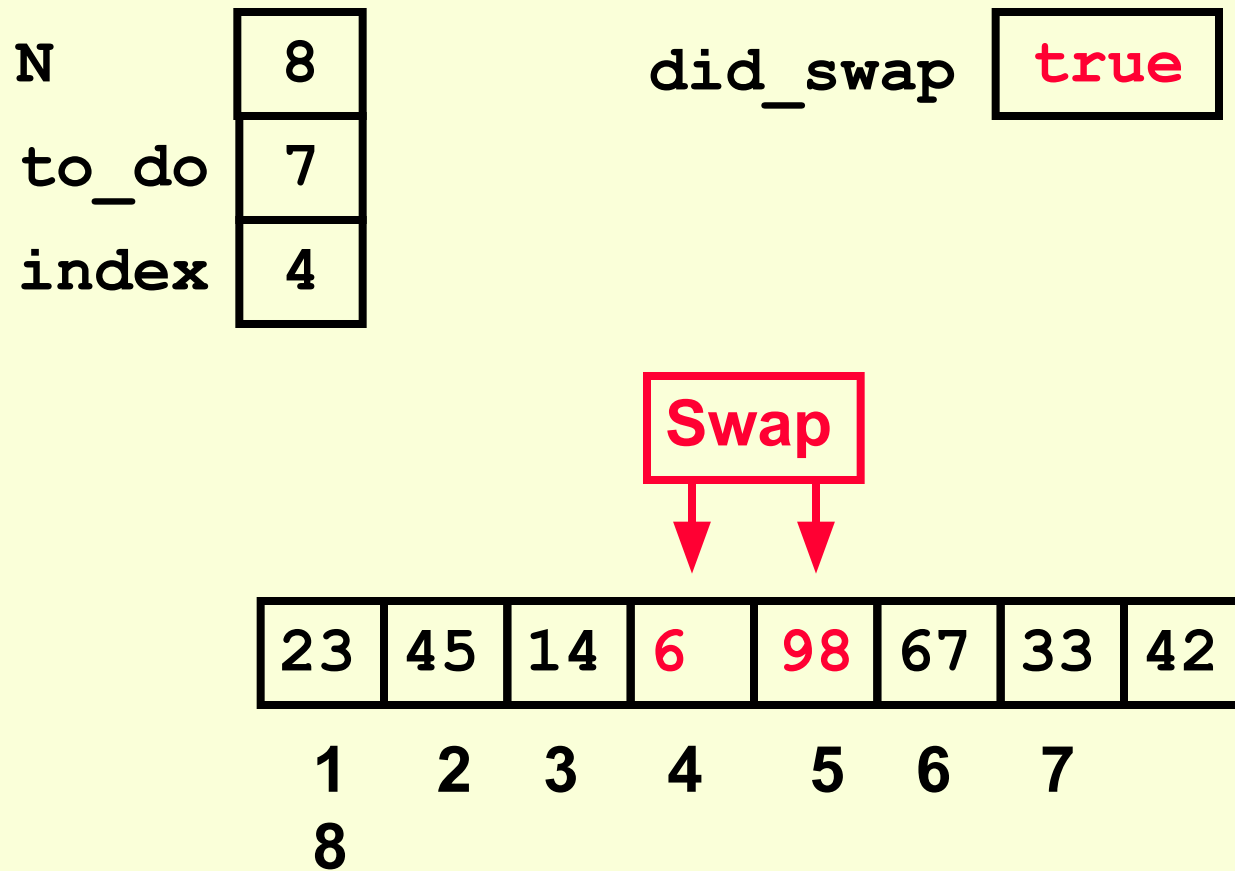
true



An Animated Example



An Animated Example



An Animated Example

N

8

to_do

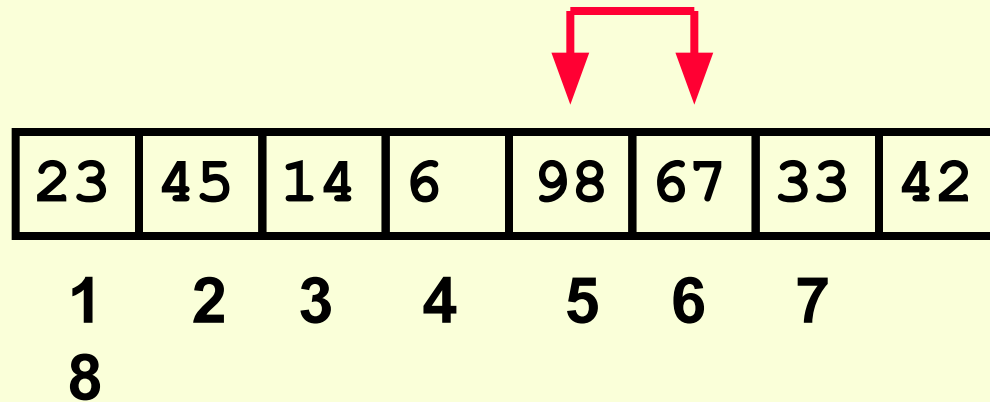
7

index

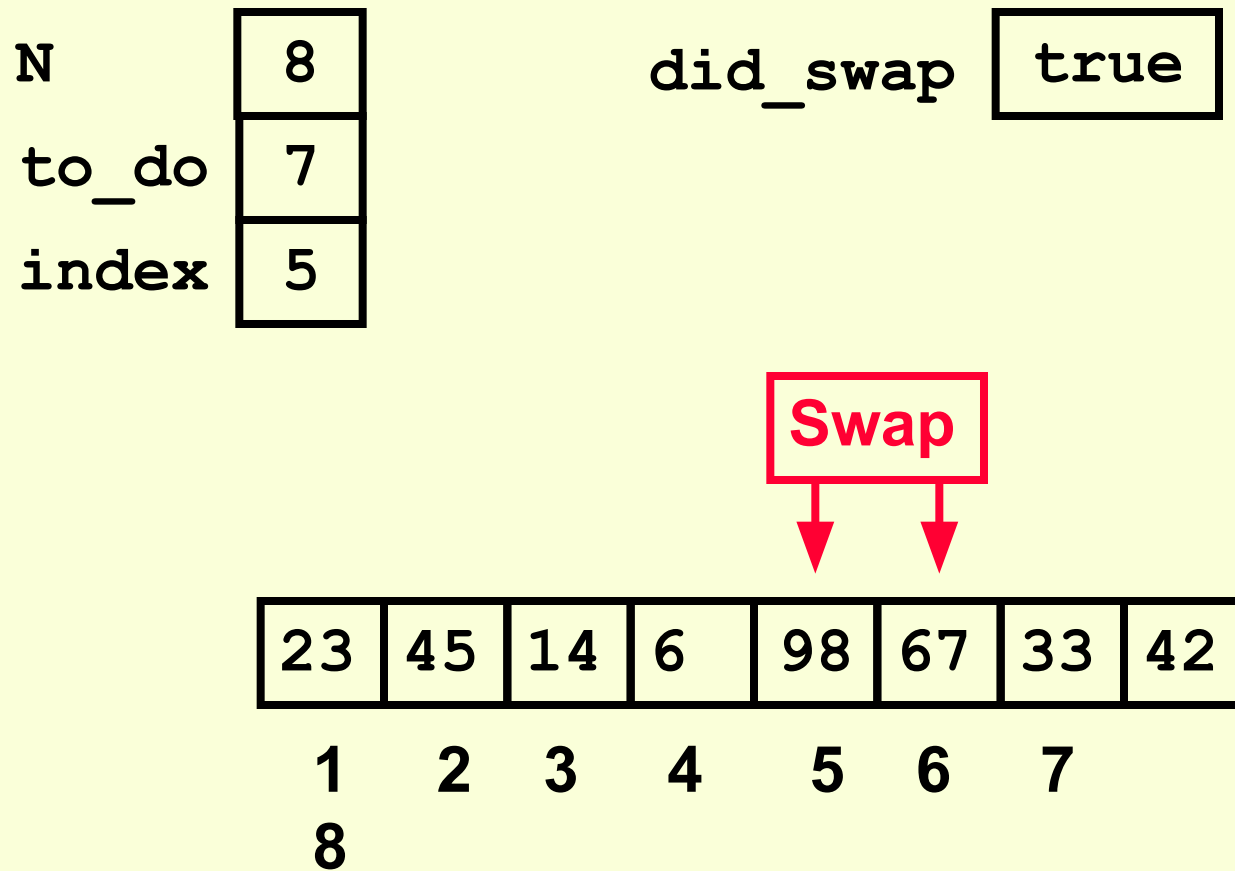
5

did_swap

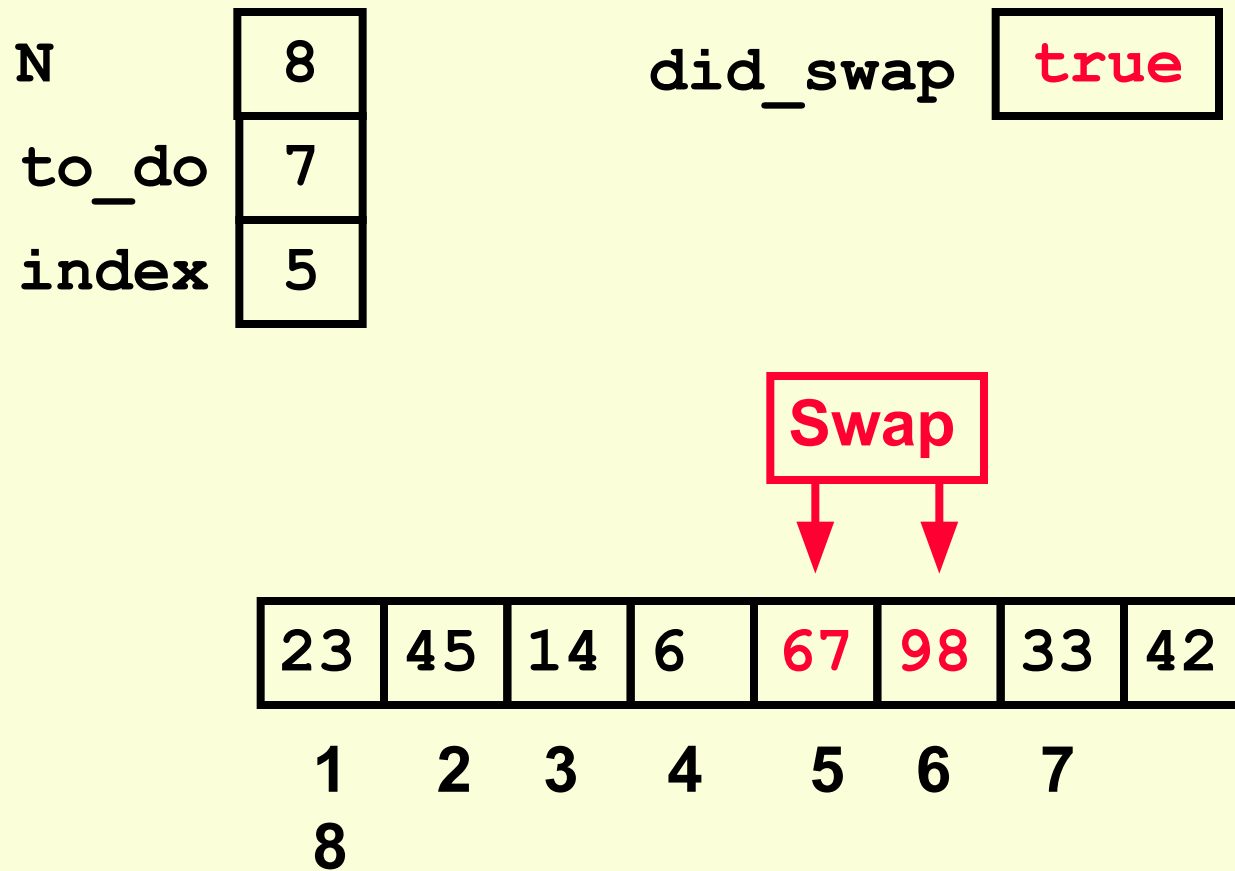
true



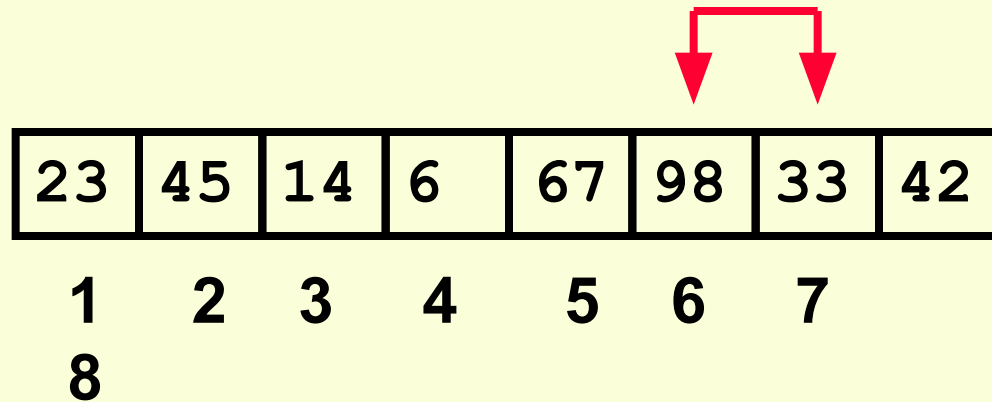
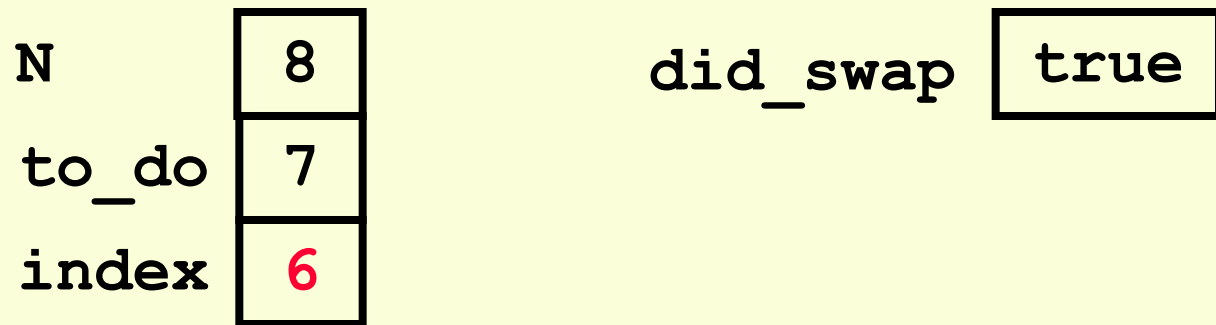
An Animated Example



An Animated Example



An Animated Example



An Animated Example

N	8
to_do	7
index	6

did_swap true

Swap



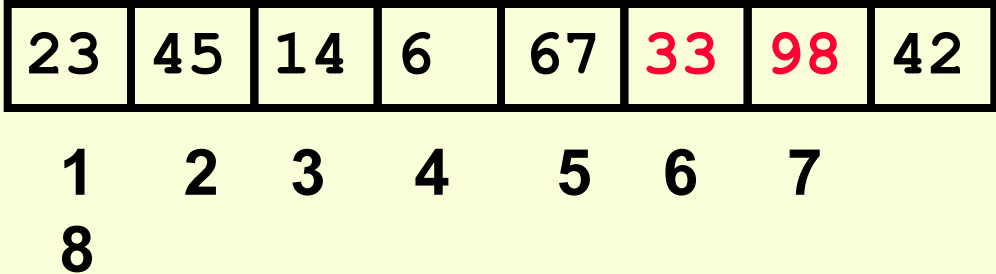
23	45	14	6	67	98	33	42
1	2	3	4	5	6	7	
8							

An Animated Example

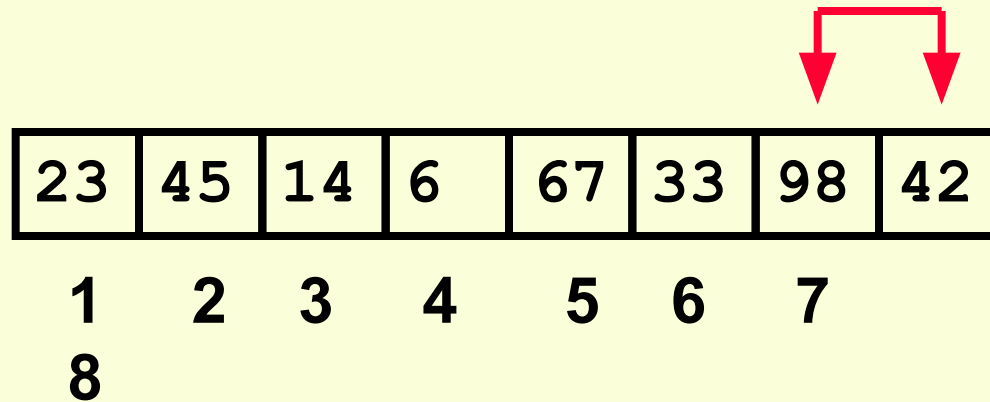
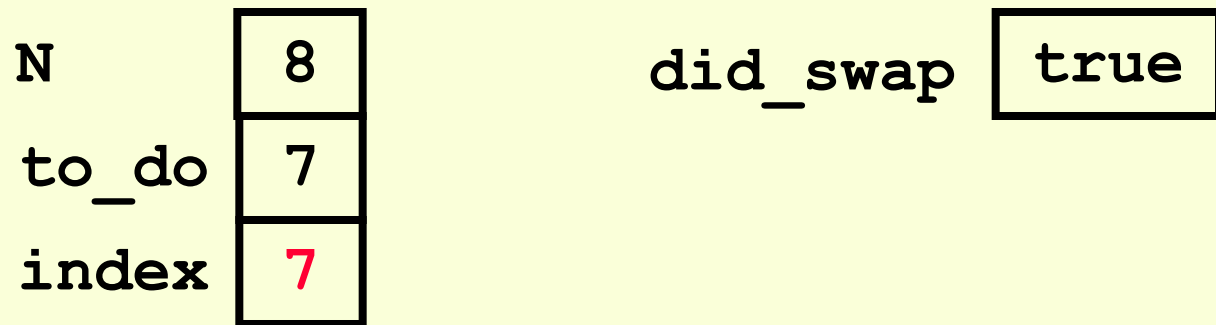
N	8
to_do	7
index	6

did_swap **true**

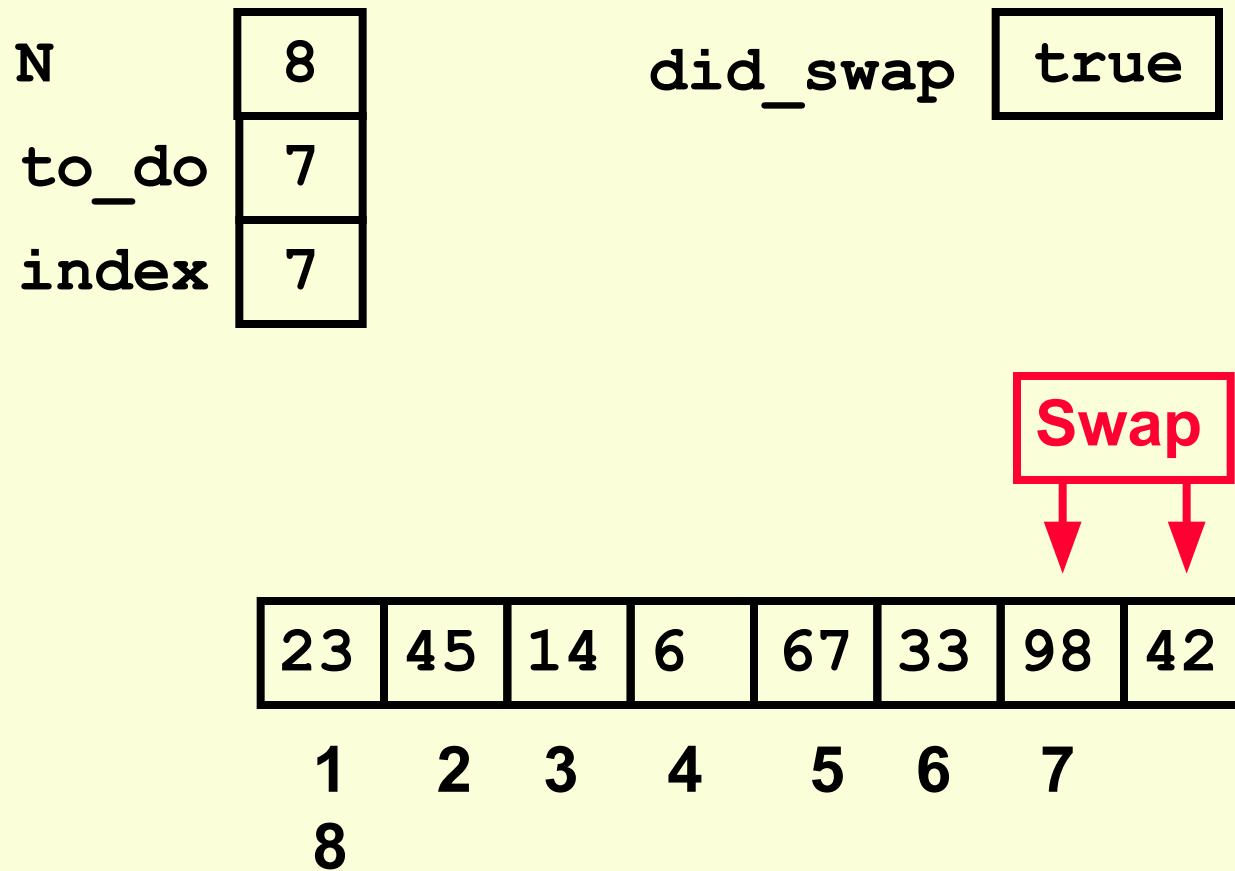
Swap



An Animated Example



An Animated Example



An Animated Example

N

8

to do

7

index

7

did_swap

true

Swap

23

45

14

6

67

33

42

98

1

2

3

4

5

6

7

8

After First Pass of Outer Loop

N

8

 did_swap

true

to_do

7

index

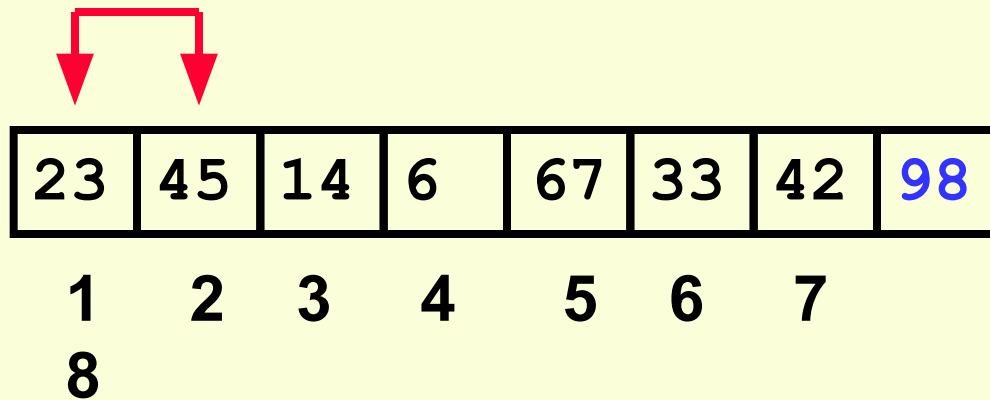
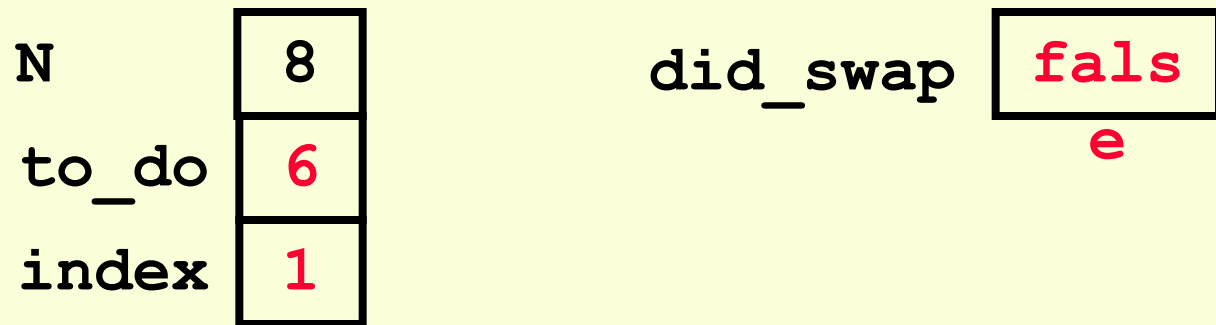
8

Finished first “Bubble Up”

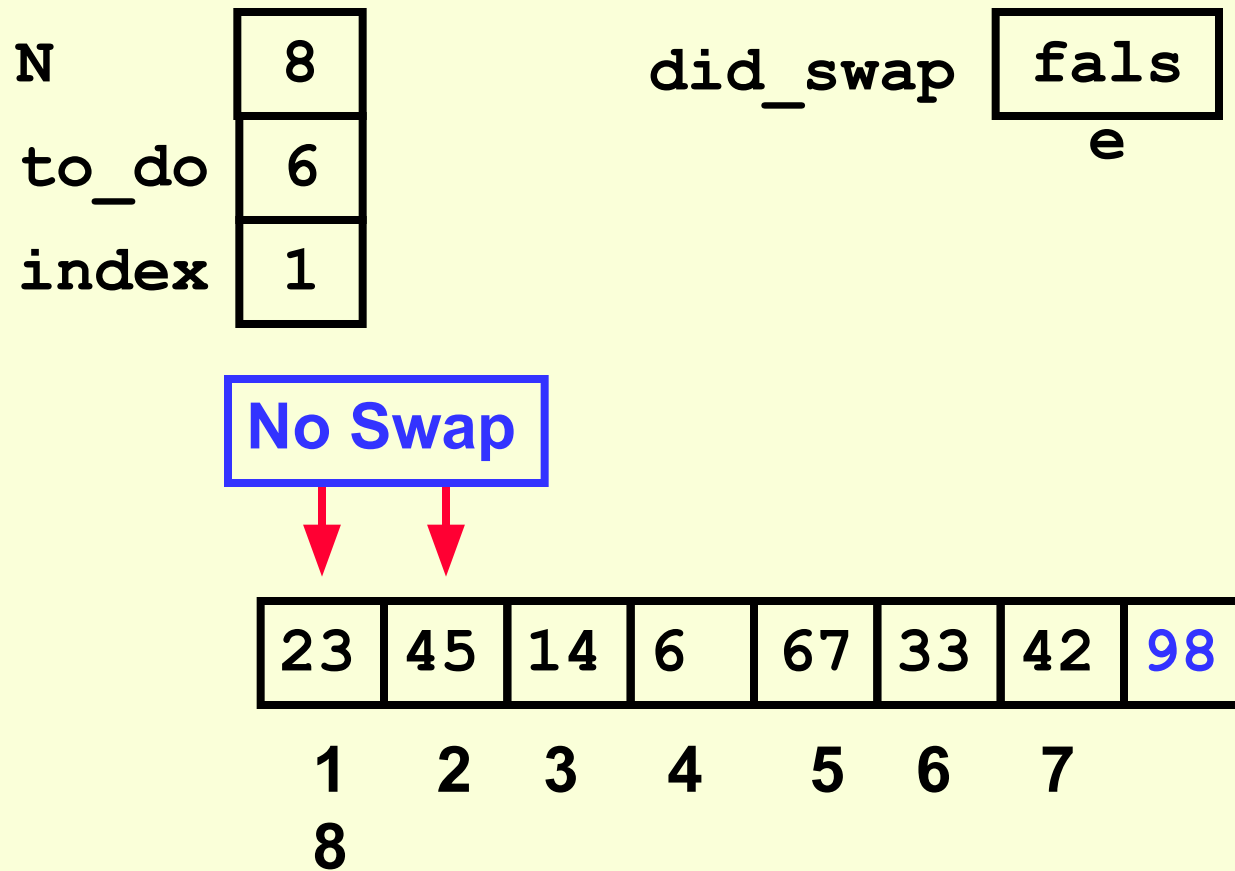
23	45	14	6	67	33	42	98
1	2	3	4	5	6	7	
8							



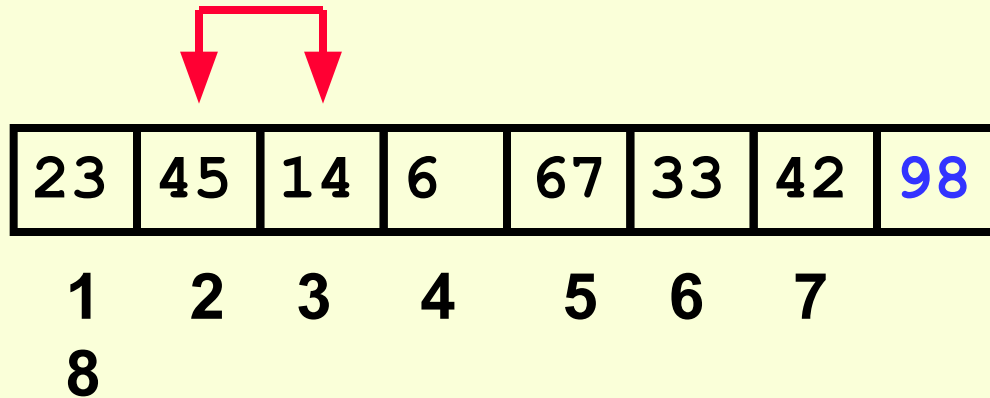
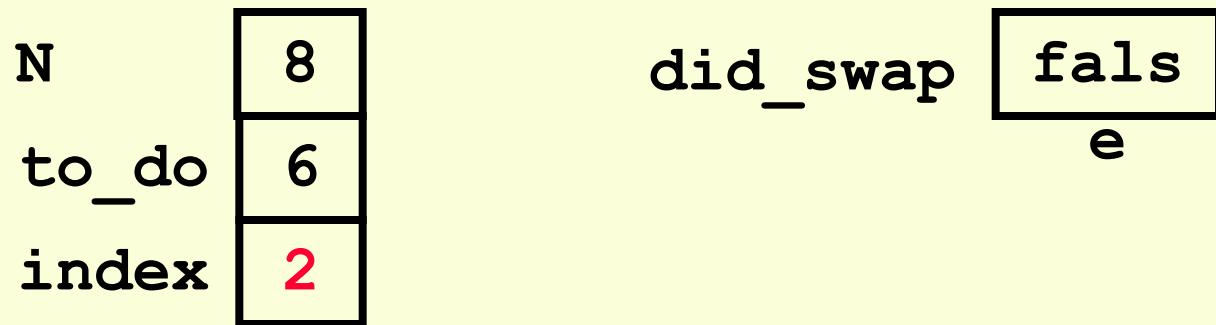
The Second “Bubble Up”



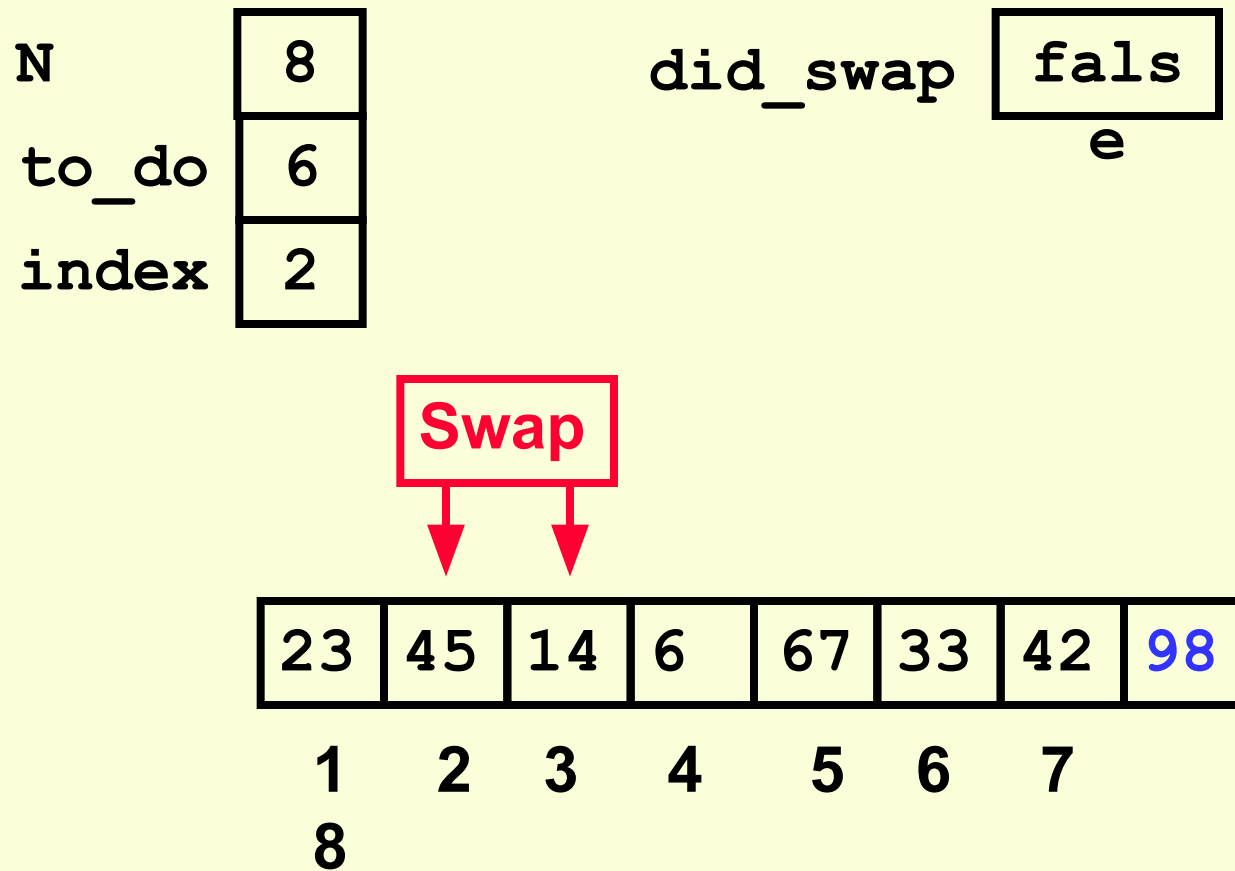
The Second “Bubble Up”



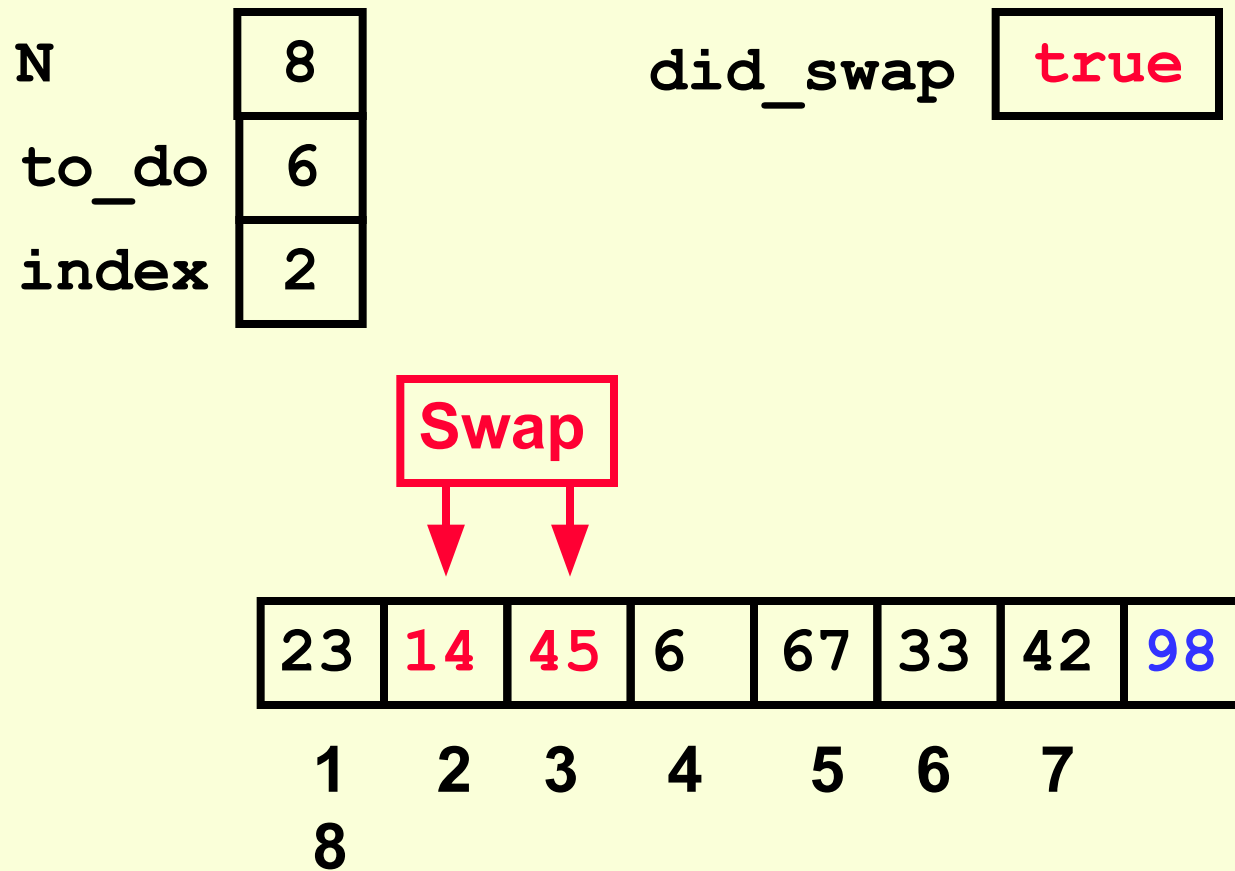
The Second “Bubble Up”



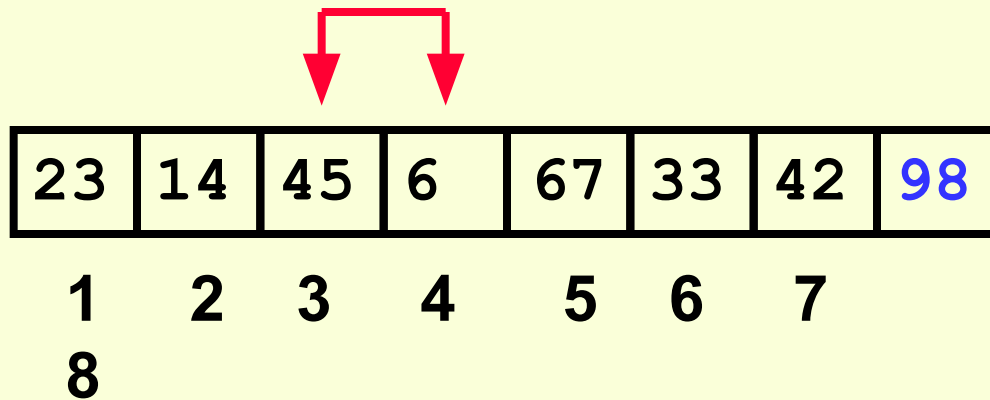
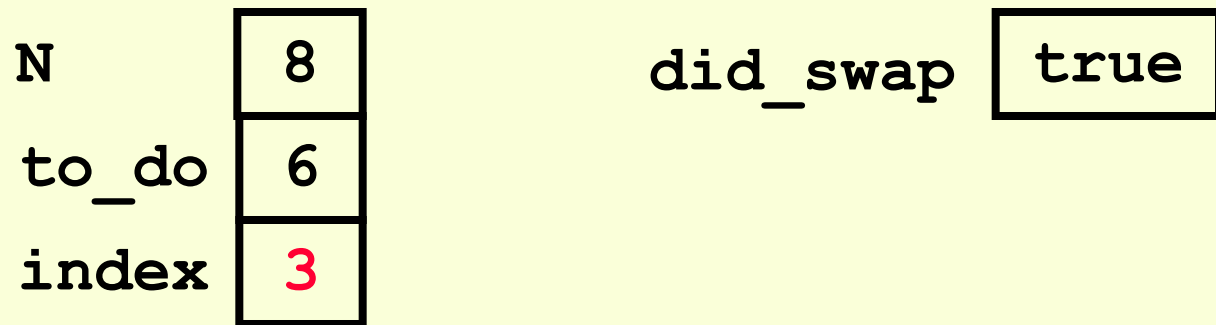
The Second “Bubble Up”



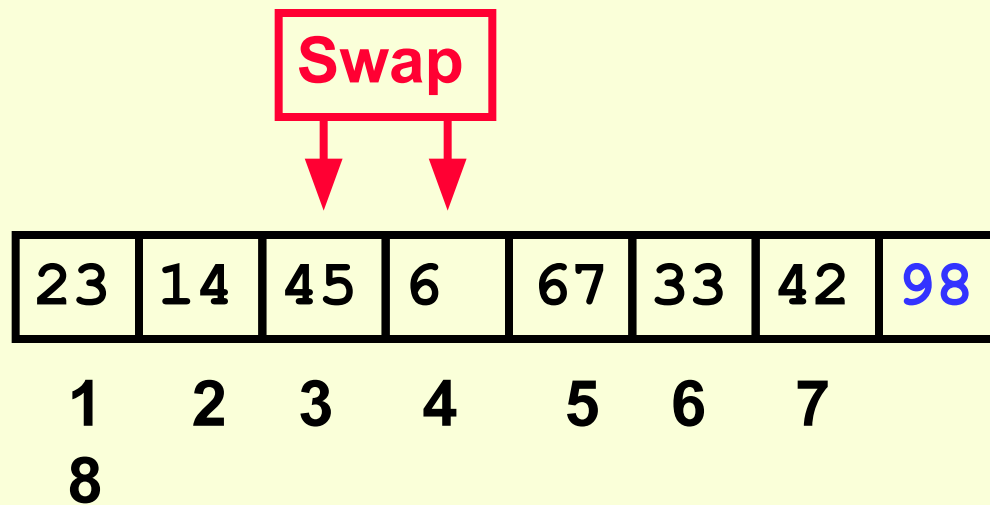
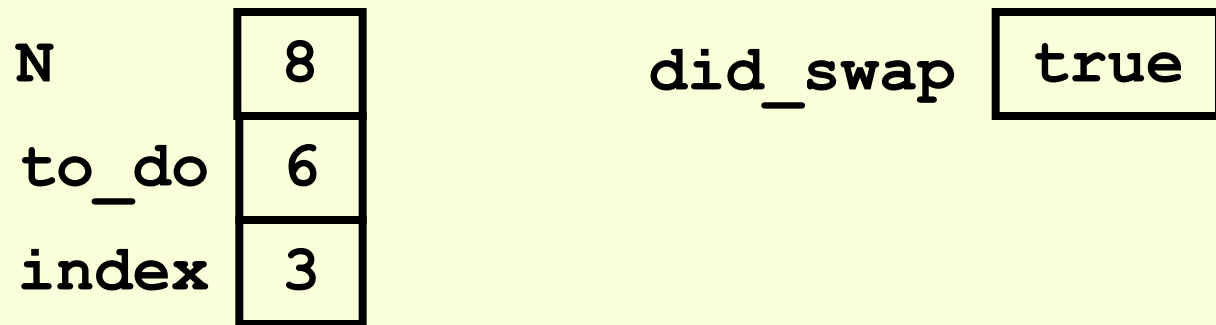
The Second “Bubble Up”



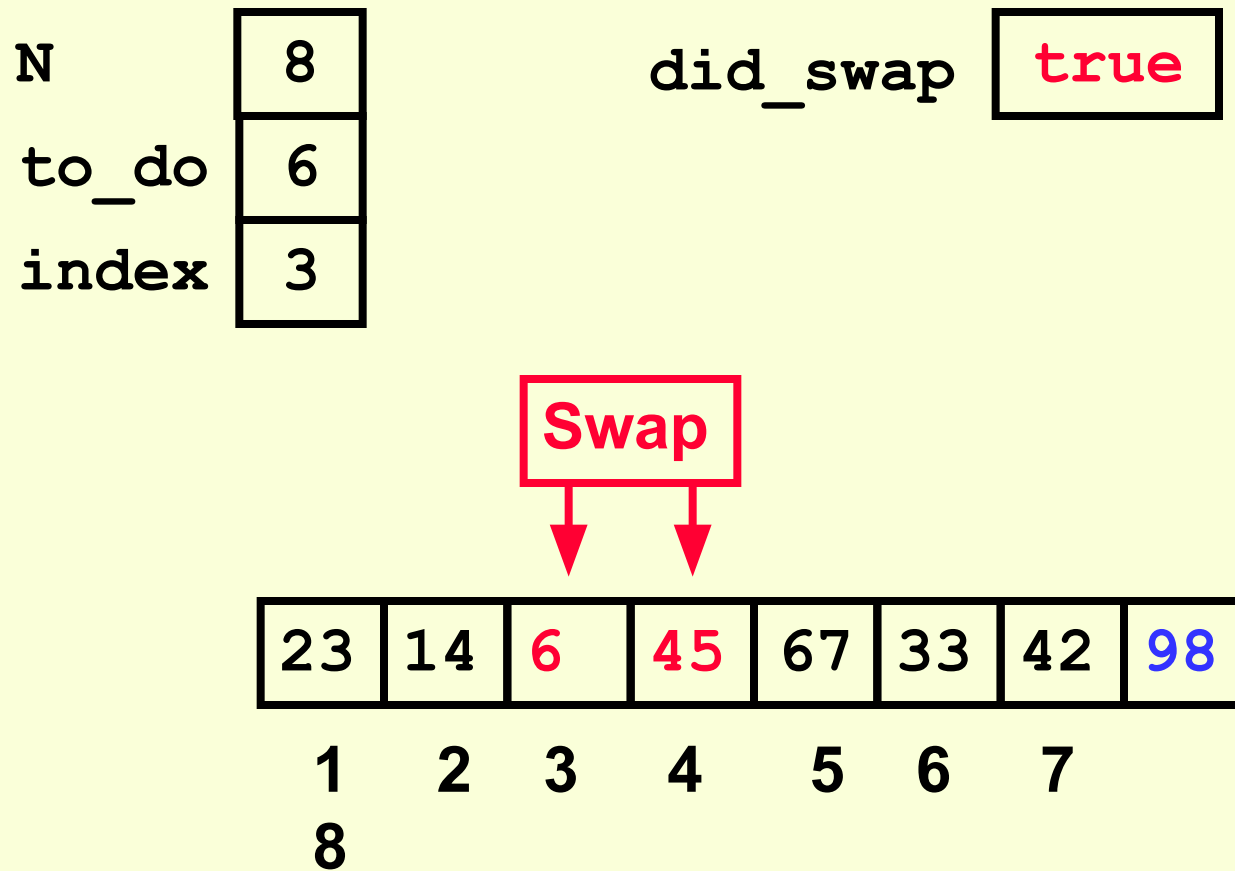
The Second “Bubble Up”



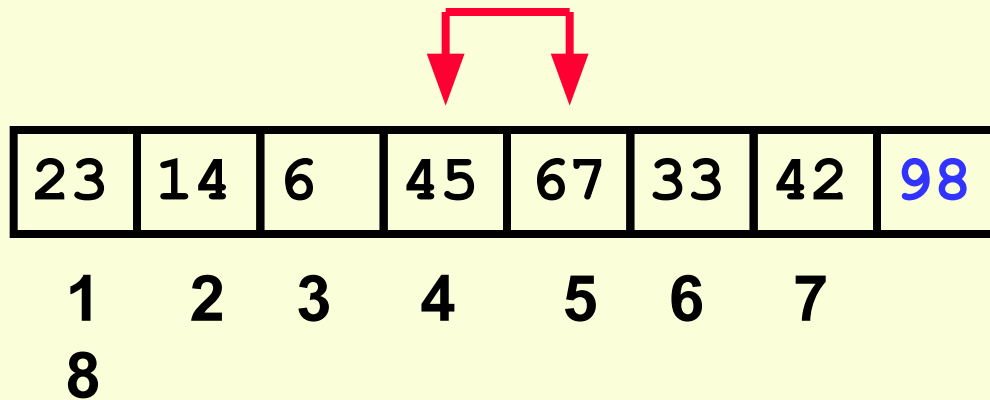
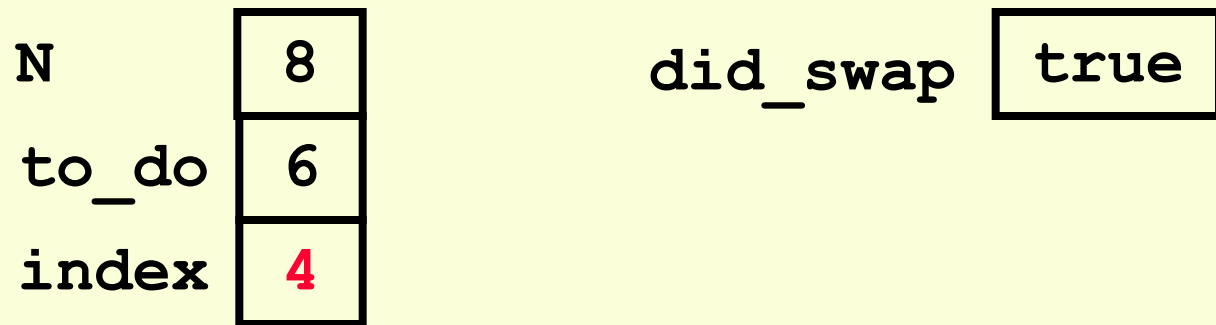
The Second “Bubble Up”



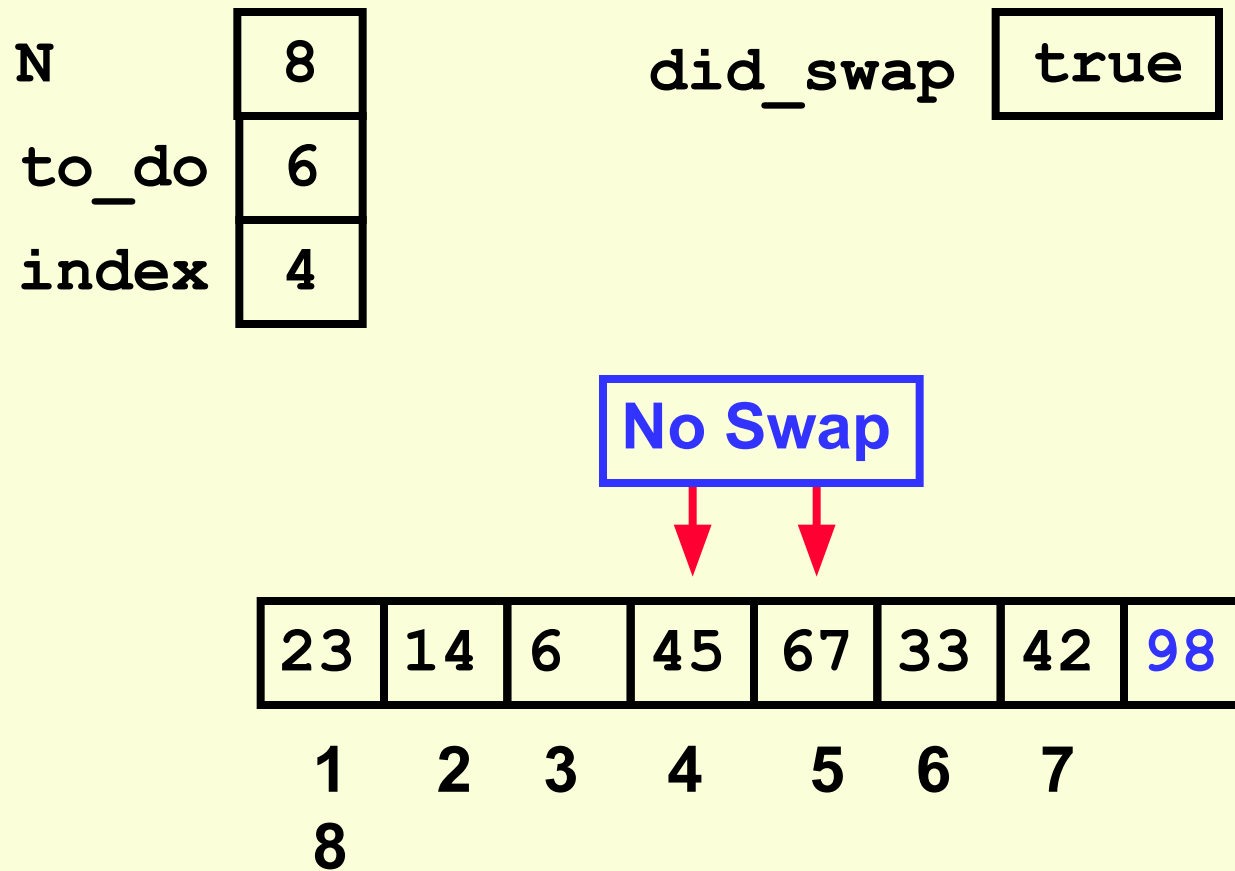
The Second “Bubble Up”



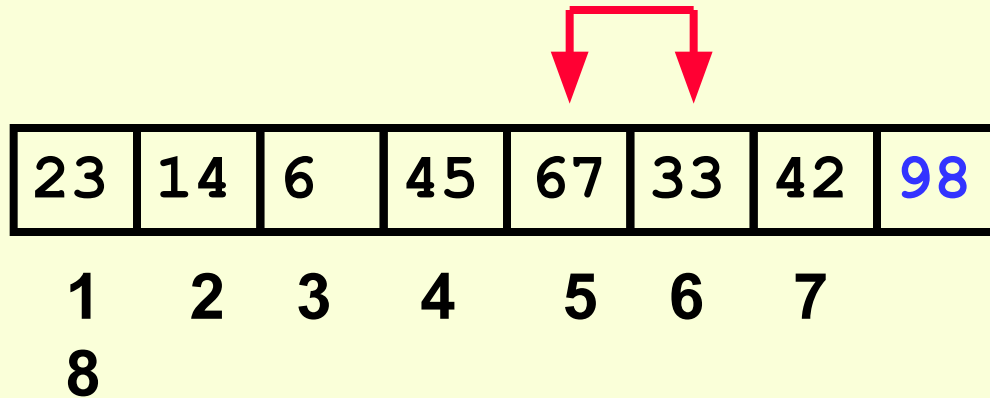
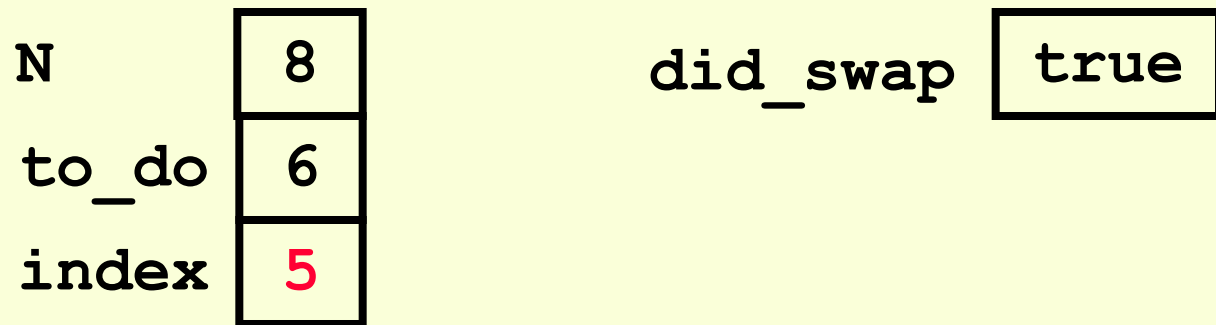
The Second “Bubble Up”



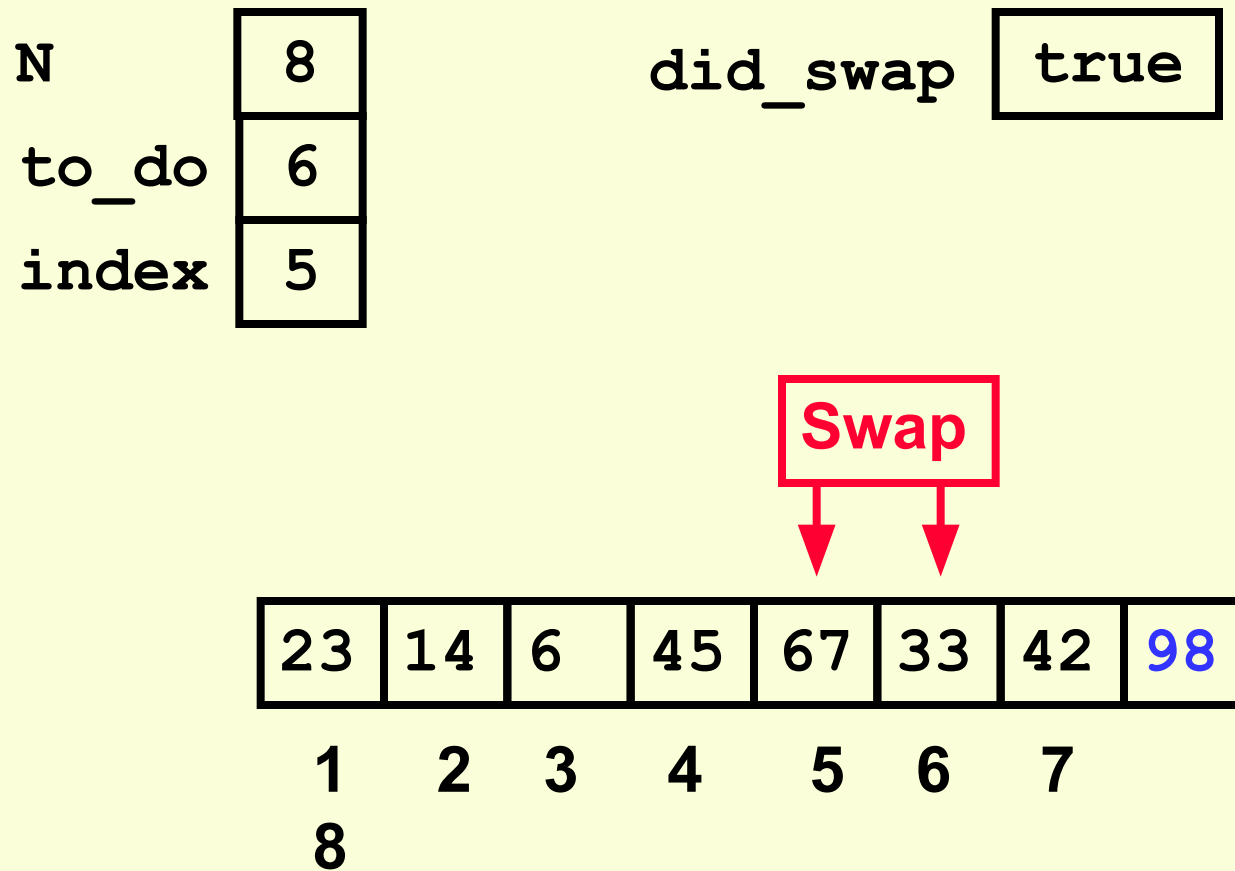
The Second “Bubble Up”



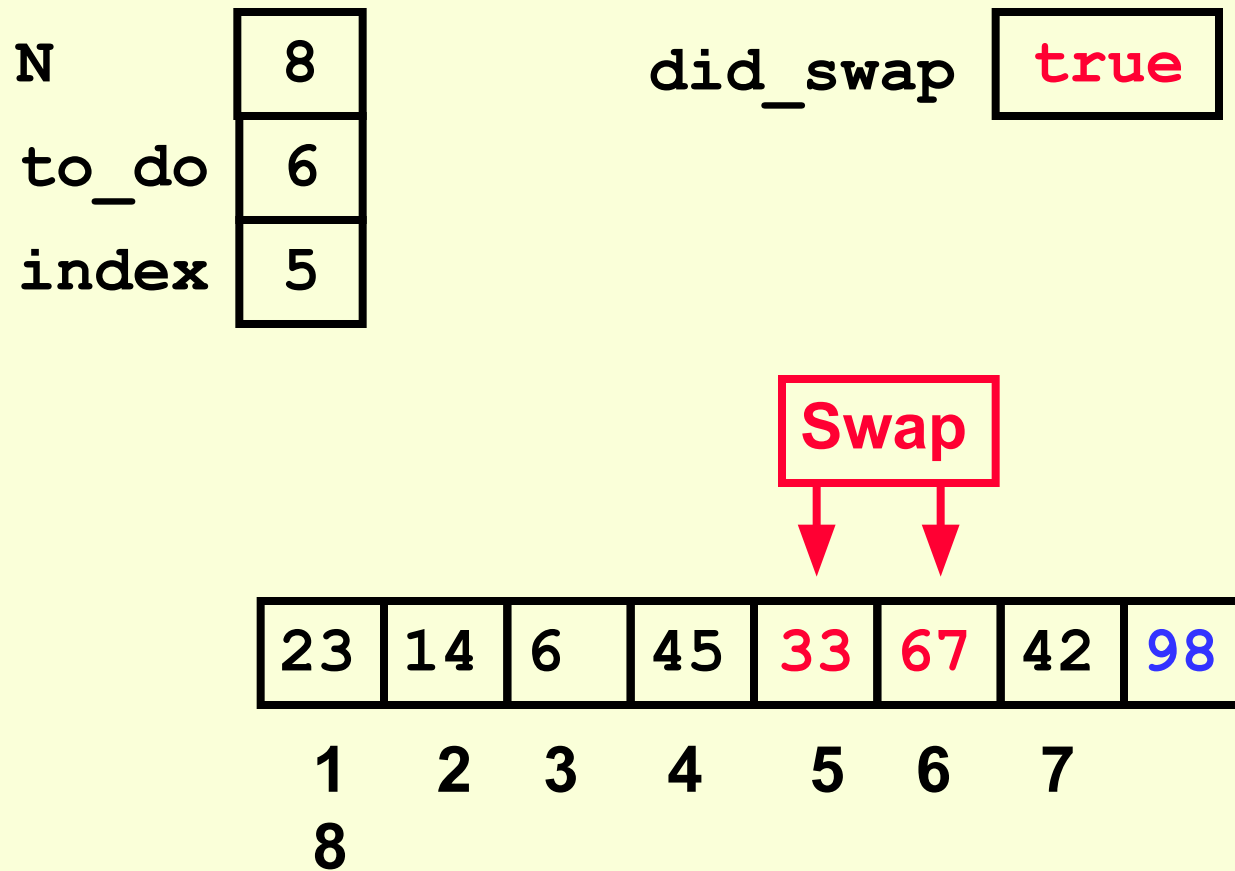
The Second “Bubble Up”



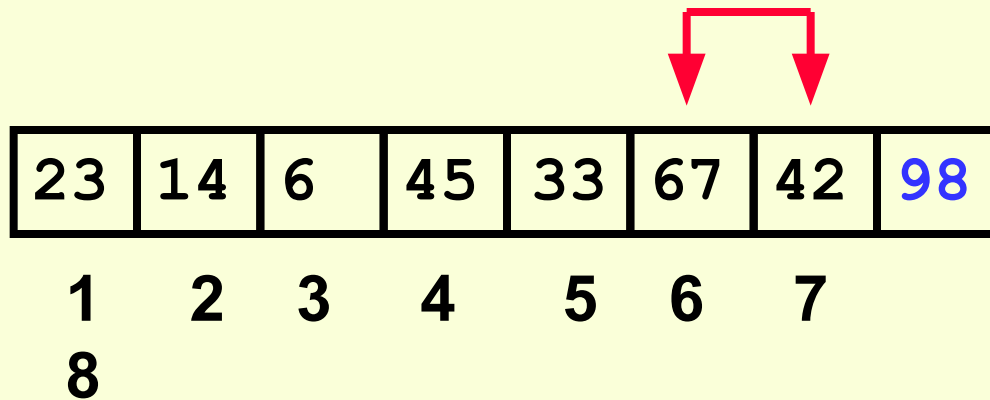
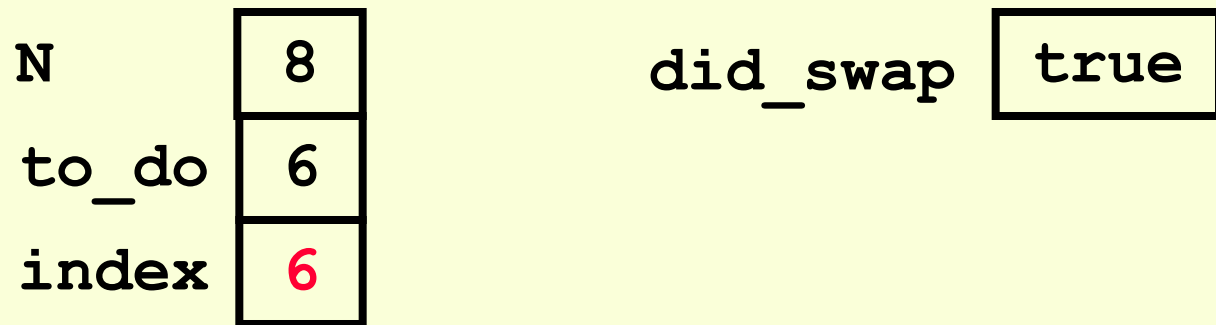
The Second “Bubble Up”



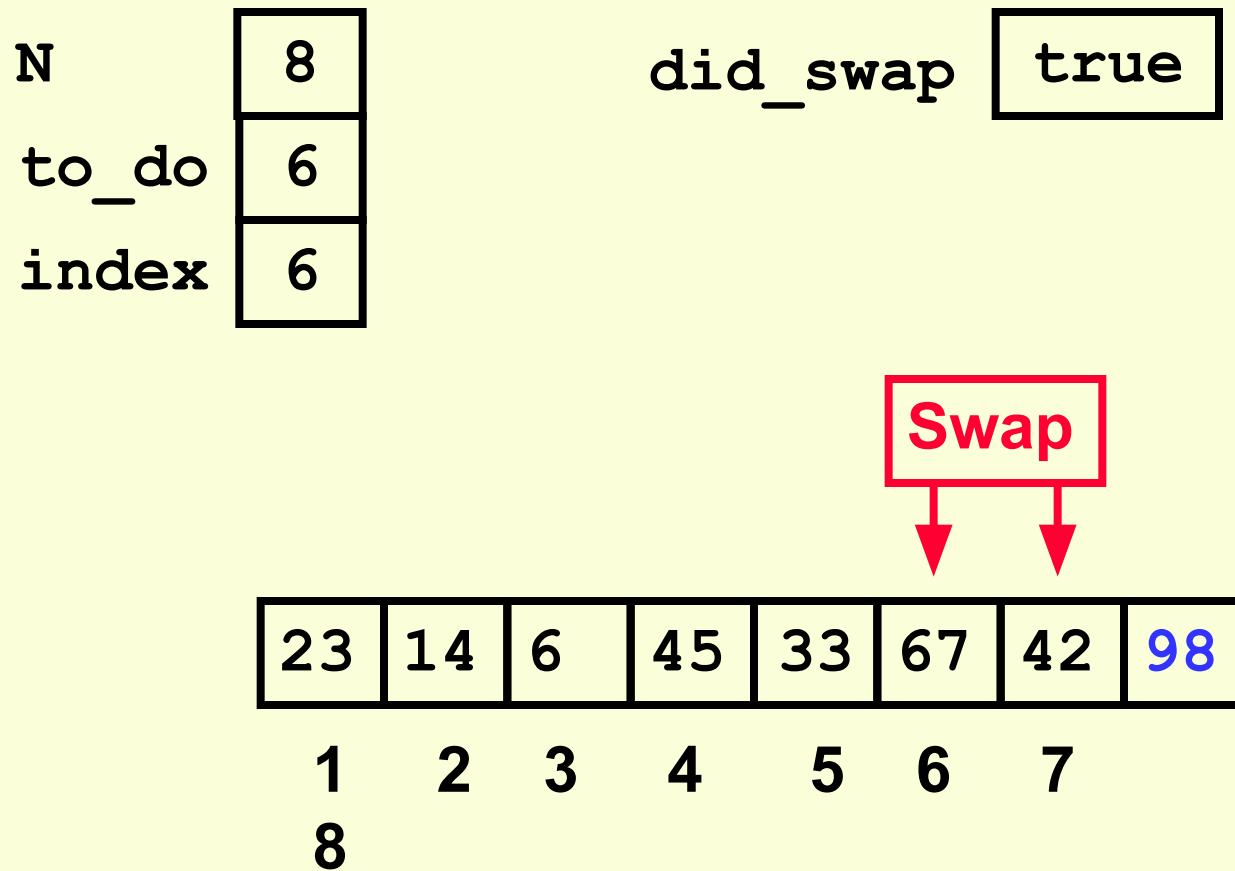
The Second “Bubble Up”



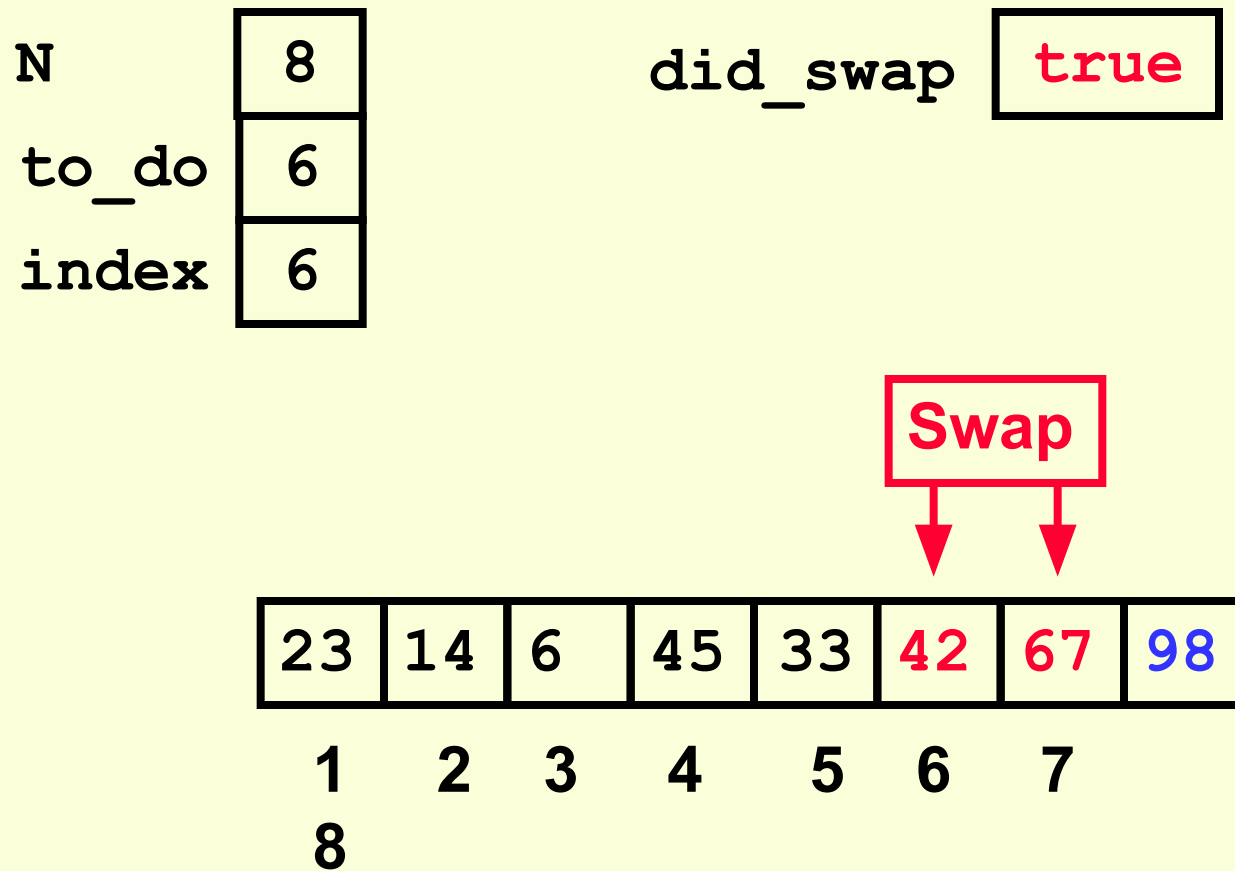
The Second “Bubble Up”



The Second “Bubble Up”



The Second “Bubble Up”



After Second Pass of Outer Loop

N

8
6
7

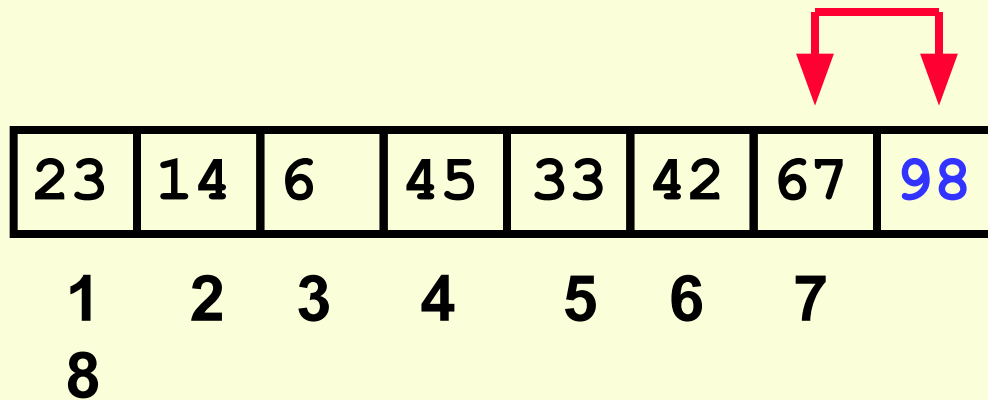
to_do

index

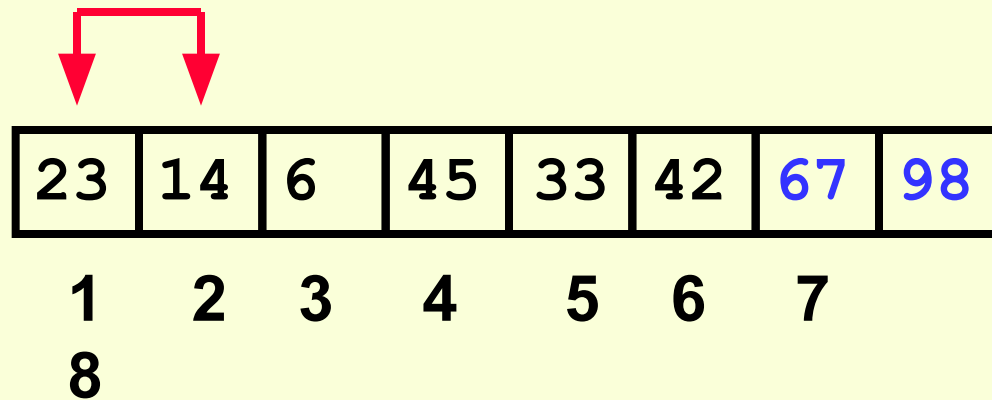
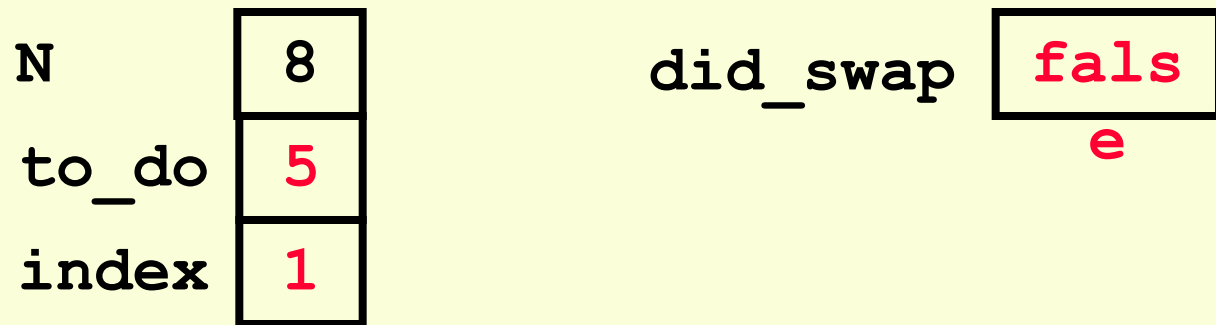
did_swap

true

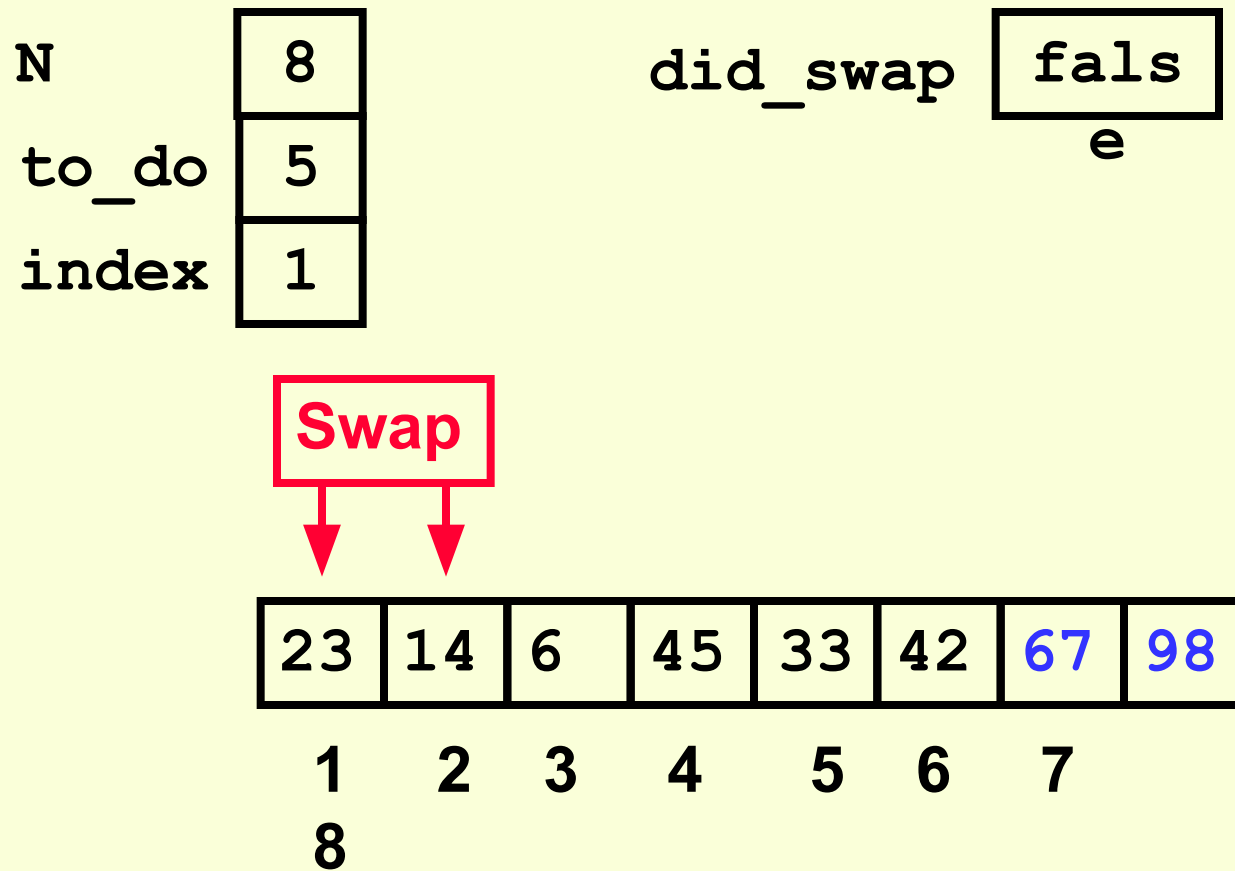
Finished second "Bubble Up"



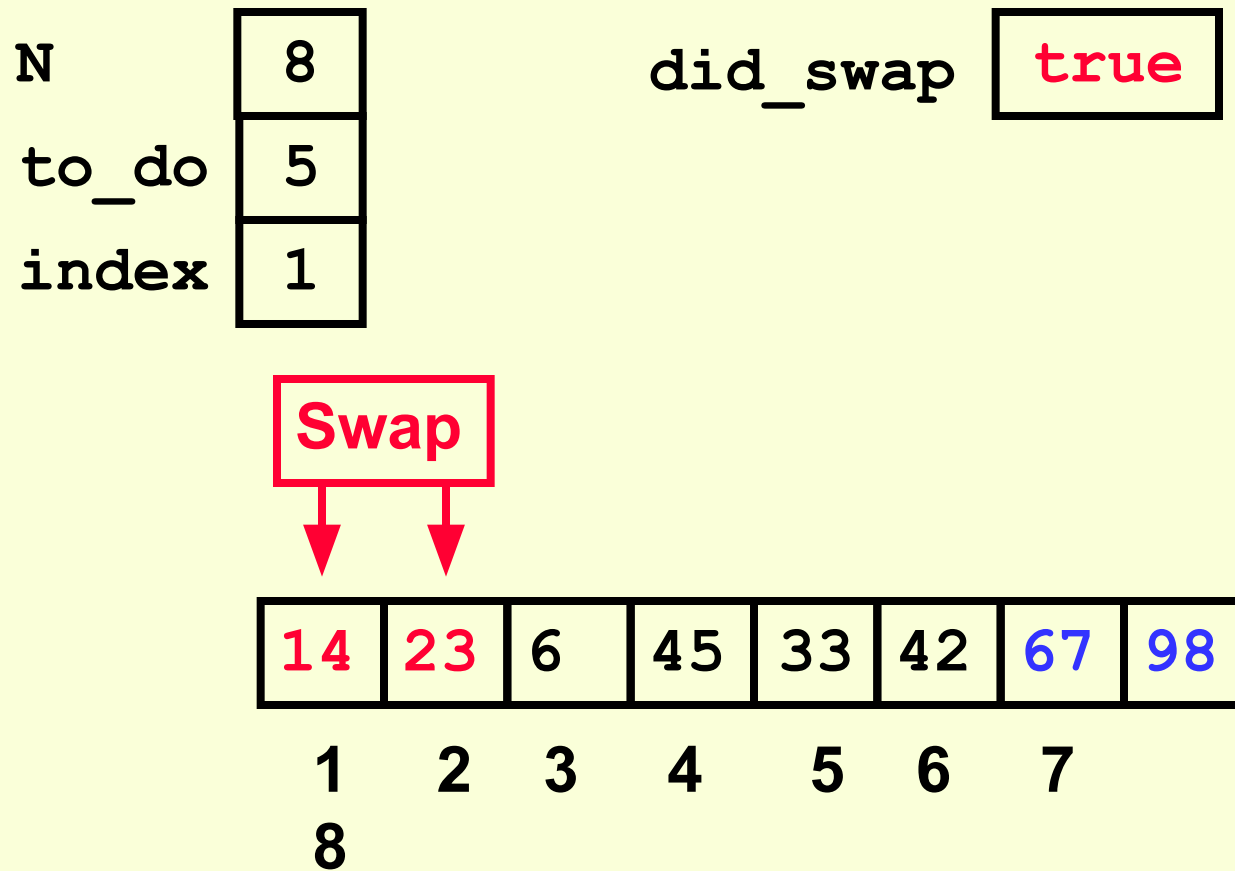
The Third “Bubble Up”



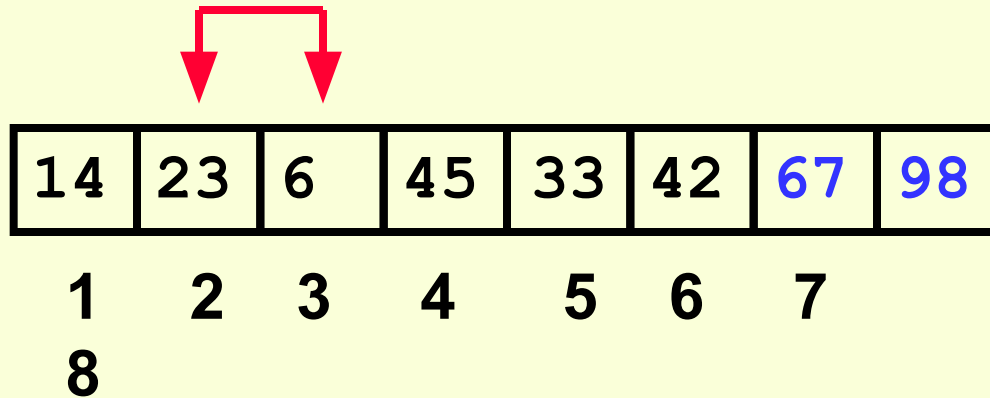
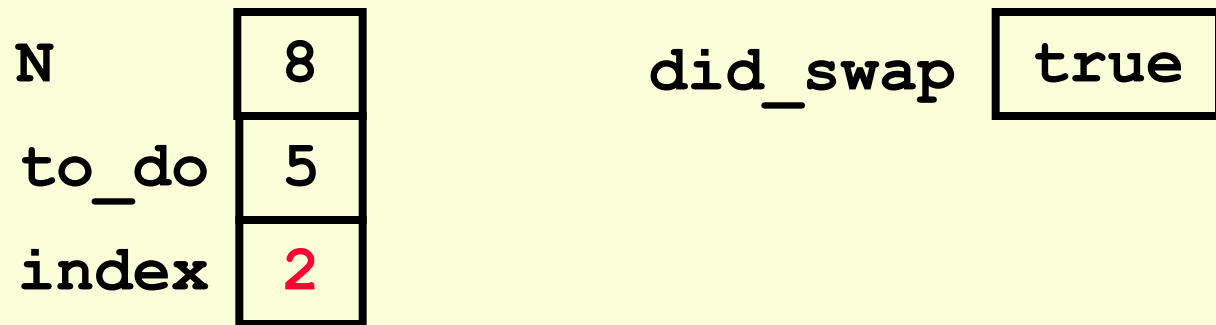
The Third “Bubble Up”



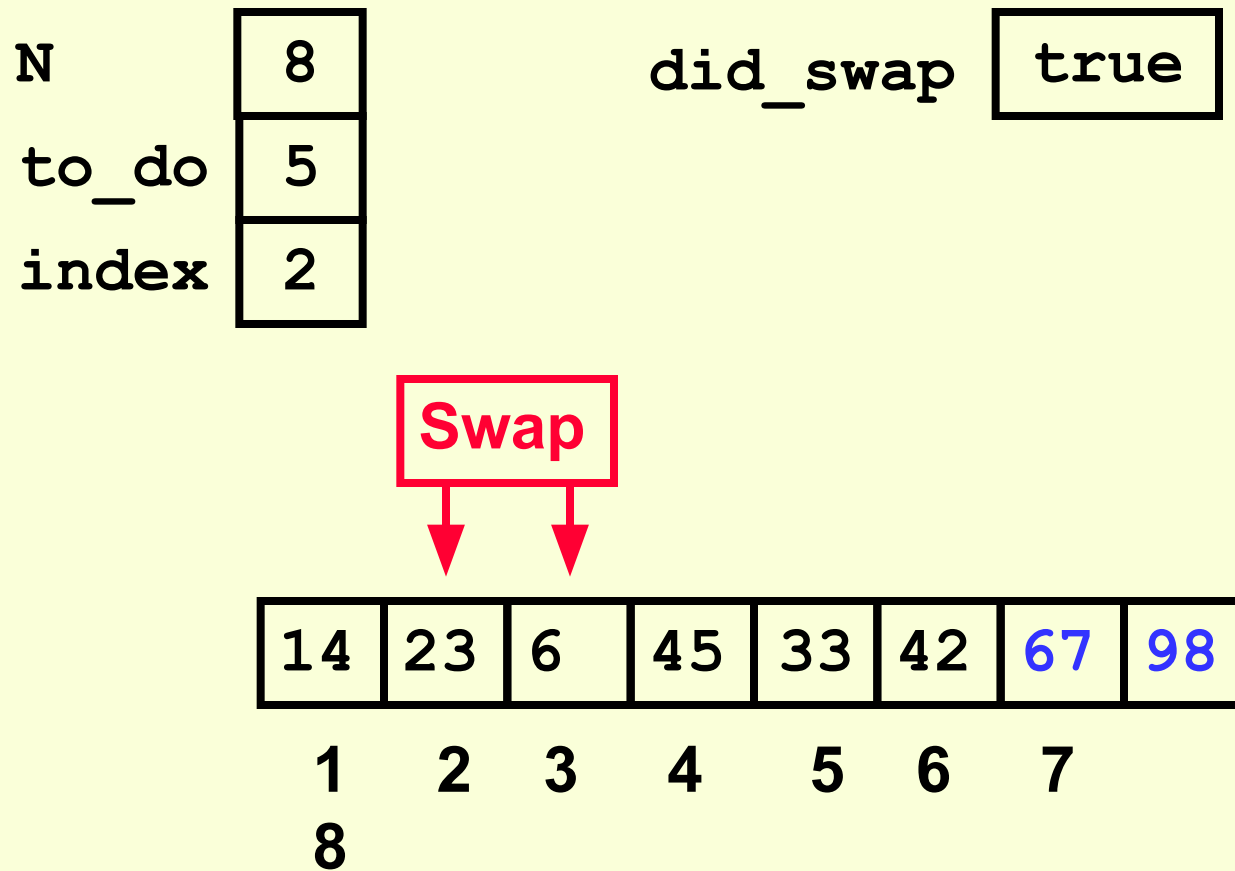
The Third “Bubble Up”



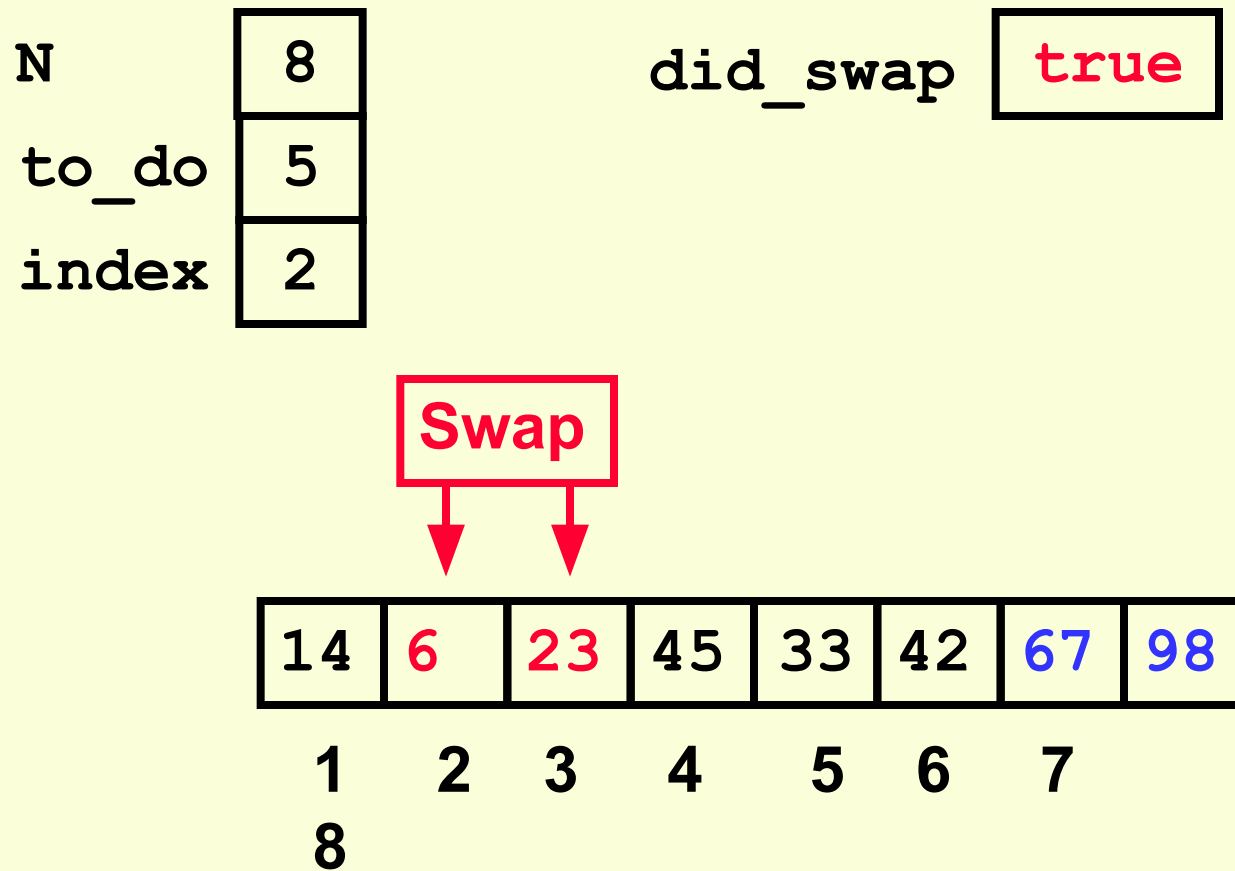
The Third “Bubble Up”



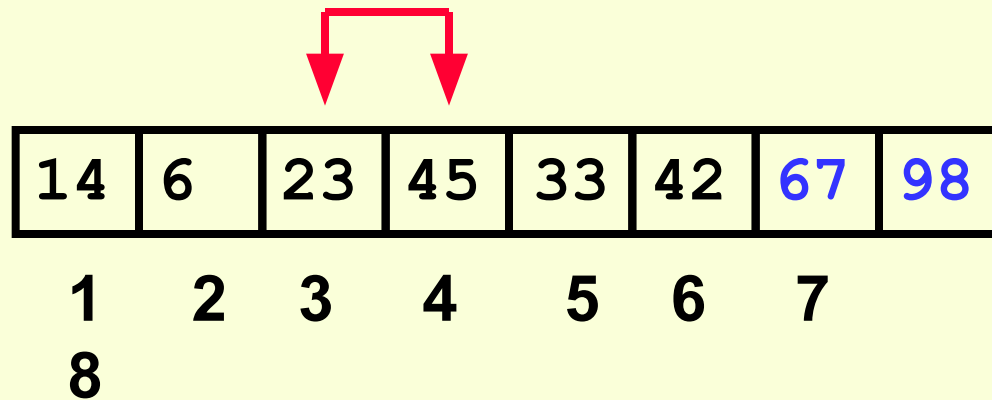
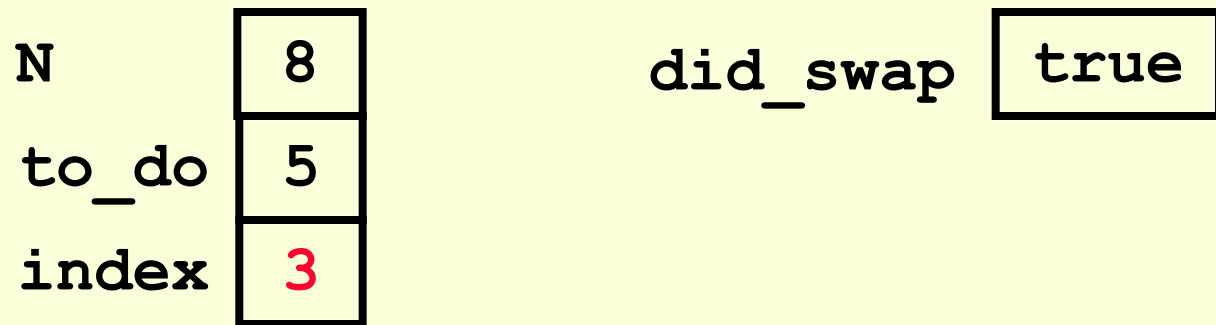
The Third “Bubble Up”



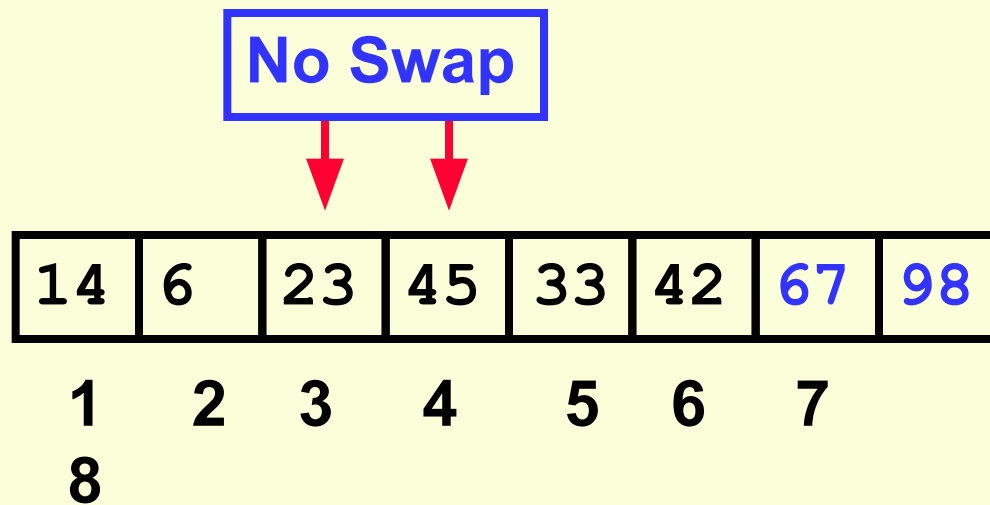
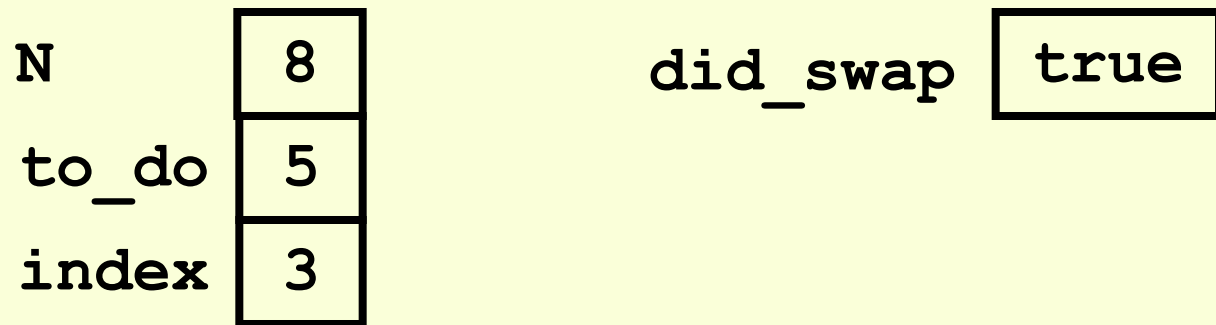
The Third “Bubble Up”



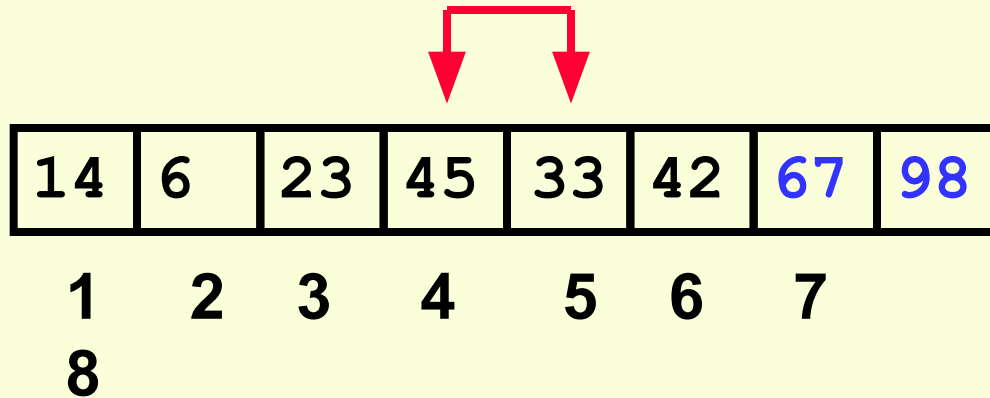
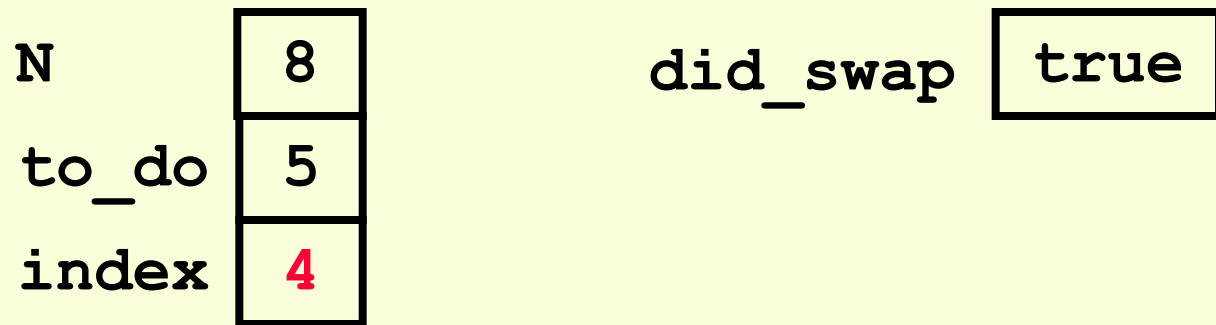
The Third “Bubble Up”



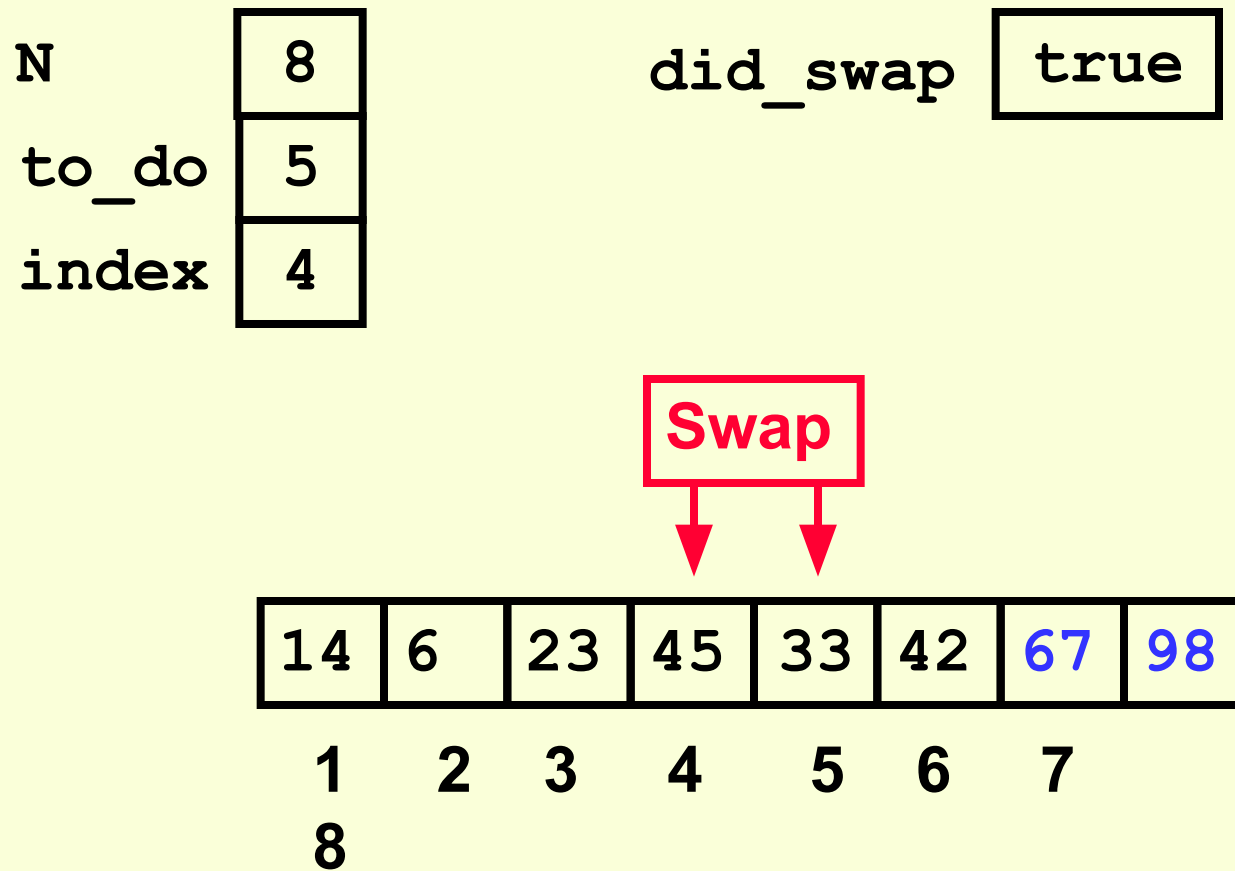
The Third “Bubble Up”



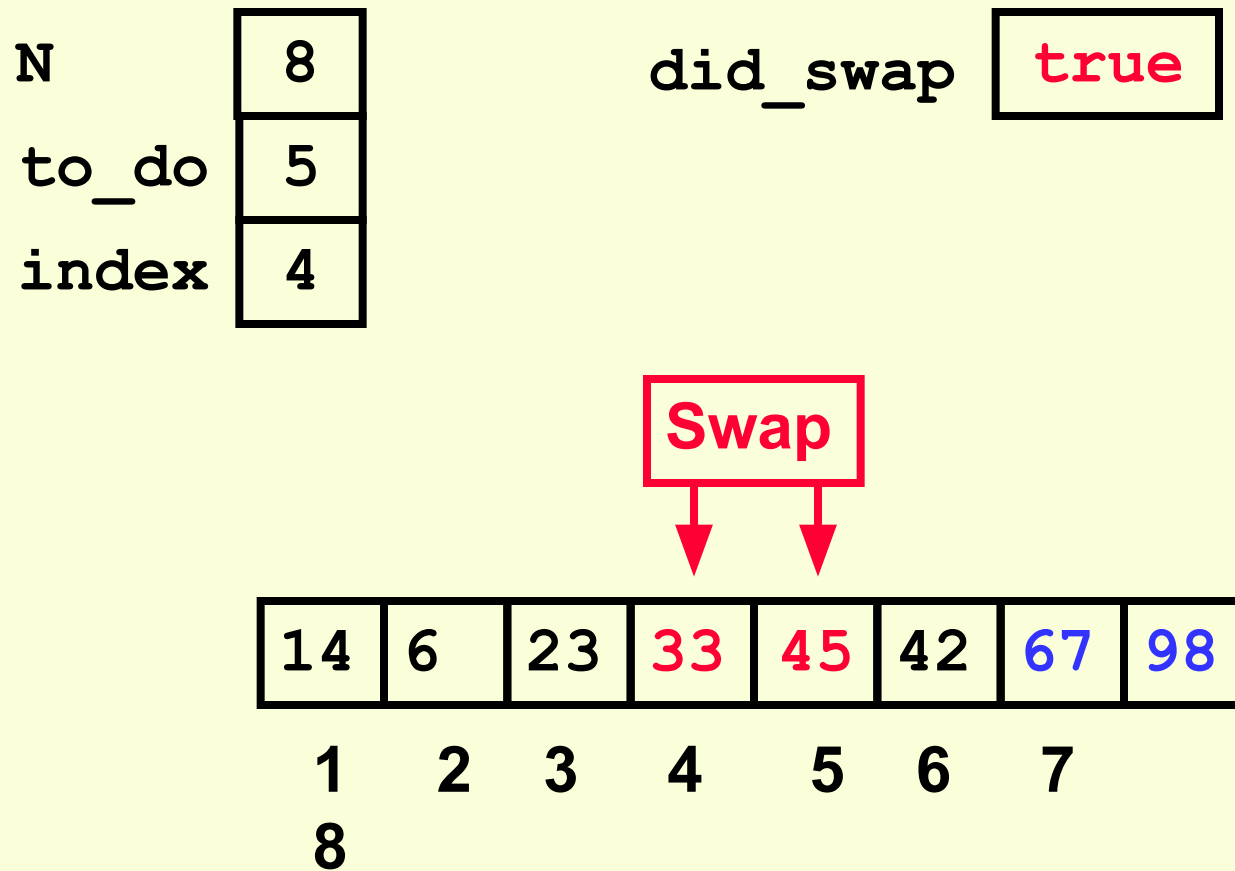
The Third “Bubble Up”



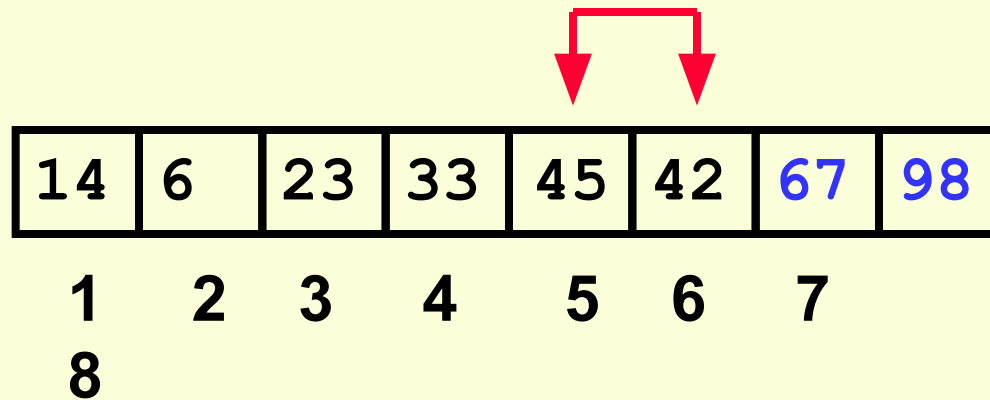
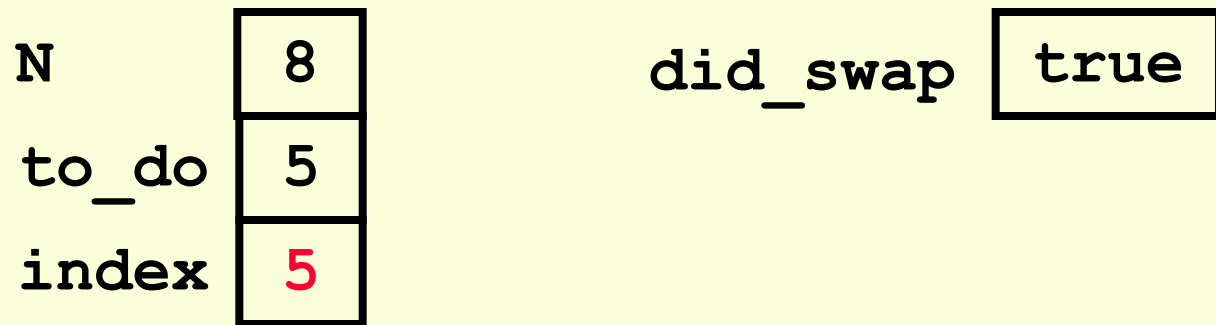
The Third “Bubble Up”



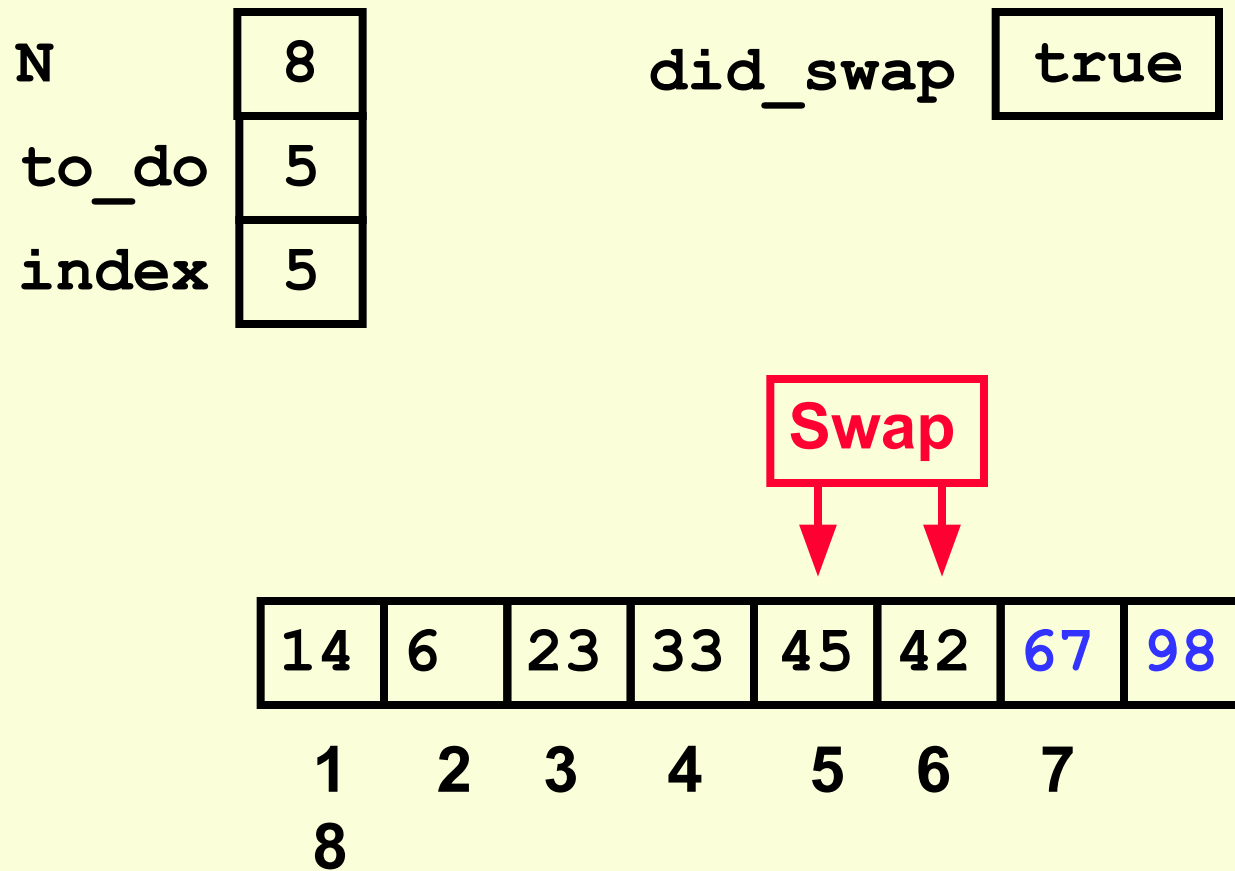
The Third “Bubble Up”



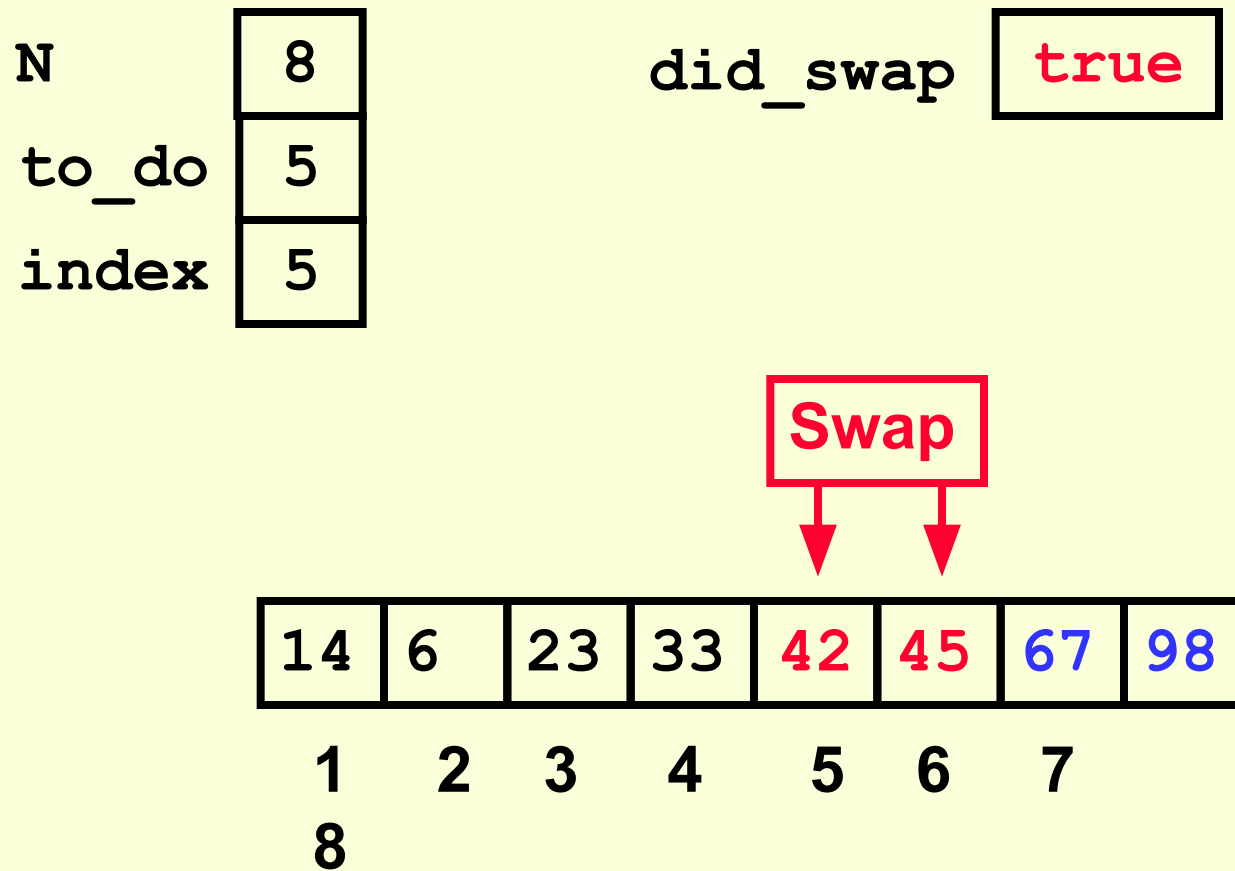
The Third “Bubble Up”



The Third “Bubble Up”



The Third “Bubble Up”



After Third Pass of Outer Loop

N

8
5
6

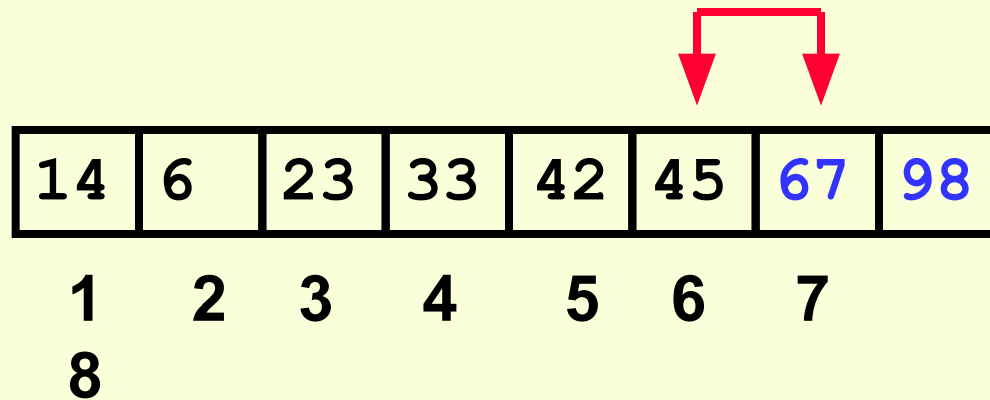
to_do

index

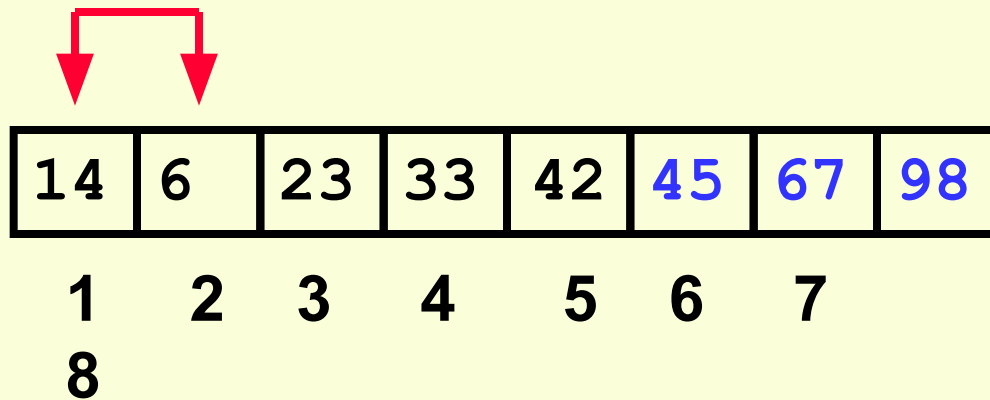
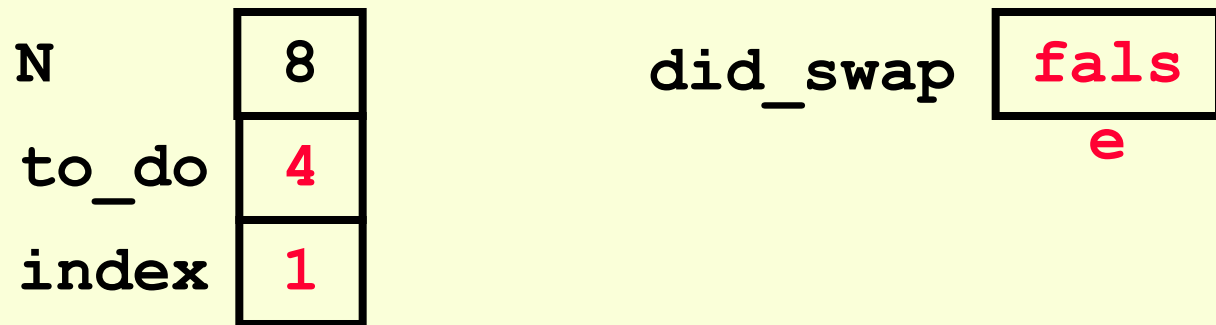
did_swap

true

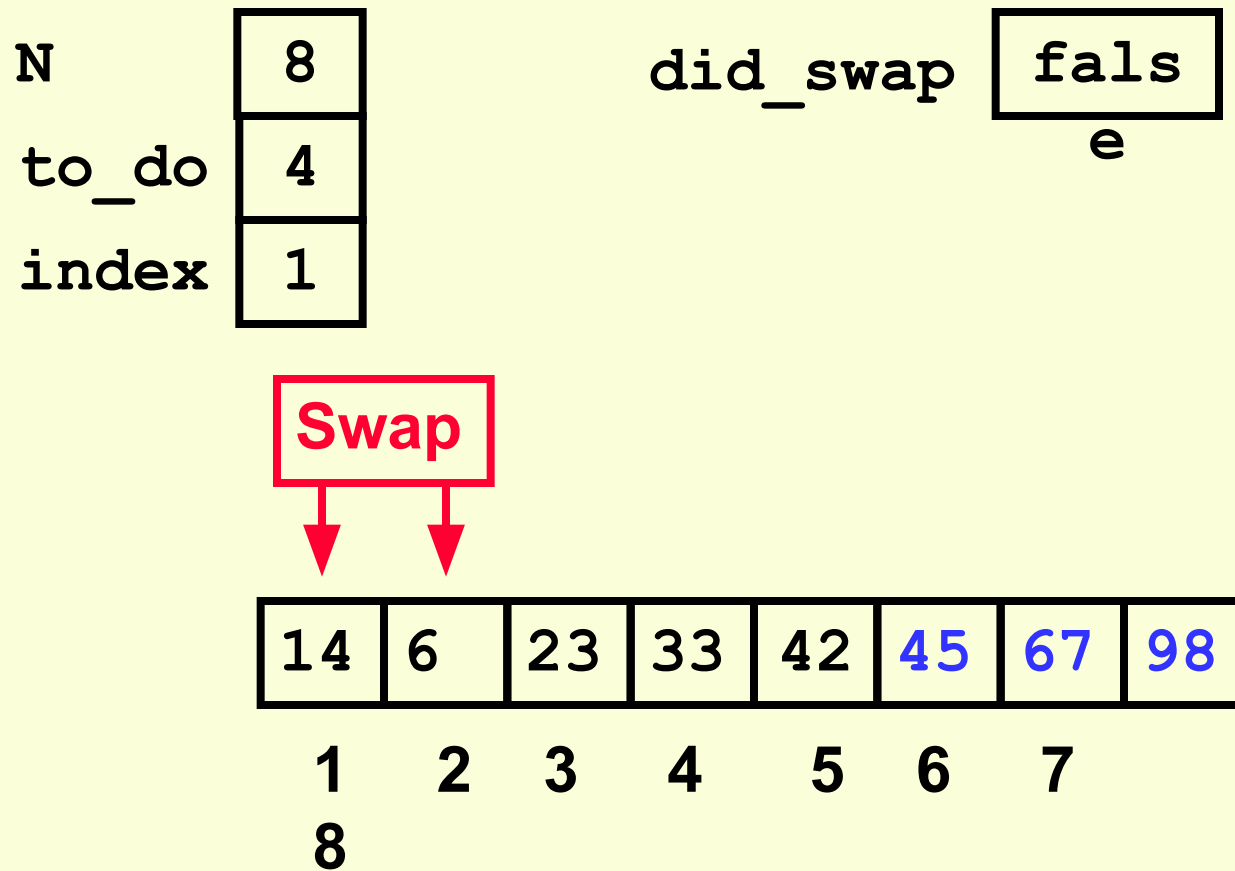
Finished third “Bubble Up”



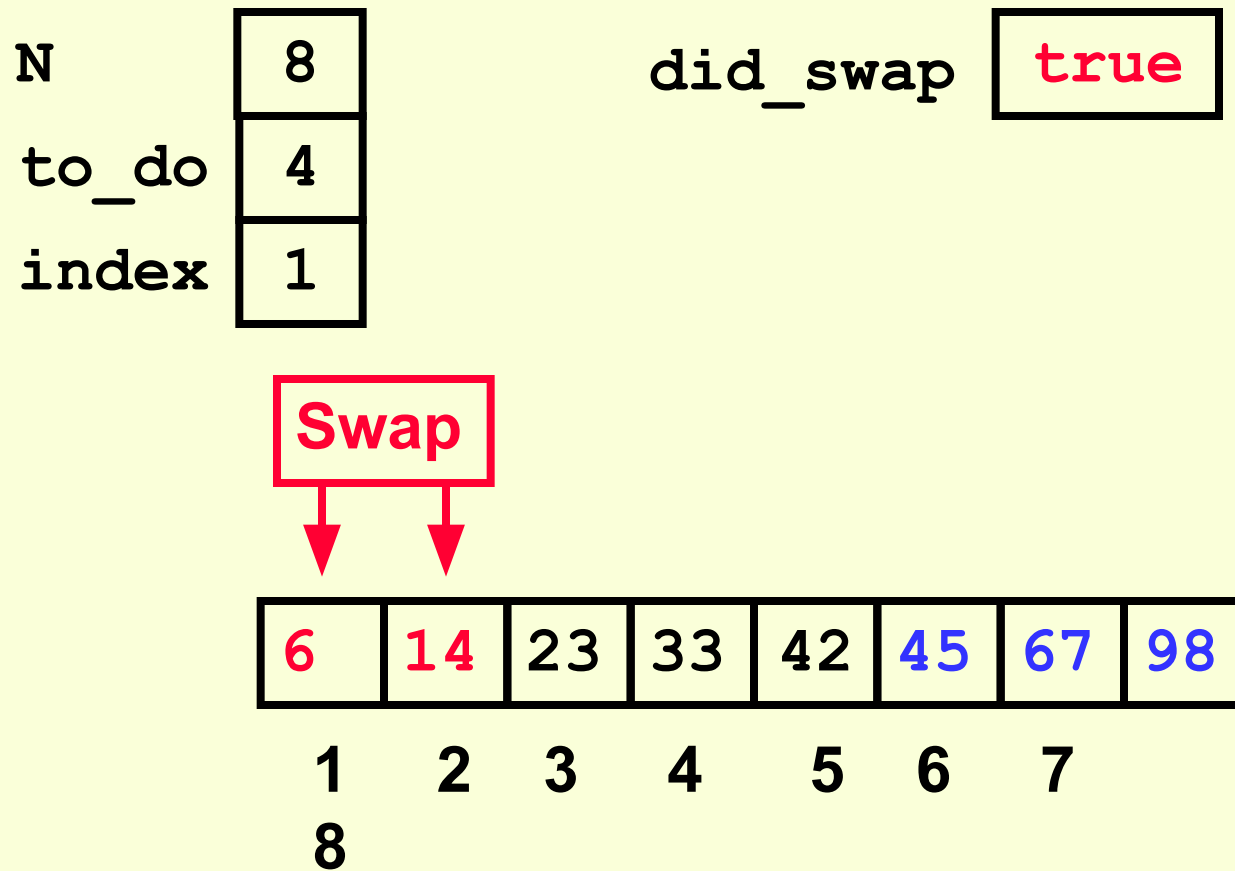
The Fourth “Bubble Up”



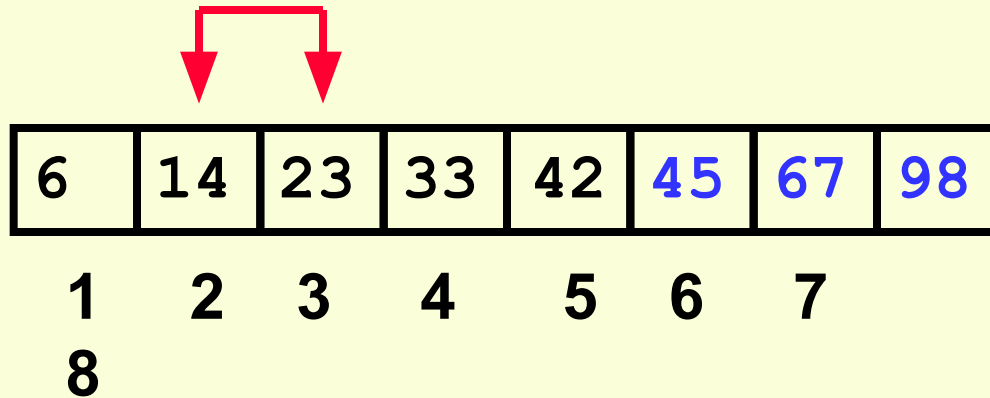
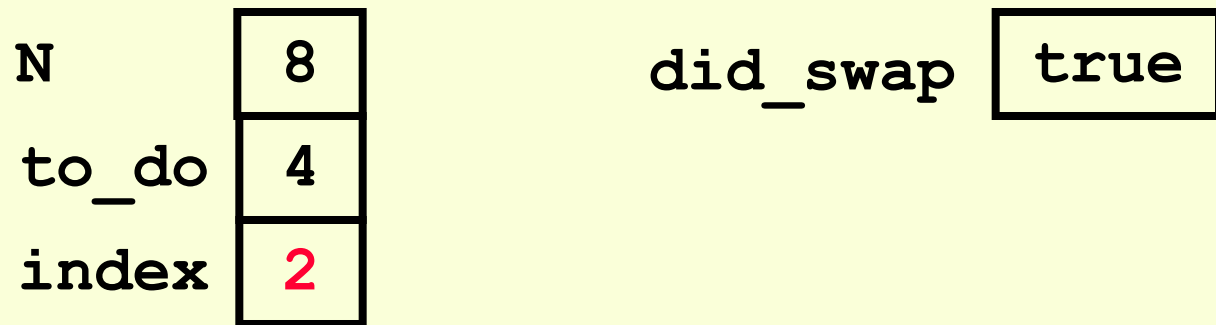
The Fourth “Bubble Up”



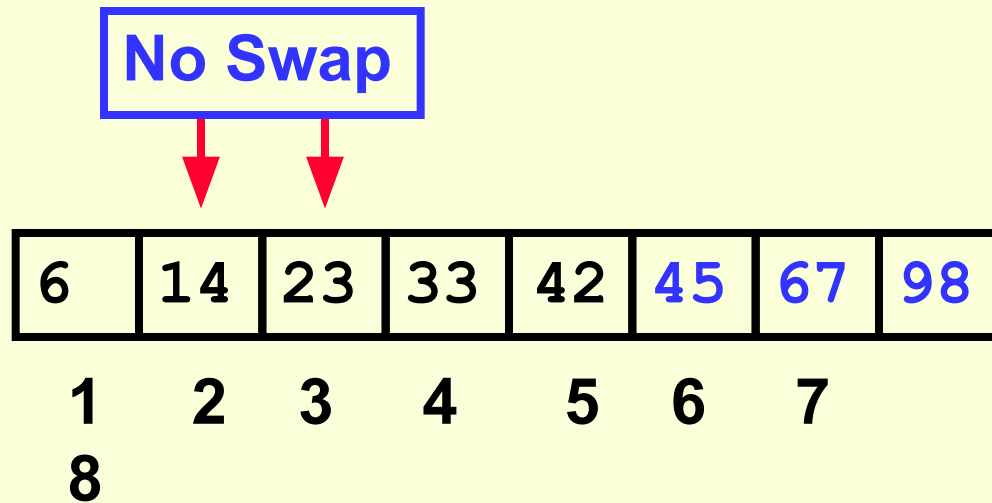
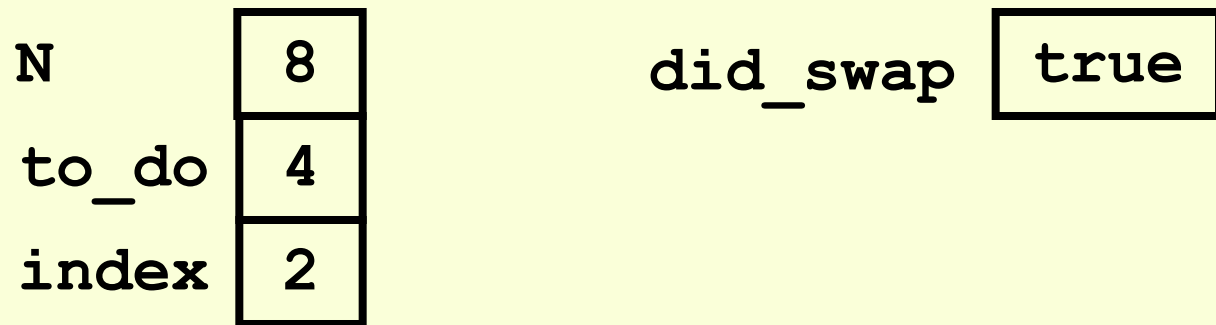
The Fourth “Bubble Up”



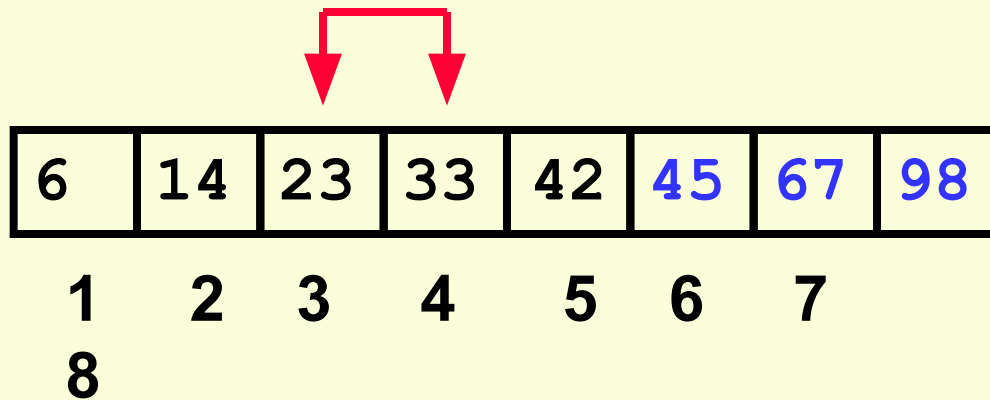
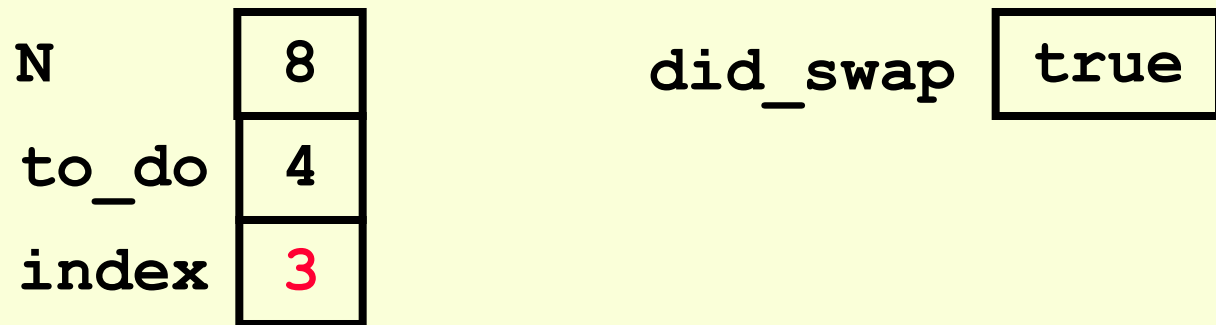
The Fourth “Bubble Up”



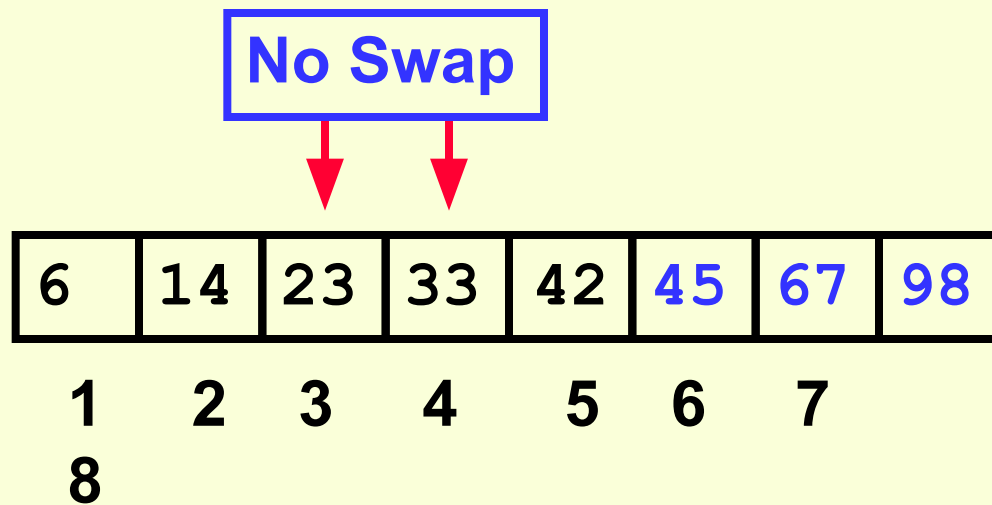
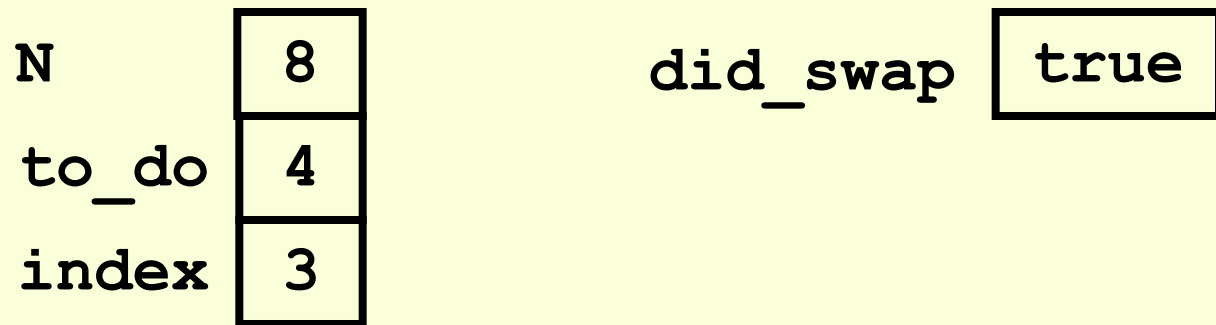
The Fourth “Bubble Up”



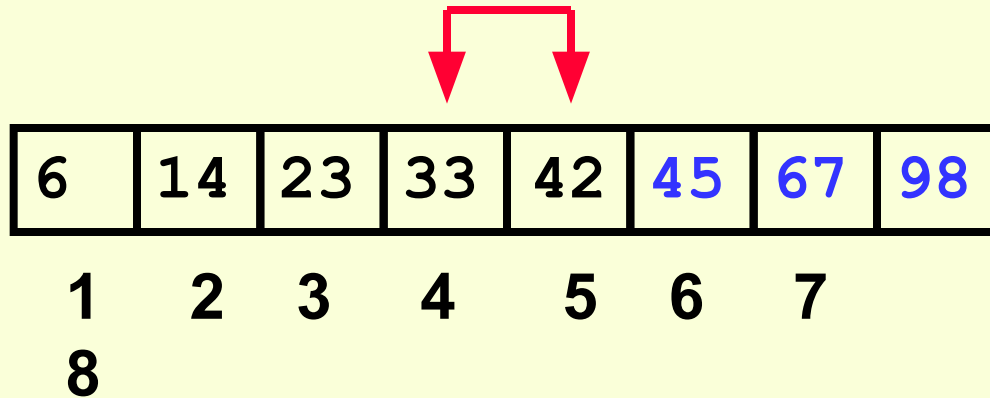
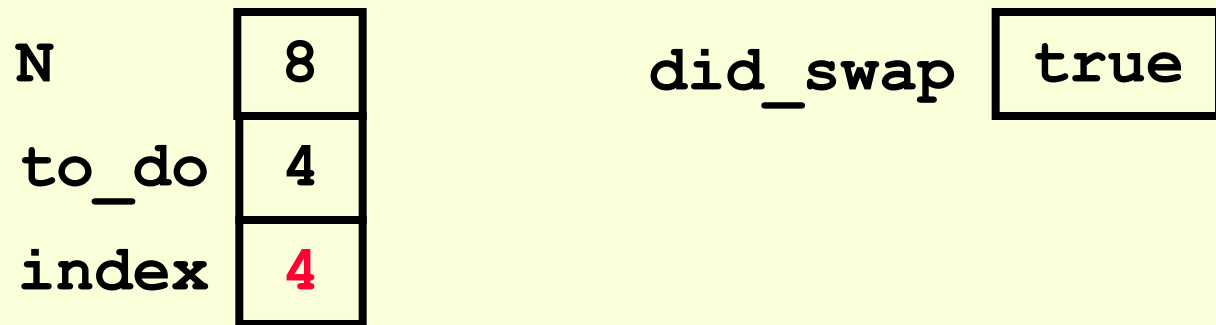
The Fourth “Bubble Up”



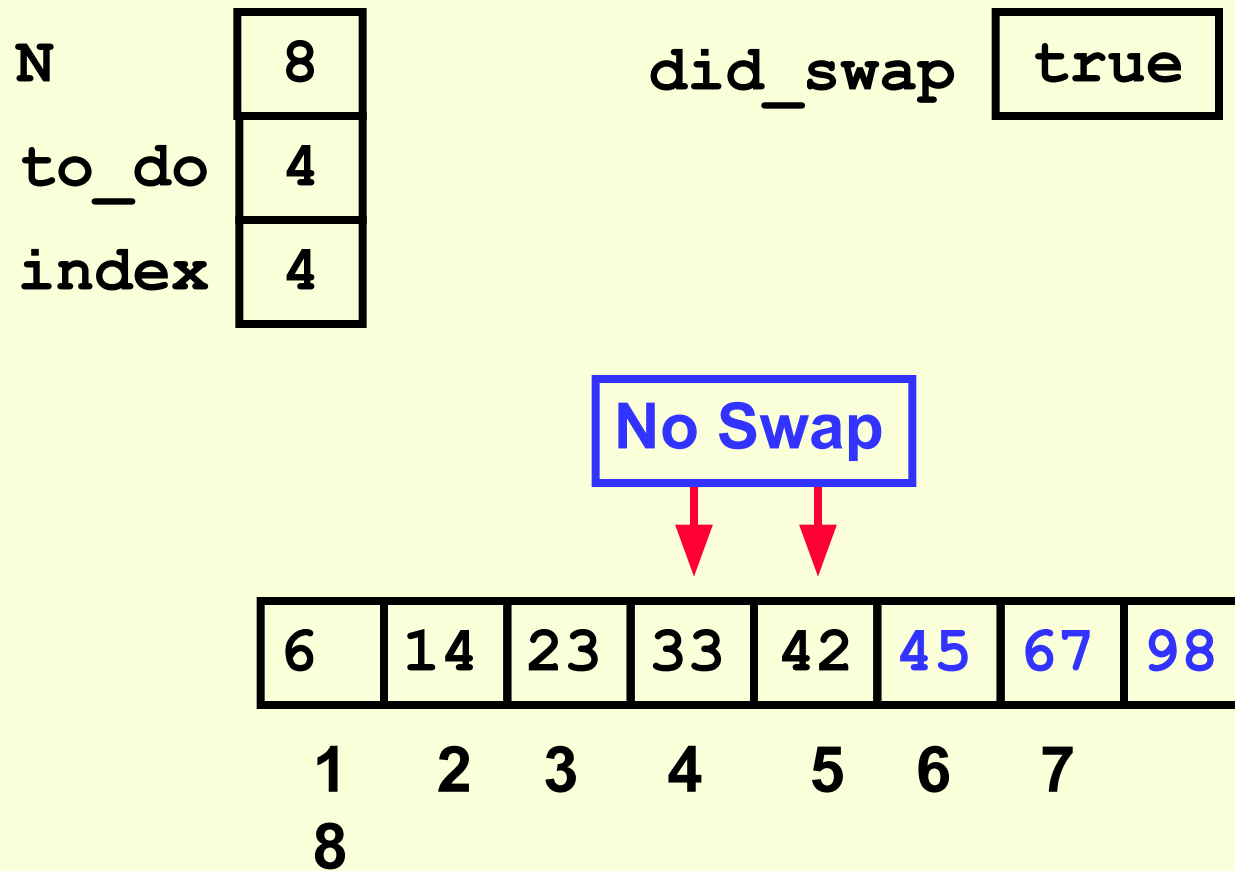
The Fourth “Bubble Up”



The Fourth “Bubble Up”



The Fourth “Bubble Up”



After Fourth Pass of Outer Loop

N

8

to_do

4

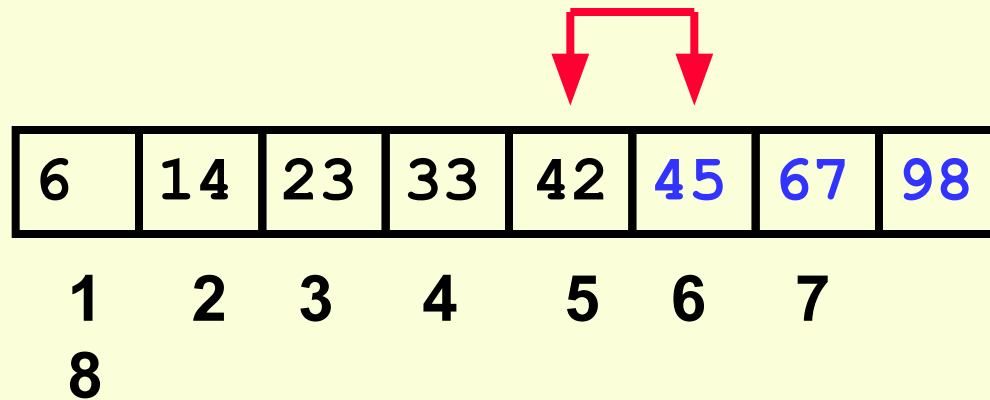
index

5

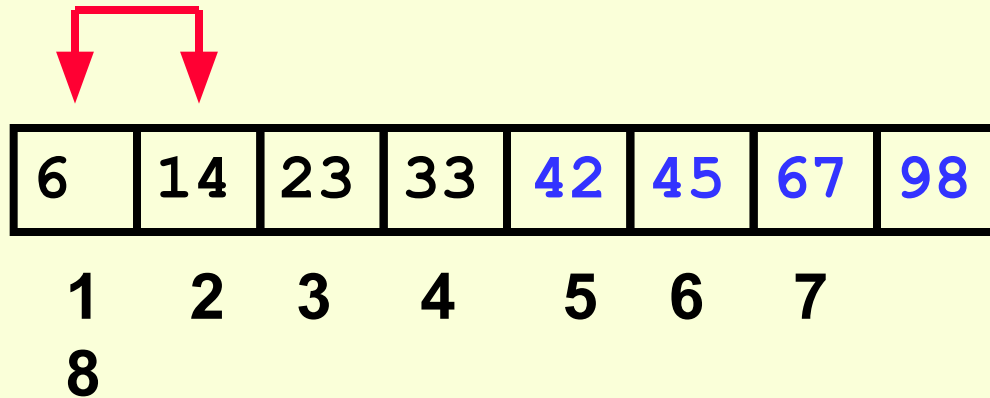
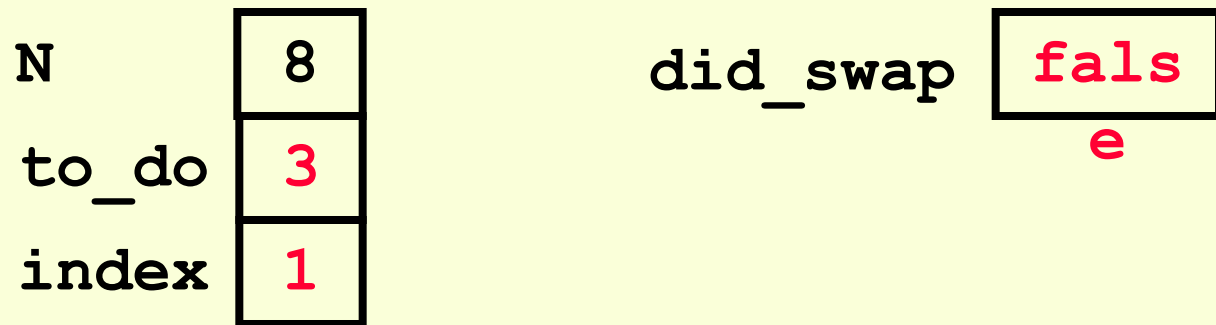
did_swap

true

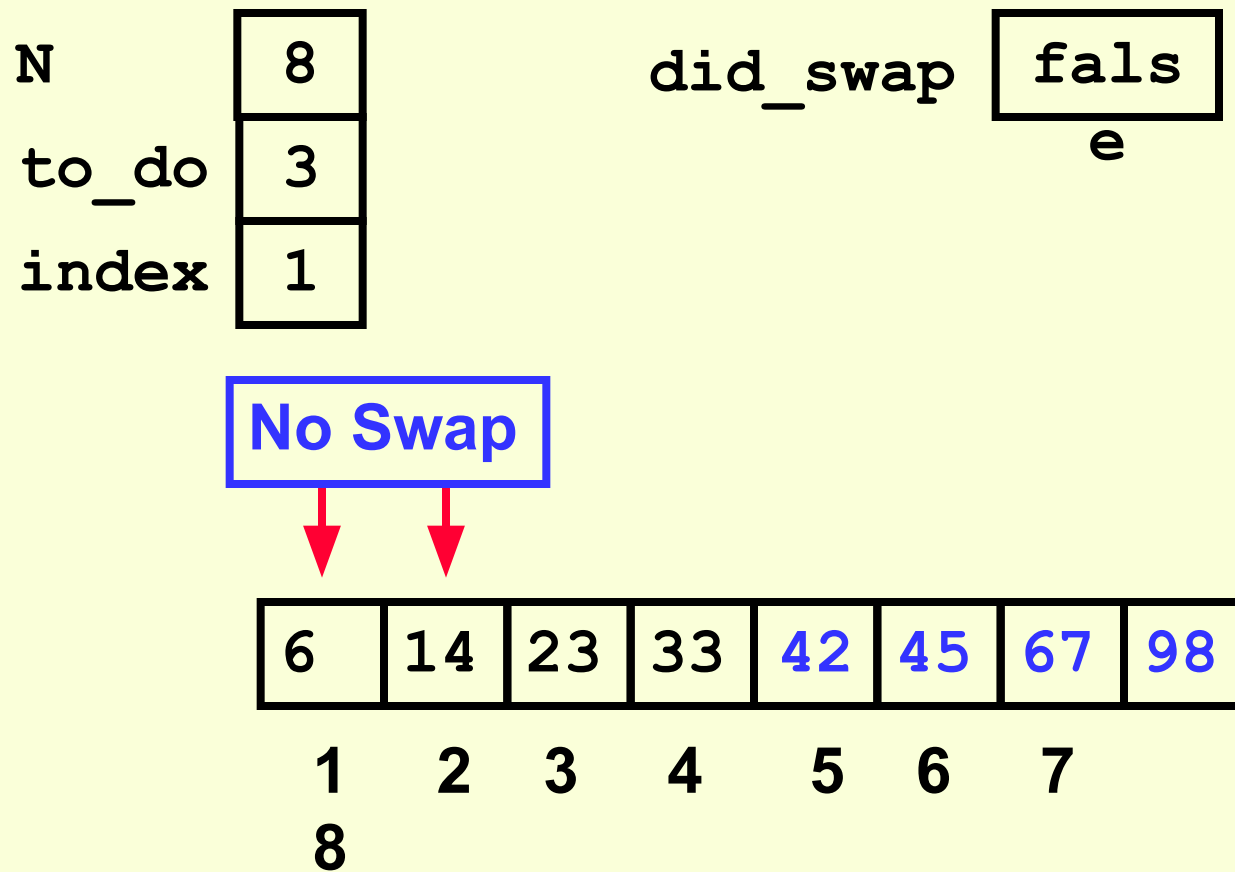
Finished fourth “Bubble Up”



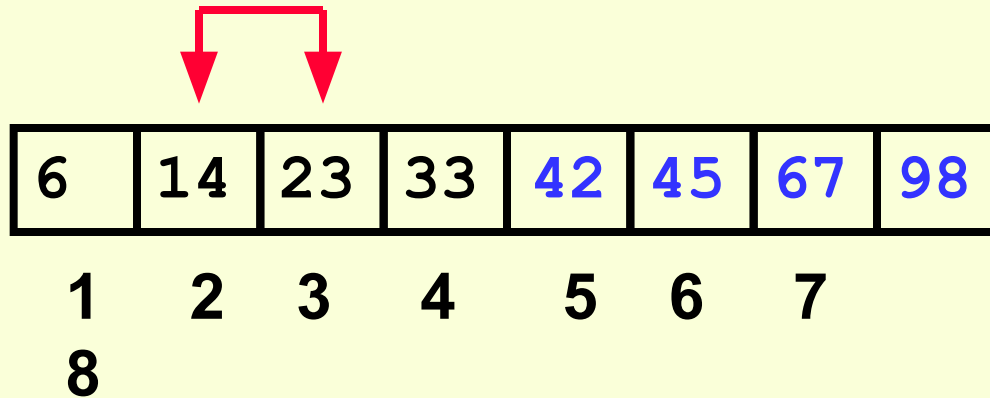
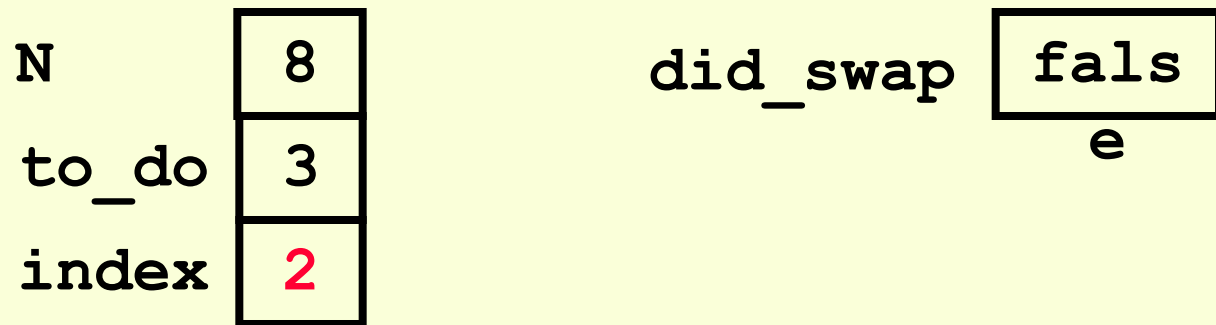
The Fifth “Bubble Up”



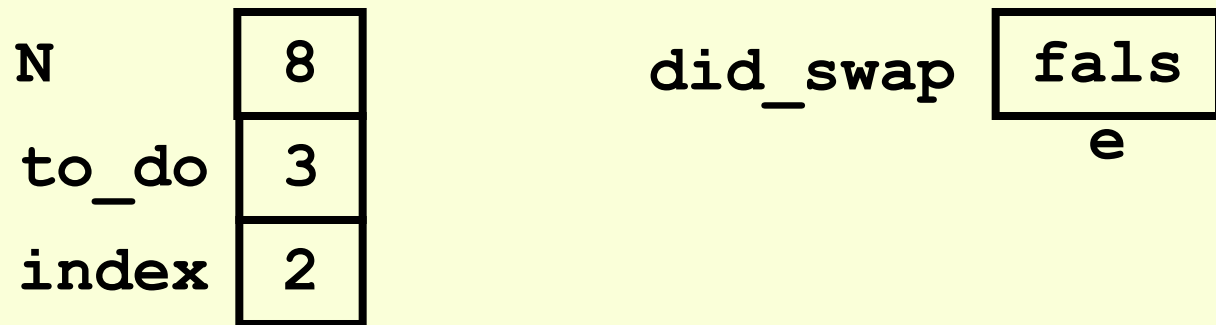
The Fifth “Bubble Up”



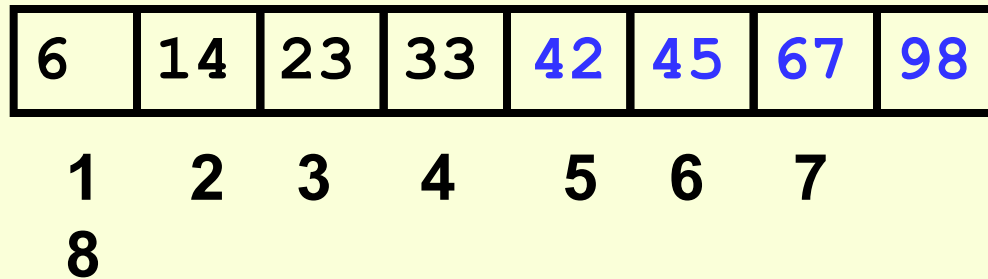
The Fifth “Bubble Up”



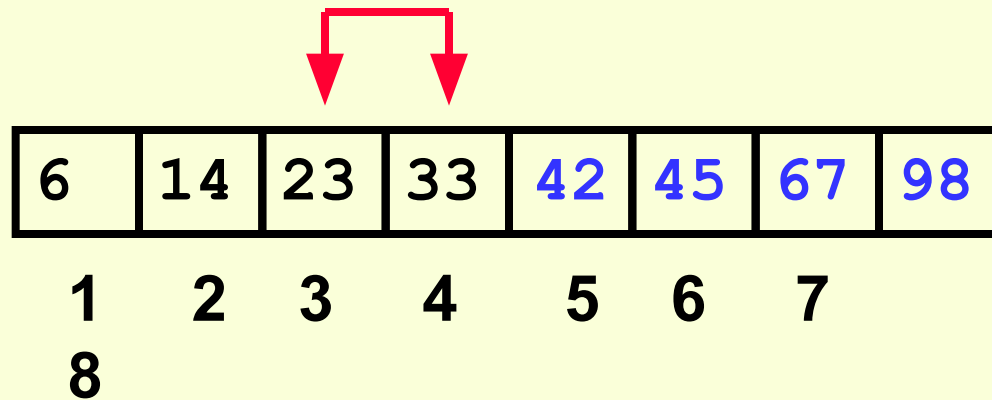
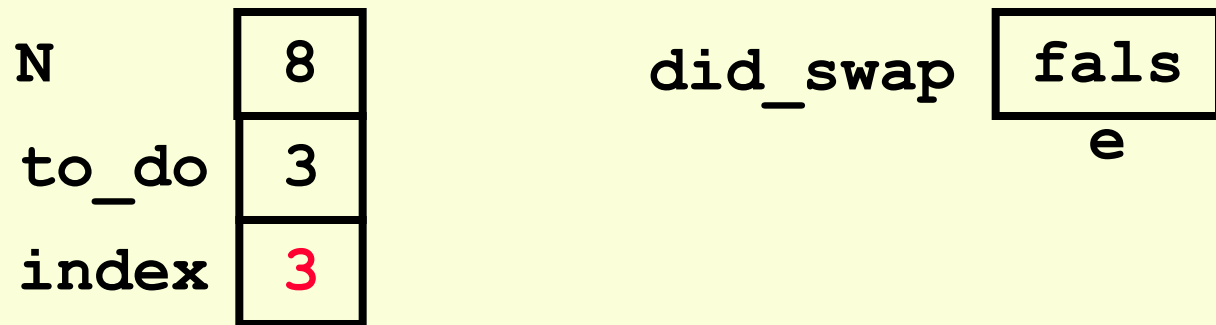
The Fifth “Bubble Up”



No Swap



The Fifth “Bubble Up”



The Fifth “Bubble Up”

N

8

to_do

3

index

3

did_swap

fals

e

No Swap

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

After Fifth Pass of Outer Loop

N

8

to_do

3

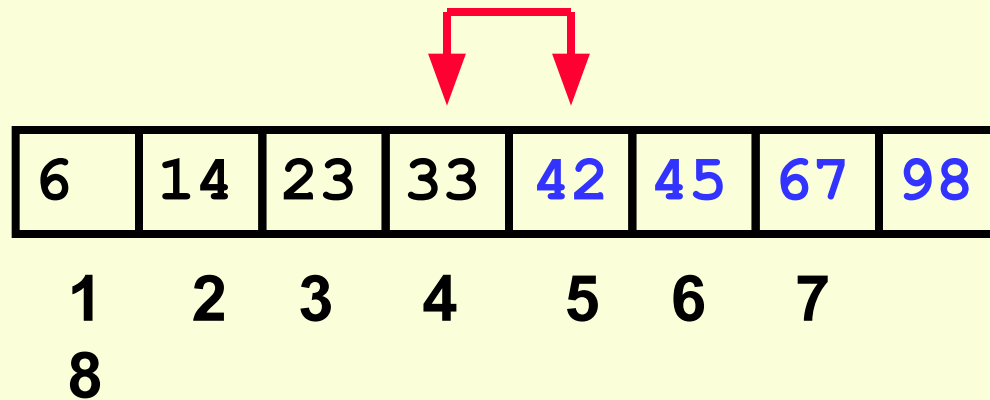
index

4

did_swap

false

Finished fifth "Bubble Up"



Finished “Early”

N	8
to_do	3
index	4

did_swap

false

e

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							

Summary

- “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

Truth in CS Act

- **NOBODY EVER USES BUBBLE SORT**
- **NOBODY**
- **NOT EVER**
- **BECAUSE IT IS EXTREMELY INEFFICIENT**

Questions?

Mergesort

Sorting

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

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

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

Divide and Conquer

- **Divide and Conquer cuts the problem in half each time, but uses the result of both halves:**
 - cut the problem in half until the problem is trivial
 - solve for both halves
 - combine the solutions

Mergesort

- A divide-and-conquer algorithm:
- Divide the unsorted array into 2 halves until the sub-arrays only contain one element
- Merge the sub-problem solutions together:
 - Compare the sub-array's first elements
 - Remove the smallest element and put it into the result array
 - Continue the process until all elements have been put into the result array

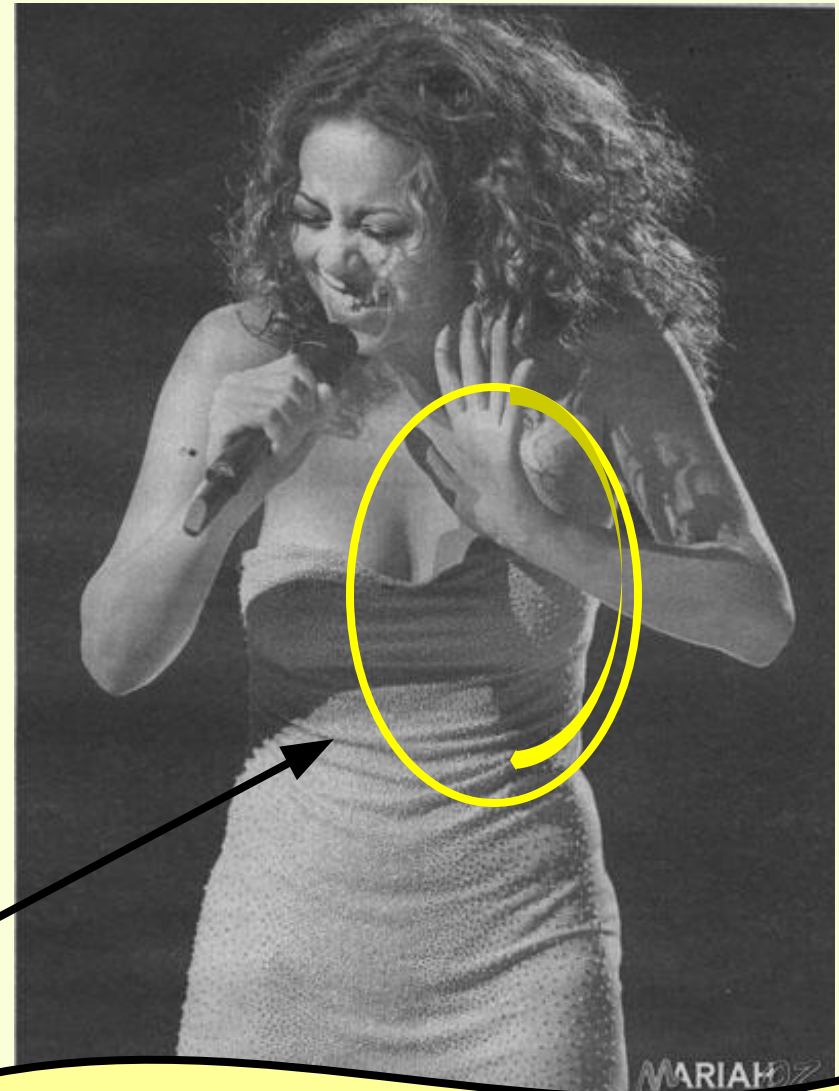
37	23	6	89	15	12	2	19
----	----	---	----	----	----	---	----

How to Remember Merge Sort?

That's easy. Just remember Mariah Carey.

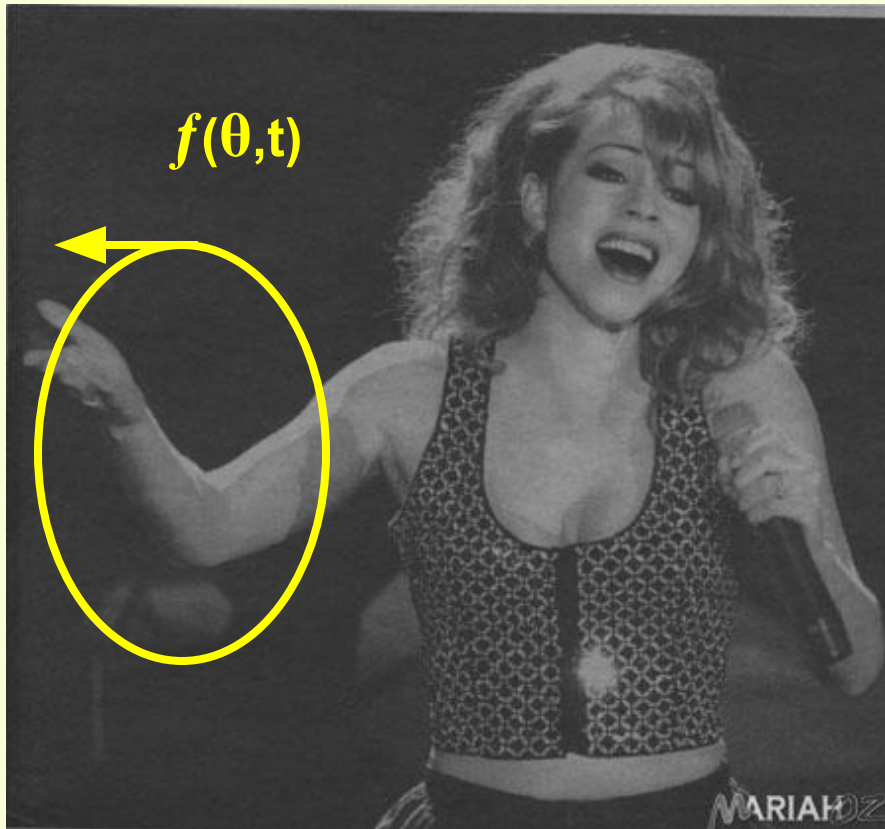
As a singing star, Ms. Carey has perfected the “wax-on” wave motion--a clockwise sweep of her hand used to emphasize lyrics.

The Maria
“Wax-on” Angle:
 $f(\theta, t)$



The Siren of Subquadratic Sorts

How To Remember Merge Sort?



Just as Mariah recursively moves her hands into smaller circles, so too does merge sort recursively split an array into smaller segments.



We need two such recursions, one for each half of the split array.

Algorithm

Mergesort(Passed an array)

if array size > 1

Divide array in half

Call Mergesort on first half.

Call Mergesort on second half.

Merge two halves.

Merge(Passed two arrays)

Compare leading element in each array

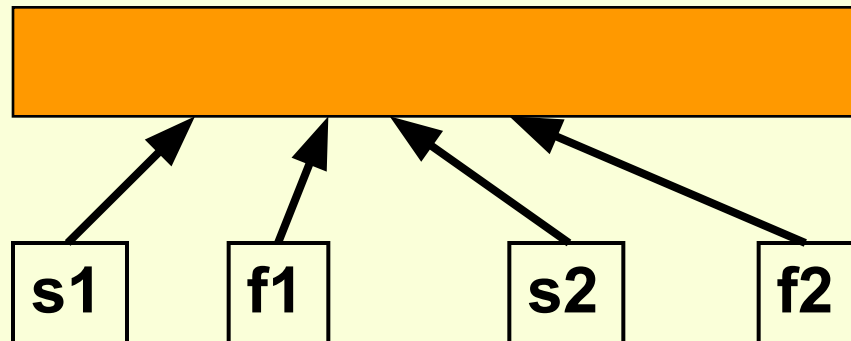
Select lower and place in new array.

(If one input array is empty then place
remainder of other array in output array)

More TRUTH in CS

- We don't really pass in two arrays!
- We pass in one array with indicator variables which tell us where one set of data starts and finishes and where the other set of data starts and finishes.

- Honest.



Algorithm

Mergesort(Passed an array)

if array size > 1

Divide array in half

Call Mergesort on first half.

Call Mergesort on second half.

Merge two halves.

Merge(Passed two arrays)

Compare leading element in each array

Select lower and place in new array.

(If one input array is empty then place
remainder of other array in output array)

98	23	45	14	6	67	33	42
----	----	----	----	---	----	----	----

98	23	45	14	6	67	33	42
----	----	----	----	---	----	----	----

98	23	45	14
----	----	----	----

6	67	33	42
---	----	----	----

98	23	45	14	6	67	33	42
----	----	----	----	---	----	----	----

98	23	45	14
----	----	----	----

6	67	33	42
---	----	----	----

98	23
----	----

45	14
----	----

98	23	45	14	6	67	33	42
----	----	----	----	---	----	----	----

98	23	45	14
----	----	----	----

6	67	33	42
---	----	----	----

98	23
----	----

45	14
----	----

98	23
----	----

98	23	45	14	6	67	33	42
----	----	----	----	---	----	----	----

98	23	45	14
----	----	----	----

6	67	33	42
---	----	----	----

98	23
----	----

45	14
----	----

98

23

Merge

98	23	45	14	6	67	33	42
----	----	----	----	---	----	----	----

98	23	45	14
----	----	----	----

6	67	33	42
---	----	----	----

98	23
----	----

45	14
----	----

98	23
----	----

23

Merge

98	23	45	14	6	67	33	42
----	----	----	----	---	----	----	----

98	23	45	14
----	----	----	----

6	67	33	42
---	----	----	----

98	23
----	----

45	14
----	----

98	23
----	----

23	98
----	----

Merge

98	23	45	14	6	67	33	42
----	----	----	----	---	----	----	----

98	23	45	14
----	----	----	----

6	67	33	42
---	----	----	----

98	23
----	----

45	14
----	----

98

23

45

14

23	98
----	----

98	23	45	14	6	67	33	42
----	----	----	----	---	----	----	----

98	23	45	14
----	----	----	----

6	67	33	42
---	----	----	----

98	23
----	----

45	14
----	----

98

23

45

14

23	98
----	----

Merge

98	23	45	14	6	67	33	42
----	----	----	----	---	----	----	----

98	23	45	14
----	----	----	----

6	67	33	42
---	----	----	----

98	23
----	----

45	14
----	----

98

23

45

14

23	98
----	----

14

Merge

98	23	45	14	6	67	33	42
----	----	----	----	---	----	----	----

98	23	45	14
----	----	----	----

6	67	33	42
---	----	----	----

98	23
----	----

45	14
----	----

98

23

45

14

23	98
----	----

14	45
----	----

Merge

98	23	45	14	6	67	33	42
----	----	----	----	---	----	----	----

98	23	45	14
----	----	----	----

6	67	33	42
---	----	----	----

98	23
----	----

45	14
----	----

98

23

45

14

23	98
----	----

14	45
----	----

Merge

98	23	45	14	6	67	33	42
----	----	----	----	---	----	----	----

98	23	45	14
----	----	----	----

6	67	33	42
---	----	----	----

98	23
----	----

45	14
----	----

98

23

45

14

23	98
----	----

14	45
----	----

14

Merge

98	23	45	14	6	67	33	42
----	----	----	----	---	----	----	----

98	23	45	14
----	----	----	----

6	67	33	42
---	----	----	----

98	23
----	----

45	14
----	----

98

23

45

14

23	98
----	----

14	45
----	----

14	23
----	----

Merge

98	23	45	14	6	67	33	42
----	----	----	----	---	----	----	----

98	23	45	14
----	----	----	----

6	67	33	42
---	----	----	----

98	23
----	----

45	14
----	----

98

23

45

14

23	98
----	----

14	45
----	----

14	23	45
----	----	----

Merge

98	23	45	14	6	67	33	42
----	----	----	----	---	----	----	----

98	23	45	14
----	----	----	----

6	67	33	42
---	----	----	----

98	23
----	----

45	14
----	----

98

23

45

14

23	98
----	----

14	45
----	----

14	23	45	98
----	----	----	----

Merge

98	23	45	14	6	67	33	42
----	----	----	----	---	----	----	----

98	23	45	14
----	----	----	----

6	67	33	42
---	----	----	----

98	23
----	----

45	14
----	----

6	67
---	----

33	42
----	----

98

23

45

14

23	98
----	----

14	45
----	----

14	23	45	98
----	----	----	----

98	23	45	14	6	67	33	42
----	----	----	----	---	----	----	----

98	23	45	14
----	----	----	----

6	67	33	42
---	----	----	----

98	23
----	----

45	14
----	----

6	67
---	----

33	42
----	----

98

23

45

14

6

67

23	98
----	----

14	45
----	----

14	23	45	98
----	----	----	----

98	23	45	14	6	67	33	42
----	----	----	----	---	----	----	----

98	23	45	14
----	----	----	----

6	67	33	42
---	----	----	----

98	23
----	----

45	14
----	----

6	67
---	----

33	42
----	----

98

23

45

14

6

67

23	98
----	----

14	45
----	----

14	23	45	98
----	----	----	----

Merge

98	23	45	14	6	67	33	42
----	----	----	----	---	----	----	----

98	23	45	14
----	----	----	----

6	67	33	42
---	----	----	----

98	23
----	----

45	14
----	----

6	67
---	----

33	42
----	----

98

23

45

14

6

67

23	98
----	----

14	45
----	----

6

14	23	45	98
----	----	----	----

Merge

98	23	45	14	6	67	33	42
----	----	----	----	---	----	----	----

98	23	45	14
----	----	----	----

6	67	33	42
---	----	----	----

98	23
----	----

45	14
----	----

6	67
---	----

33	42
----	----

98

23

45

14

6

67

23	98
----	----

14	45
----	----

6	67
---	----

14	23	45	98
----	----	----	----

Merge

98	23	45	14	6	67	33	42
----	----	----	----	---	----	----	----

98	23	45	14
----	----	----	----

6	67	33	42
---	----	----	----

98	23
----	----

45	14
----	----

6	67
---	----

33	42
----	----

98

23

45

14

6

67

33

42

23	98
----	----

14	45
----	----

6	67
---	----

14	23	45	98
----	----	----	----

98	23	45	14	6	67	33	42
----	----	----	----	---	----	----	----

98	23	45	14
----	----	----	----

6	67	33	42
---	----	----	----

98	23
----	----

45	14
----	----

6	67
---	----

33	42
----	----

98

23

45

14

6

67

33

42

23	98
----	----

14	45
----	----

6	67
---	----

14	23	45	98
----	----	----	----

Merge

98	23	45	14	6	67	33	42
----	----	----	----	---	----	----	----

98	23	45	14
----	----	----	----

6	67	33	42
---	----	----	----

98	23
----	----

45	14
----	----

6	67
---	----

33	42
----	----

98

23

45

14

6

67

33

42

23	98
----	----

14	45
----	----

6	67
---	----

33

14	23	45	98
----	----	----	----

Merge

98	23	45	14	6	67	33	42
----	----	----	----	---	----	----	----

98	23	45	14
----	----	----	----

6	67	33	42
---	----	----	----

98	23
----	----

45	14
----	----

6	67
---	----

33	42
----	----

98

23

45

14

6

67

33

42

23	98
----	----

14	45
----	----

6	67
---	----

33	42
----	----

14	23	45	98
----	----	----	----

Merge

98	23	45	14	6	67	33	42
----	----	----	----	---	----	----	----

98	23	45	14
----	----	----	----

6	67	33	42
---	----	----	----

98	23
----	----

45	14
----	----

6	67
---	----

33	42
----	----

98

23

45

14

6

67

33

42

23	98
----	----

14	45
----	----

6	67
---	----

33	42
----	----

14	23	45	98
----	----	----	----

Merge

98	23	45	14	6	67	33	42
----	----	----	----	---	----	----	----

98	23	45	14
----	----	----	----

6	67	33	42
---	----	----	----

98	23
----	----

45	14
----	----

6	67
---	----

33	42
----	----

98

23

45

14

6

67

33

42

23	98
----	----

14	45
----	----

6	67
---	----

33	42
----	----

14	23	45	98
----	----	----	----

6

Merge

98	23	45	14	6	67	33	42
----	----	----	----	---	----	----	----

98	23	45	14
----	----	----	----

6	67	33	42
---	----	----	----

98	23
----	----

45	14
----	----

6	67
---	----

33	42
----	----

98

23

45

14

6

67

33

42

23	98
----	----

14	45
----	----

6	67
---	----

33	42
----	----

14	23	45	98
----	----	----	----

6	33
---	----

Merge

98	23	45	14	6	67	33	42
----	----	----	----	---	----	----	----

98	23	45	14
----	----	----	----

6	67	33	42
---	----	----	----

98	23
----	----

45	14
----	----

6	67
---	----

33	42
----	----

98

23

45

14

6

67

33

42

23	98
----	----

14	45
----	----

6	67
---	----

33	42
----	----

14	23	45	98
----	----	----	----

6	33	42
---	----	----

Merge

98	23	45	14	6	67	33	42
----	----	----	----	---	----	----	----

98	23	45	14
----	----	----	----

6	67	33	42
---	----	----	----

98	23
----	----

45	14
----	----

6	67
---	----

33	42
----	----

98

23

45

14

6

67

33

42

23	98
----	----

14	45
----	----

6	67
---	----

33	42
----	----

14	23	45	98
----	----	----	----

6	33	42	67
---	----	----	----

Merge

98	23	45	14	6	67	33	42
----	----	----	----	---	----	----	----

98	23	45	14
----	----	----	----

6	67	33	42
---	----	----	----

98	23
----	----

45	14
----	----

6	67
---	----

33	42
----	----

98

23

45

14

6

67

33

42

23	98
----	----

14	45
----	----

6	67
---	----

33	42
----	----

14	23	45	98
----	----	----	----

6	33	42	67
---	----	----	----

Merge

98	23	45	14	6	67	33	42
----	----	----	----	---	----	----	----

98	23	45	14
----	----	----	----

6	67	33	42
---	----	----	----

98	23
----	----

45	14
----	----

6	67
---	----

33	42
----	----

98

23

45

14

6

67

33

42

23	98
----	----

14	45
----	----

6	67
---	----

33	42
----	----

14	23	45	98
----	----	----	----

6	33	42	67
---	----	----	----

6

Merge

98	23	45	14	6	67	33	42
----	----	----	----	---	----	----	----

98	23	45	14
----	----	----	----

6	67	33	42
---	----	----	----

98	23
----	----

45	14
----	----

6	67
---	----

33	42
----	----

98

23

45

14

6

67

33

42

23	98
----	----

14	45
----	----

6	67
---	----

33	42
----	----

14	23	45	98
----	----	----	----

6	33	42	67
---	----	----	----

6	14
---	----

Merge

98	23	45	14	6	67	33	42
----	----	----	----	---	----	----	----

98	23	45	14
----	----	----	----

6	67	33	42
---	----	----	----

98	23
----	----

45	14
----	----

6	67
---	----

33	42
----	----

98

23

45

14

6

67

33

42

23	98
----	----

14	45
----	----

6	67
---	----

33	42
----	----

14	23	45	98
----	----	----	----

6	33	42	67
---	----	----	----

6	14	23
---	----	----

Merge

98	23	45	14	6	67	33	42
----	----	----	----	---	----	----	----

98	23	45	14
----	----	----	----

6	67	33	42
---	----	----	----

98	23
----	----

45	14
----	----

6	67
---	----

33	42
----	----

98

23

45

14

6

67

33

42

23	98
----	----

14	45
----	----

6	67
---	----

33	42
----	----

14	23	45	98
----	----	----	----

6	33	42	67
---	----	----	----

6	14	23	33
---	----	----	----

Merge

98	23	45	14	6	67	33	42
----	----	----	----	---	----	----	----

98	23	45	14
----	----	----	----

6	67	33	42
---	----	----	----

98	23
----	----

45	14
----	----

6	67
---	----

33	42
----	----

98

23

45

14

6

67

33

42

23	98
----	----

14	45
----	----

6	67
---	----

33	42
----	----

14	23	45	98
----	----	----	----

6	33	42	67
---	----	----	----

6	14	23	33	42
---	----	----	----	----

Merge

98	23	45	14	6	67	33	42
----	----	----	----	---	----	----	----

98	23	45	14
----	----	----	----

6	67	33	42
---	----	----	----

98	23
----	----

45	14
----	----

6	67
---	----

33	42
----	----

98

23

45

14

6

67

33

42

23	98
----	----

14	45
----	----

6	67
---	----

33	42
----	----

14	23	45	98
----	----	----	----

6	33	42	67
---	----	----	----

6	14	23	33	42	45
---	----	----	----	----	----

Merge

98	23	45	14	6	67	33	42
----	----	----	----	---	----	----	----

98	23	45	14
----	----	----	----

6	67	33	42
---	----	----	----

98	23
----	----

45	14
----	----

6	67
---	----

33	42
----	----

98

23

45

14

6

67

33

42

23	98
----	----

14	45
----	----

6	67
---	----

33	42
----	----

14	23	45	98
----	----	----	----

6	33	42	67
---	----	----	----

6	14	23	33	42	45	67
---	----	----	----	----	----	----

Merge

98	23	45	14	6	67	33	42
----	----	----	----	---	----	----	----

98	23	45	14
----	----	----	----

6	67	33	42
---	----	----	----

98	23
----	----

45	14
----	----

6	67
---	----

33	42
----	----

98

23

45

14

6

67

33

42

23	98
----	----

14	45
----	----

6	67
---	----

33	42
----	----

14	23	45	98
----	----	----	----

6	33	42	67
---	----	----	----

6	14	23	33	42	45	67	98
---	----	----	----	----	----	----	----

Merge

98	23	45	14	6	67	33	42
----	----	----	----	---	----	----	----

98	23	45	14
----	----	----	----

6	67	33	42
---	----	----	----

98	23
----	----

45	14
----	----

6	67
---	----

33	42
----	----

98

23

45

14

6

67

33

42

23	98
----	----

14	45
----	----

6	67
---	----

33	42
----	----

14	23	45	98
----	----	----	----

6	33	42	67
---	----	----	----

6	14	23	33	42	45	67	98
---	----	----	----	----	----	----	----

98	23	45	14	6	67	33	42
----	----	----	----	---	----	----	----



6	14	23	33	42	45	67	98
---	----	----	----	----	----	----	----

Summary

- **Divide** the unsorted collection **into two**
- **Until the sub-arrays only contain one element**
- **Then merge the sub-problem solutions together**

Questions?

Review?

