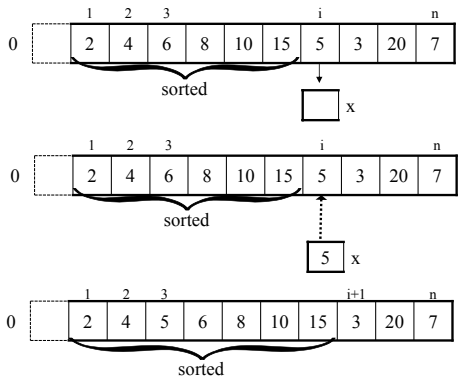
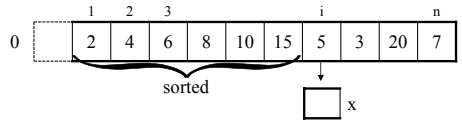
Sorting in Place

- Insertion Sort with an array
- Selection Sort with an array
- Bubblesort with an array

Insertion Sort (array)



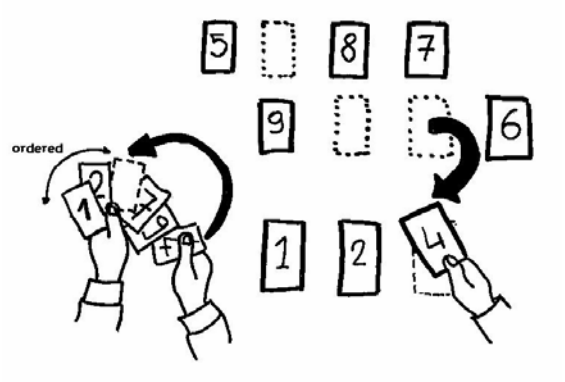

```
for (i=2; i=n, i++)
  x ← A[i]
  A[0] ← x
  j ← i-1
    while x.key < A[j].key
            A[j+1] ← A[j]
            j ← j-1
  A[j+1] ← x
```



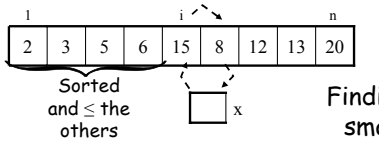
Complexity
Insertion Sort

- For n elements, we need to index through n-1 entries
- For each entry, we need to examine and shift up to n-1 other entries
- Complexity => $O(n^2)$
- Best case complexity is $O(n)$ when elements are already in order
- Worst case complexity is $O(n^2)$ when elements are in reverse order

Selection Sort (array)



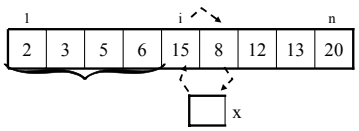
Selection Sort (array)



Finding the smallest element

```
for (i=1; i=n-1, i++)
  k ← i
  x ← A[i]
  for (j=i+1; j=n, j++)
    if A[j].key < x.key
           k ← j
           x ← A[j]
  A[k] ← A[i]
  A[i] ← x
```

```
for (i=1; i=n-1, i++)
  k ← i
  x ← A[i]
  for (j=i+1; j=n, j++)
    if A[j].key < x.key
           k ← j
           x ← A[j]
  A[k] ← A[i]
  A[i] ← x
```



Complexity of Selection Sort

-For each i from 1 to n-1, there is one exchange and (n-i) comparisons.

- In total, there are (n-1) exchanges and (n-1)+(n-2)+ … +2+1 = n(n-1)/2 comparisons.

- Complexity => $O(n^2)$.

- Worst case occurs if all the elements are reverse order.

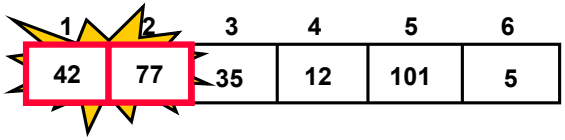
Bubble Sort

"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**

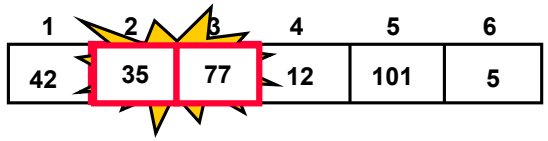
"Bubbling Up" the Largest Element

- **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**



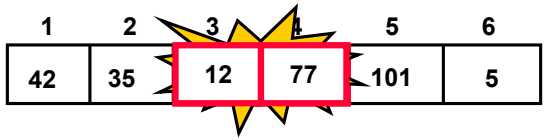
"Bubbling Up" the Largest Element

- **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**



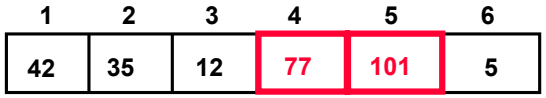
"Bubbling Up" the Largest Element

- **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

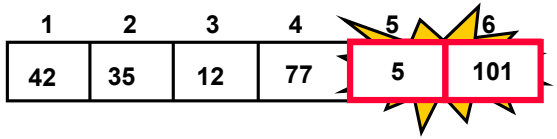
- **Traverse a collection of elements**
  - **Move from the front to the end**
  - **"Bubble" the largest value to the end using pair-wise comparisons and swapping**



No need to swap

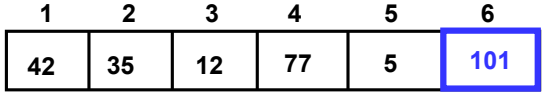
"Bubbling Up" the Largest Element

- **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**



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

N - 1

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

Complexity of Bubblesort (arrays)

Comparisons

$$C = \sum_{i=1}^{n-1} (n-i) = n(n-1)/2 = O(n^2)$$

Movements

$D_{min} = 0$ (already in order)

$D_{max} = 3 \cdot C = O(n^2)$ (in reverse order)

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

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
to_do	7
index	

did_swap | true |

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

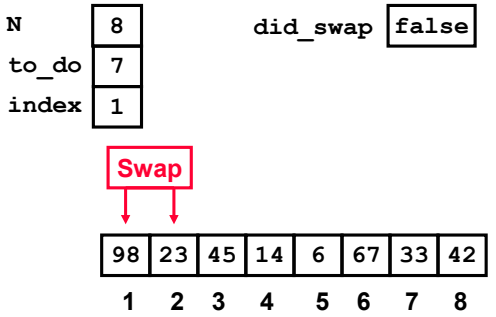
An Animated Example

N	8
to_do	7
index	1

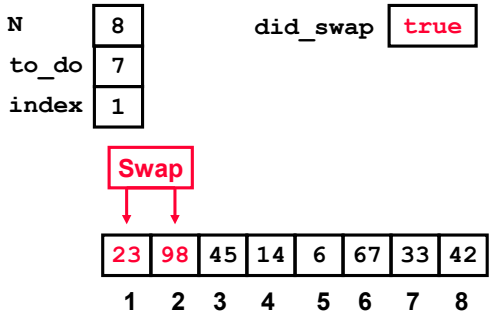
did_swap | false |

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

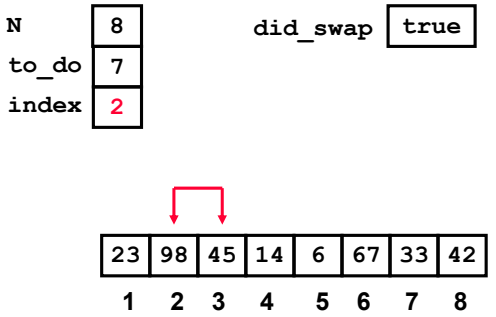
An Animated Example



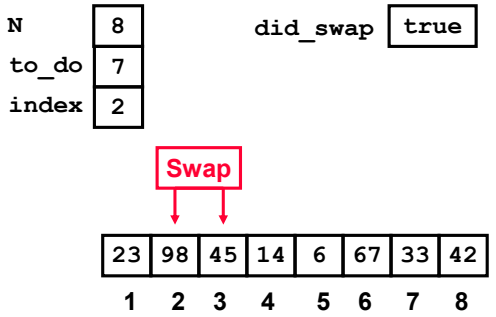
An Animated Example



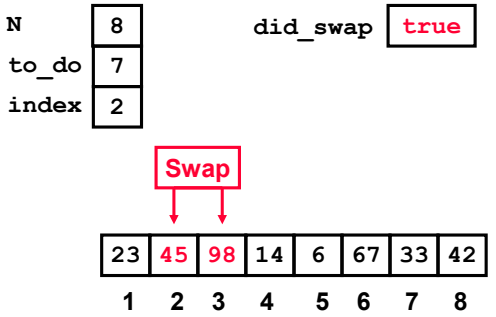
An Animated Example



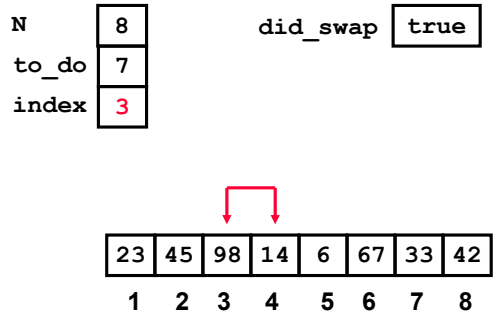
An Animated Example



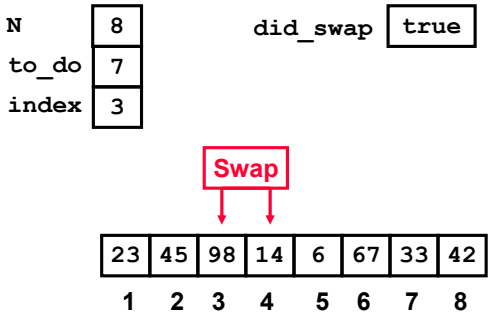
An Animated Example



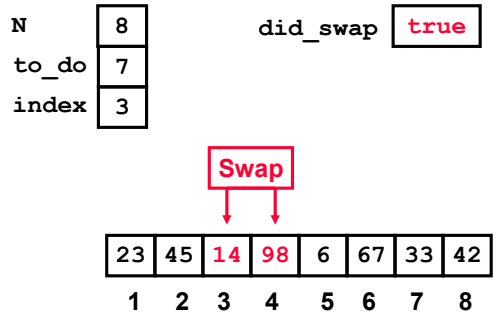
An Animated Example



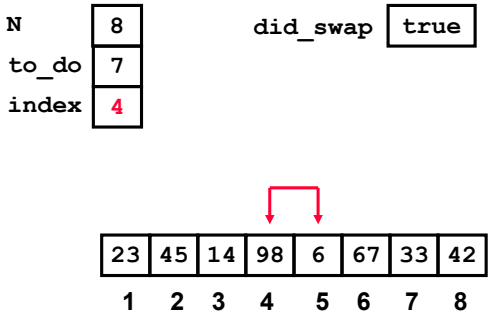
An Animated Example



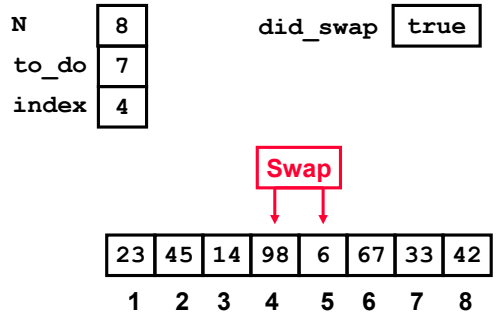
An Animated Example



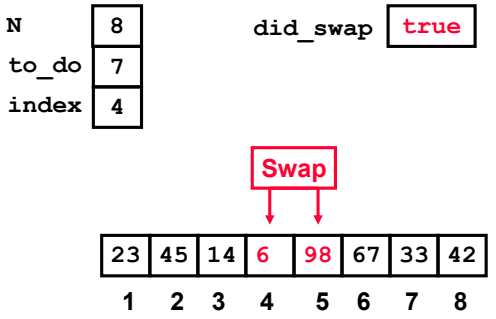
An Animated Example



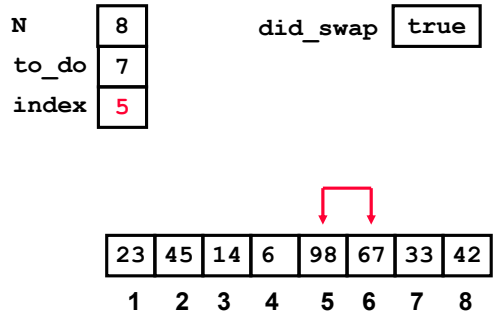
An Animated Example



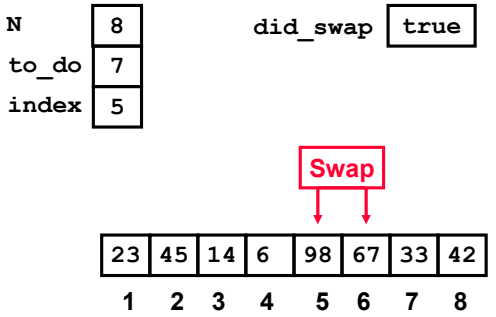
An Animated Example



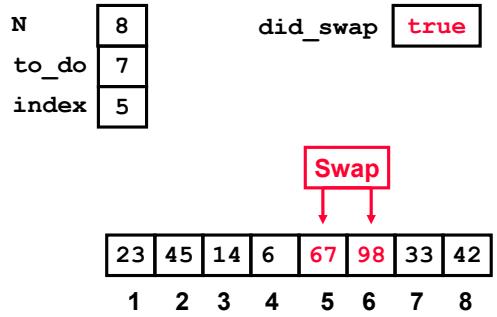
An Animated Example



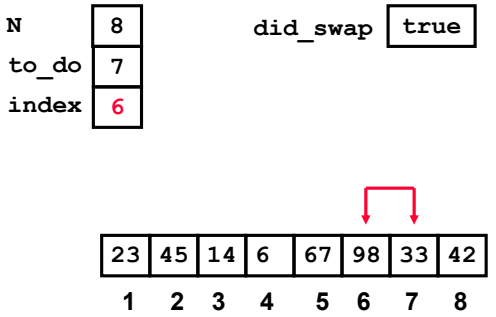
An Animated Example



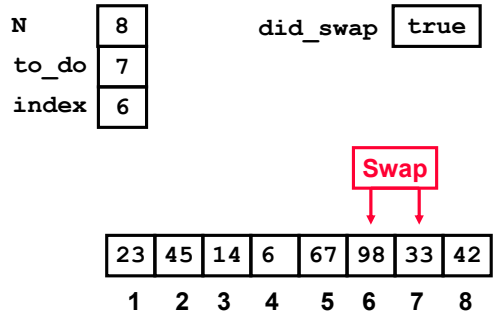
An Animated Example



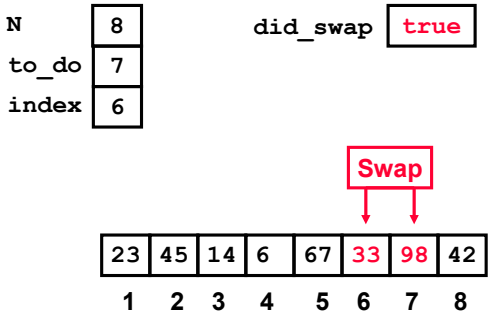
An Animated Example



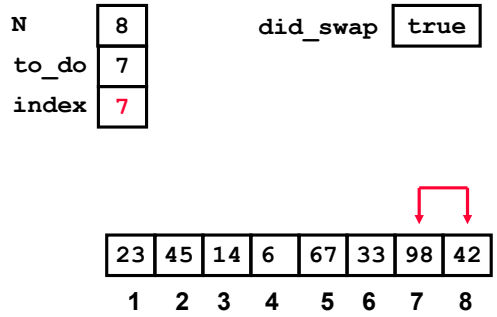
An Animated Example



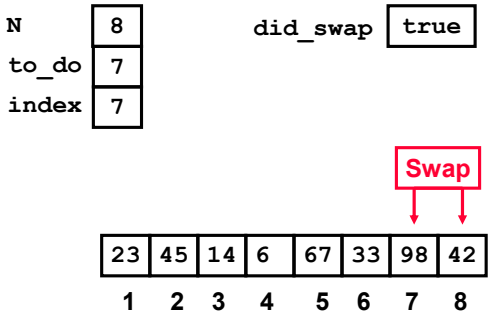
An Animated Example



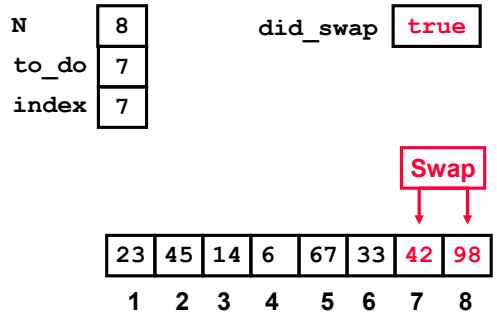
An Animated Example



An Animated Example



An Animated Example

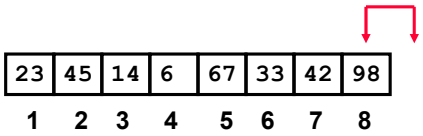


After First Pass of Outer Loop

N	8
to_do	7
index	8

did_swap: true

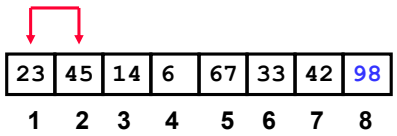
Finished first "Bubble Up"



The Second "Bubble Up"

N	8
to_do	6
index	1

did_swap: false

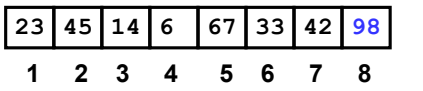


The Second "Bubble Up"

N	8
to_do	6
index	1

did_swap: false

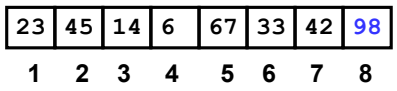
No Swap



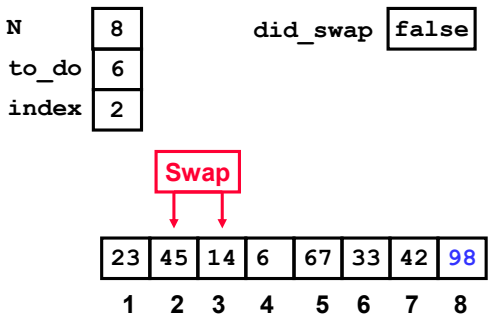
The Second "Bubble Up"

N	8
to_do	6
index	2

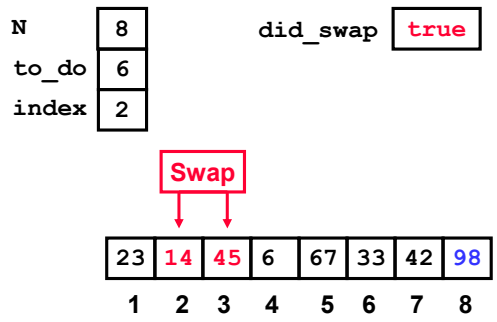
did_swap: false



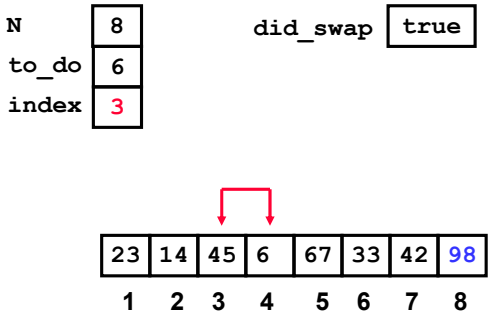
The Second "Bubble Up"



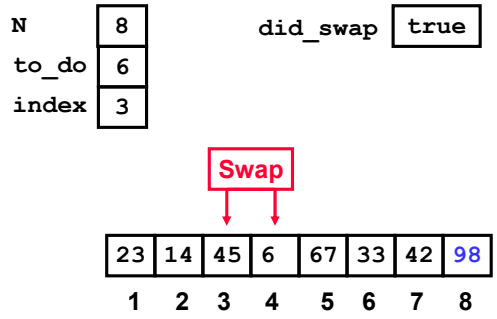
The Second "Bubble Up"



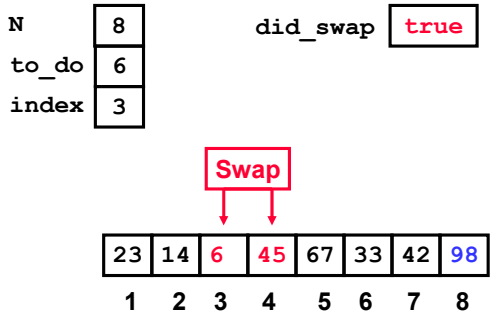
The Second "Bubble Up"



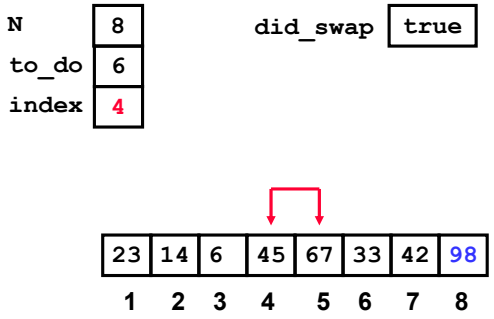
The Second "Bubble Up"



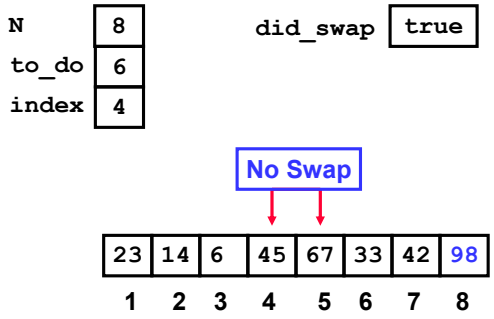
The Second "Bubble Up"



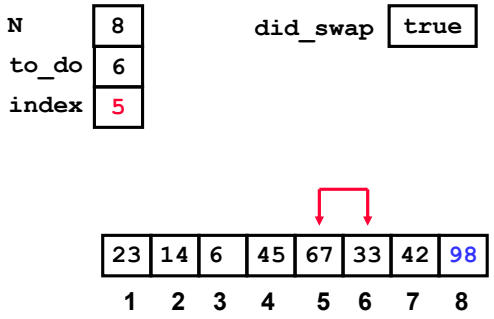
The Second "Bubble Up"



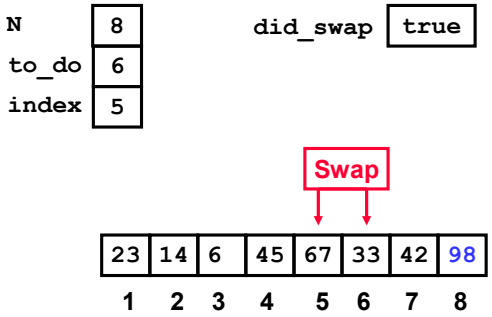
The Second "Bubble Up"



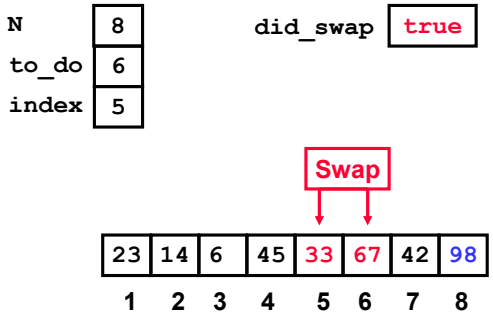
The Second "Bubble Up"



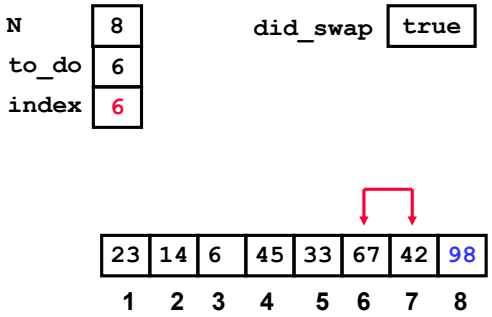
The Second "Bubble Up"



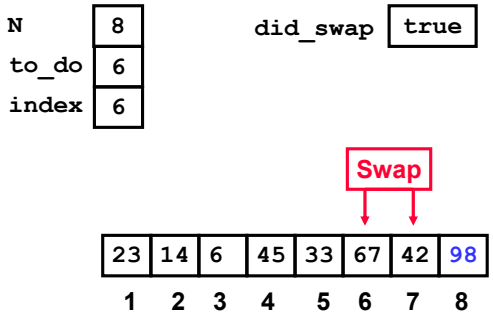
The Second "Bubble Up"



The Second "Bubble Up"



The Second "Bubble Up"



The Second "Bubble Up"

N	8
to_do	6
index	6

did_swap: true

Swap

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

After Second Pass of Outer Loop

N	8
to_do	6
index	7

did_swap: true

Finished second "Bubble Up"

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

The Third "Bubble Up"

N	8
to_do	5
index	1

did_swap: false

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

The Third "Bubble Up"

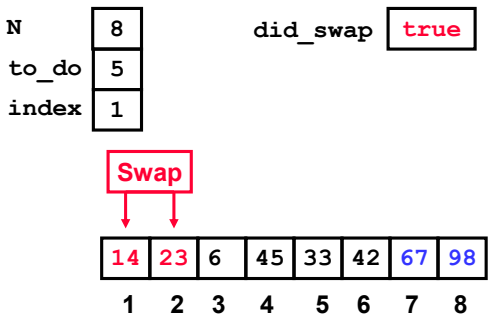
N	8
to_do	5
index	1

did_swap: false

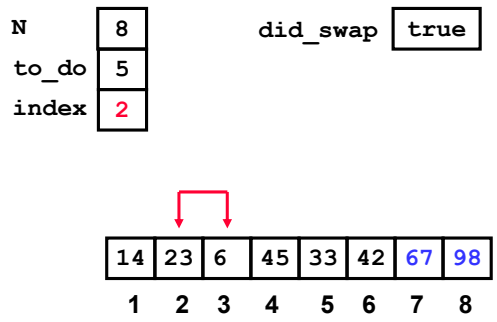
Swap

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

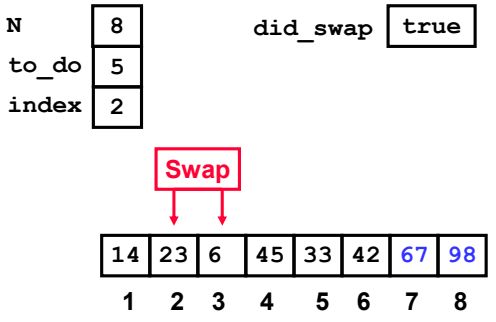
The Third "Bubble Up"



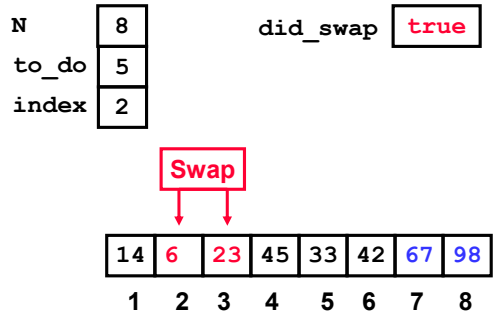
The Third "Bubble Up"



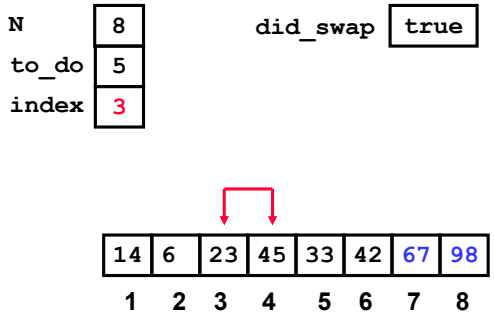
The Third "Bubble Up"



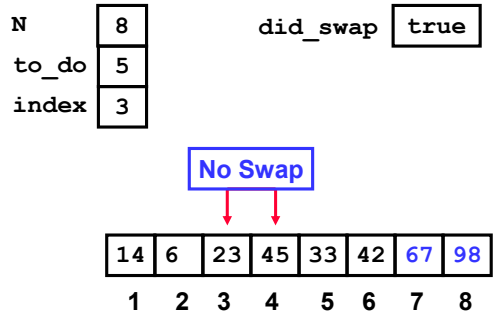
The Third "Bubble Up"



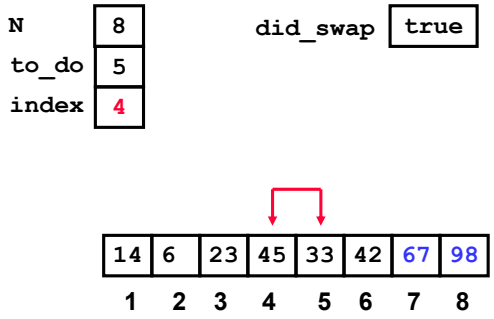
The Third "Bubble Up"



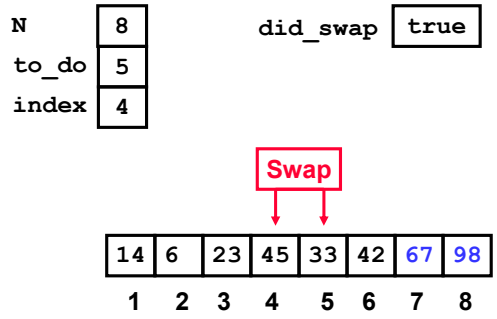
The Third "Bubble Up"



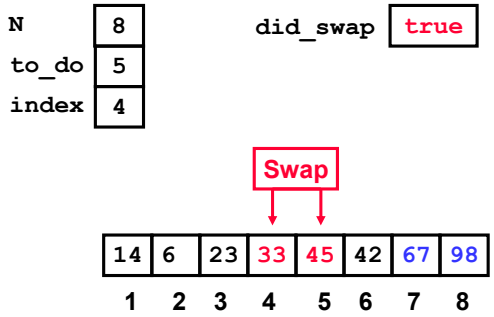
The Third "Bubble Up"



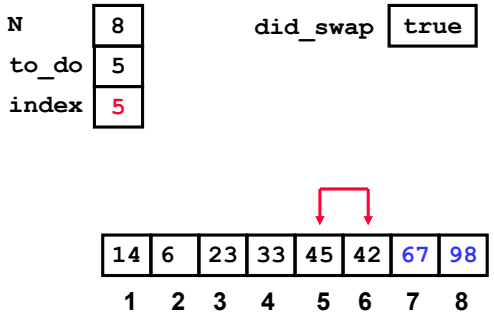
The Third "Bubble Up"



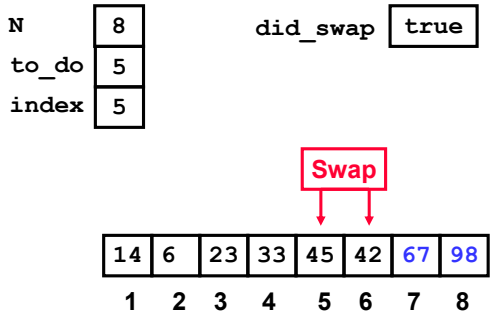
The Third "Bubble Up"



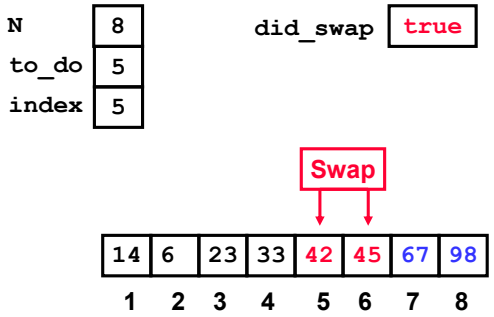
The Third "Bubble Up"



The Third "Bubble Up"



The Third "Bubble Up"

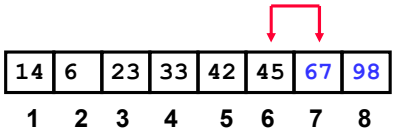


After Third Pass of Outer Loop

N	8
to_do	5
index	6

did_swap: true

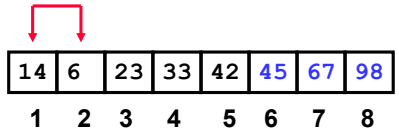
Finished third "Bubble Up"



The Fourth "Bubble Up"

N	8
to_do	4
index	1

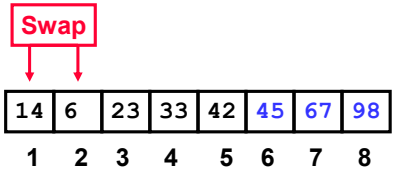
did_swap: false



The Fourth "Bubble Up"

N	8
to_do	4
index	1

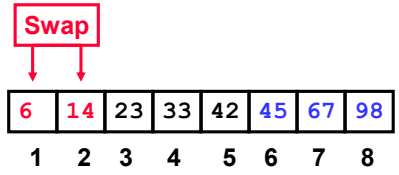
did_swap: false



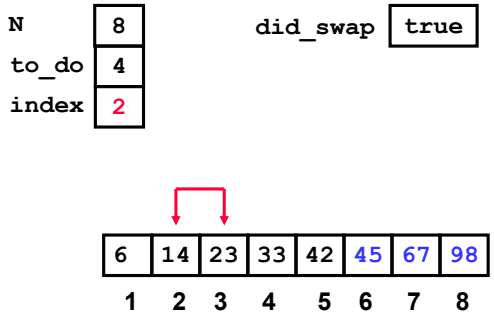
The Fourth "Bubble Up"

N	8
to_do	4
index	1

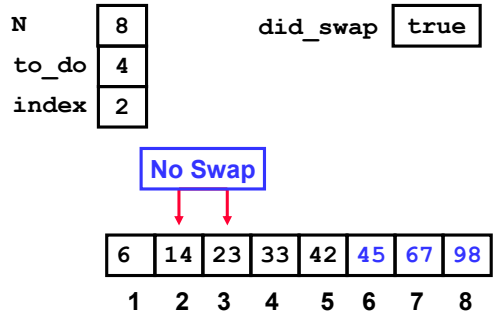
did_swap: true



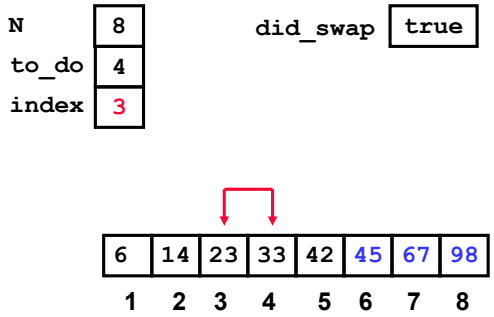
The Fourth "Bubble Up"



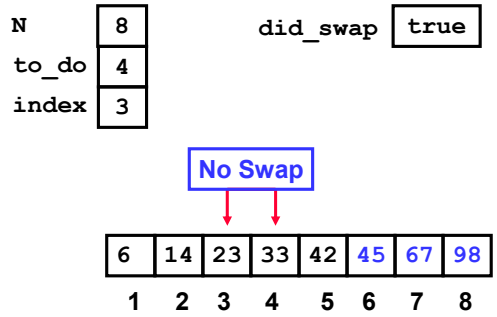
The Fourth "Bubble Up"



The Fourth "Bubble Up"



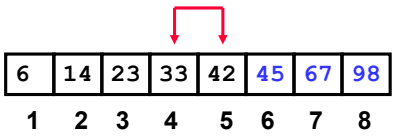
The Fourth "Bubble Up"



The Fourth "Bubble Up"

N	8
to_do	4
index	4

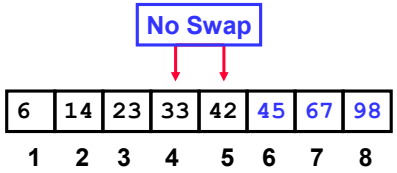
did_swap: true



The Fourth "Bubble Up"

N	8
to_do	4
index	4

did_swap: true

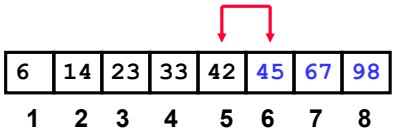


After Fourth Pass of Outer Loop

N	8
to_do	4
index	5

did_swap: true

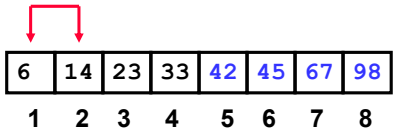
Finished fourth "Bubble Up"



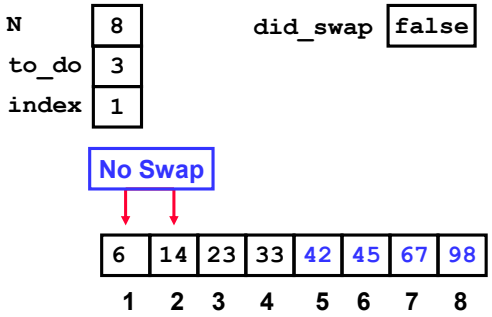
The Fifth "Bubble Up"

N	8
to_do	3
index	1

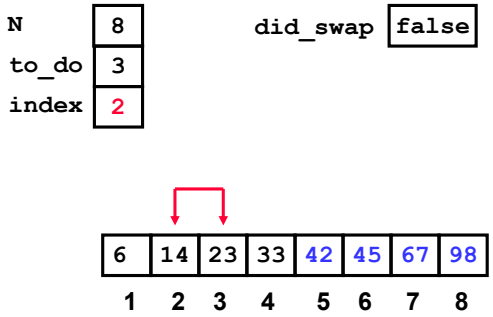
did_swap: false



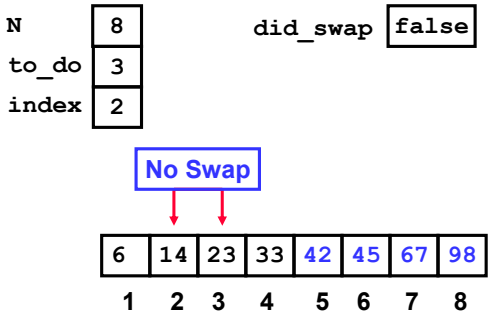
The Fifth "Bubble Up"



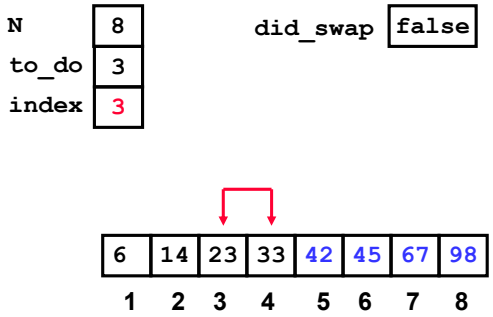
The Fifth "Bubble Up"



The Fifth "Bubble Up"



The Fifth "Bubble Up"

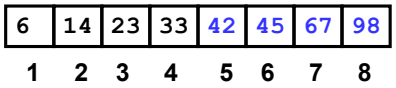


The Fifth "Bubble Up"

N	8
to_do	3
index	3

did_swap: false

No Swap

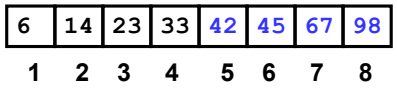


After Fifth Pass of Outer Loop

N	8
to_do	3
index	4

did_swap: false

Finished fifth "Bubble Up"



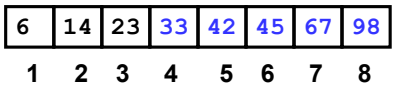
Finished "Early"

N	8
to_do	3
index	4

did_swap: false

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.



Summary of bubblesort

- **"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**

$O(n^2)$