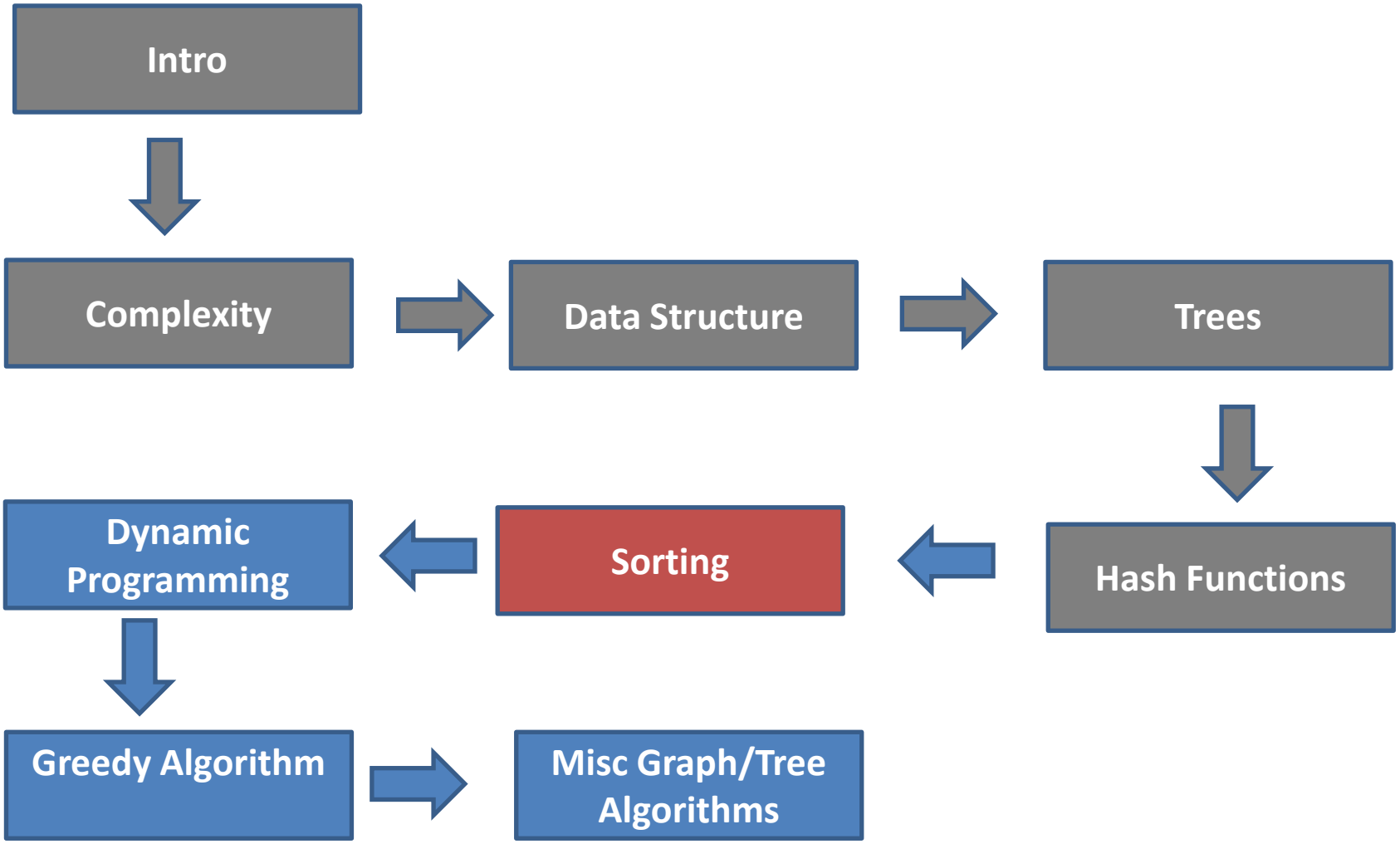
An Introduction to Algorithms

By

Hossein Rahmani

h_rahmani@iust.ac.ir

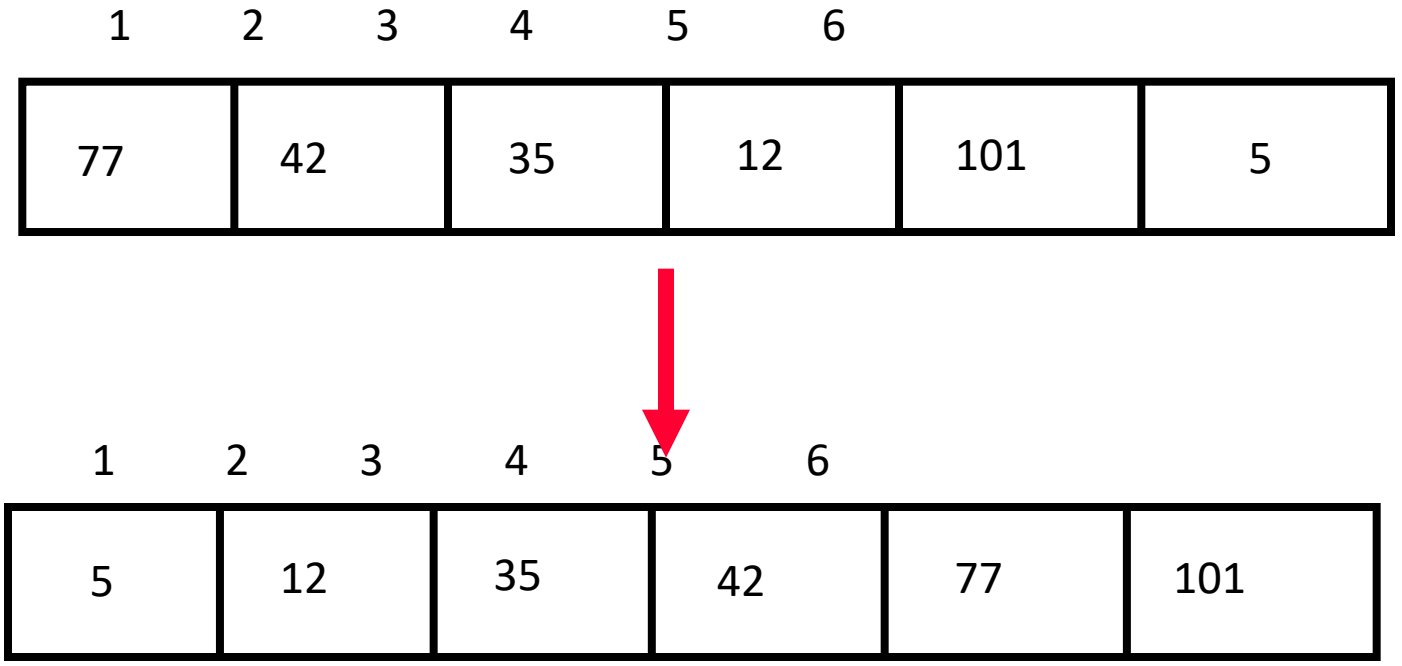
http://webpages.iust.ac.ir/h_rahmani/


Intro
Complexity
Data Structure
Trees
Hash Functions
Sorting
Dynamic Programming
Greedy Algorithm
Misc Graph/Tree Algorithms

Bubble sort

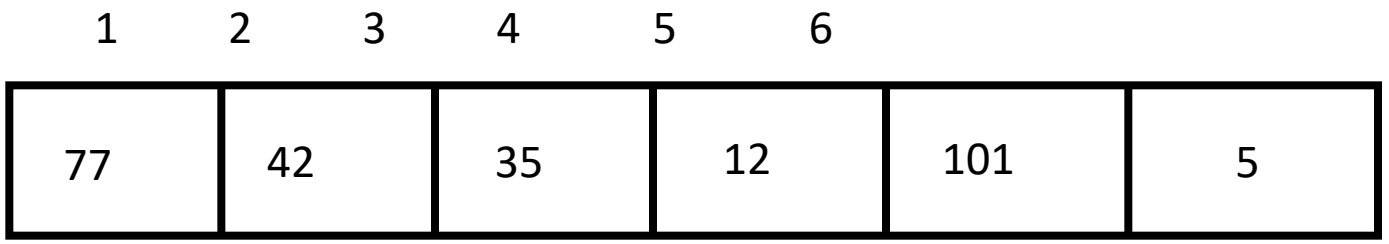
Sorting

- **Sorting takes an <u>unordered</u> collection and makes it an <u>ordered</u> one.**



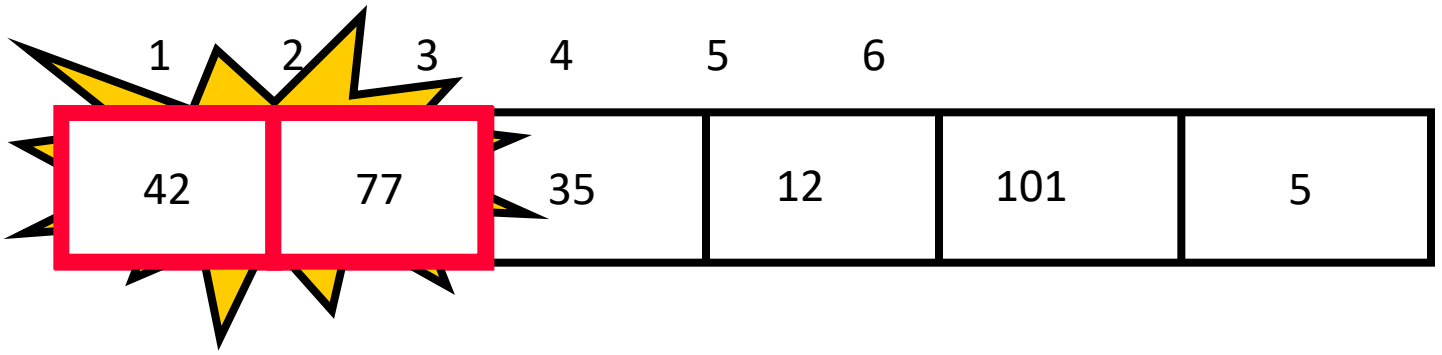
"Bubbling Up" the Largest Element

- **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**



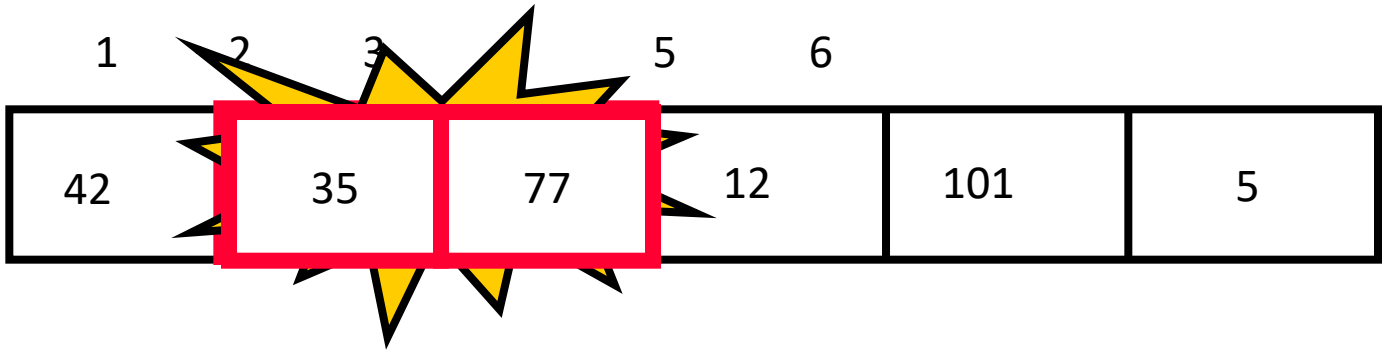
"Bubbling Up" the Largest Element

- **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**



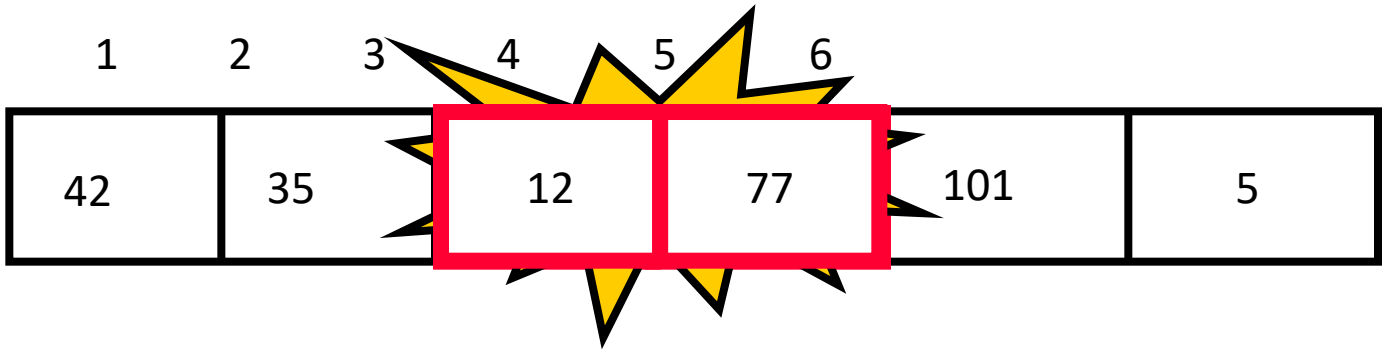
"Bubbling Up" the Largest Element

- **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**



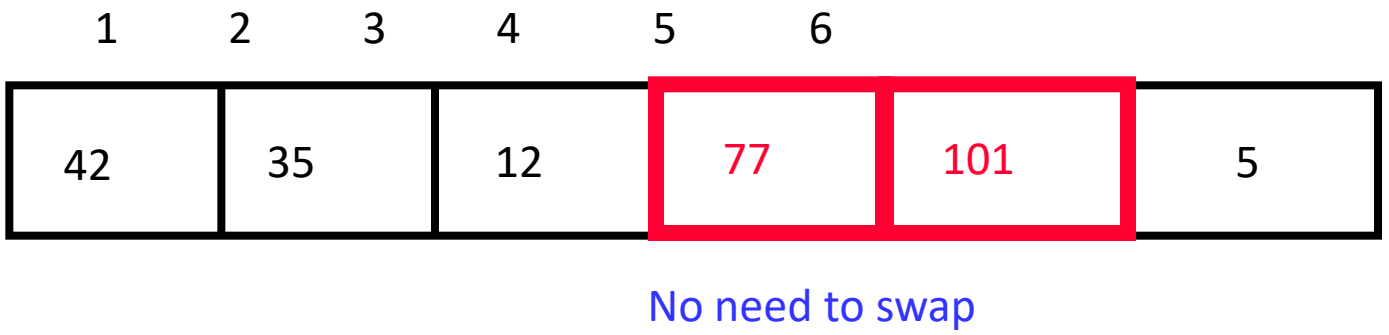
"Bubbling Up" the Largest Element

- **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**



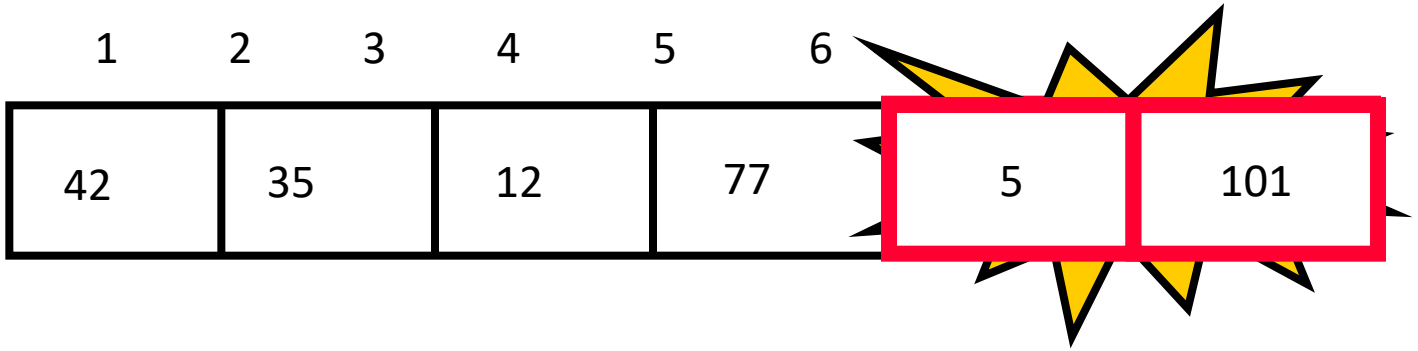
"Bubbling Up" the Largest Element

- **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**



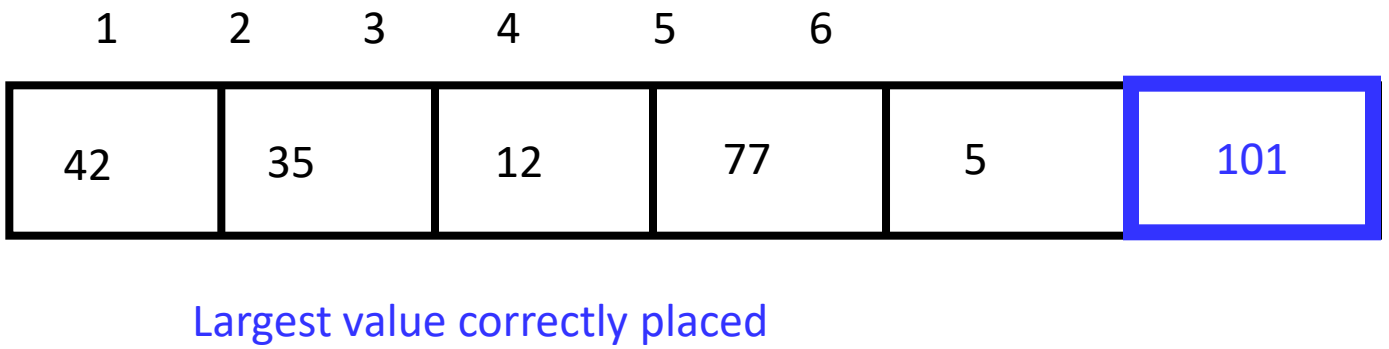
"Bubbling Up" the Largest Element

- **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**



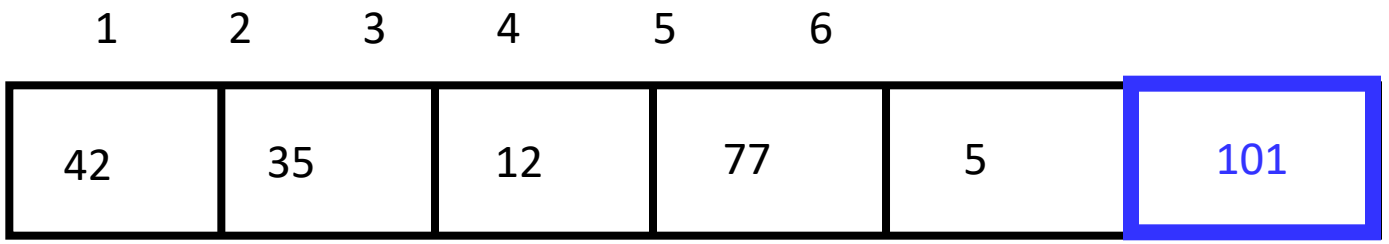
Bubble Sort

Pseudo-code Algorithm

```
public static void bubbleSort(Comparable a[]) {
   for (int p=a.length-1; p>0; p--) {
      for (int j=0; j<p; j++)
         if (a[j].compareTo(a[j+1])>0)
            swap(a,j,j+1);
   } // p
} // bubbleSort
```

Items of Interest

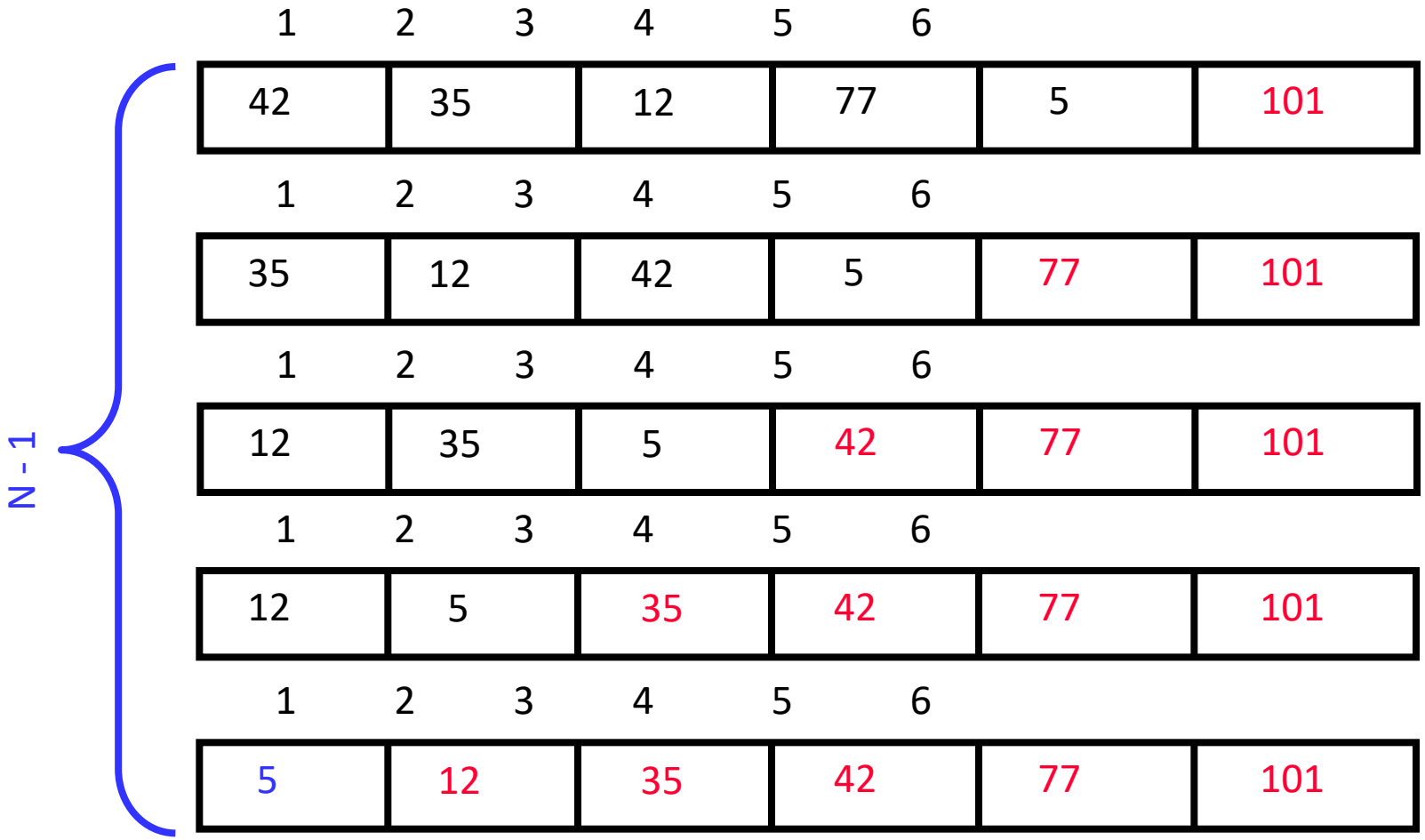
- **Notice that only the largest value is correctly placed**
- **All other values are still out of order**
- **So we need to repeat this process**



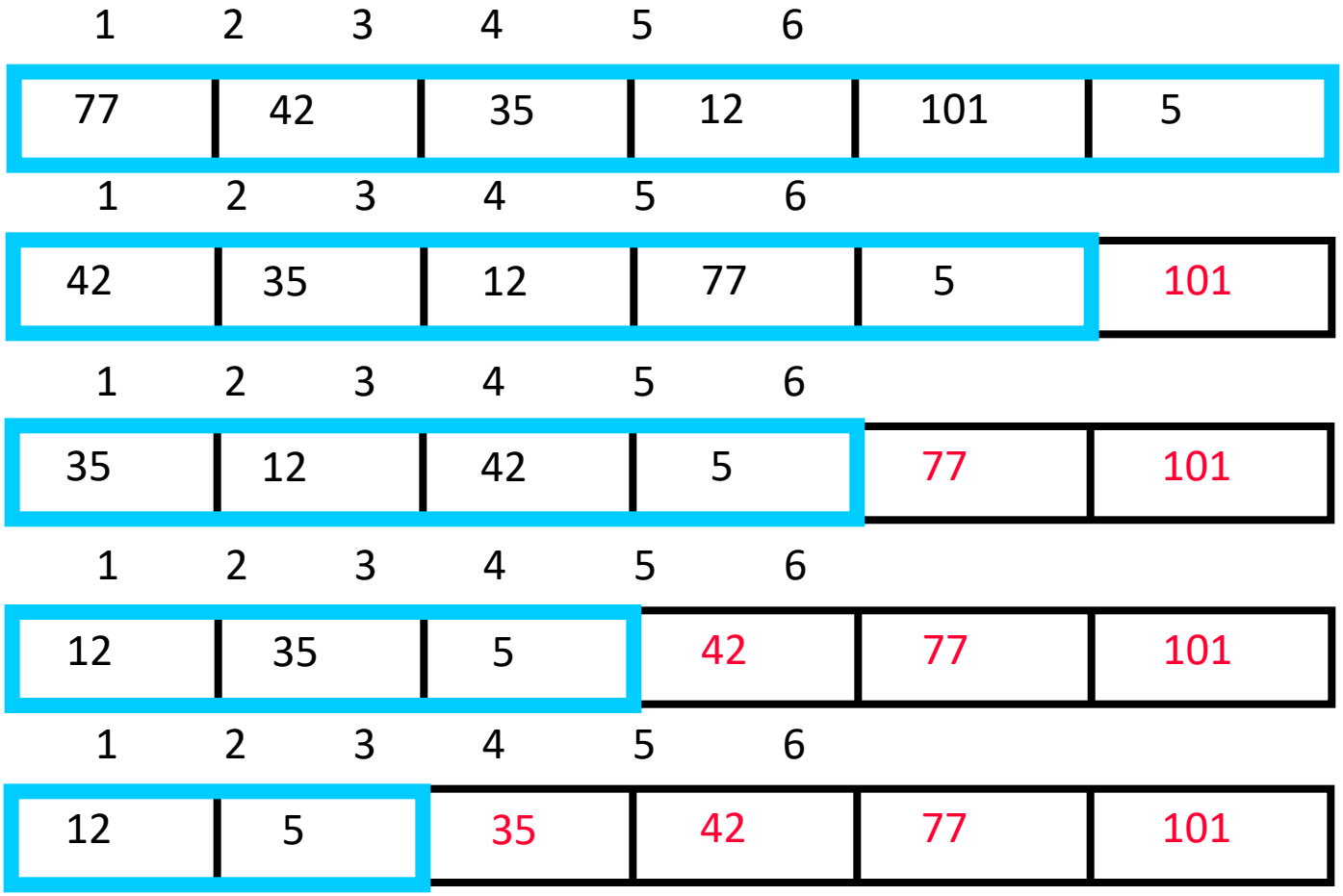
Repeat “Bubble Up” How Many Times?

- **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

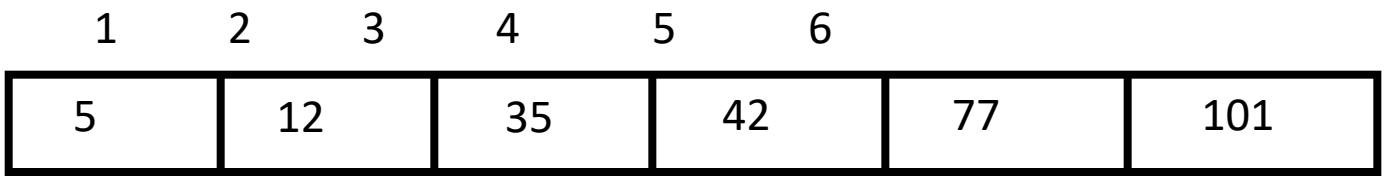


Reducing the Number of Comparisons



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"!**



Using a Boolean "Flag"

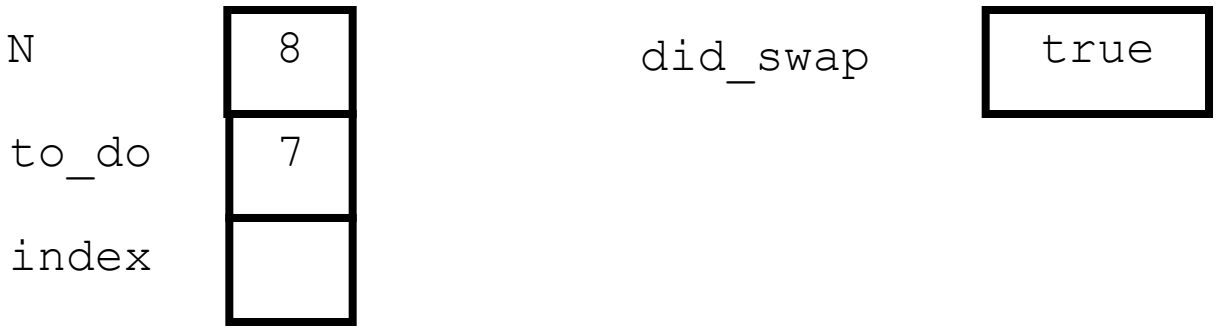
- **We can use a boolean variable to determine if any swapping occurred during the "bubble up."**

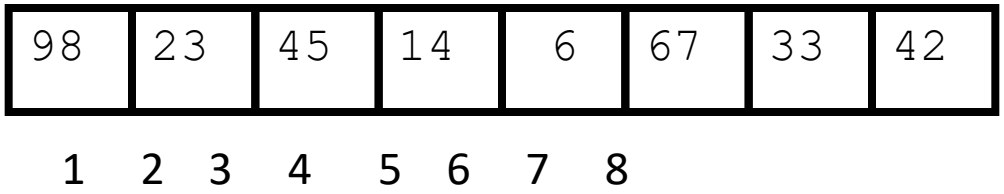
- **If no swapping occurred, then we know that the collection is already sorted!**

- **This boolean "flag" needs to be reset after each "bubble up."**

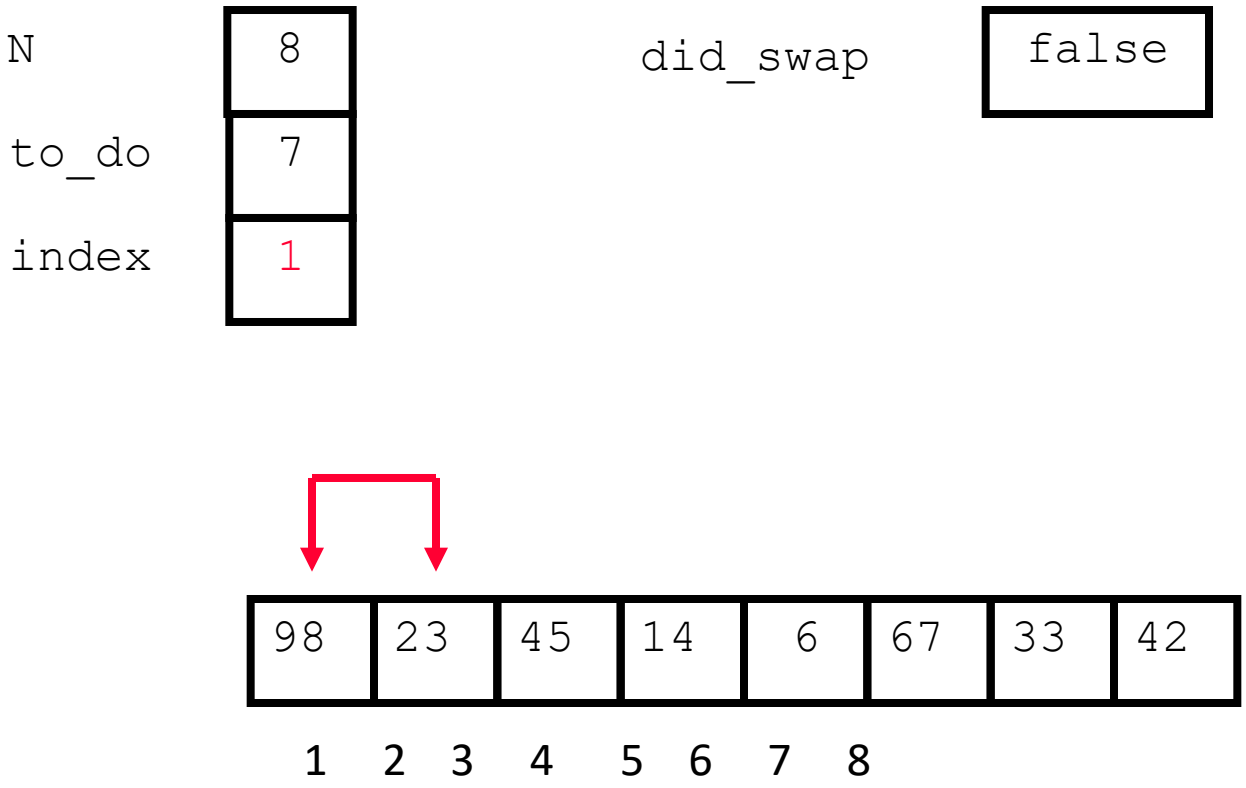
```
did_swap isoftype Boolean
did_swap <- true

loop
  exitif ((to_do = 0) OR NOT(did_swap))
  index <- 1
  did_swap <- false
  loop
    exitif(index > to_do)
    if(A[index] > A[index + 1]) then
      Swap(A[index], A[index + 1])
      did_swap <- true
    endif
    index <- index + 1
  endloop
  to_do <- to_do - 1
endloop
```

An Animated Example

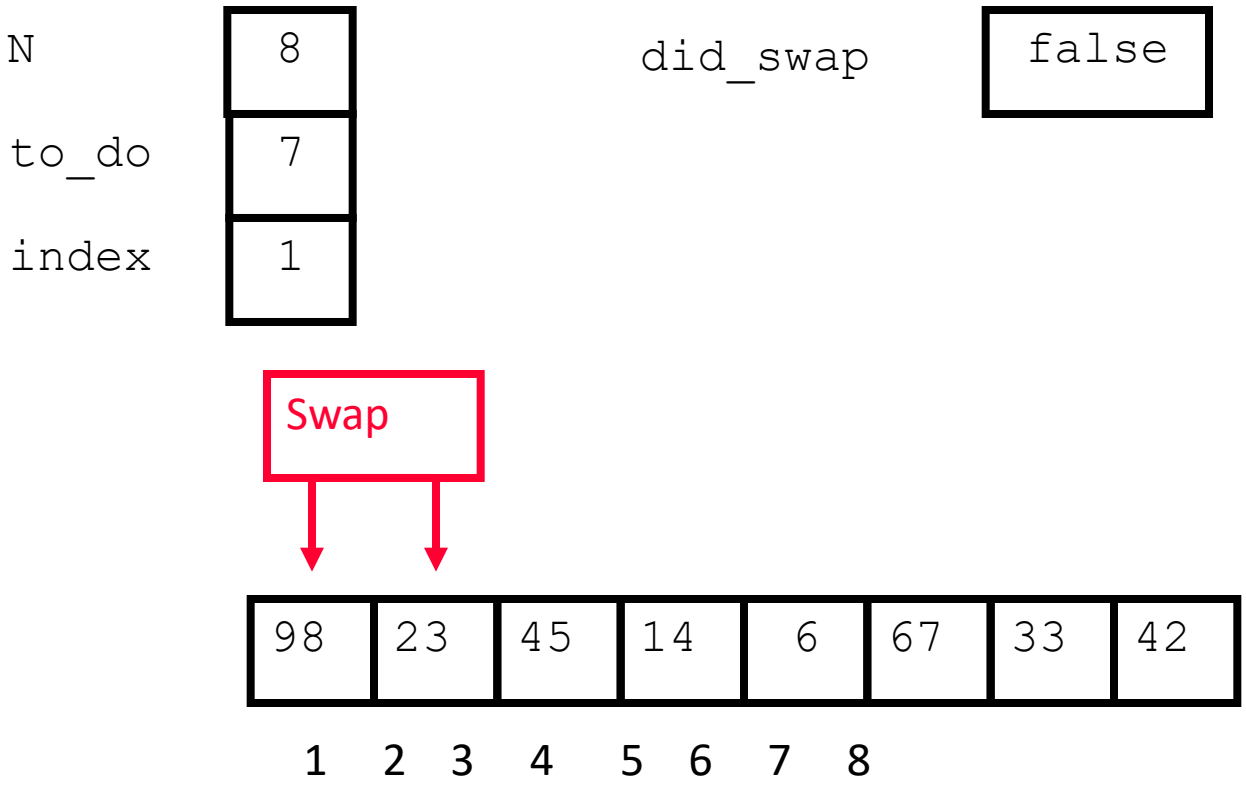




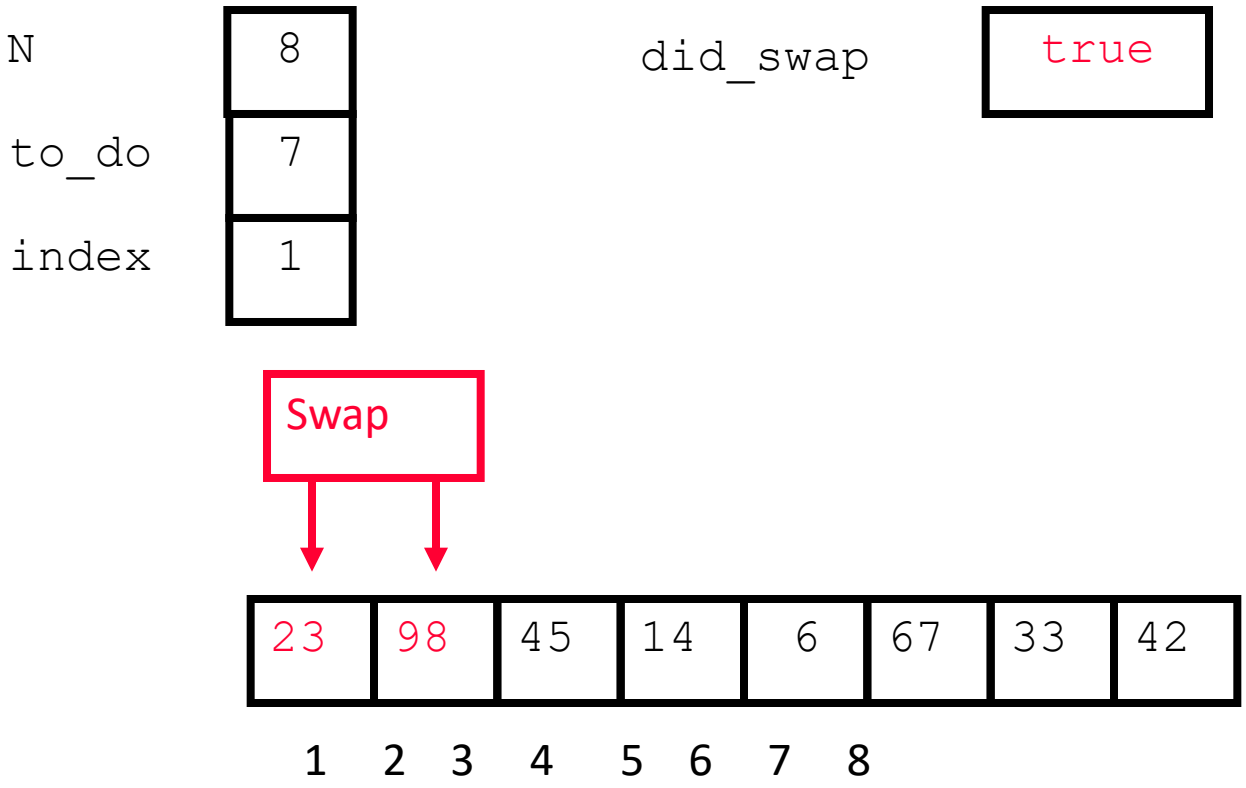
An Animated Example



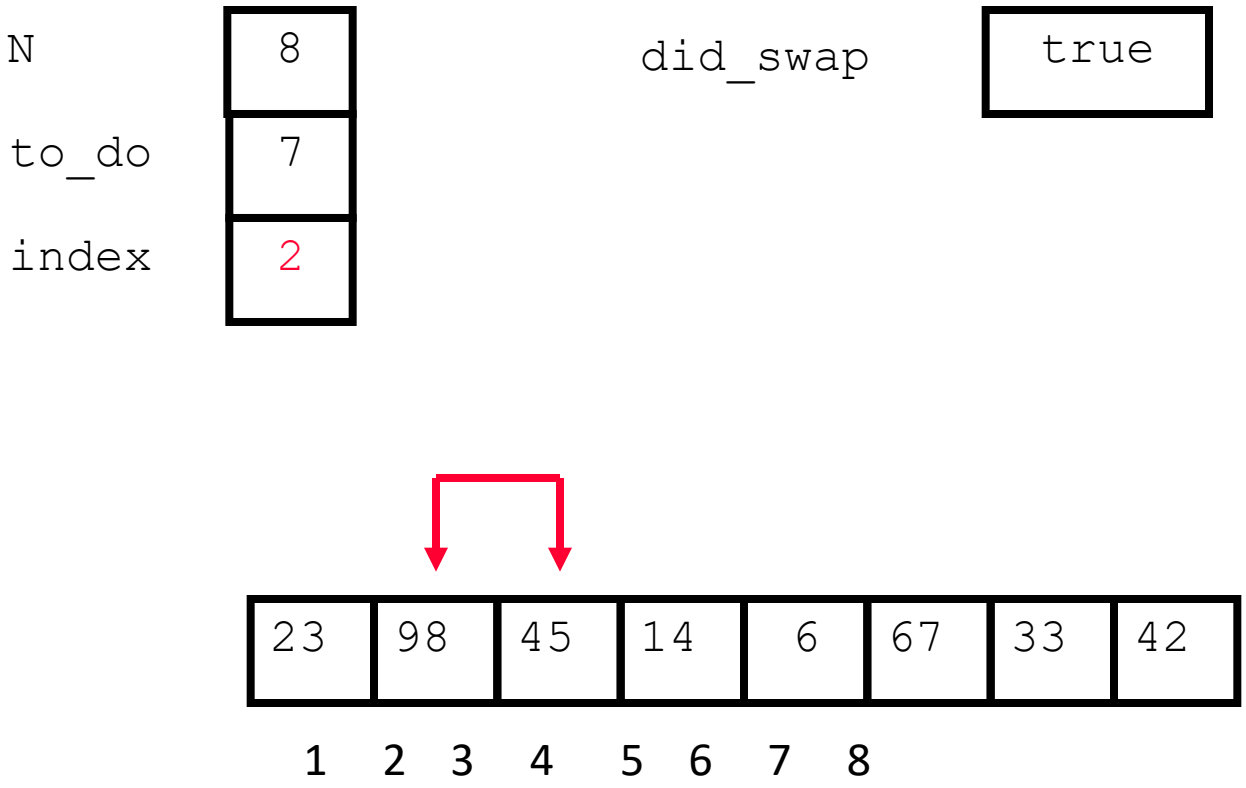
An Animated Example



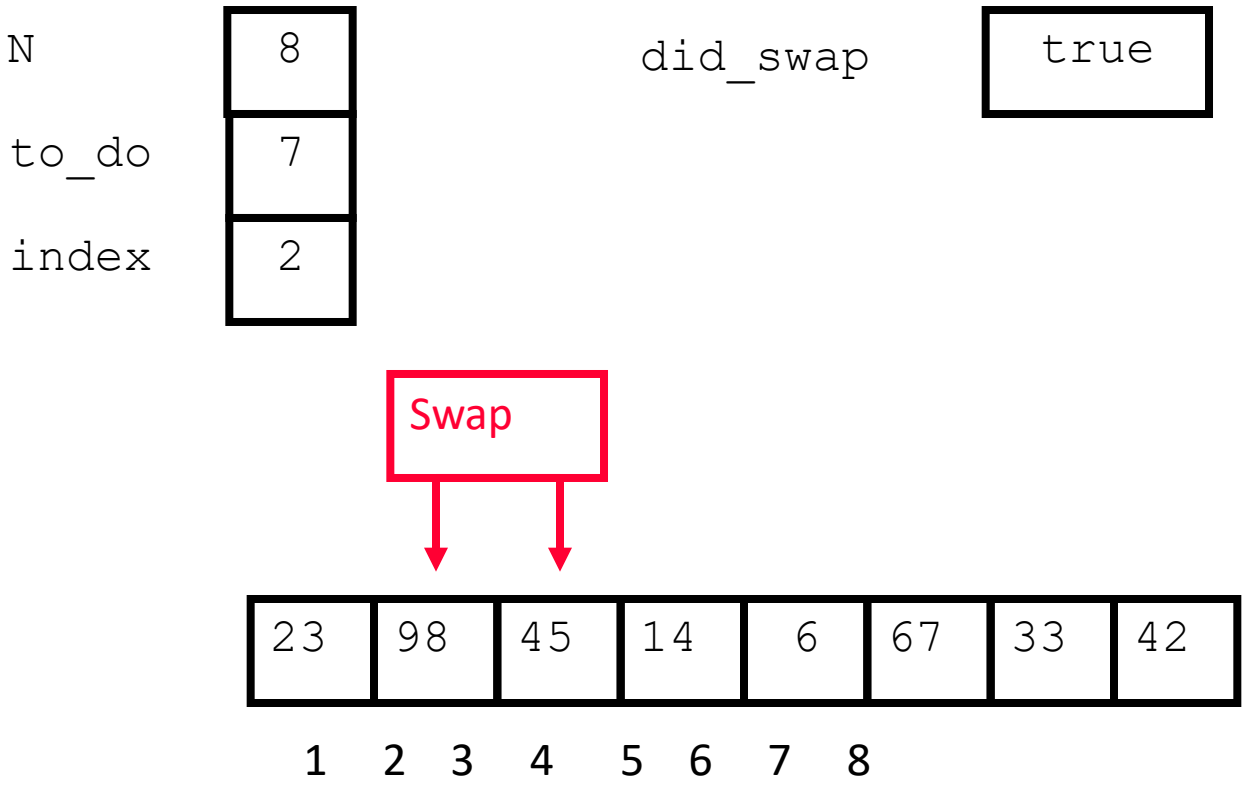
An Animated Example



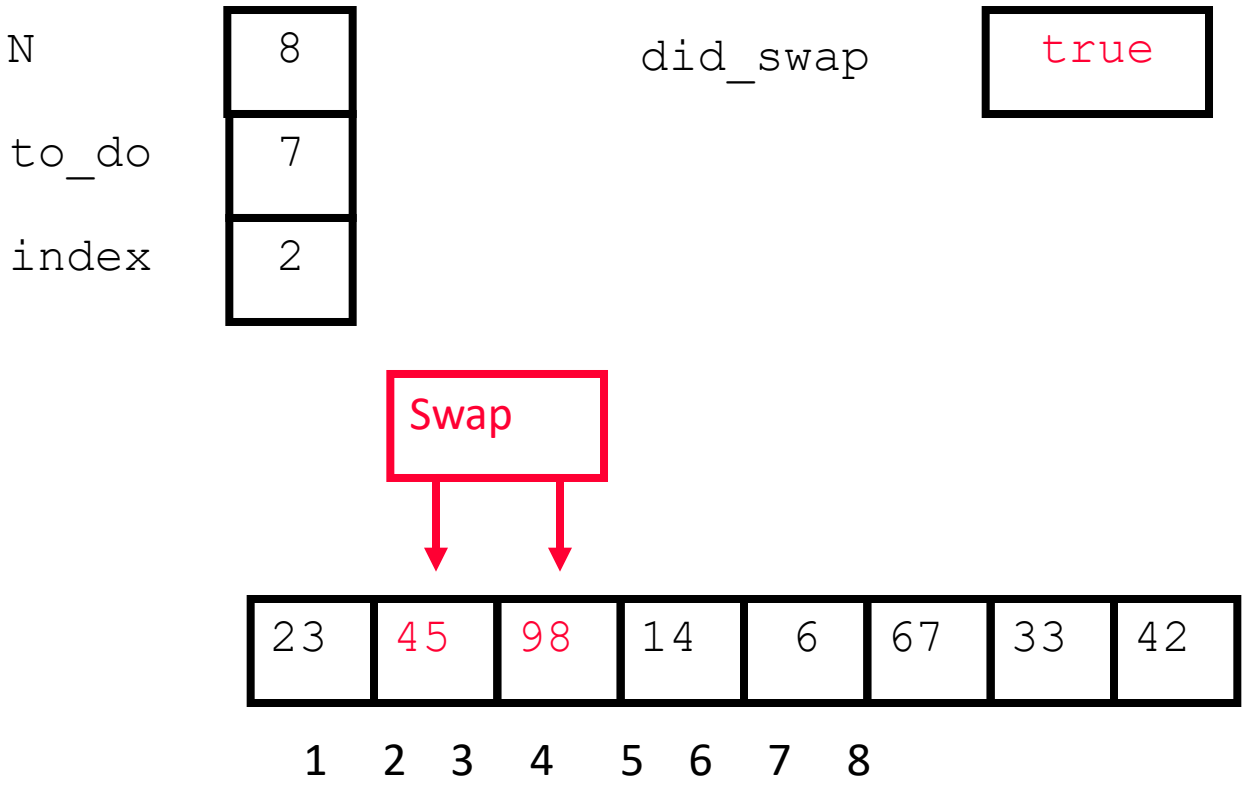
An Animated Example



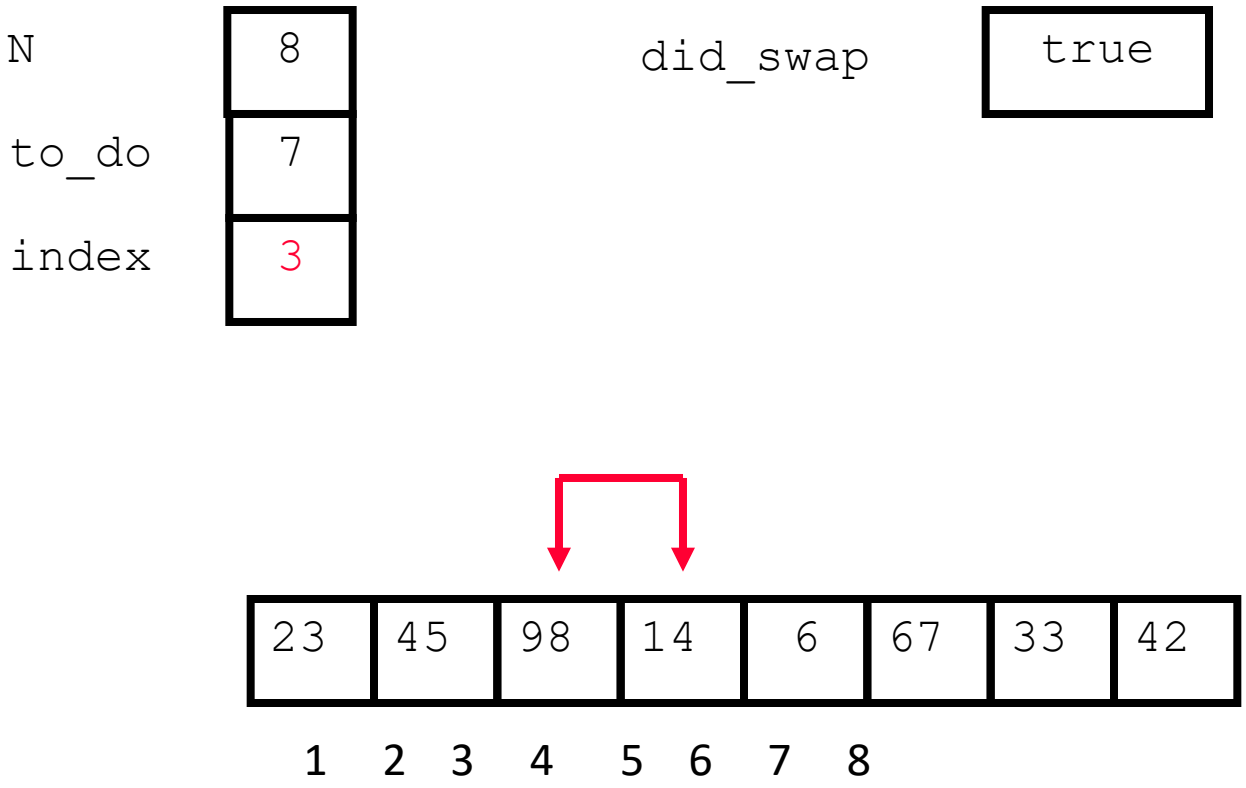
An Animated Example



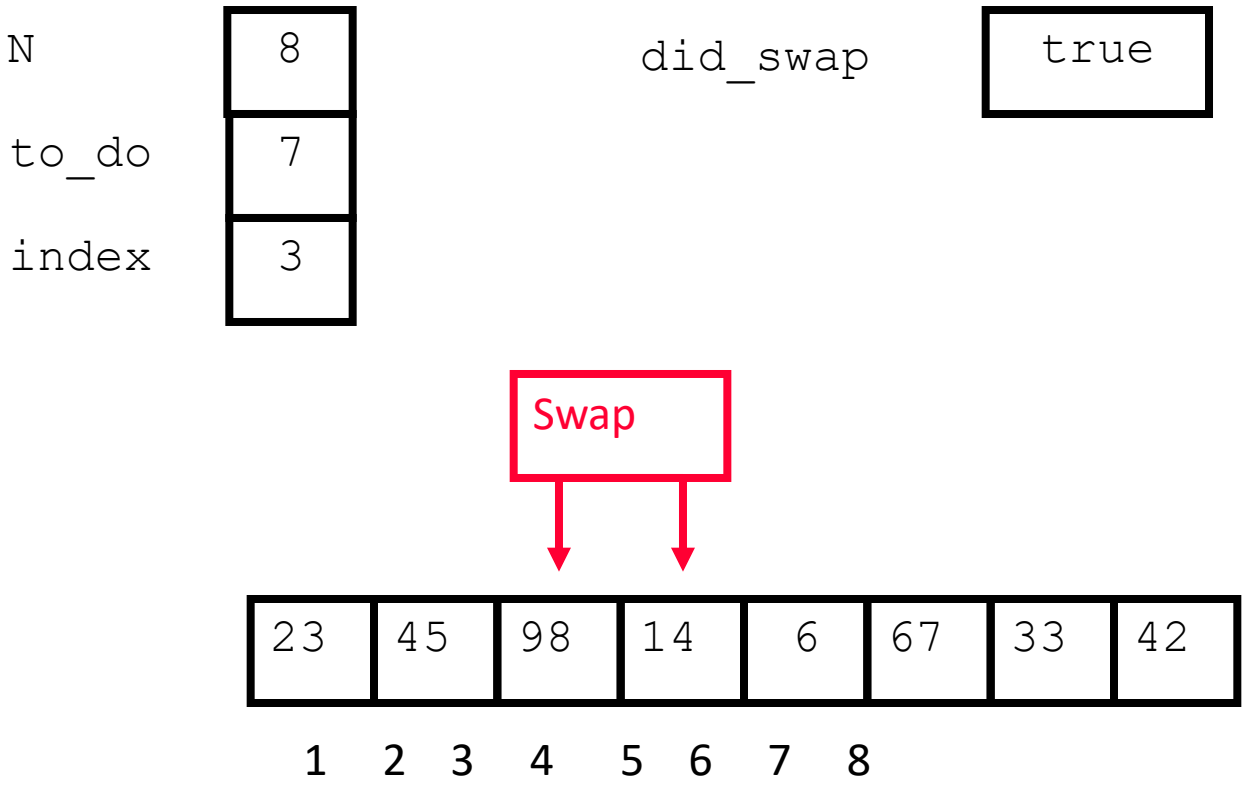
An Animated Example



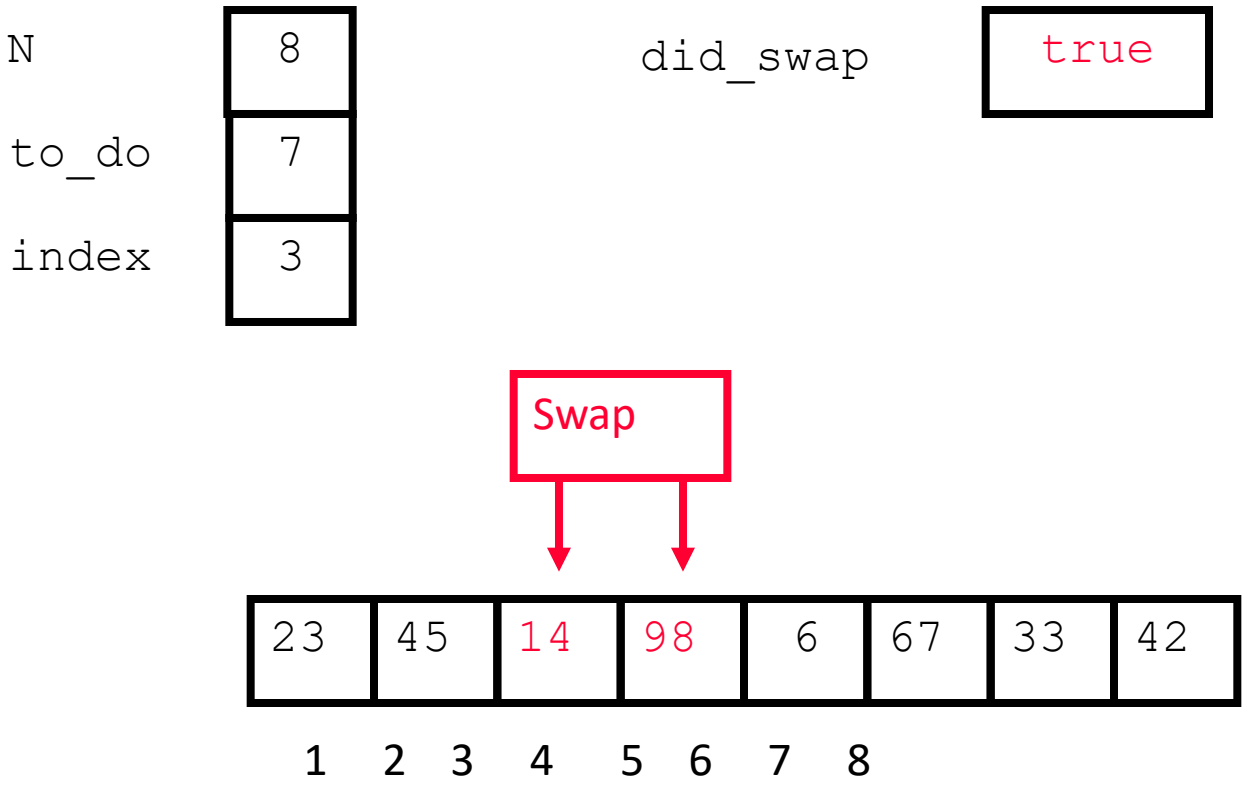
An Animated Example



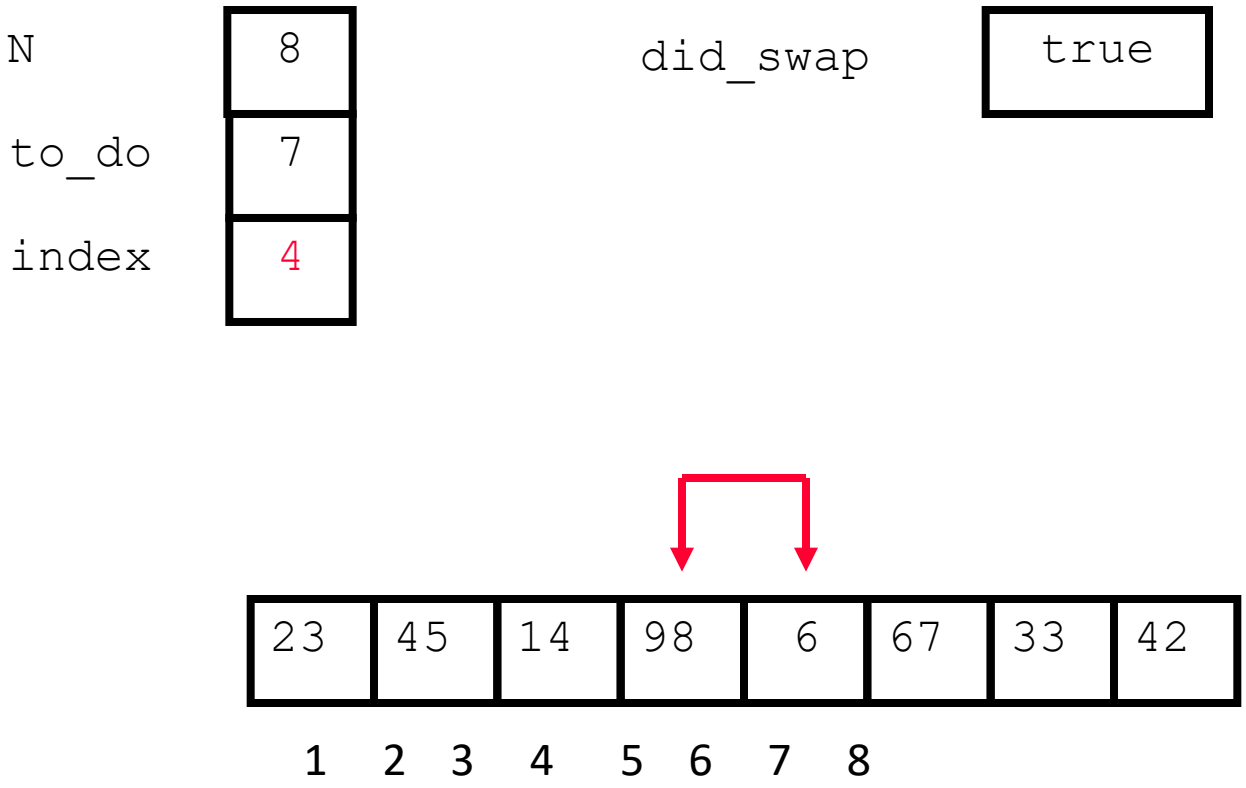
An Animated Example



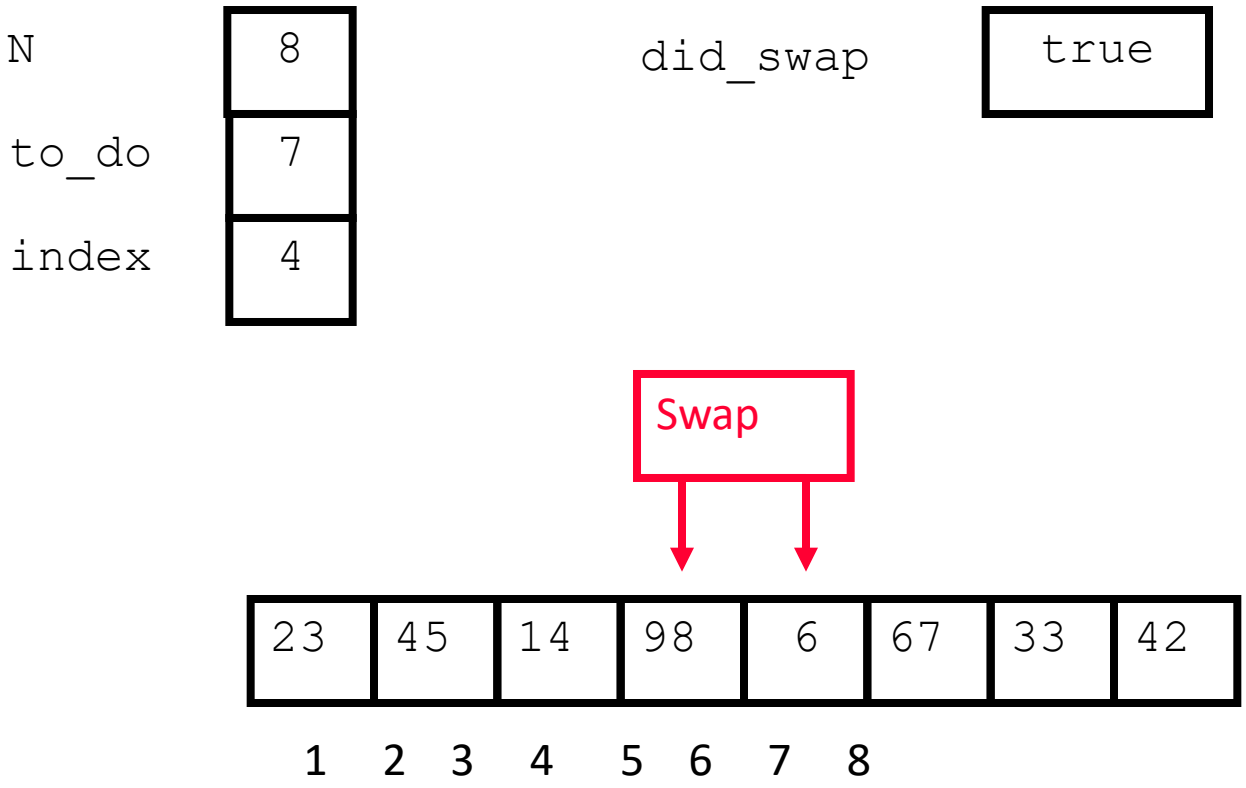
An Animated Example



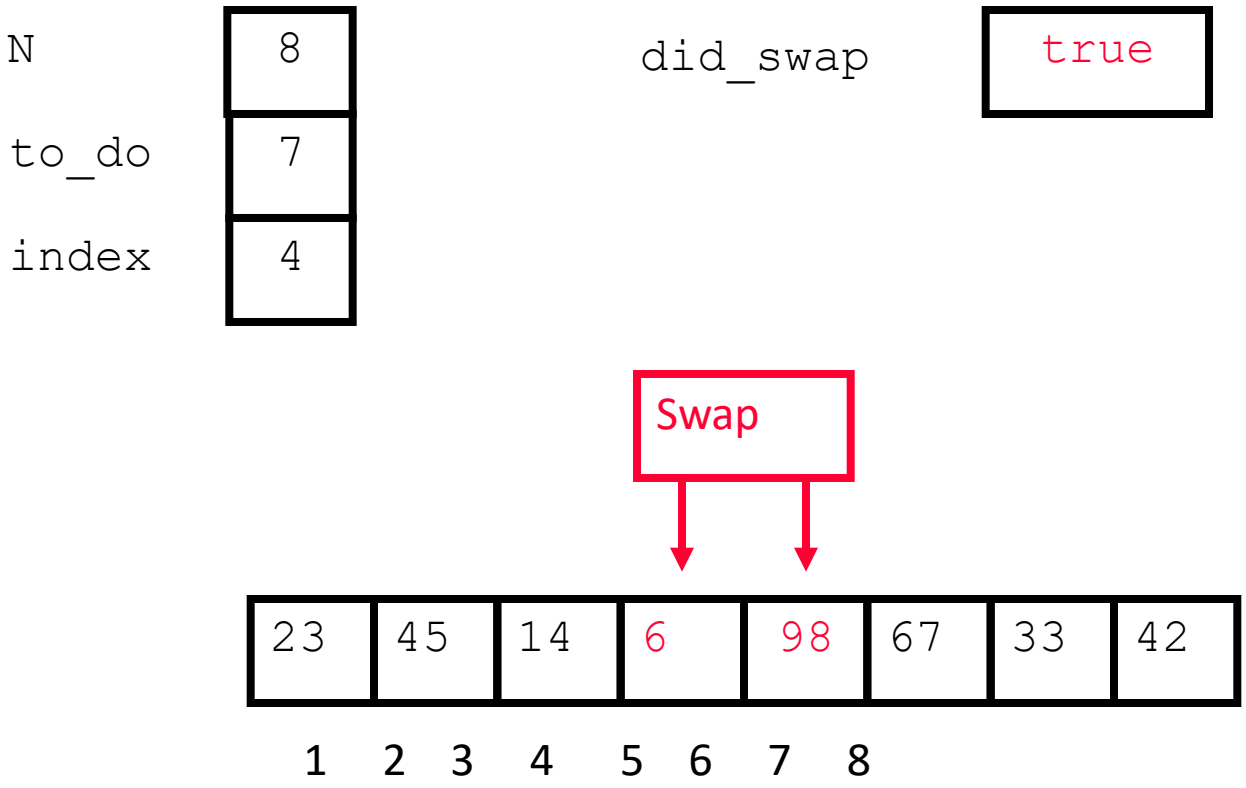
An Animated Example



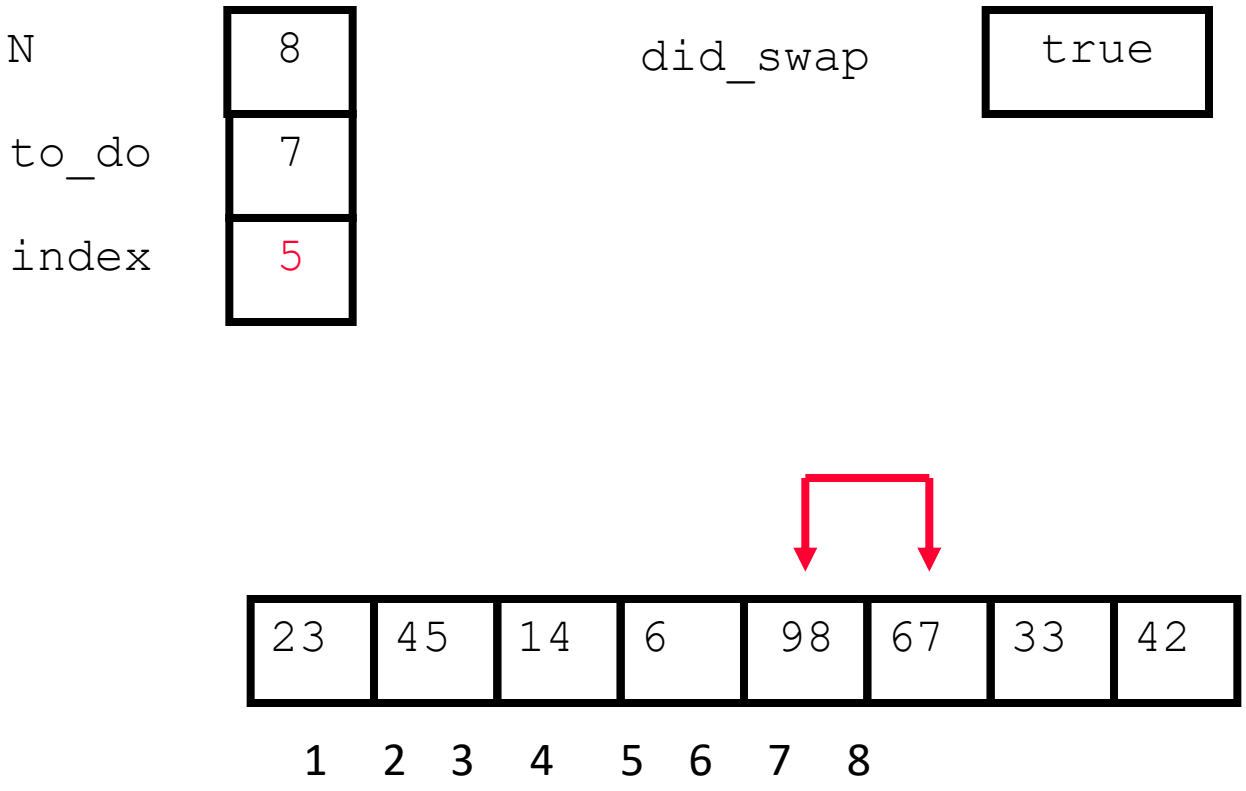
An Animated Example



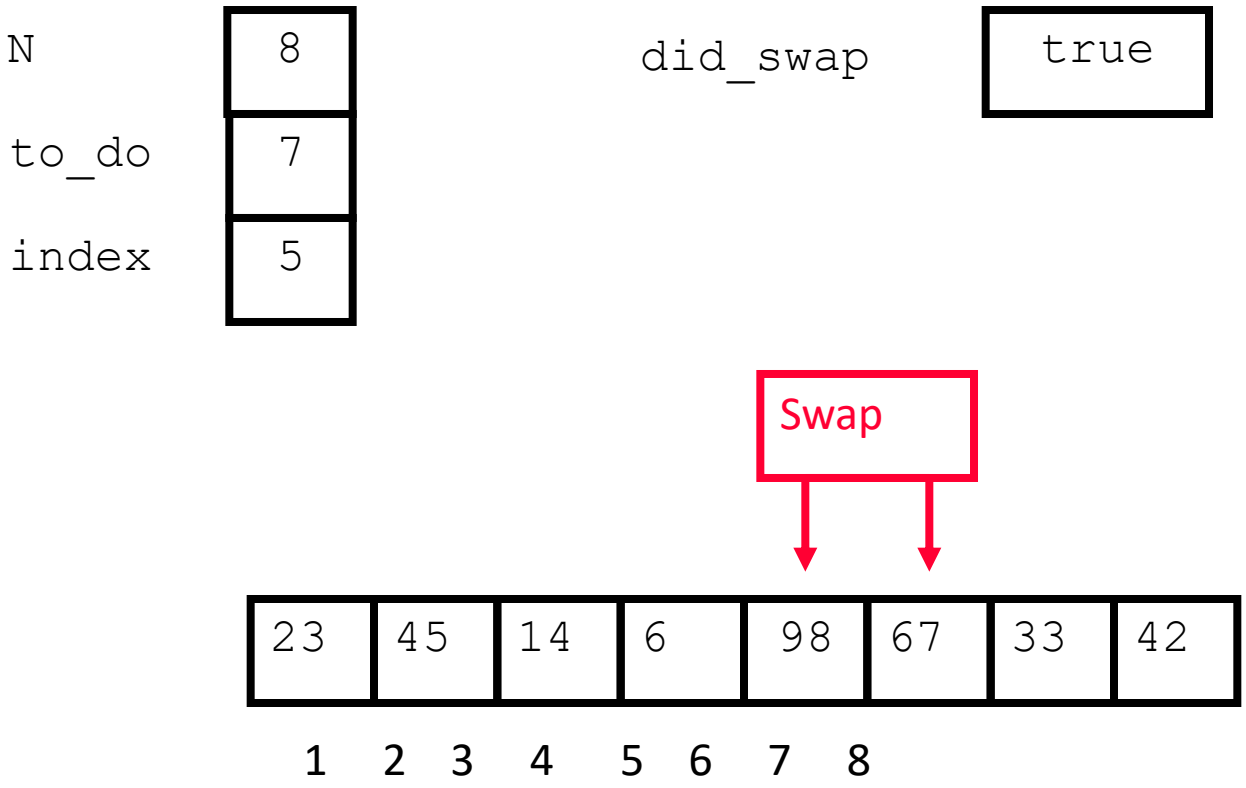
An Animated Example



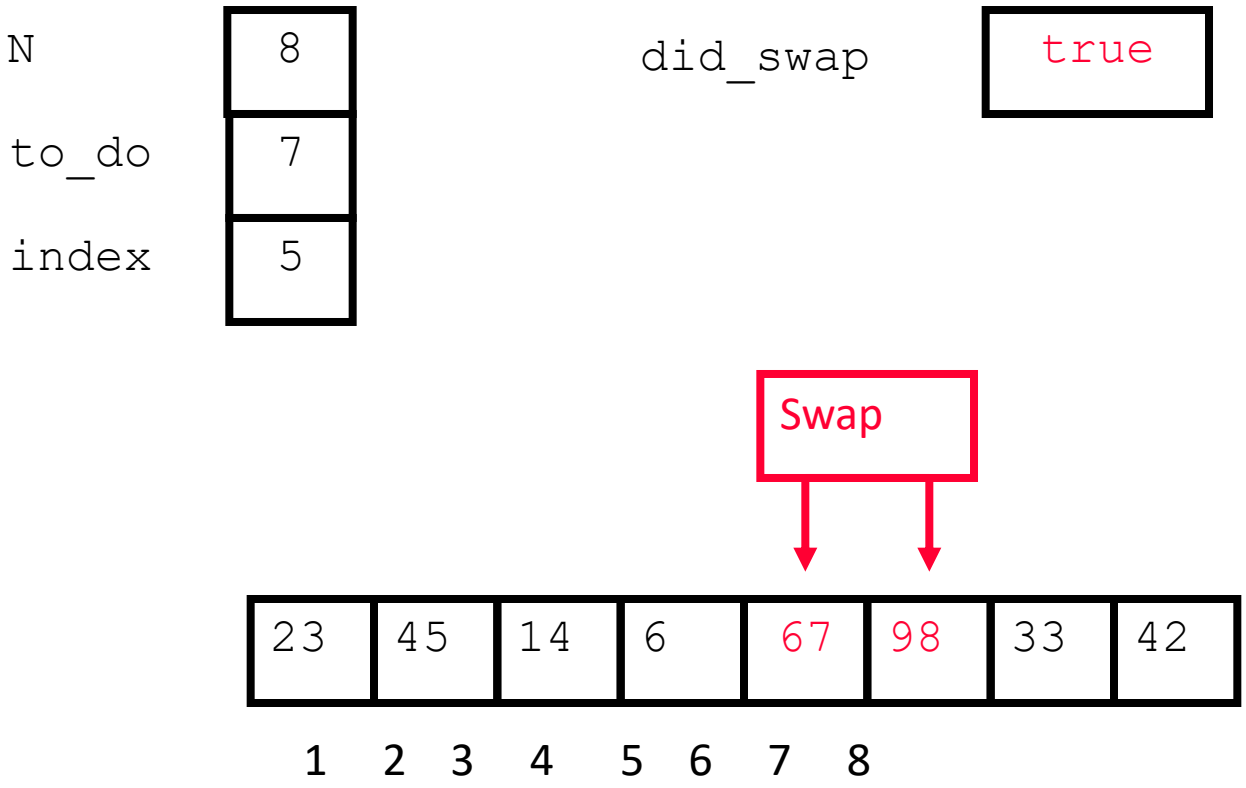
An Animated Example



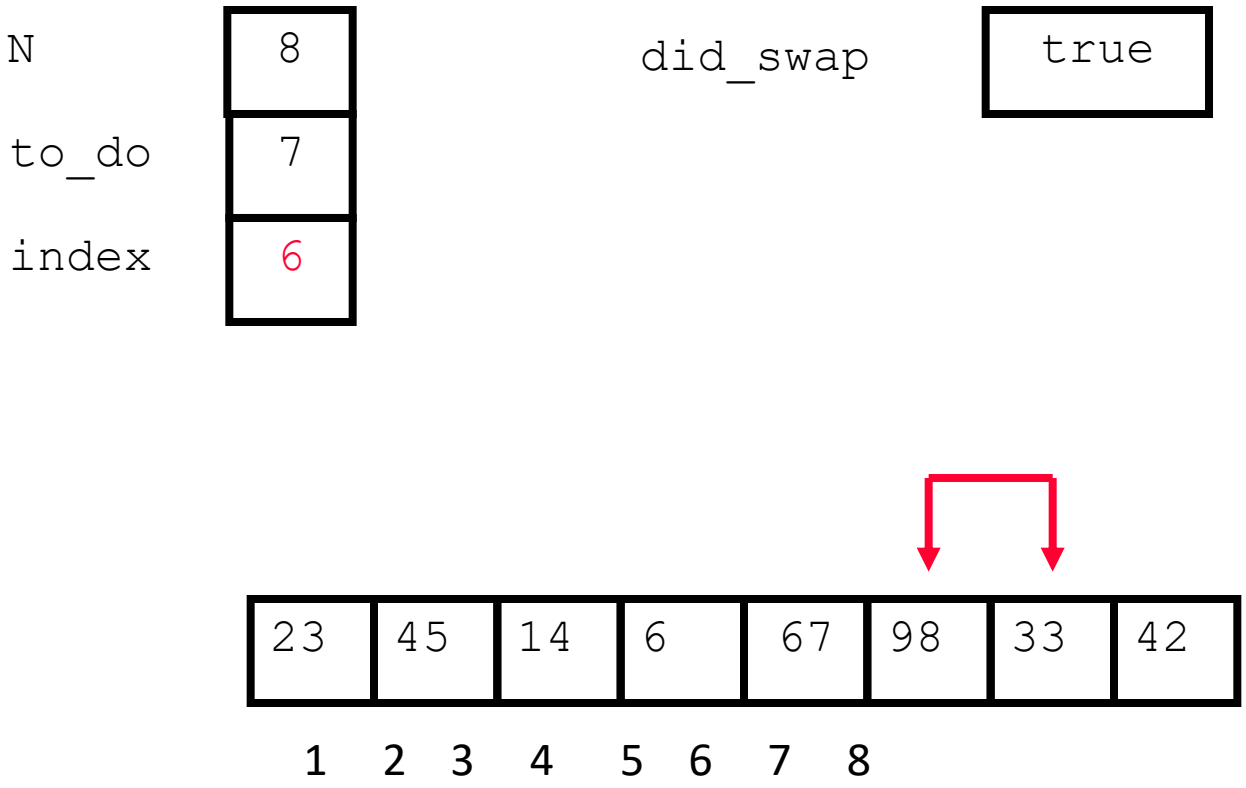
An Animated Example



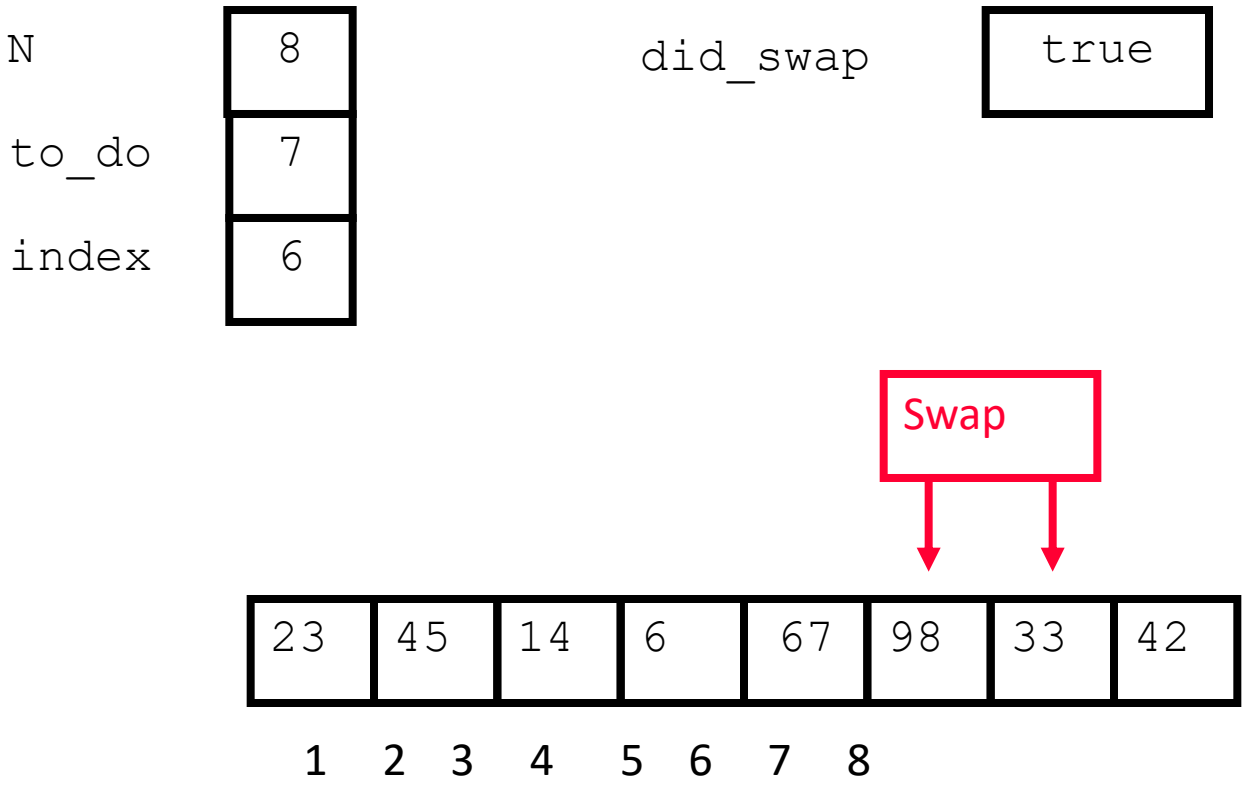
An Animated Example



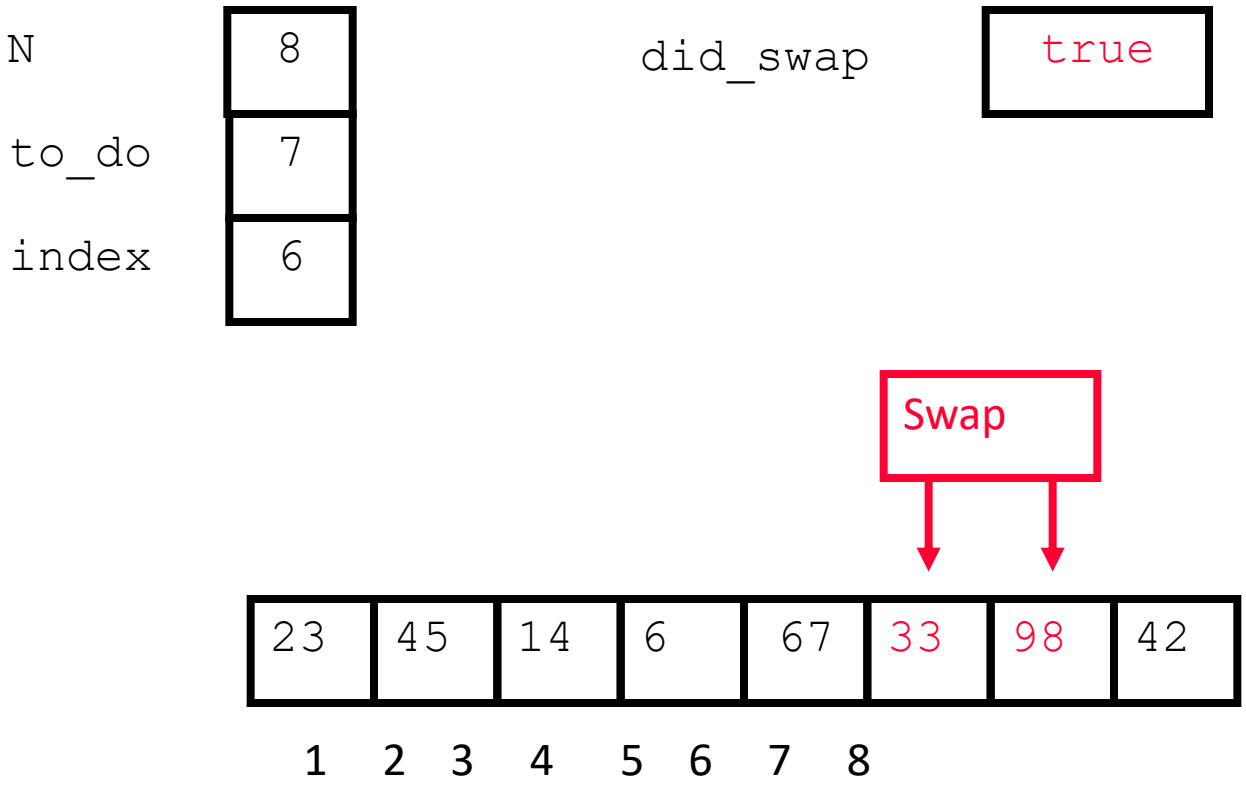
An Animated Example



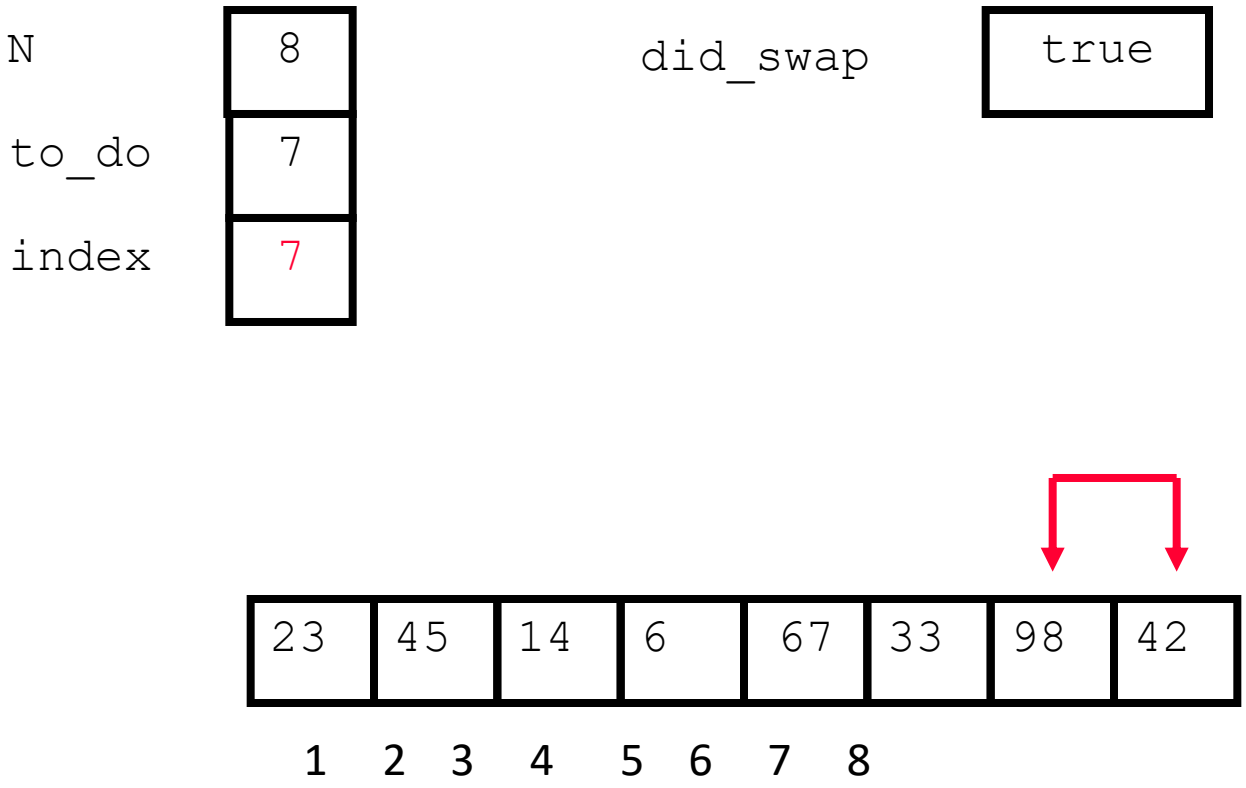
An Animated Example



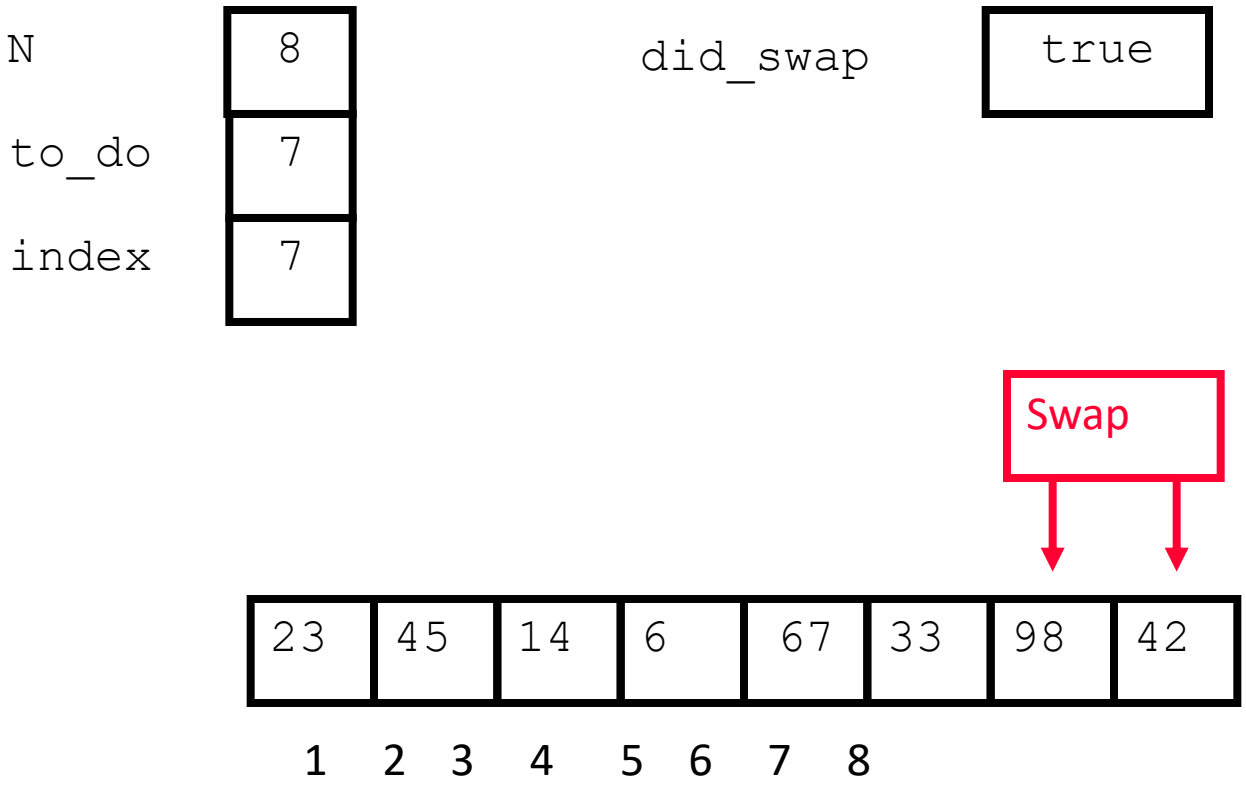
An Animated Example



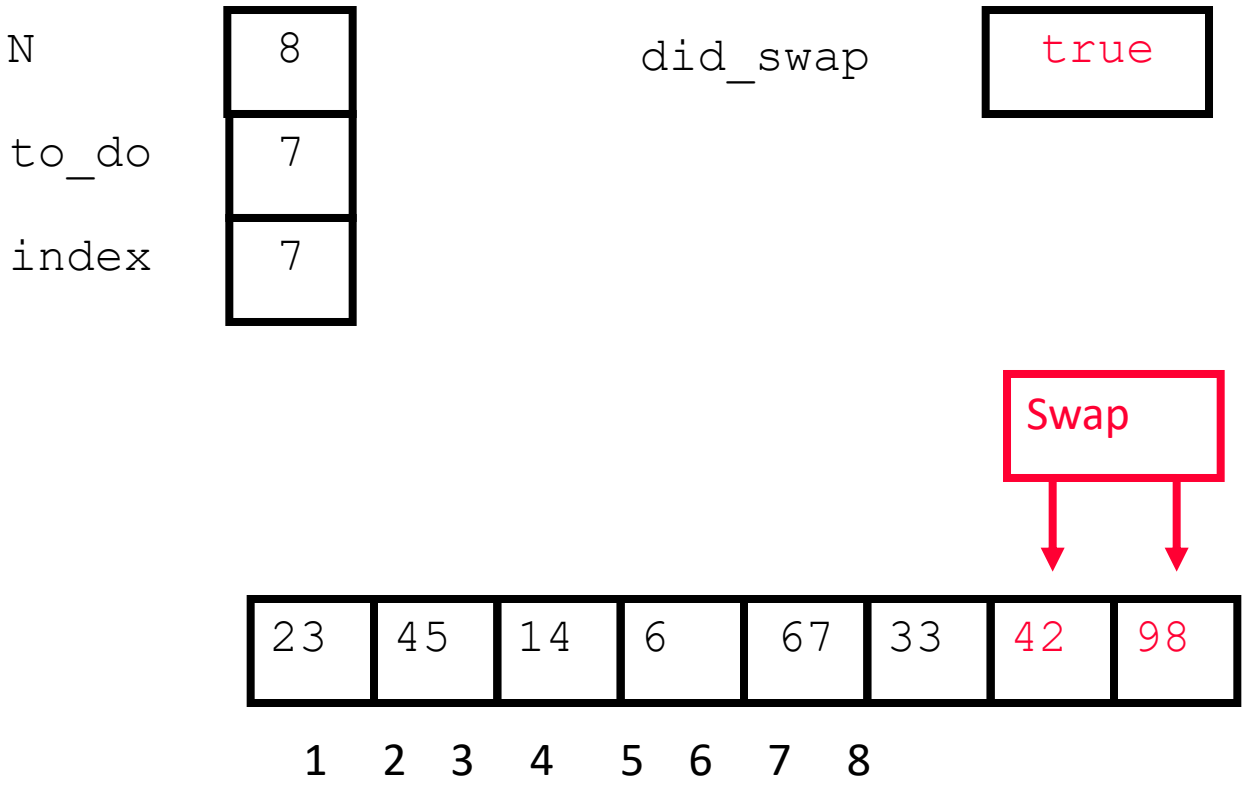
An Animated Example



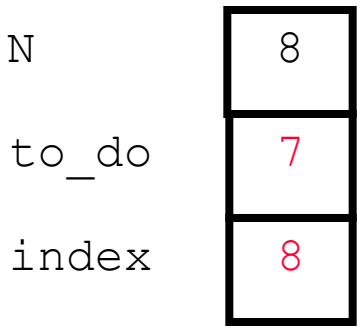
An Animated Example



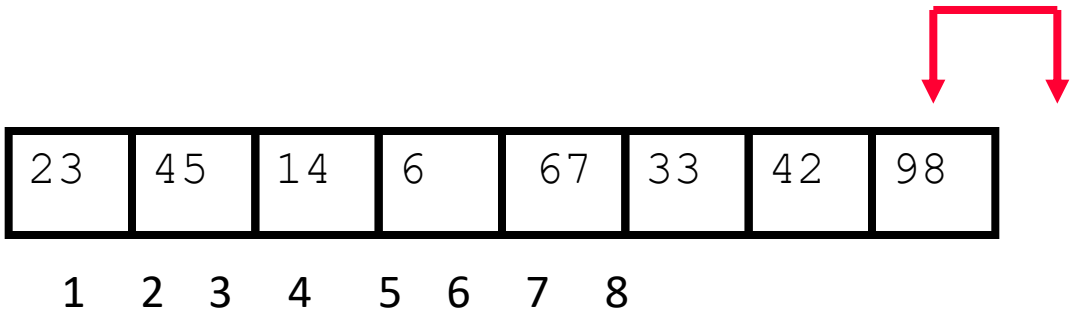
An Animated Example



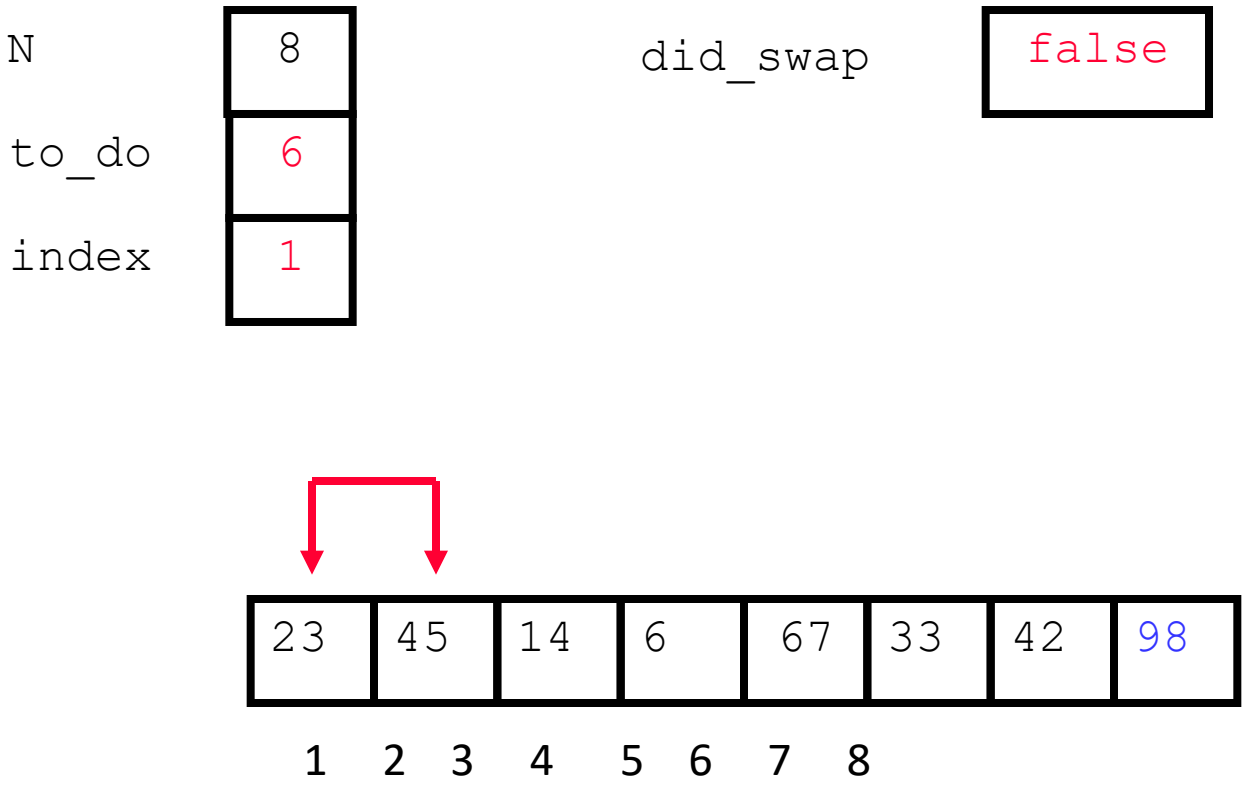
After First Pass of Outer Loop



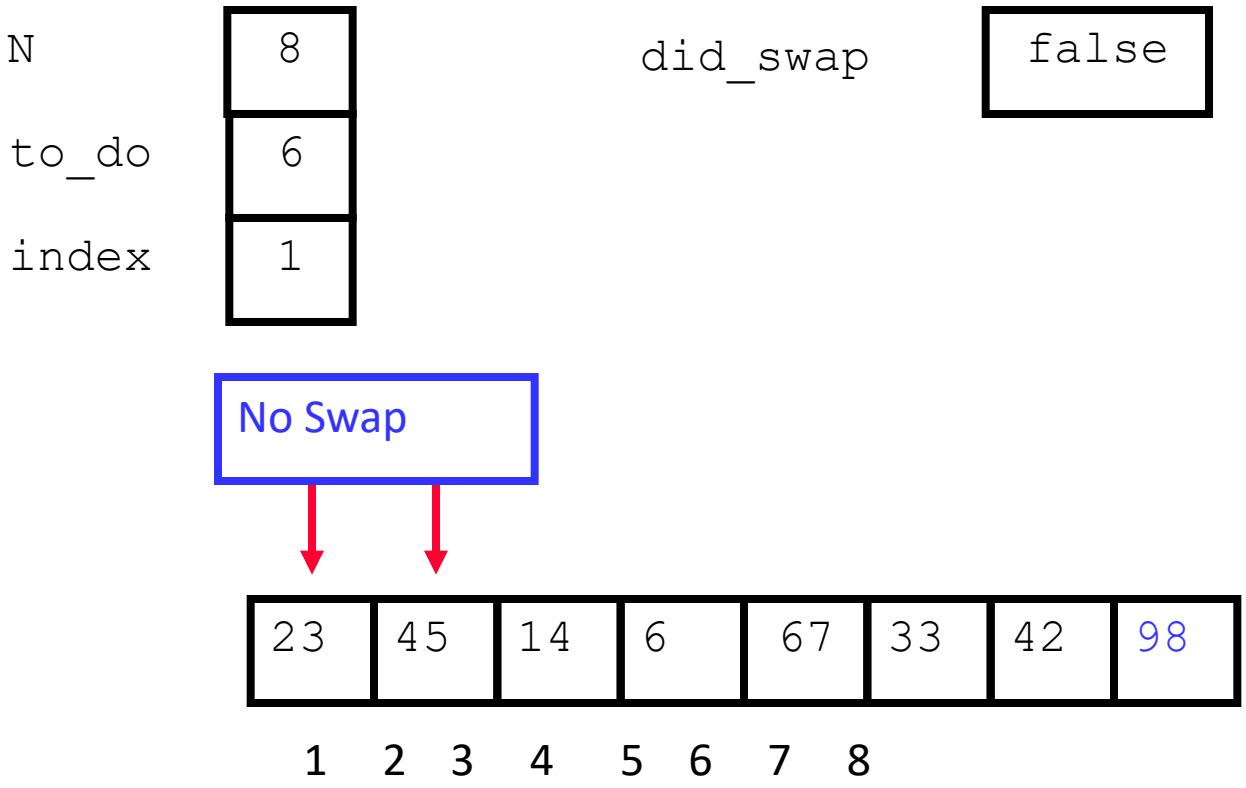
Finished first “Bubble Up”



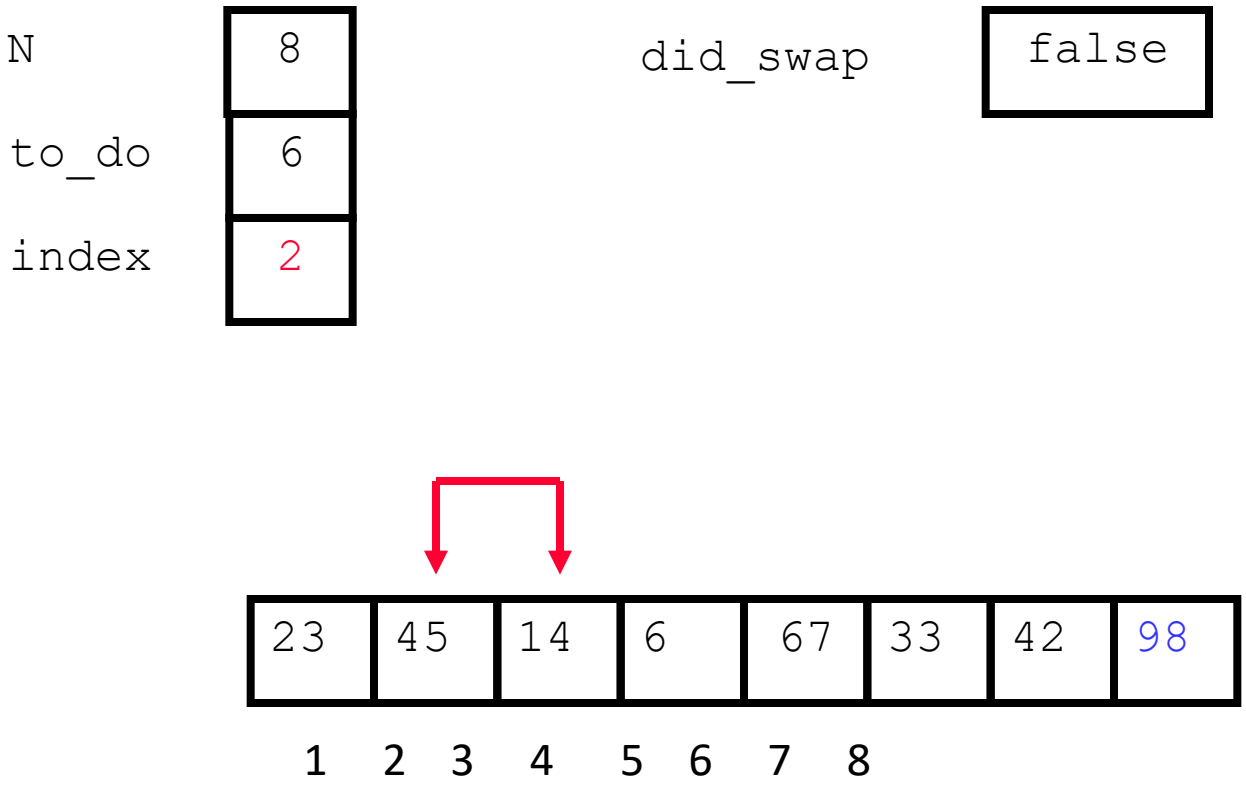
The Second “Bubble Up”



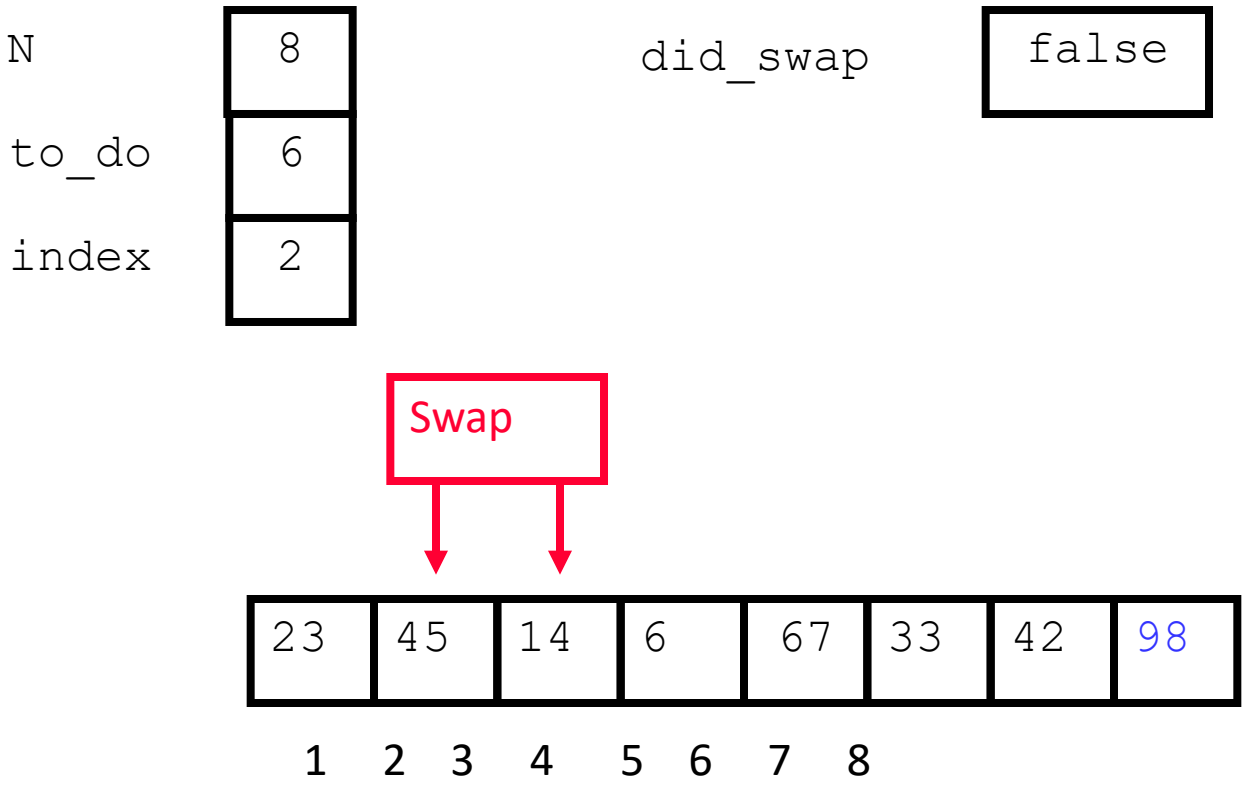
The Second “Bubble Up”



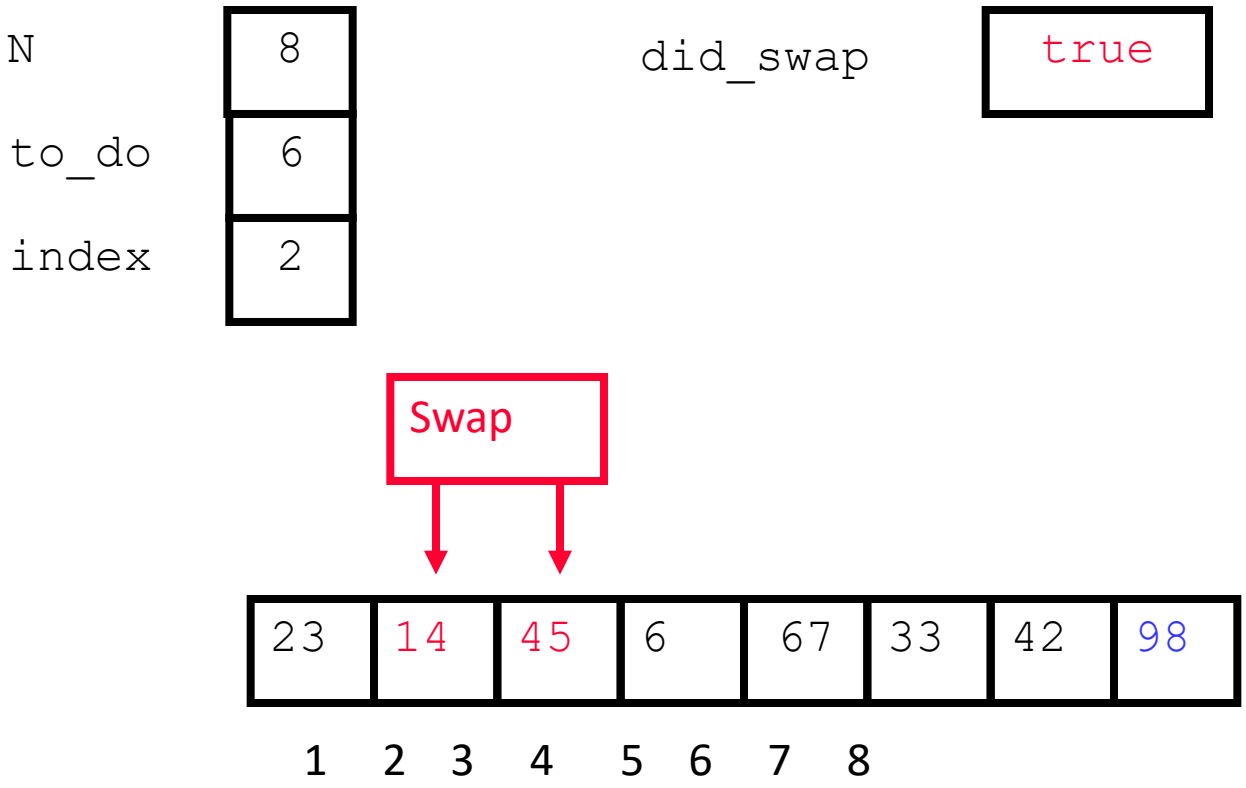
The Second “Bubble Up”



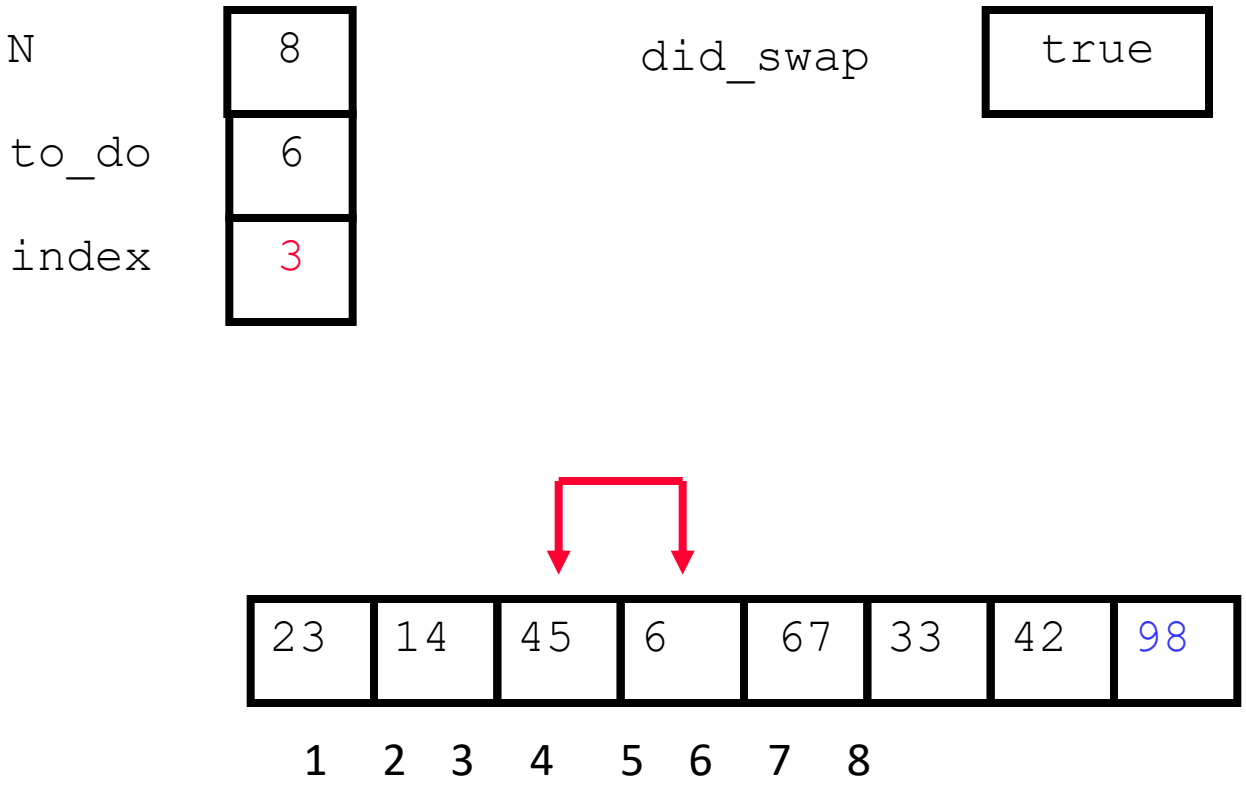
The Second “Bubble Up”



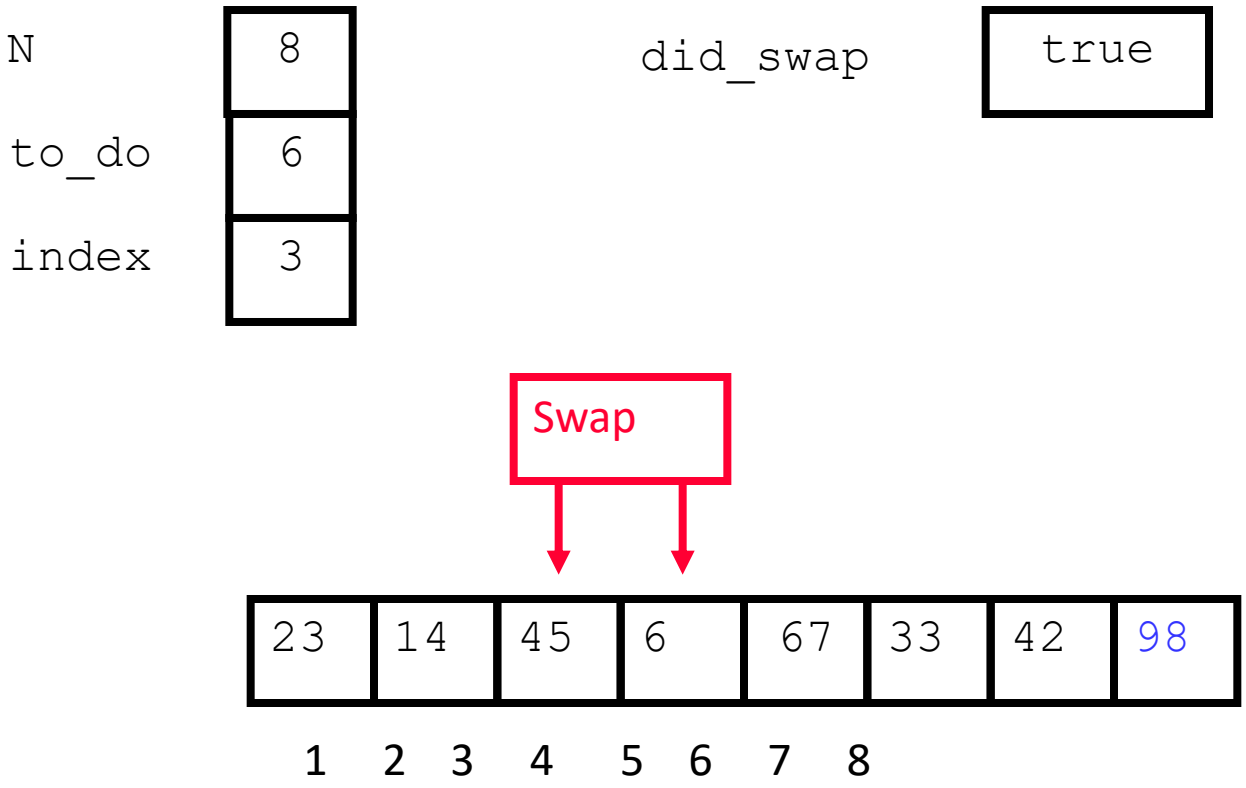
The Second “Bubble Up”



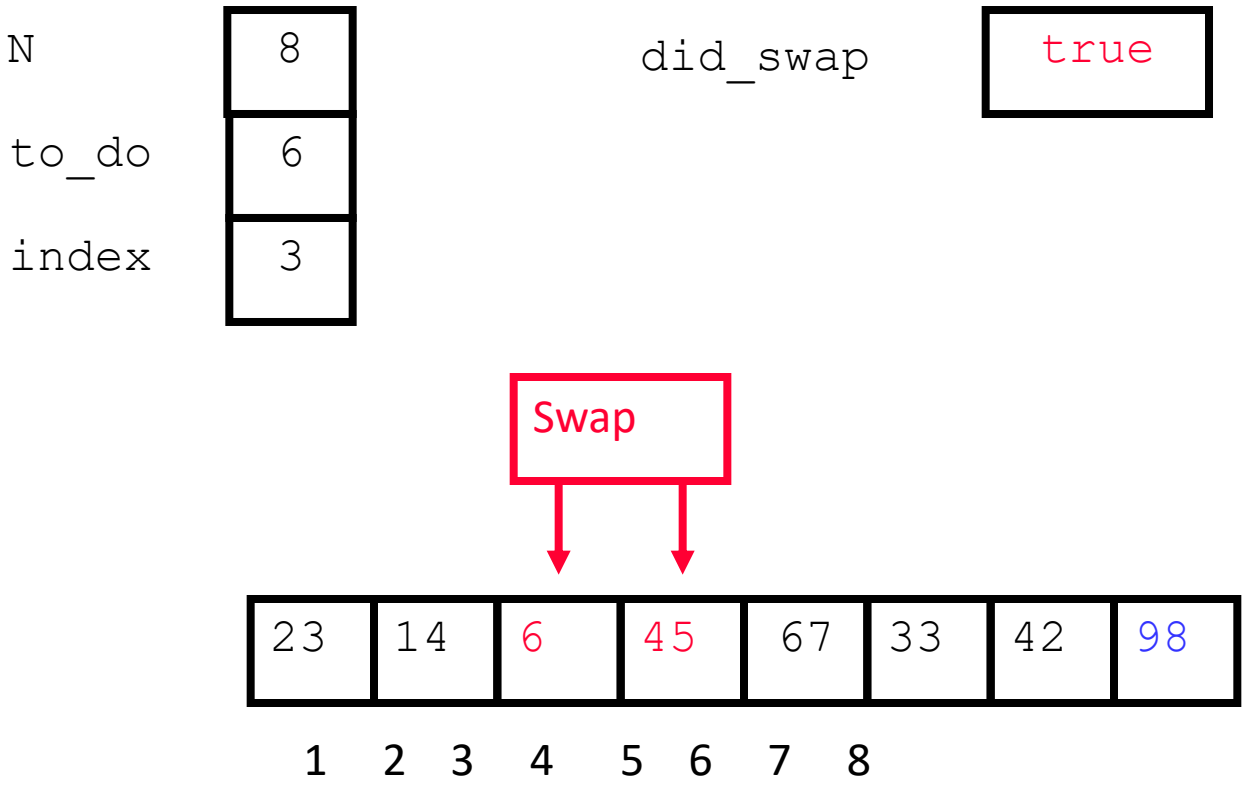
The Second “Bubble Up”



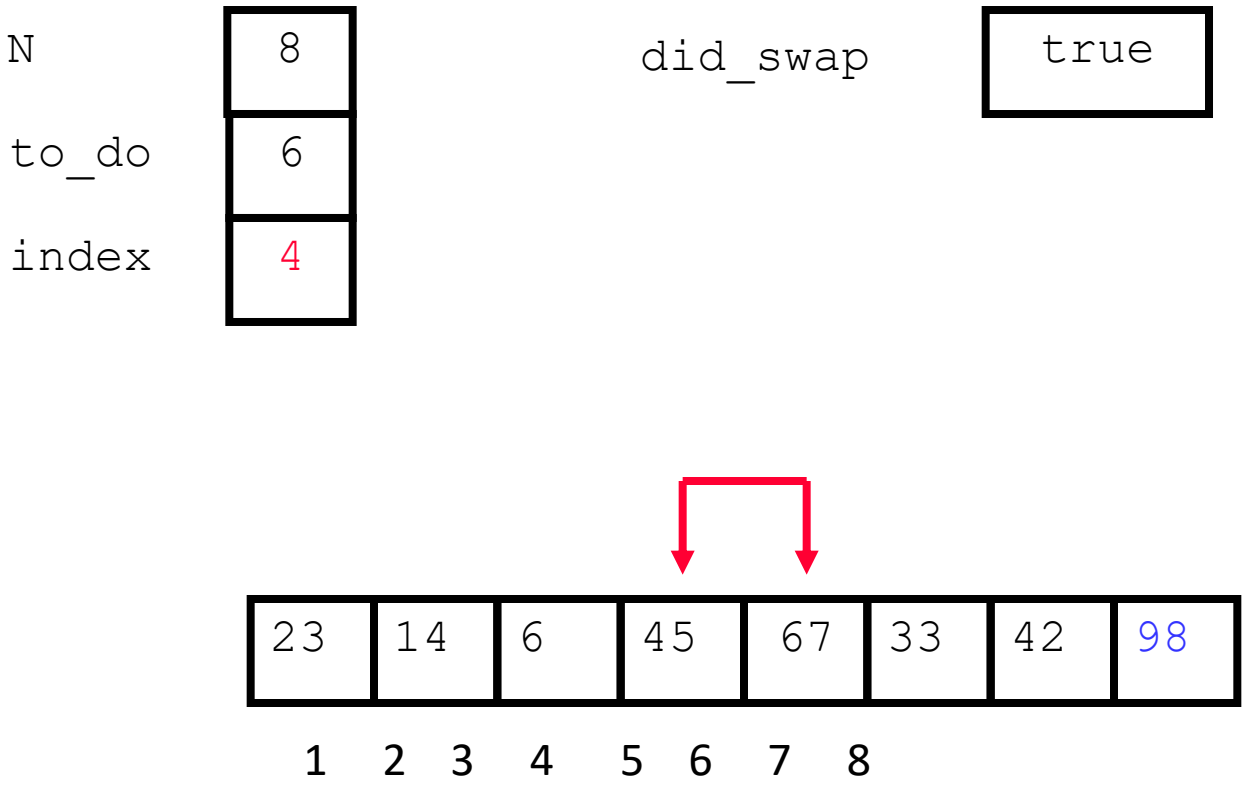
The Second “Bubble Up”



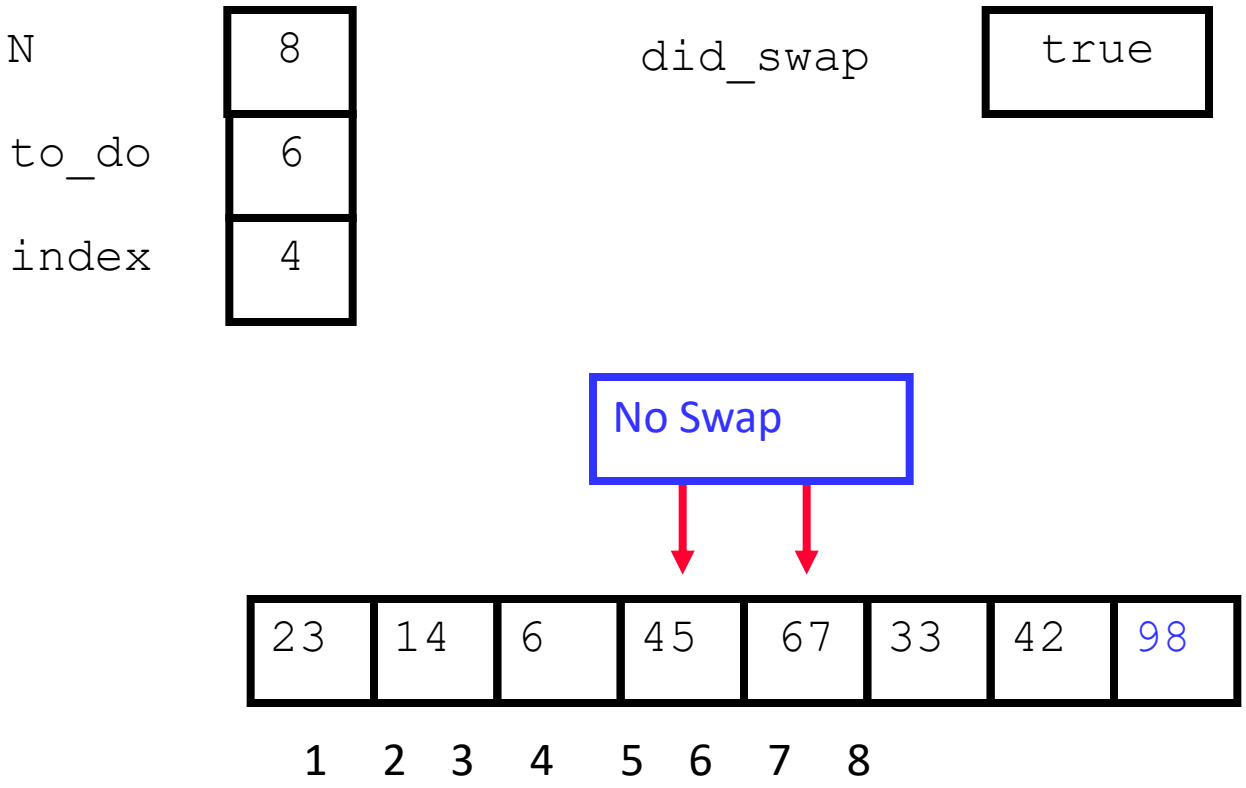
The Second “Bubble Up”



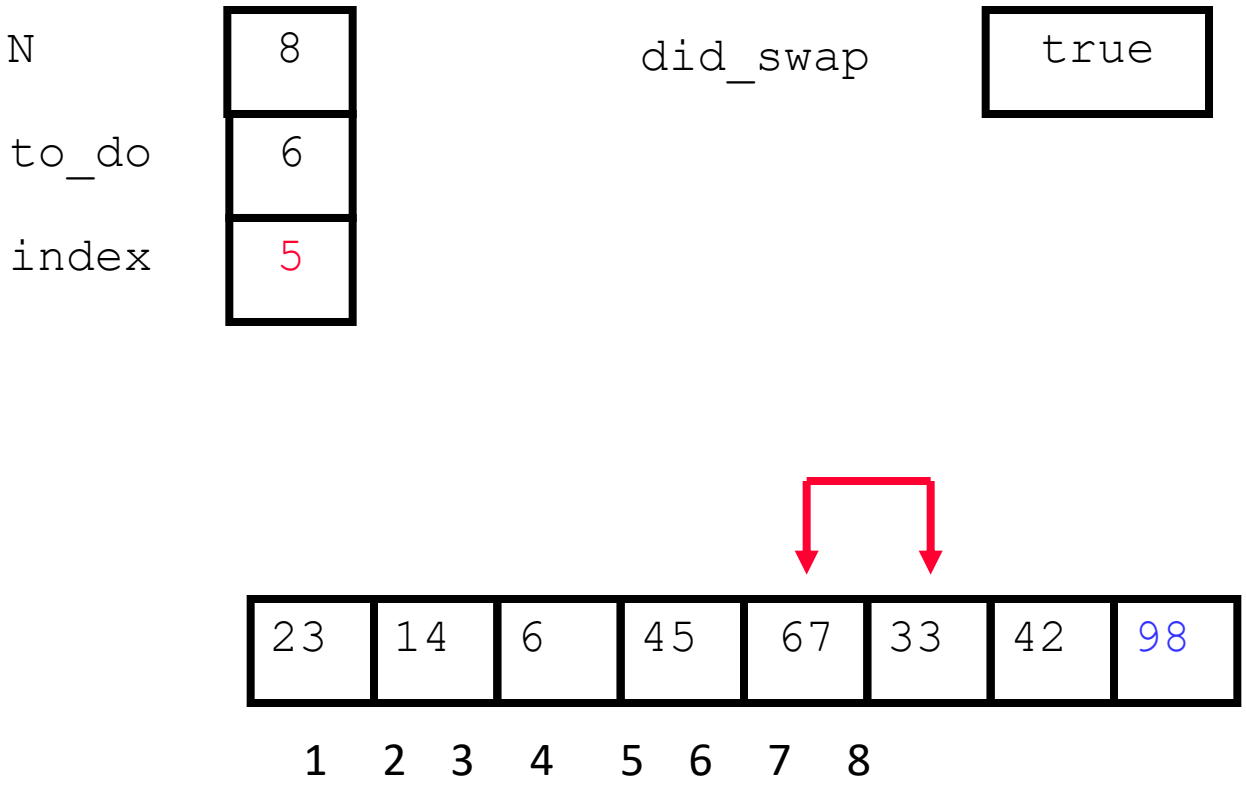
The Second “Bubble Up”



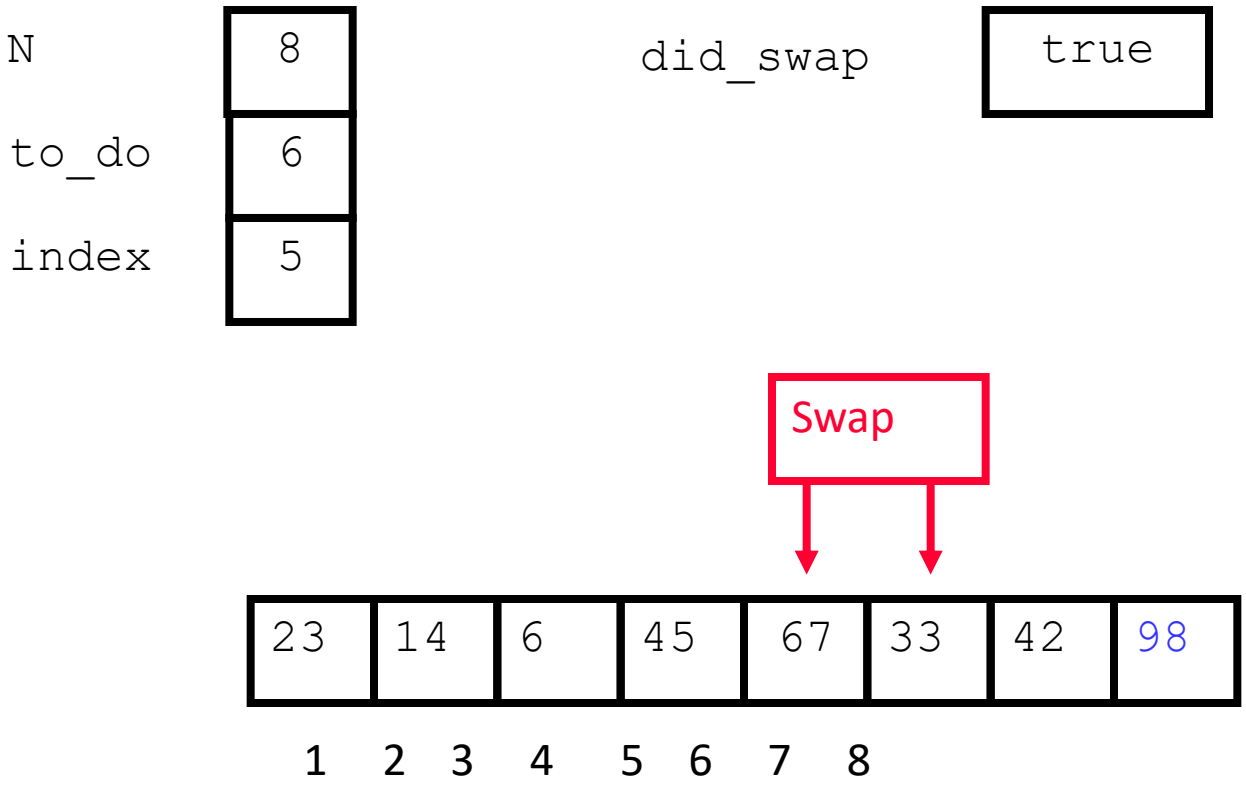
The Second “Bubble Up”



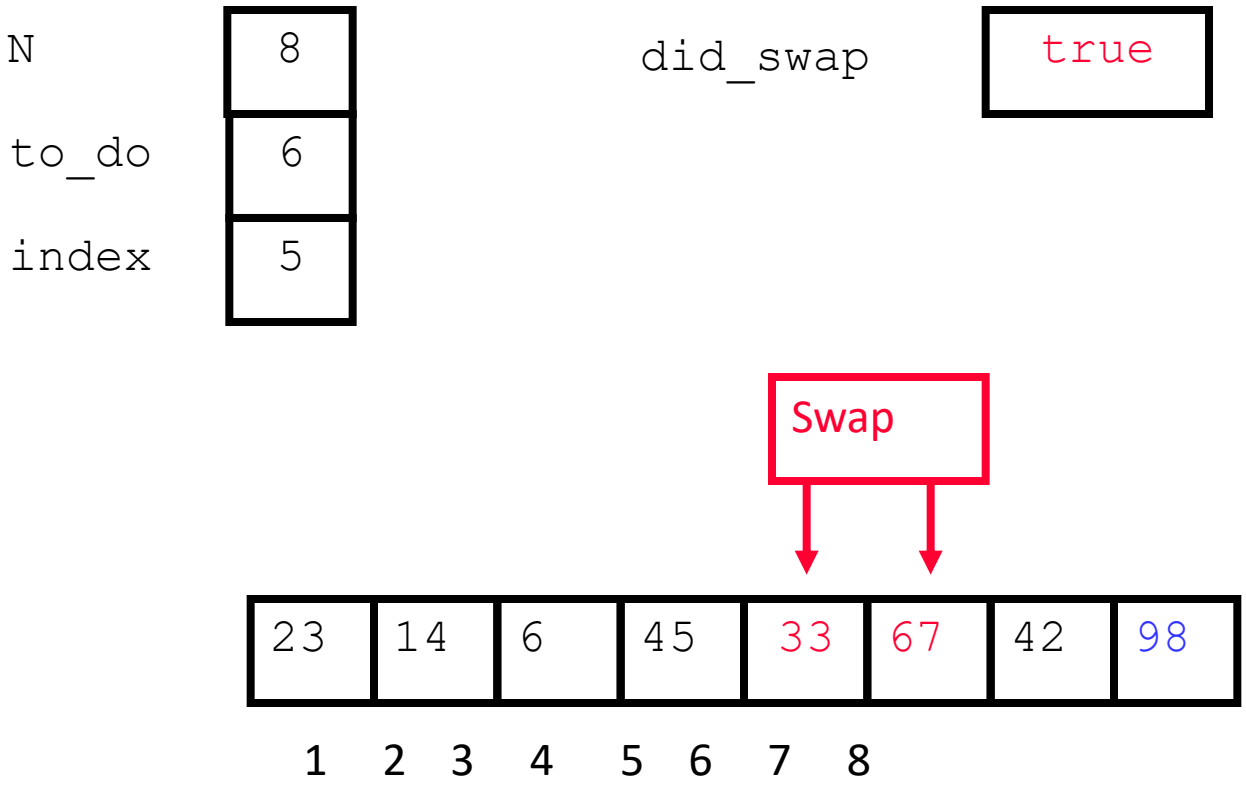
The Second “Bubble Up”



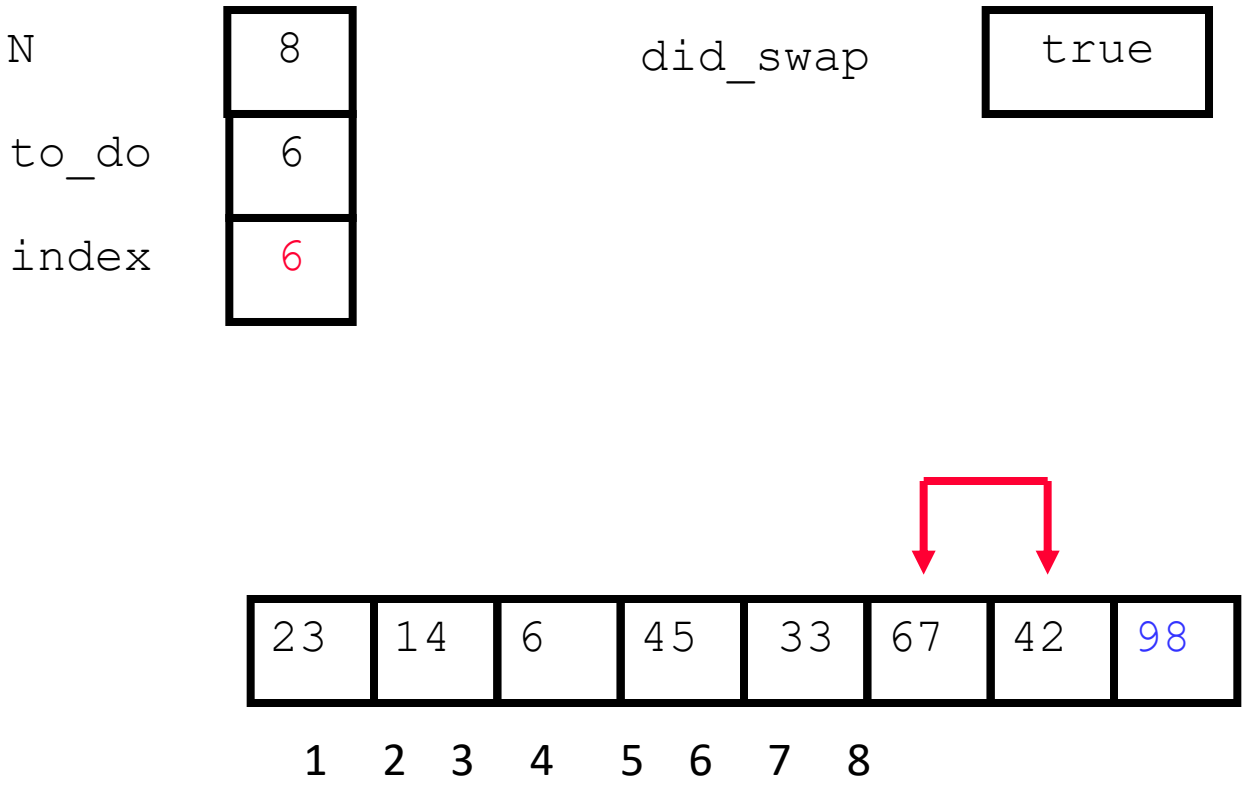
The Second “Bubble Up”



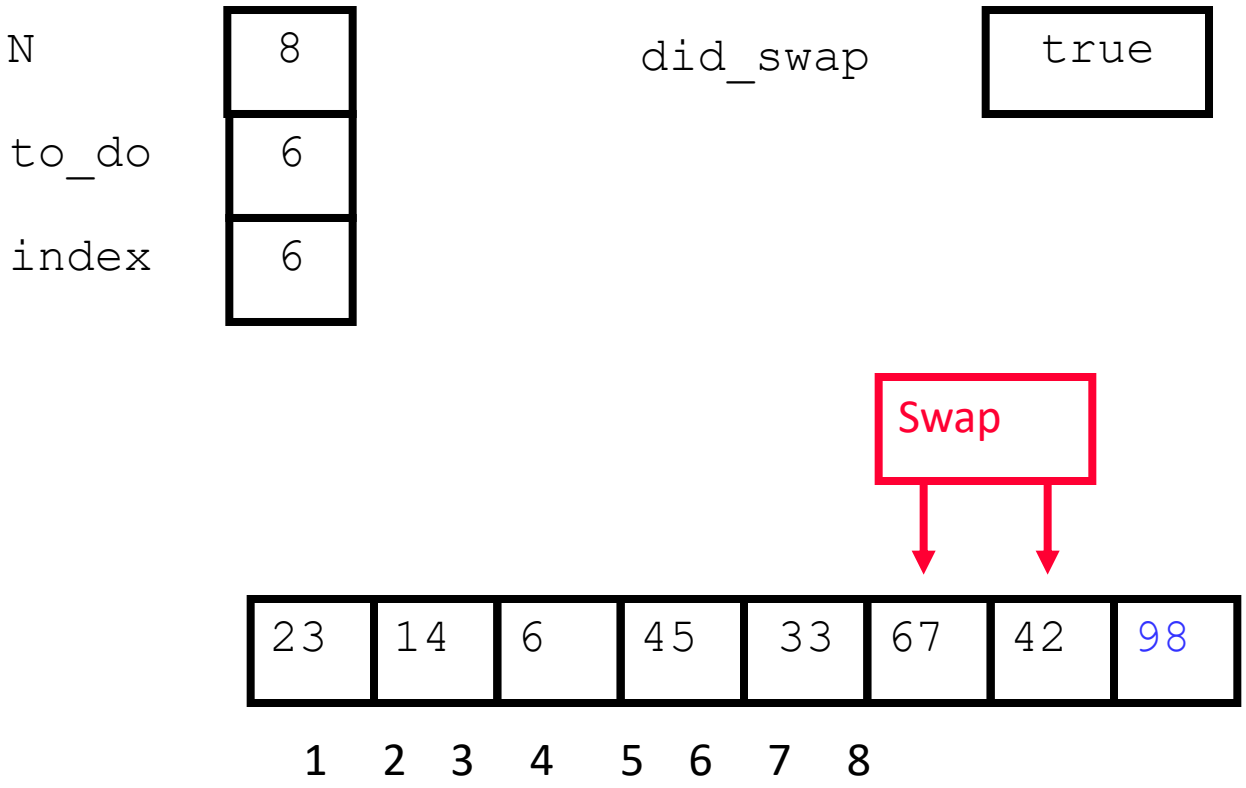
The Second “Bubble Up”



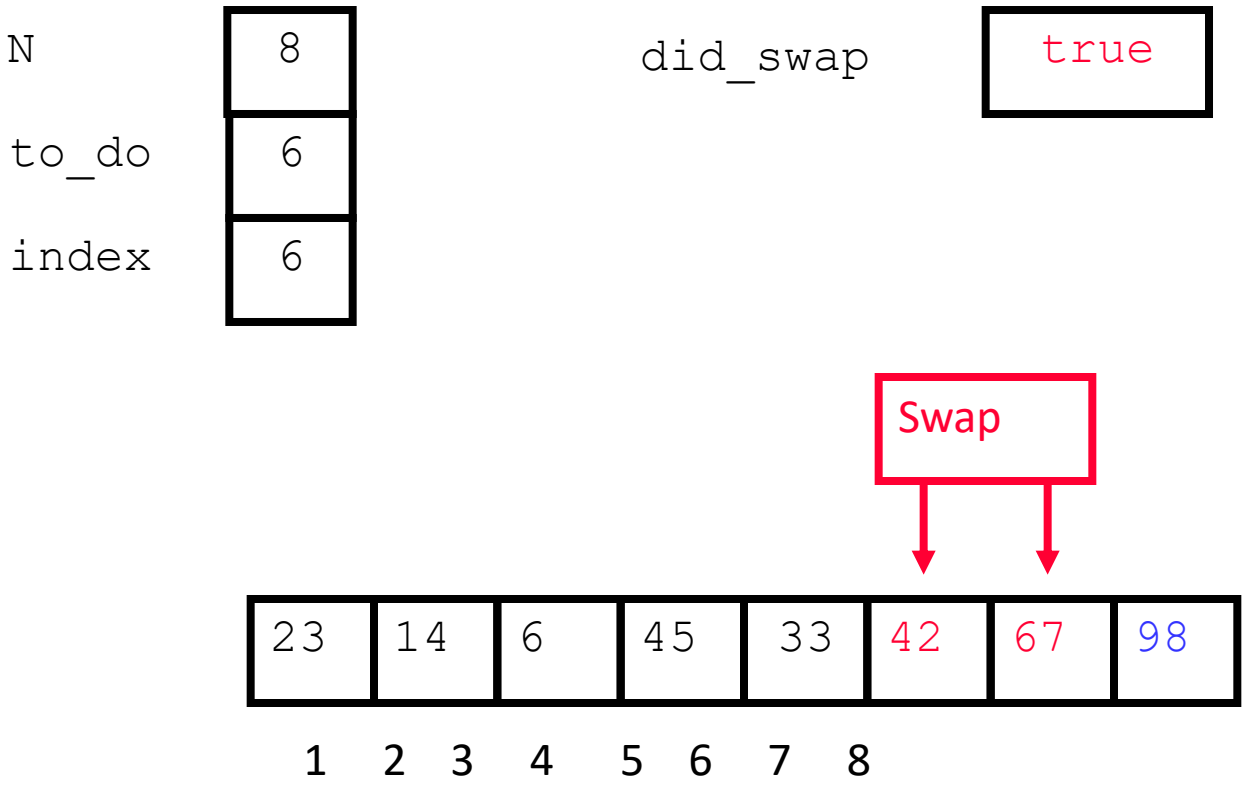
The Second “Bubble Up”



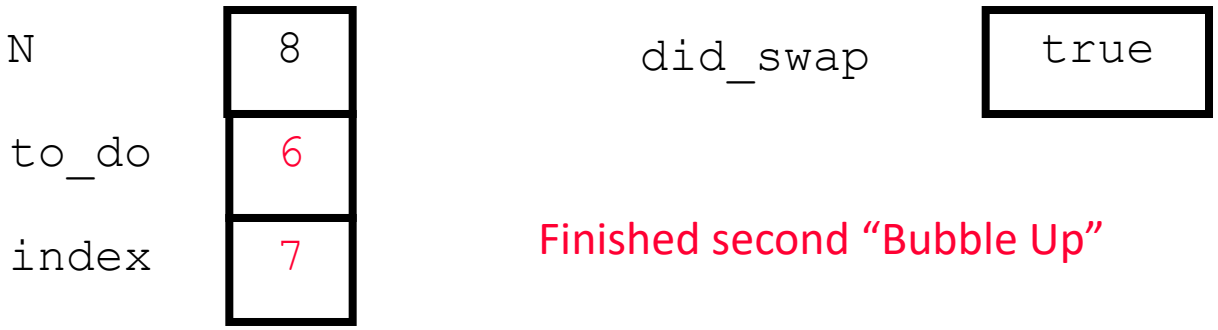
The Second “Bubble Up”



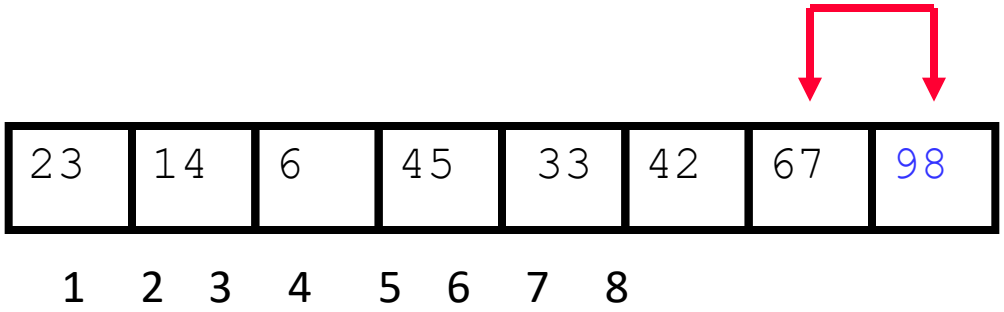
The Second “Bubble Up”



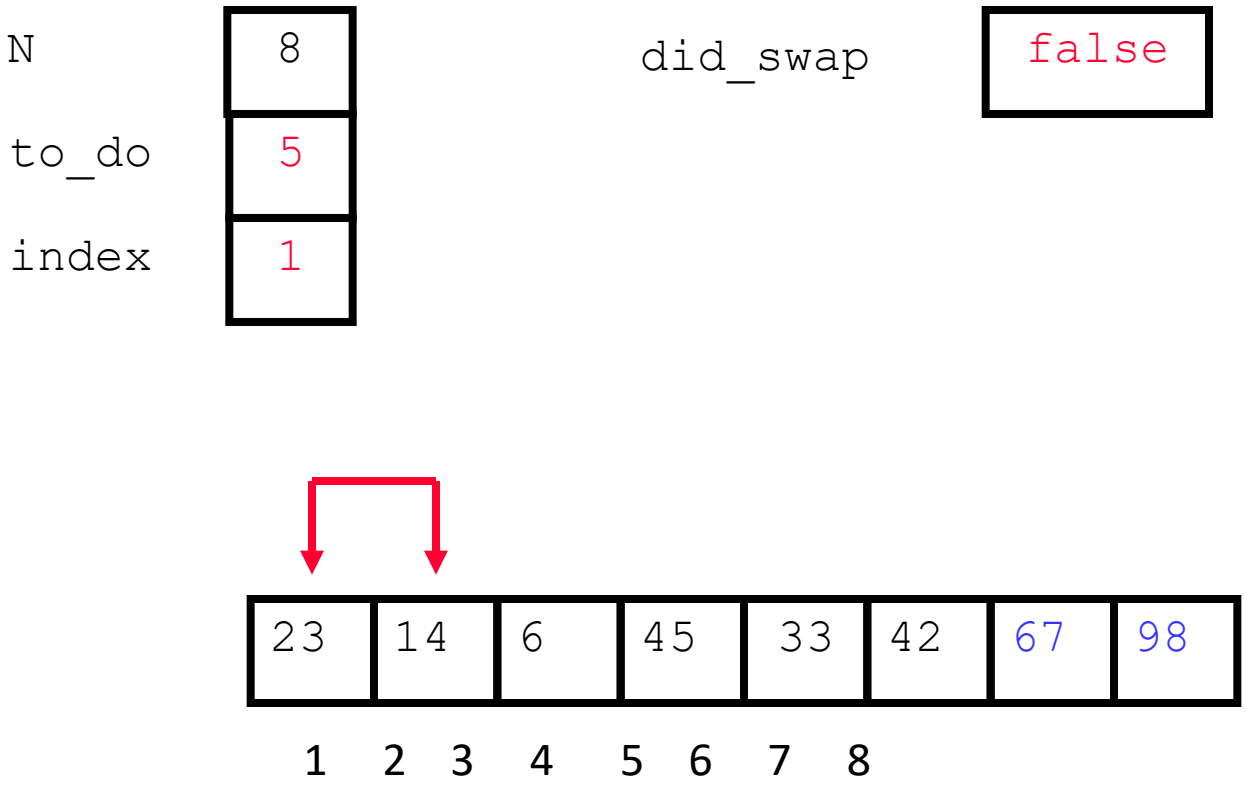
After Second Pass of Outer Loop



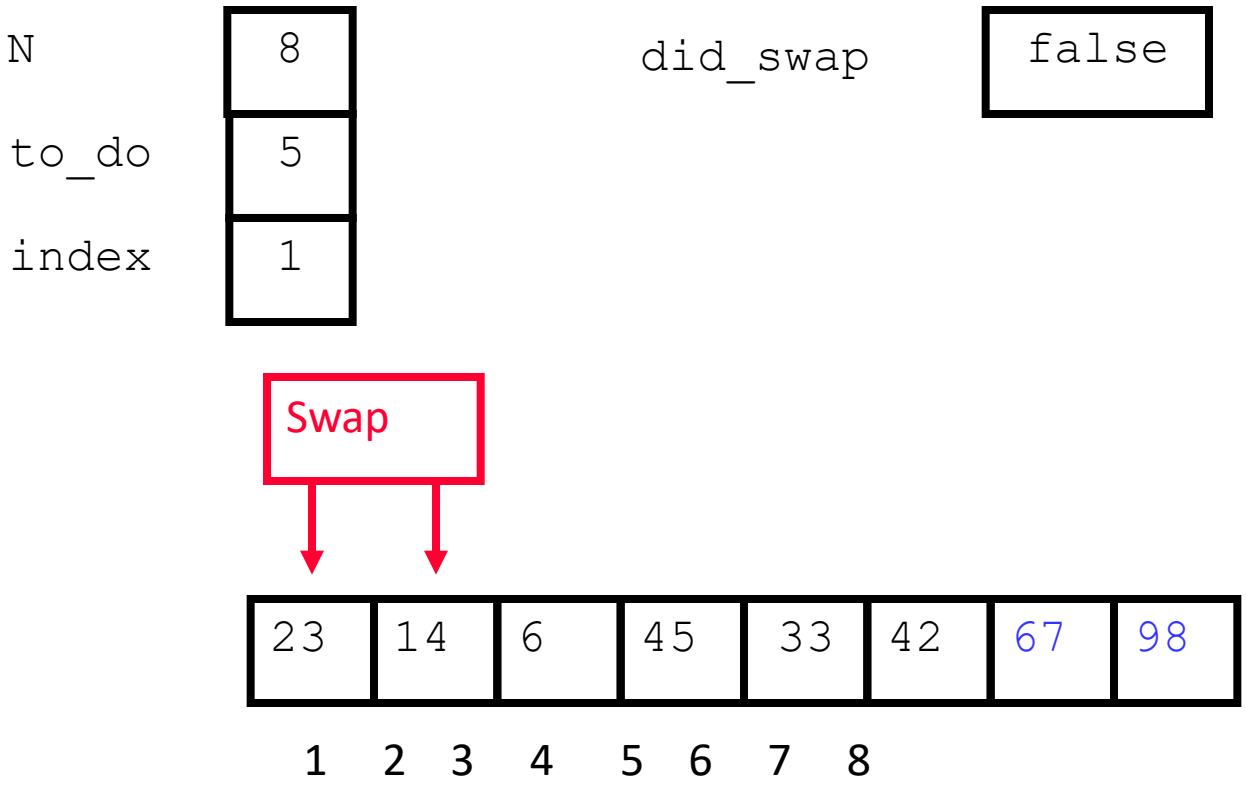
Finished second “Bubble Up”



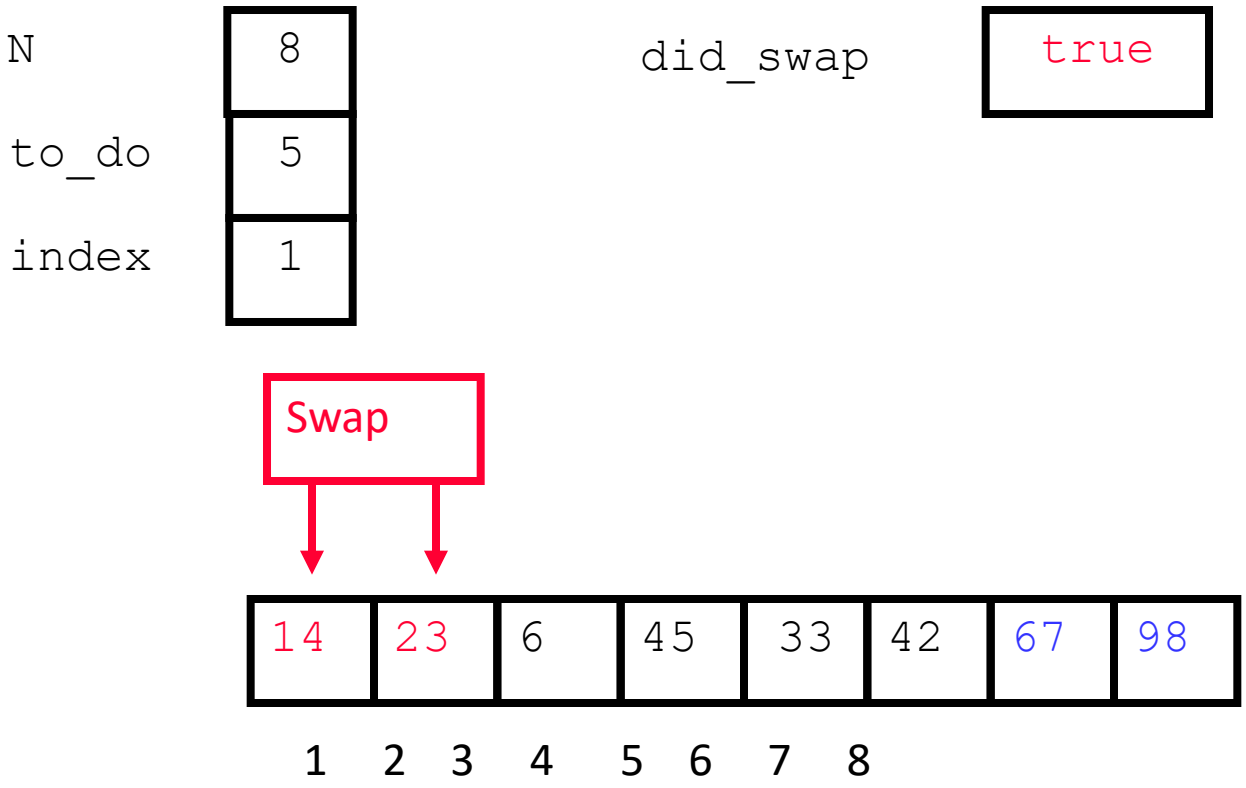
The Third “Bubble Up”



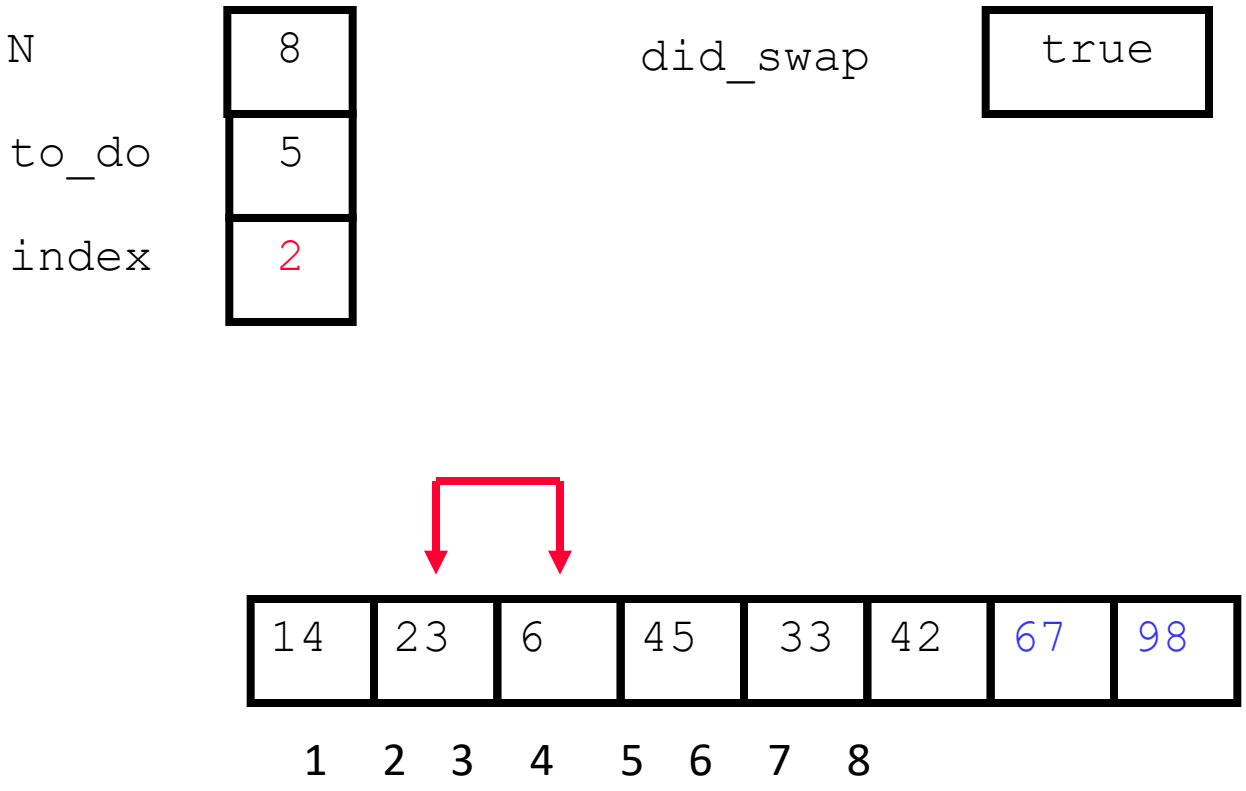
The Third “Bubble Up”



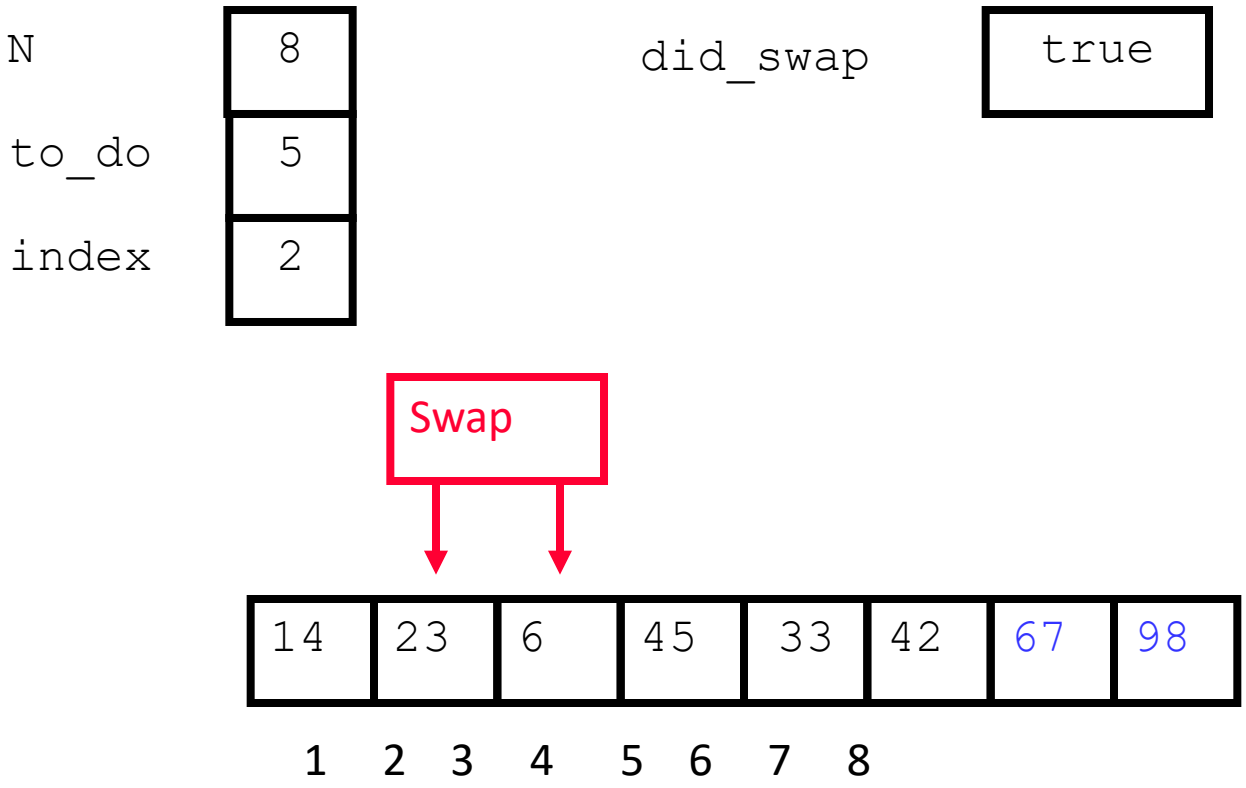
The Third “Bubble Up”



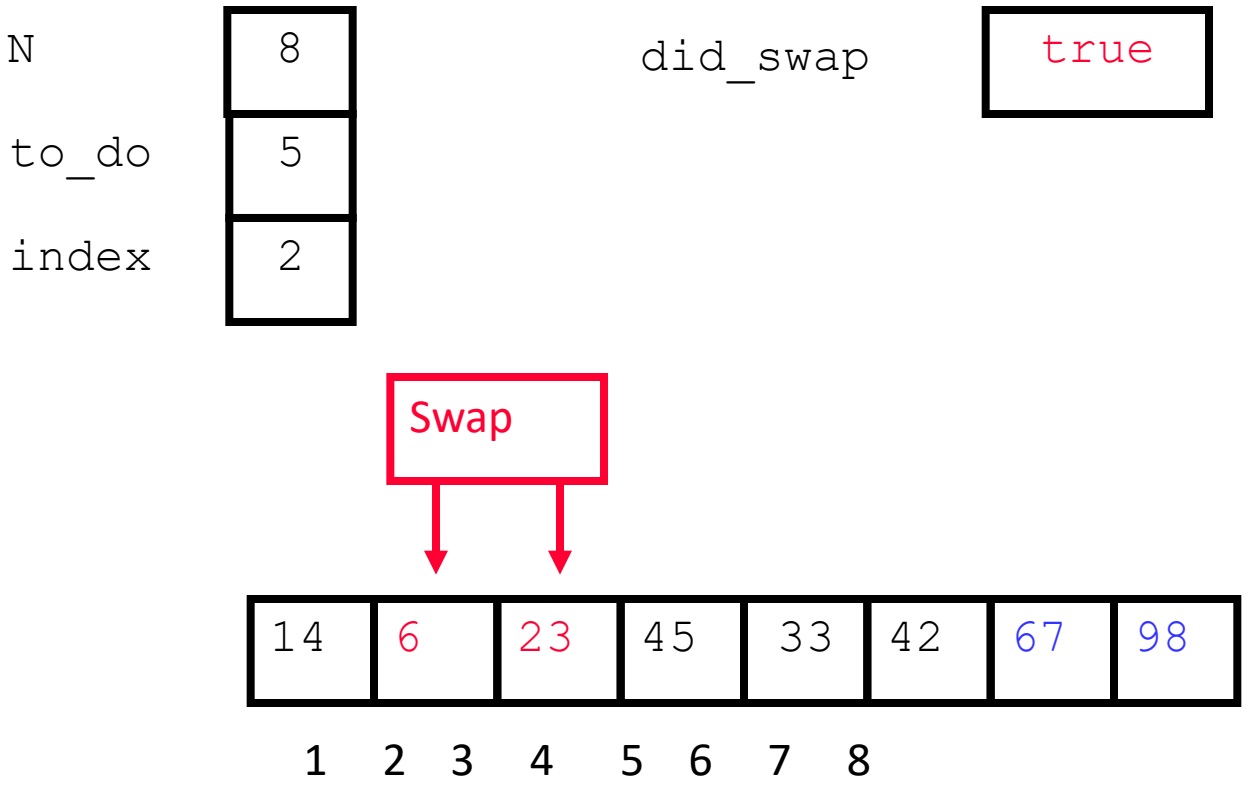
The Third “Bubble Up”



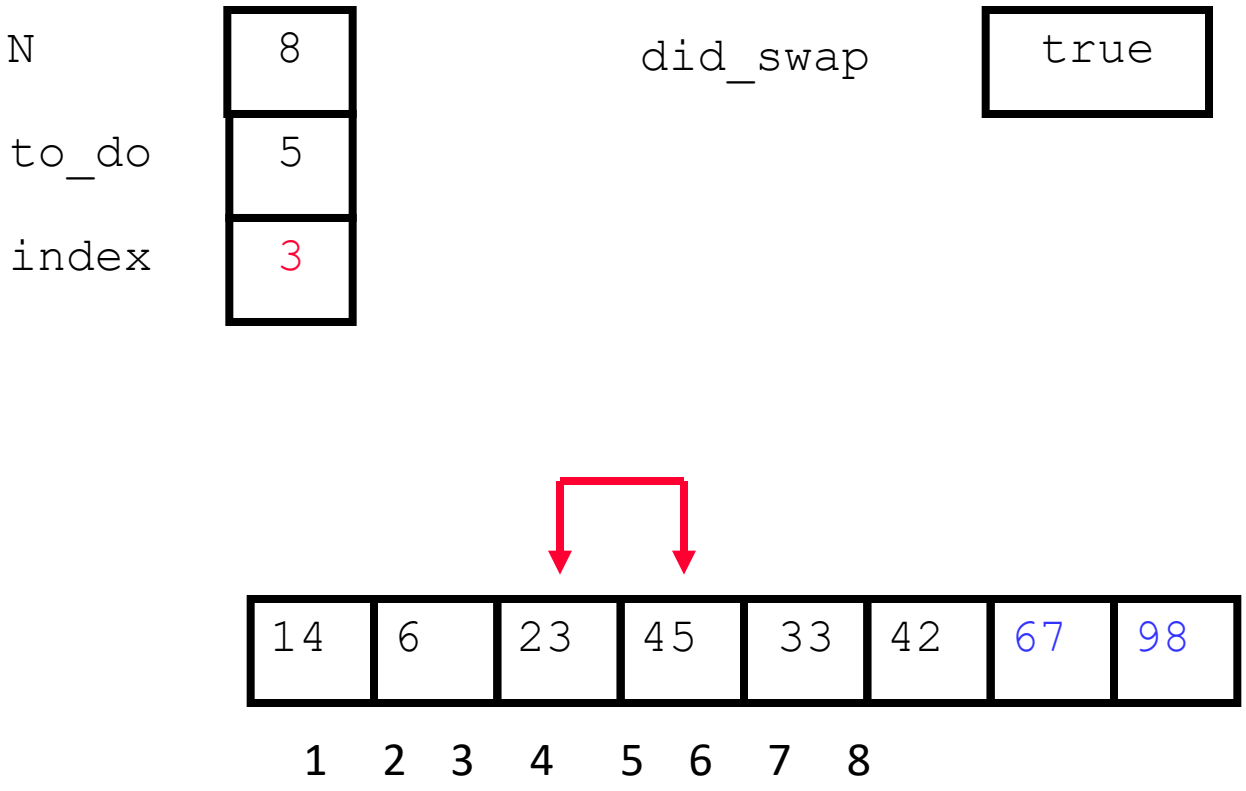
The Third “Bubble Up”



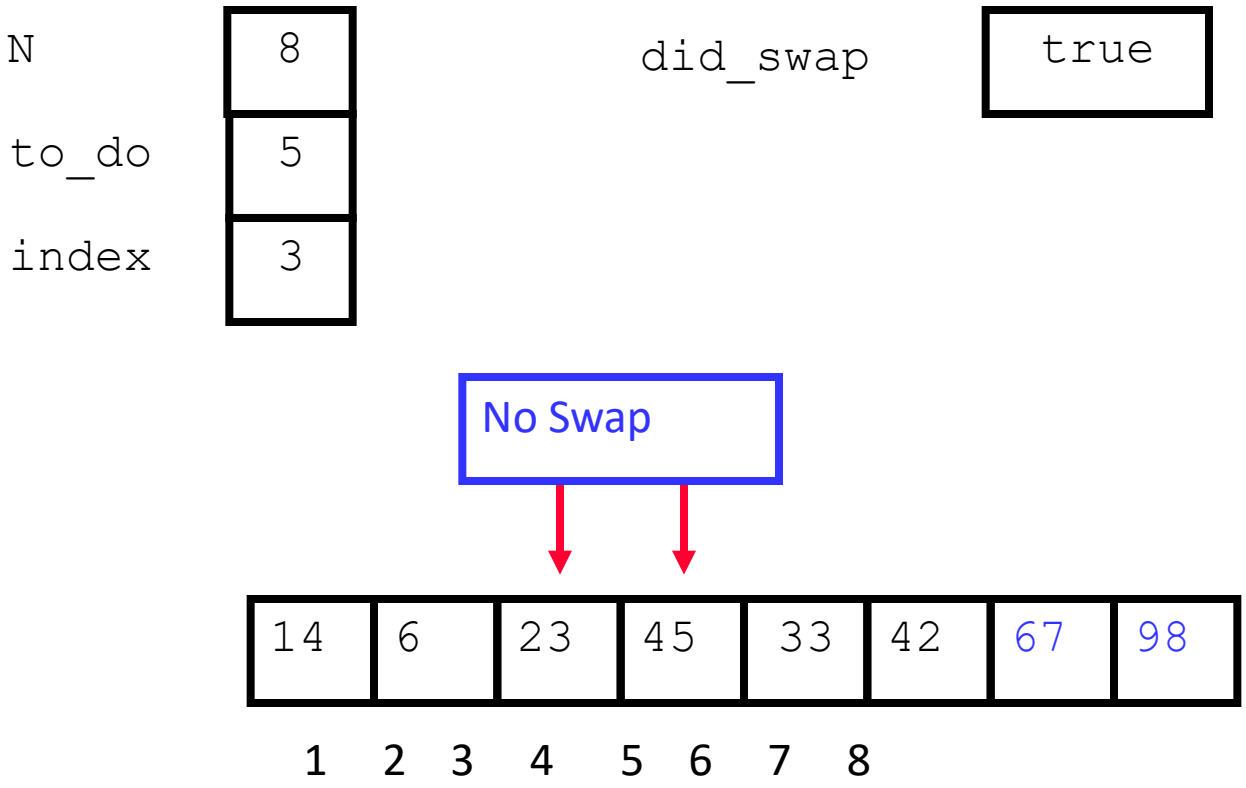
The Third “Bubble Up”



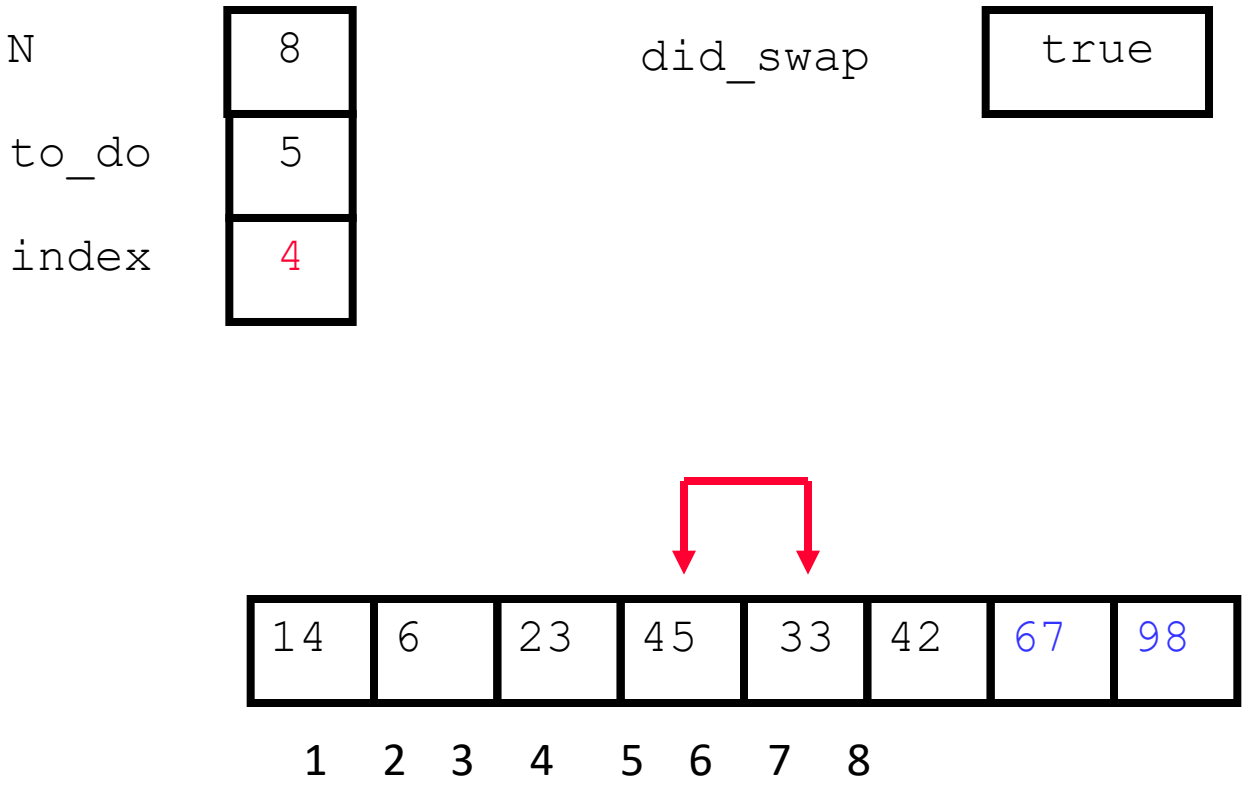
The Third “Bubble Up”



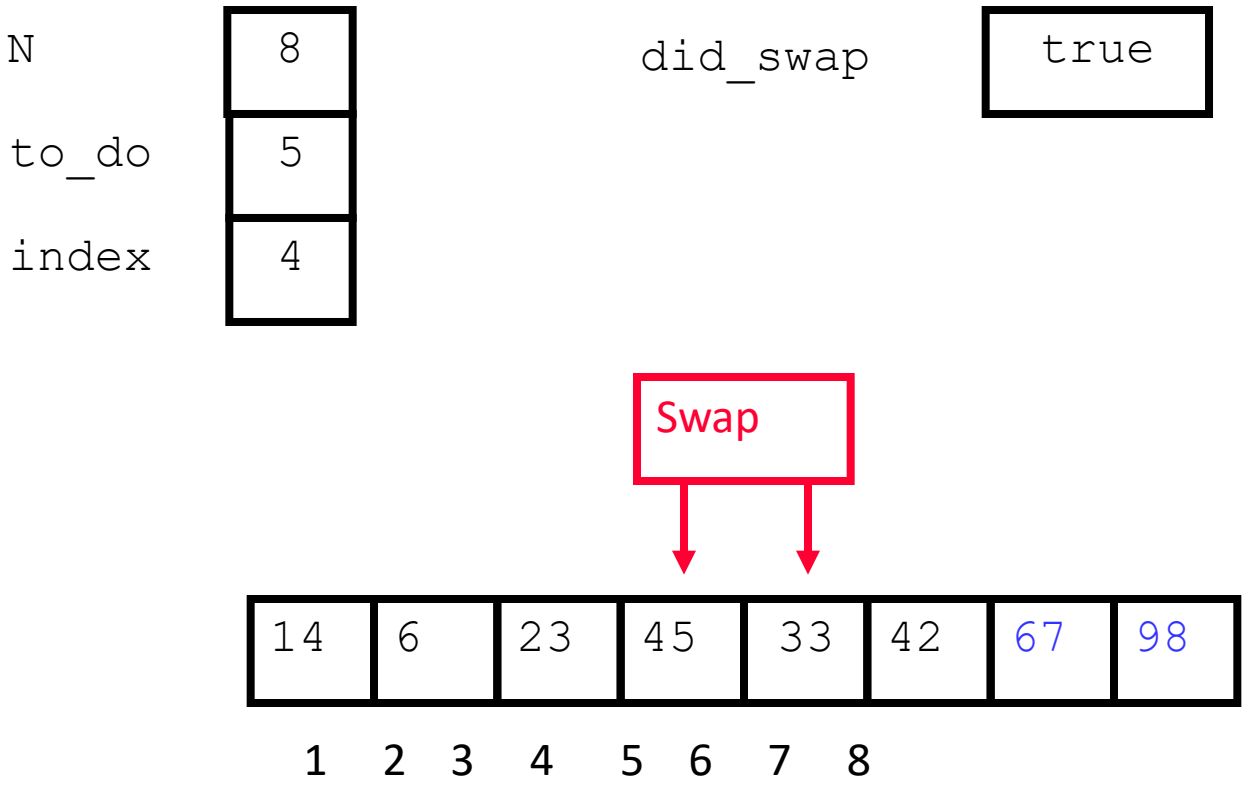
The Third “Bubble Up”



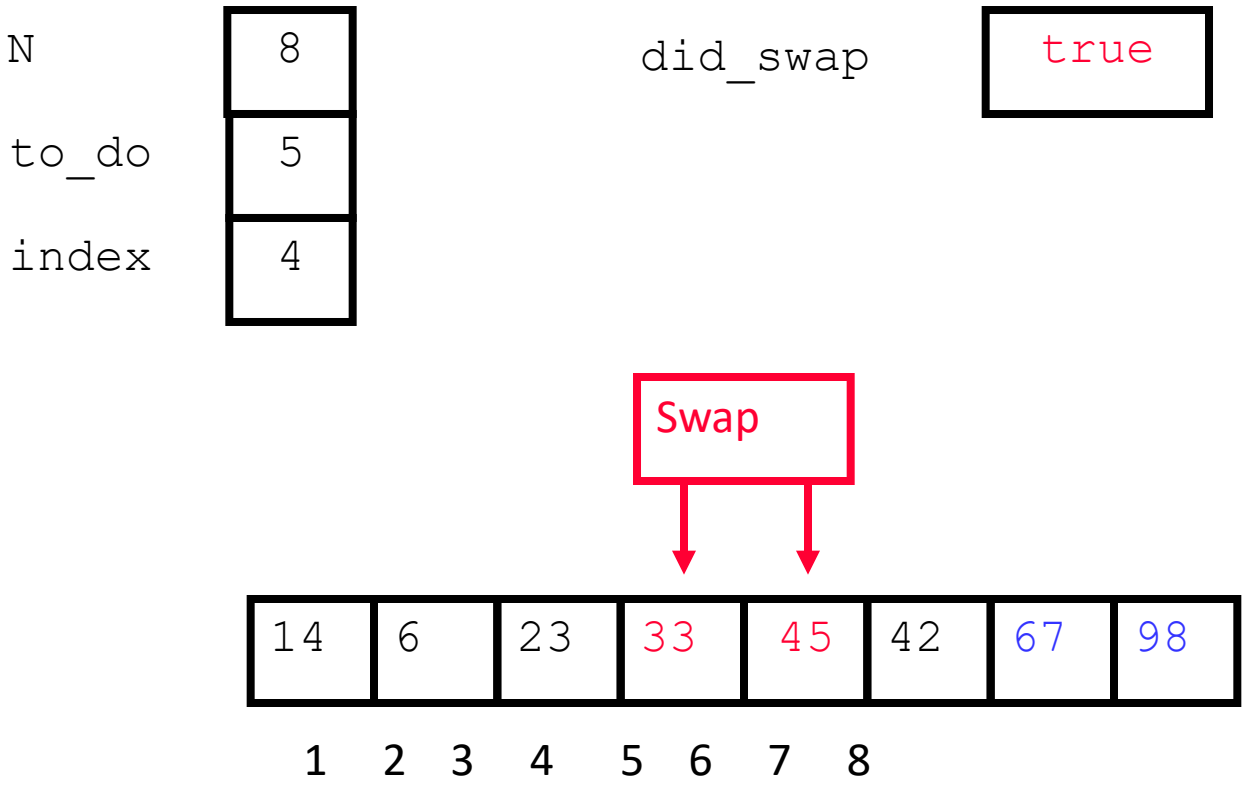
The Third “Bubble Up”



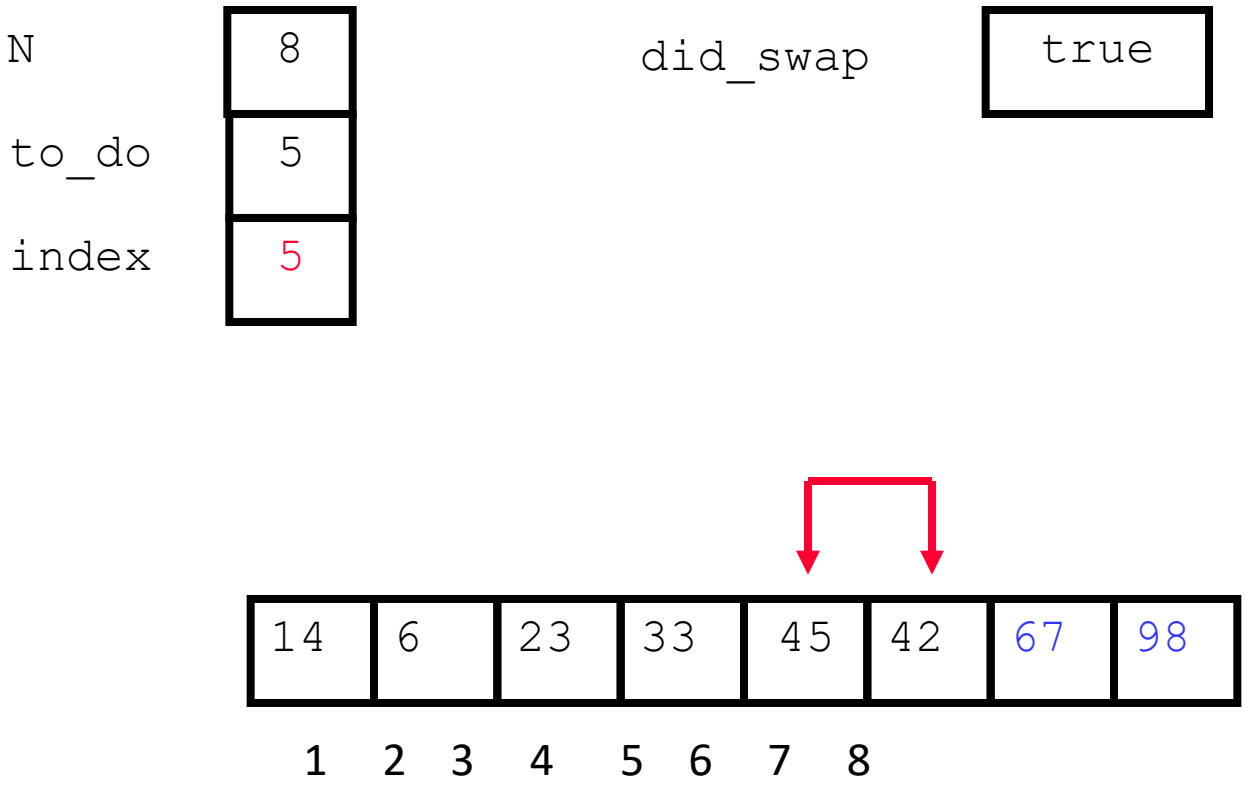
The Third “Bubble Up”



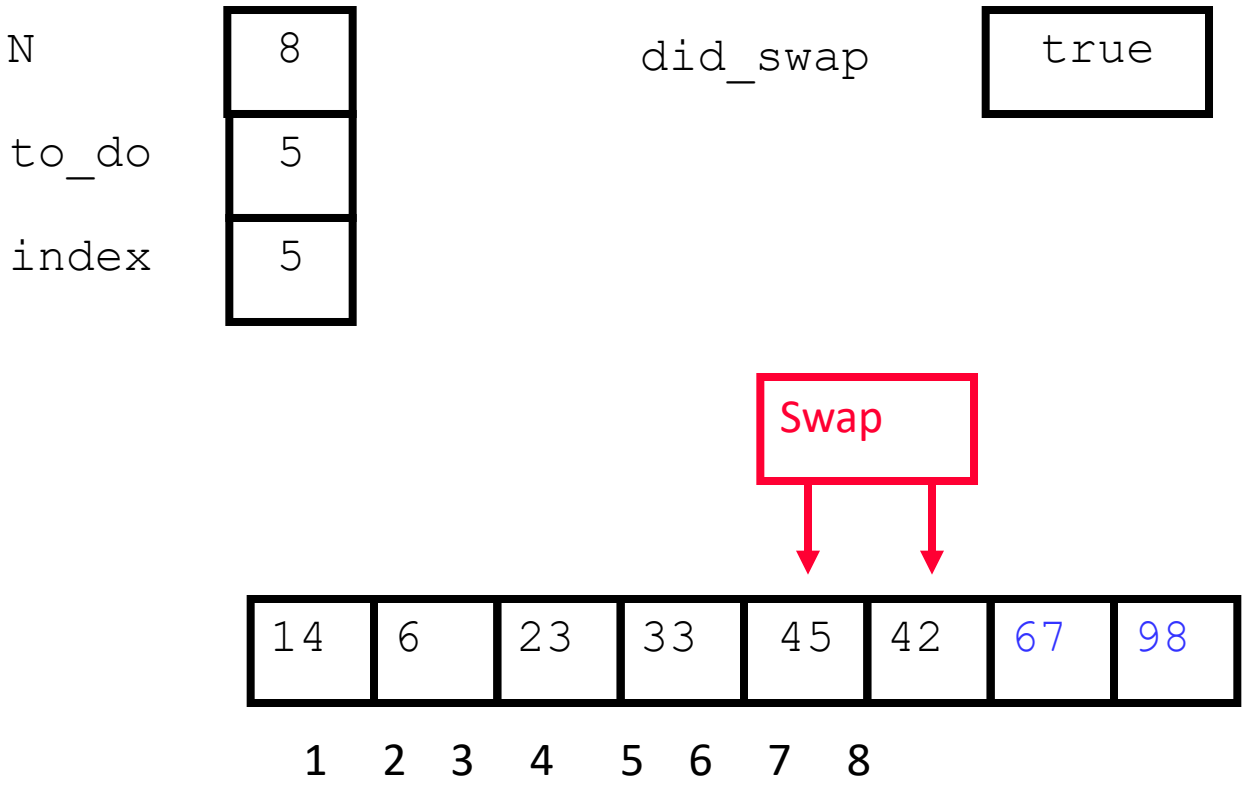
The Third “Bubble Up”



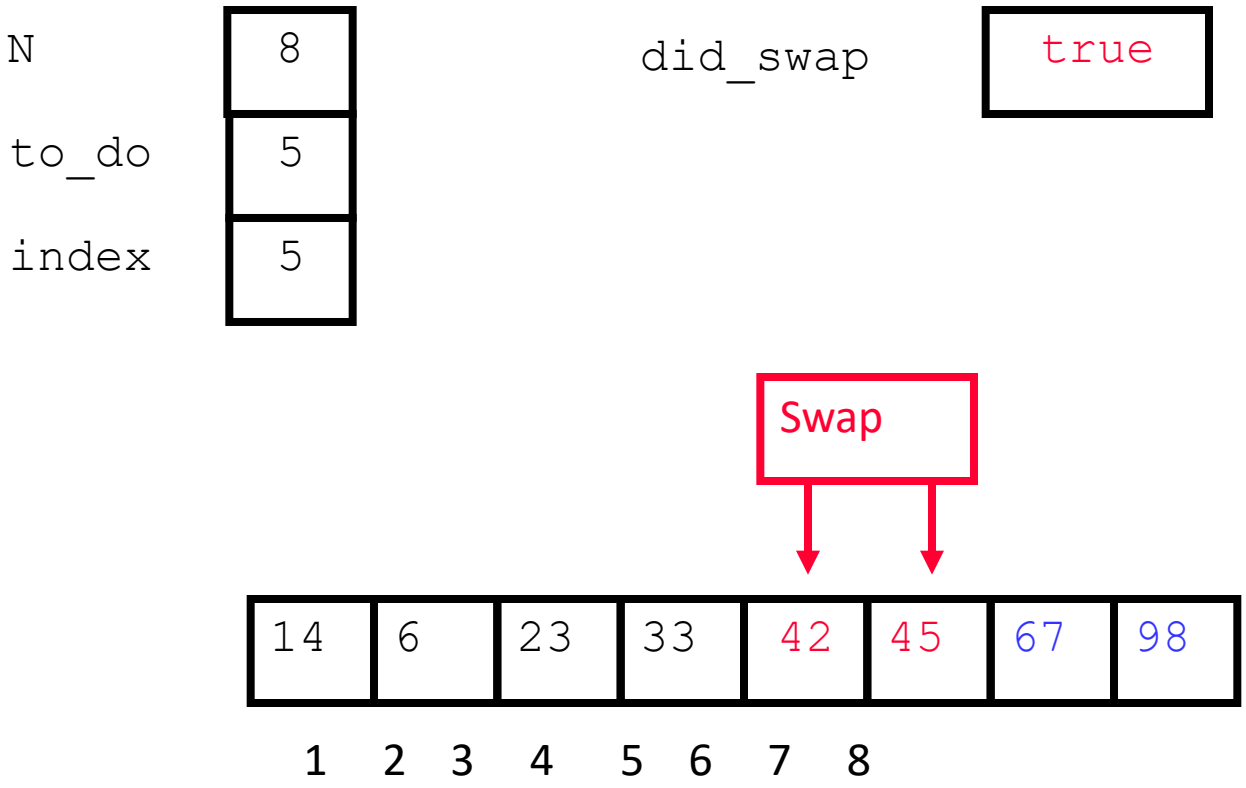
The Third “Bubble Up”



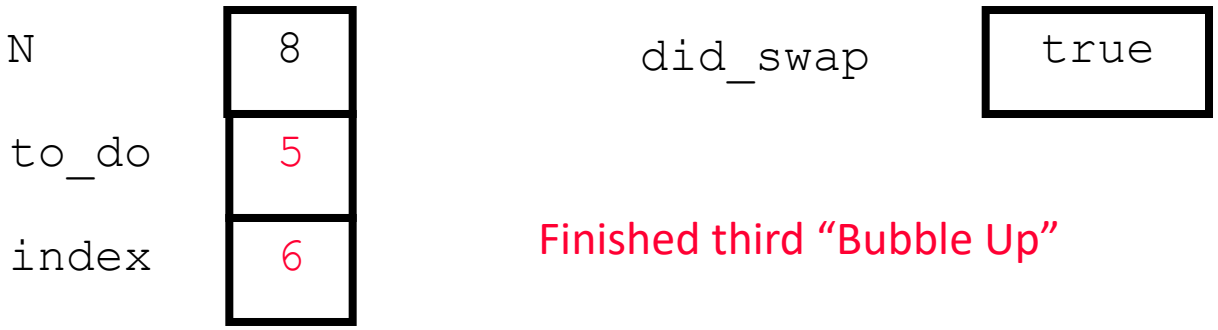
The Third “Bubble Up”



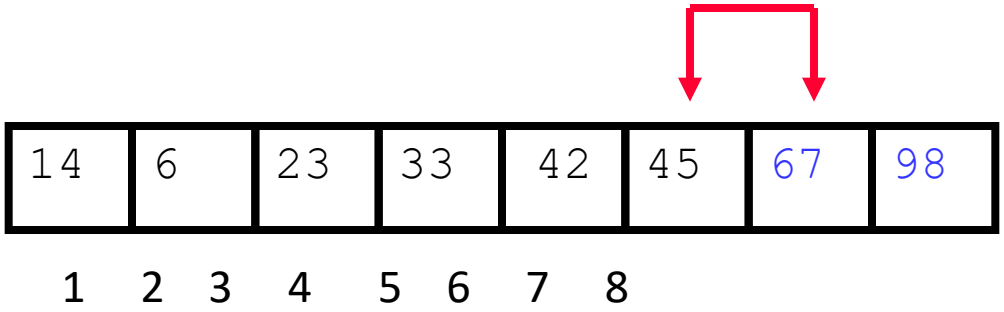
The Third “Bubble Up”



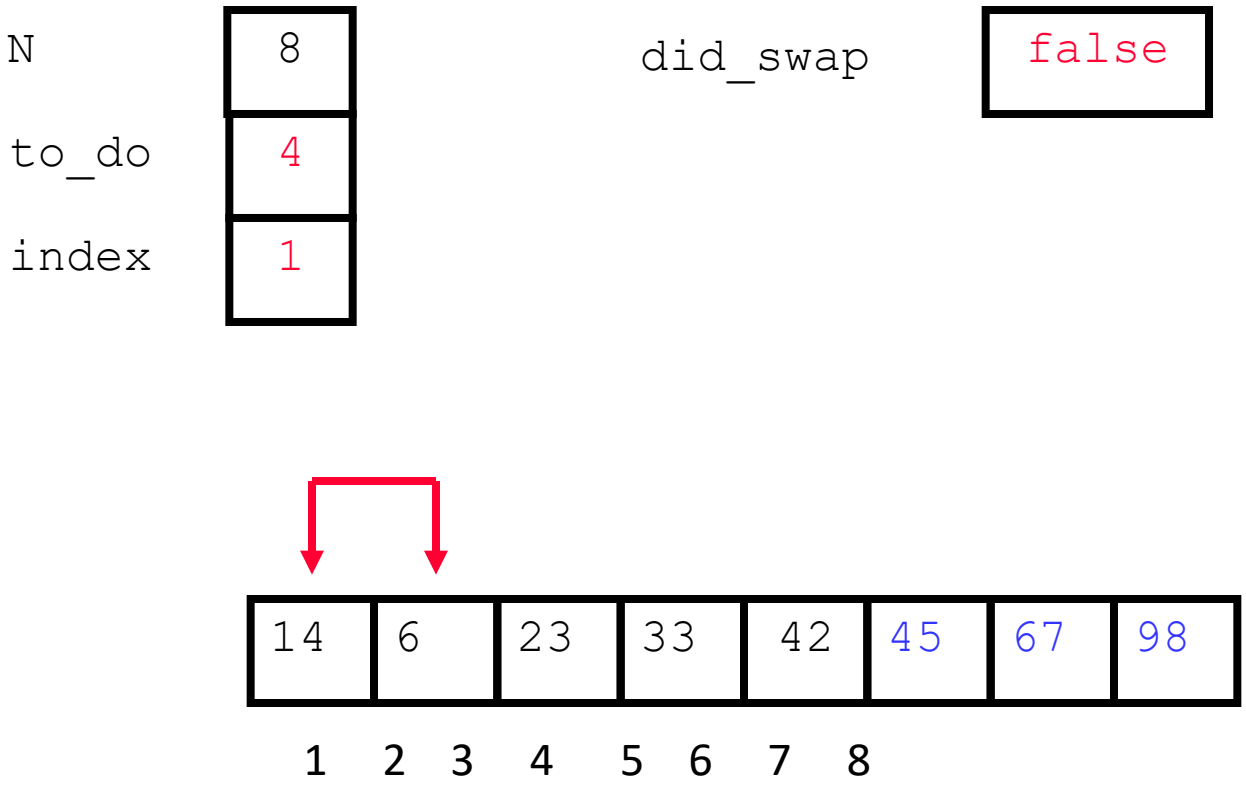
After Third Pass of Outer Loop



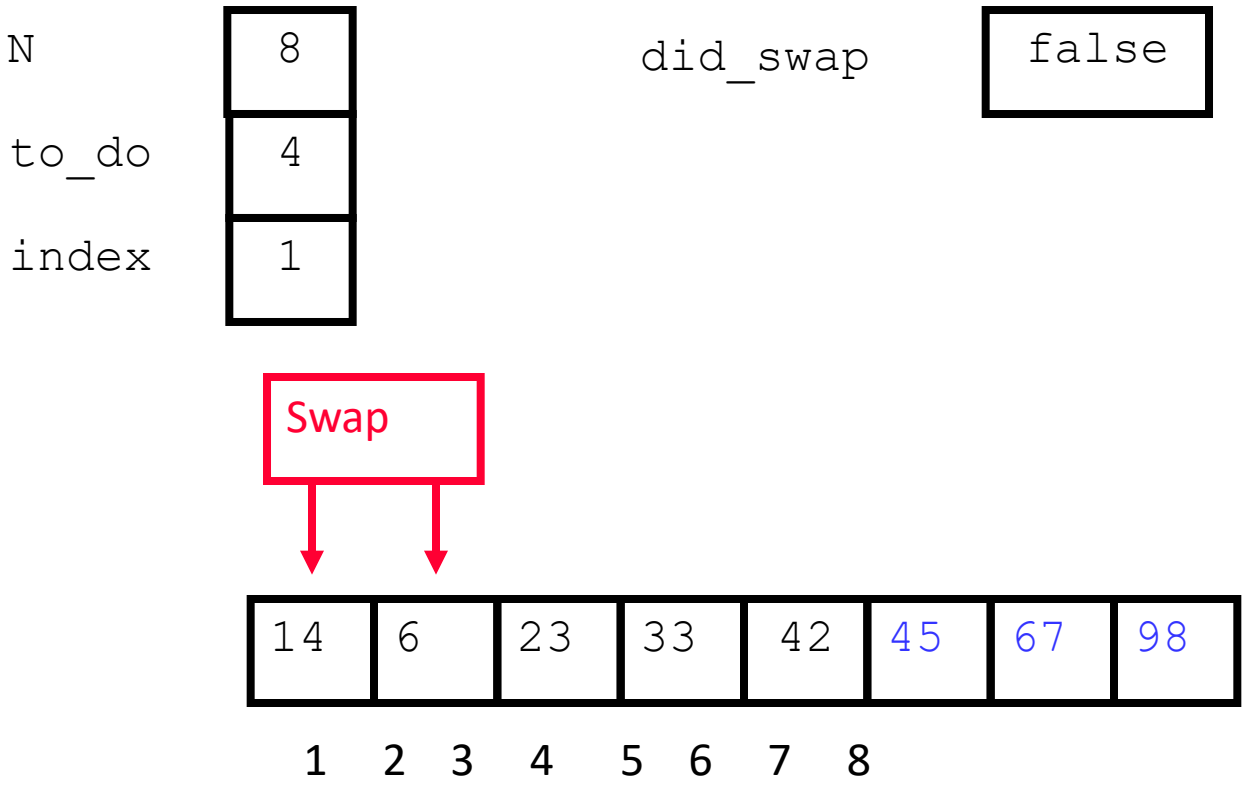
Finished third “Bubble Up”



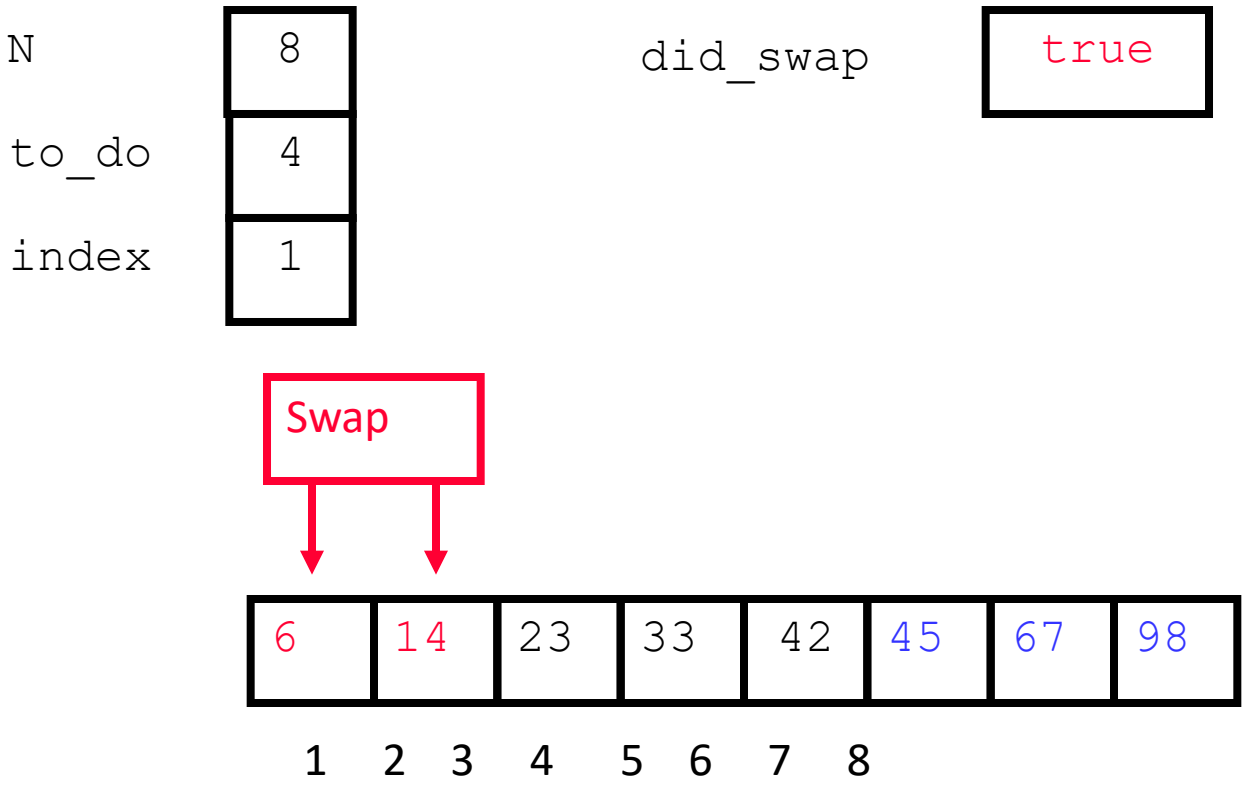
The Fourth “Bubble Up”



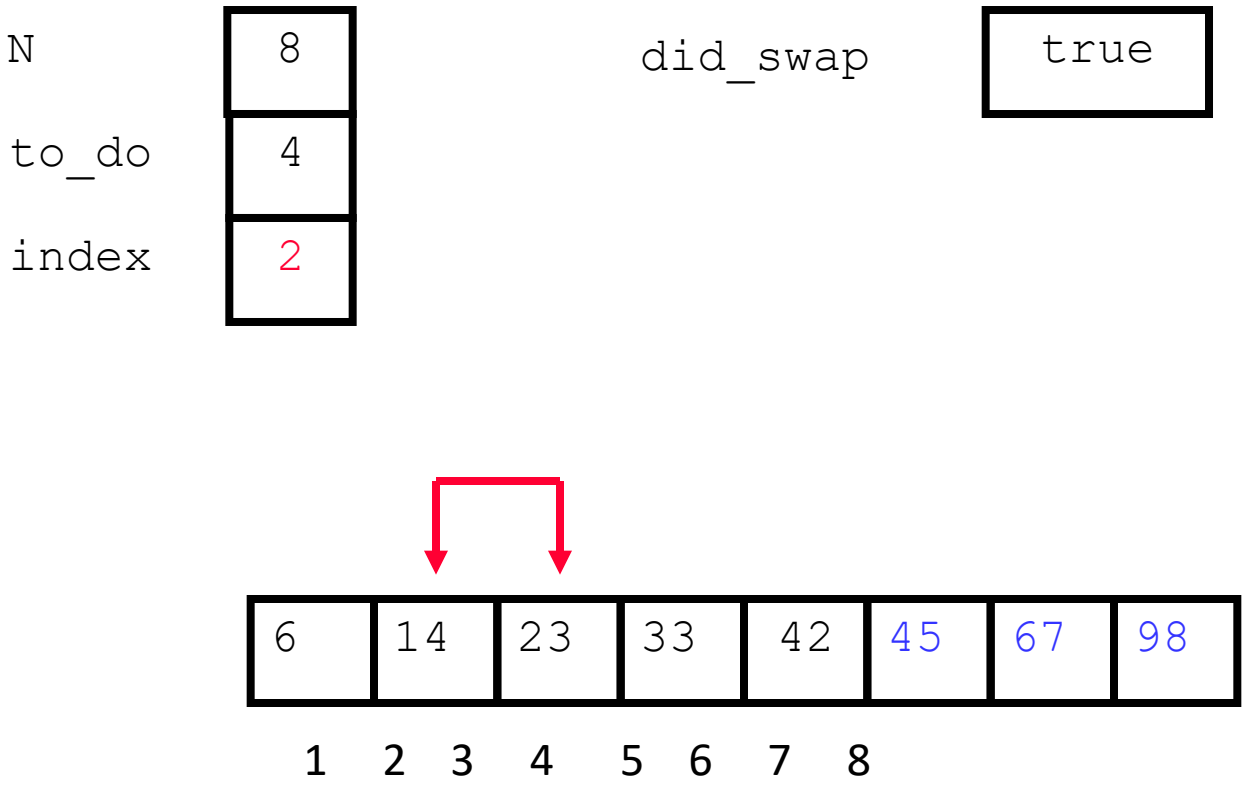
The Fourth “Bubble Up”



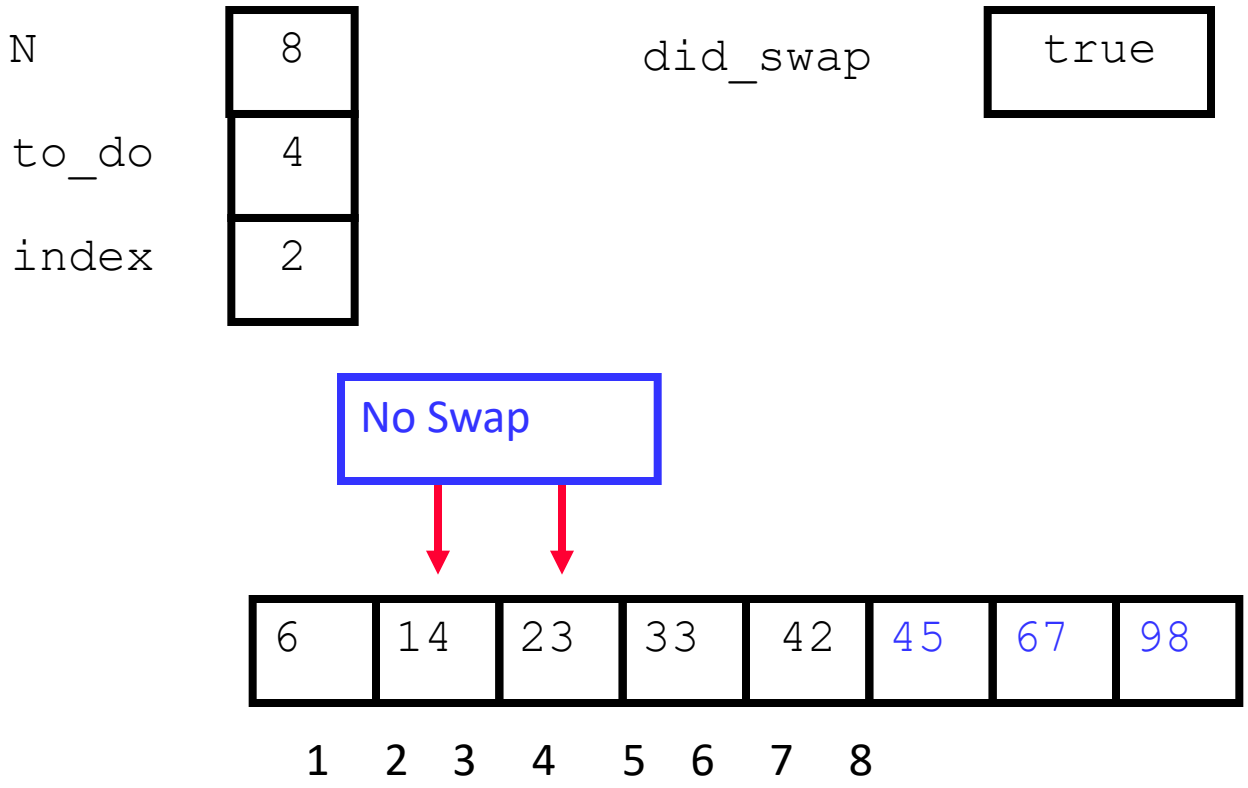
The Fourth “Bubble Up”



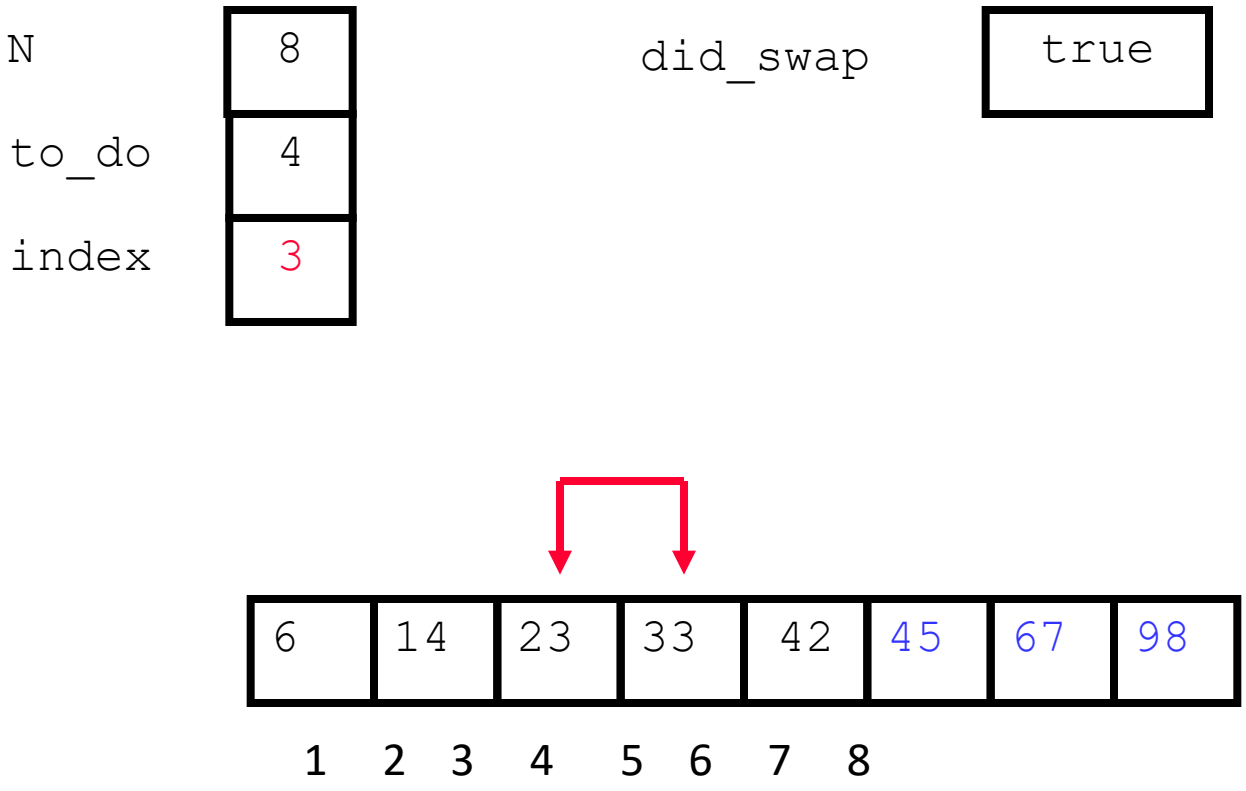
The Fourth “Bubble Up”



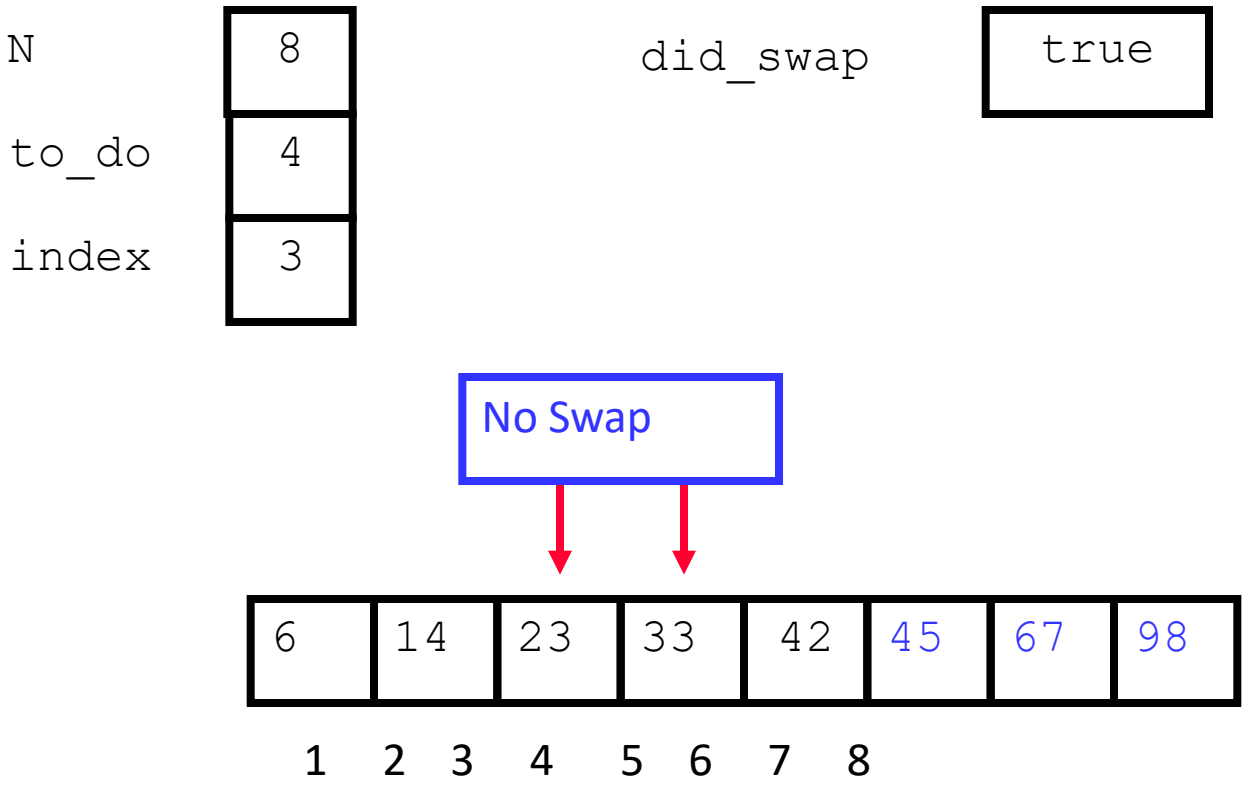
The Fourth “Bubble Up”



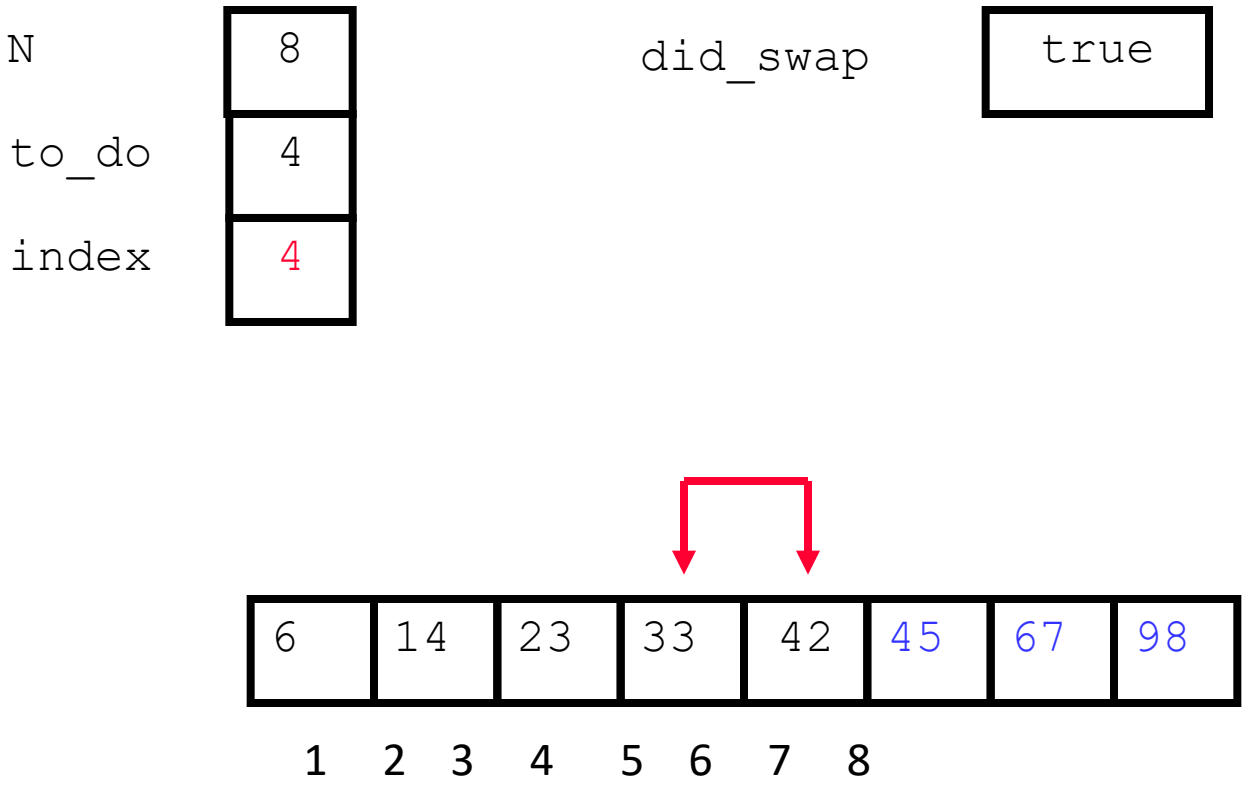
The Fourth “Bubble Up”



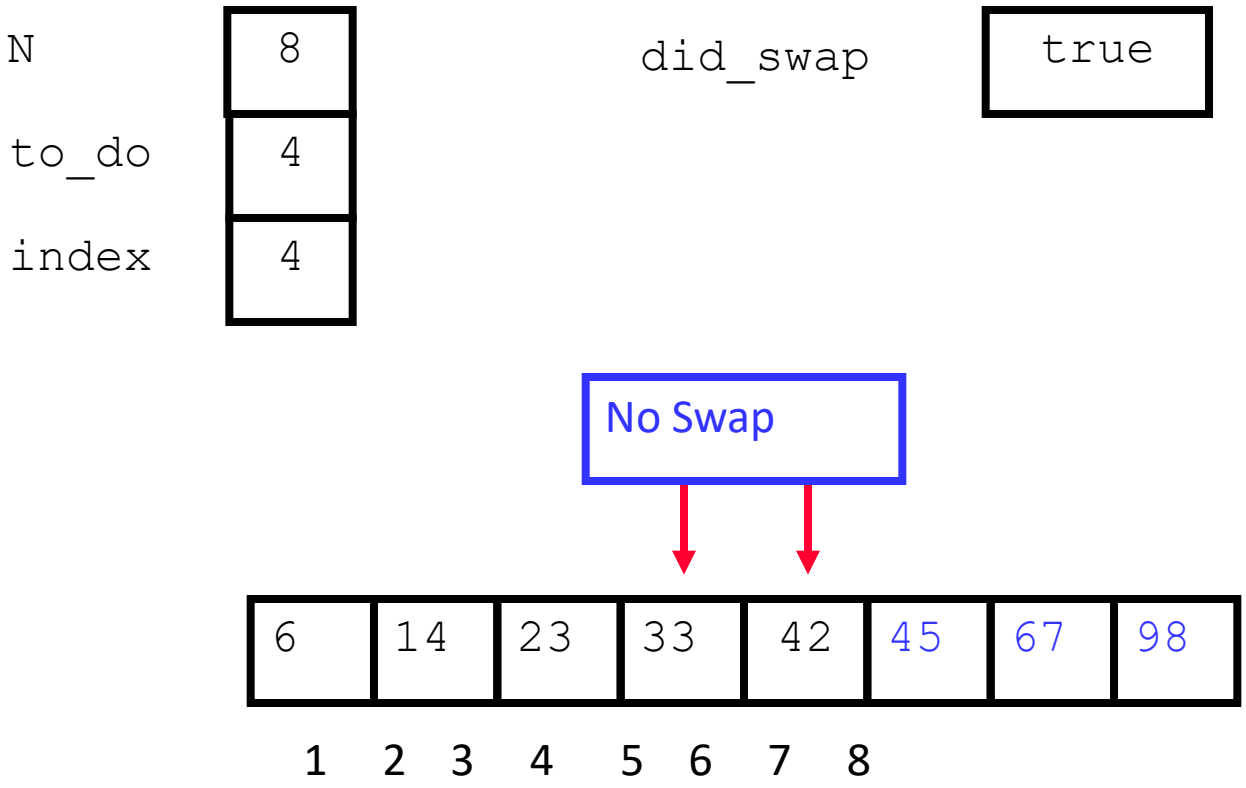
The Fourth “Bubble Up”



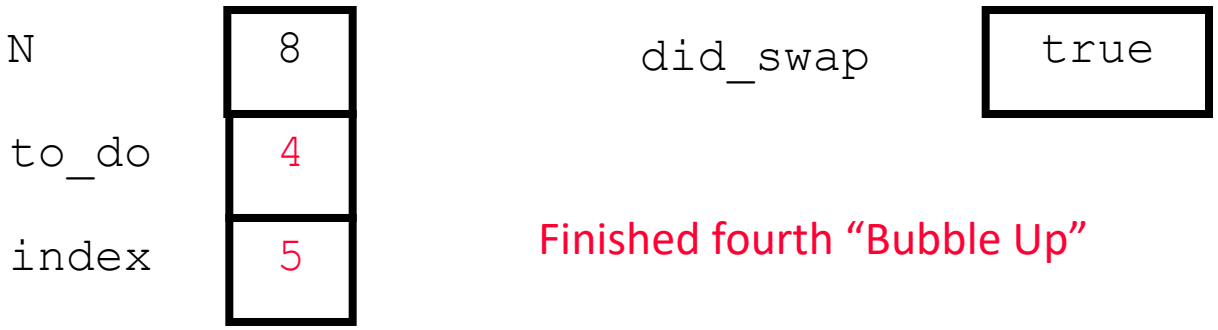
The Fourth “Bubble Up”



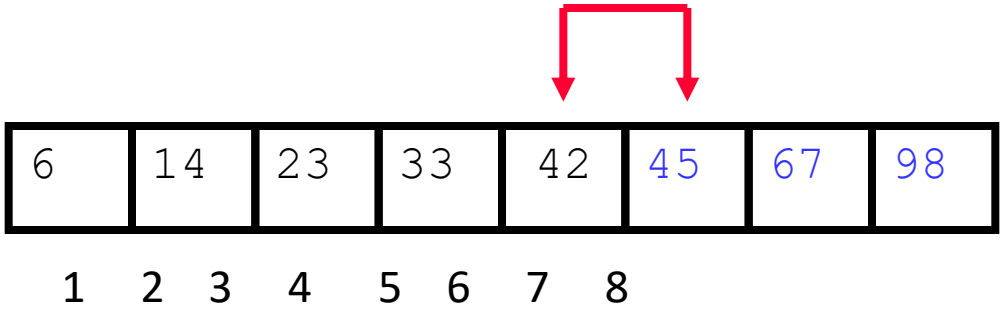
The Fourth “Bubble Up”



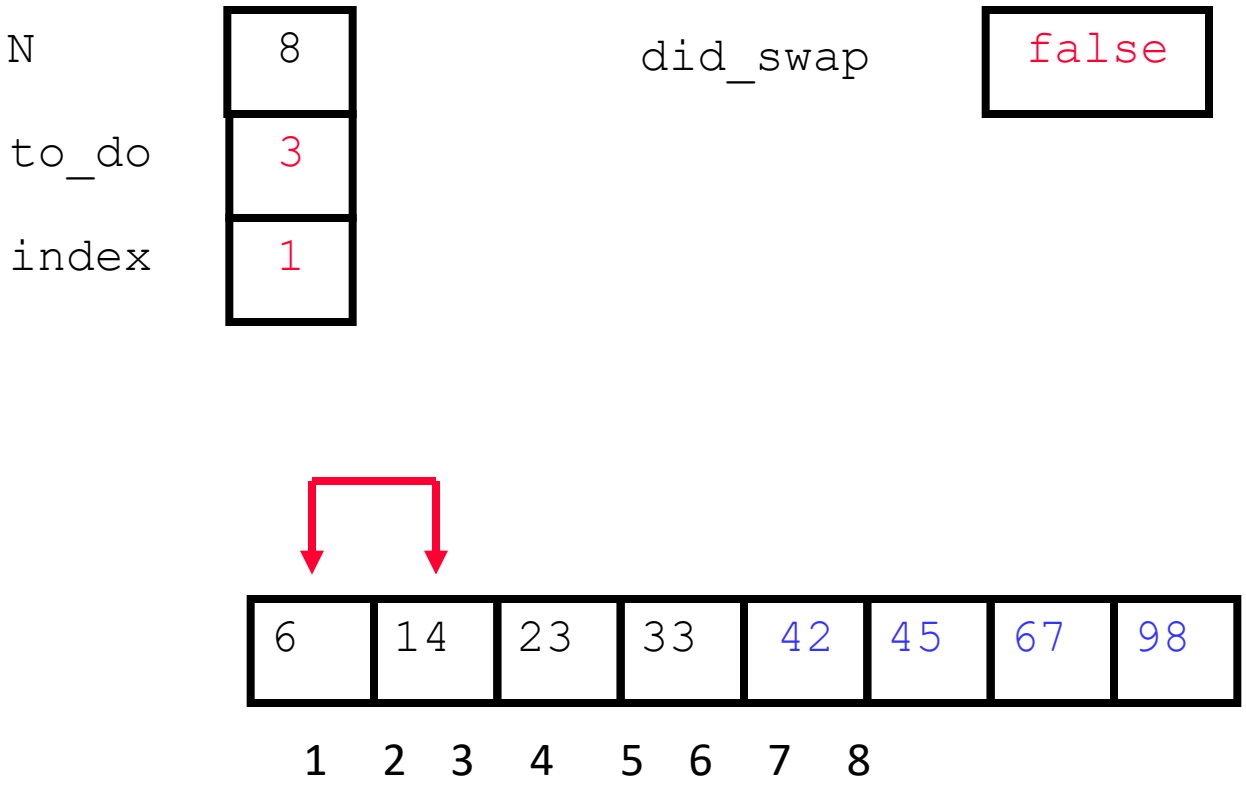
After Fourth Pass of Outer Loop



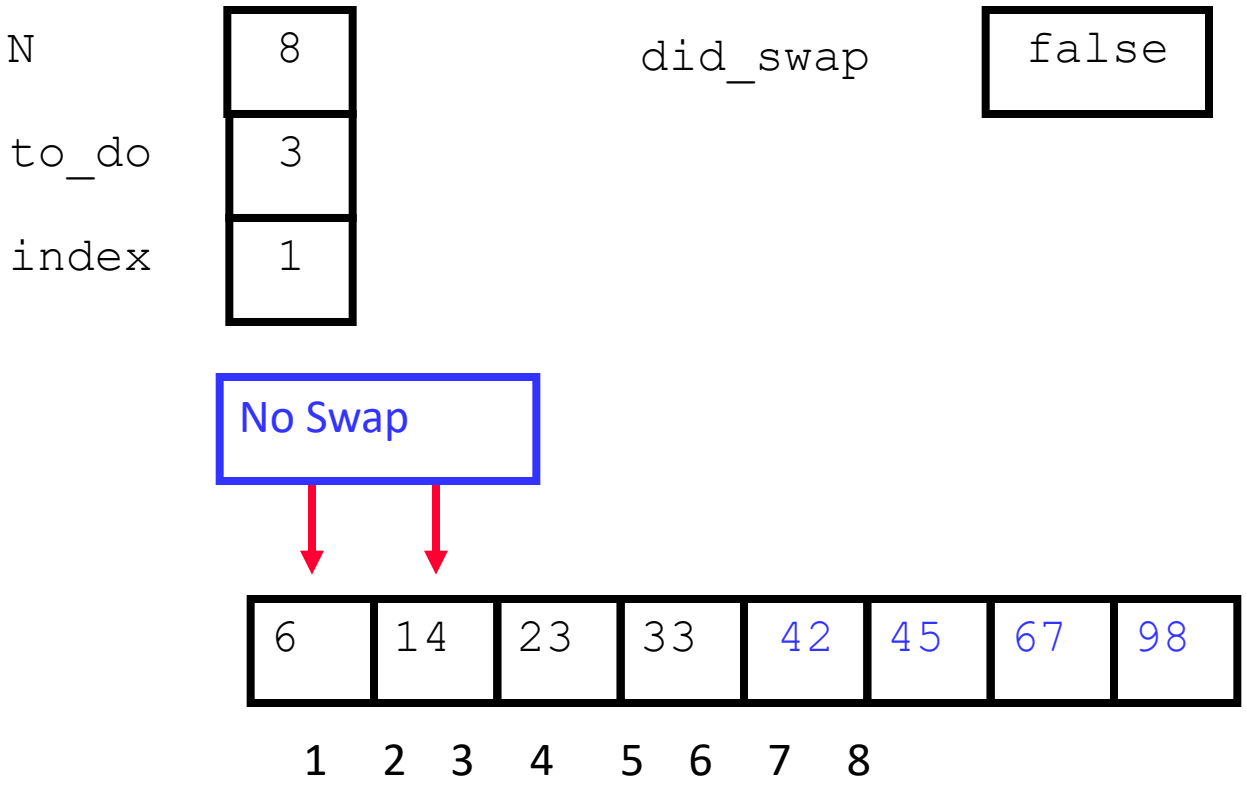
Finished fourth “Bubble Up”



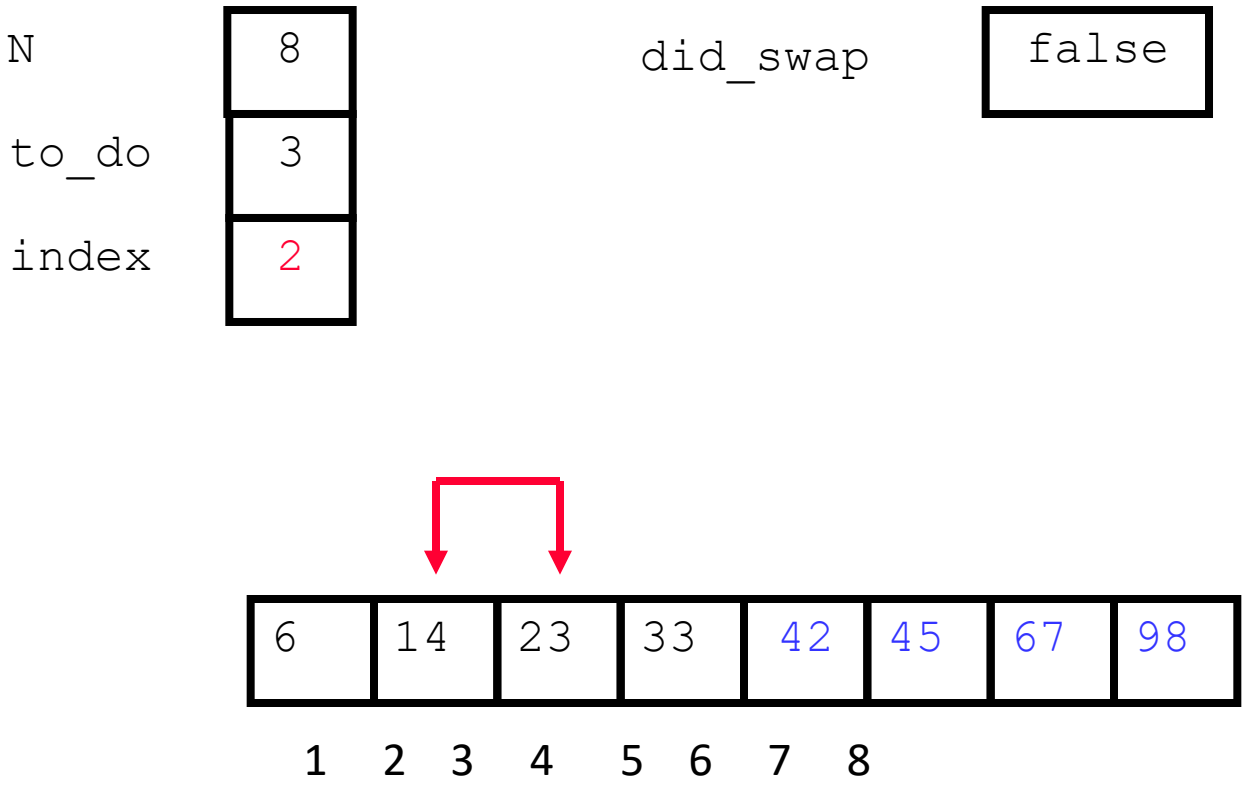
The Fifth “Bubble Up”



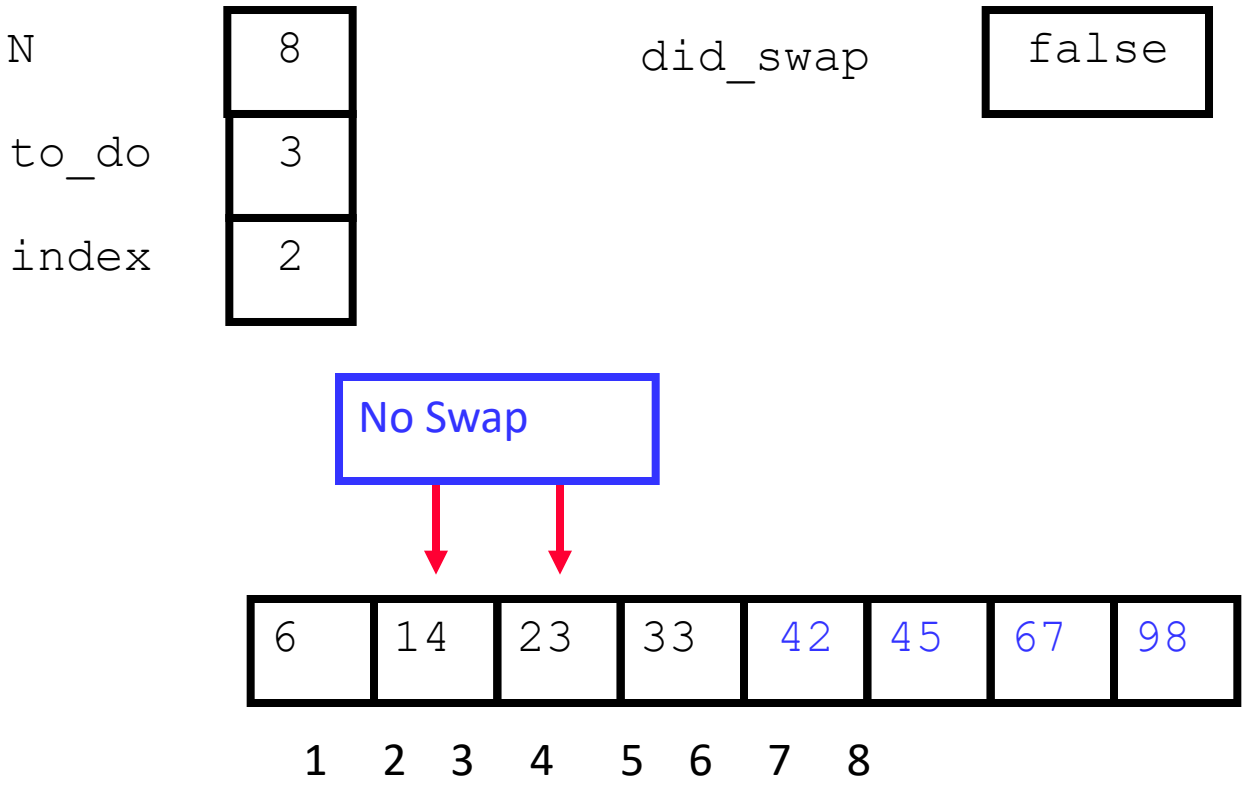
The Fifth “Bubble Up”



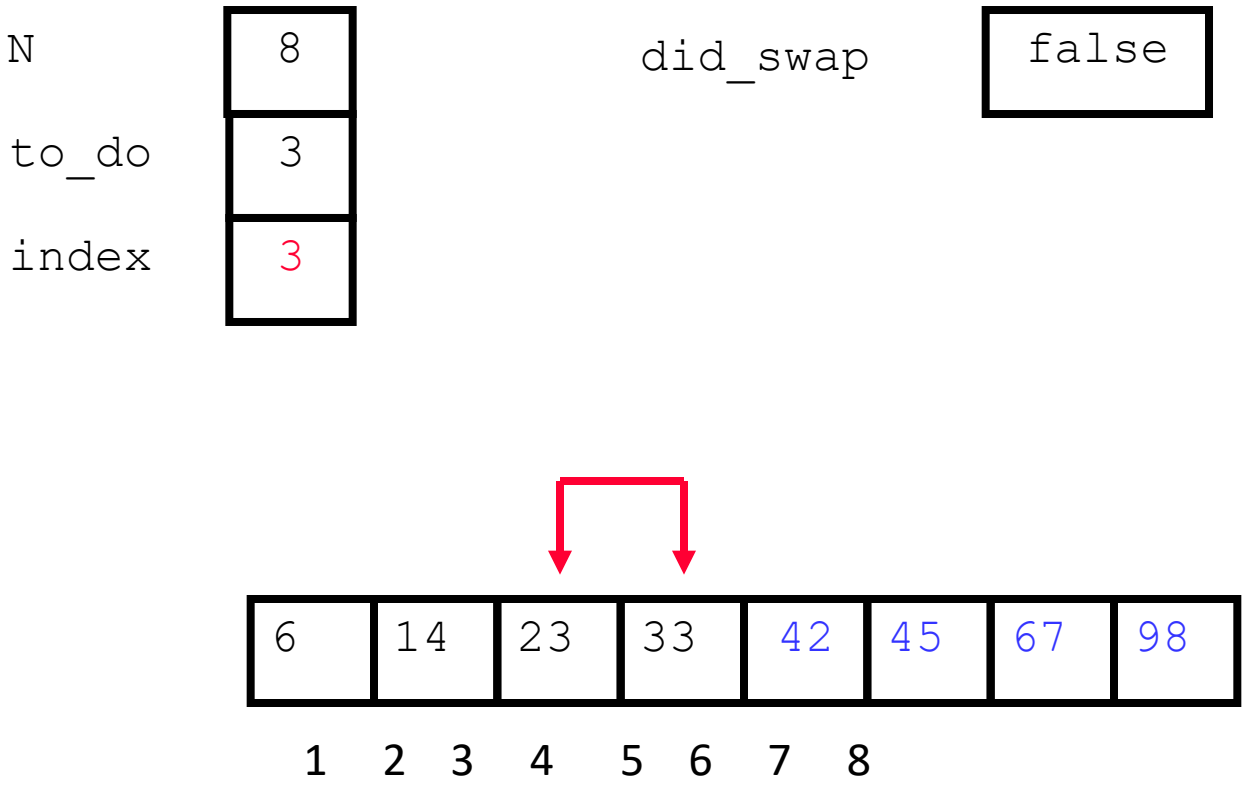
The Fifth “Bubble Up”



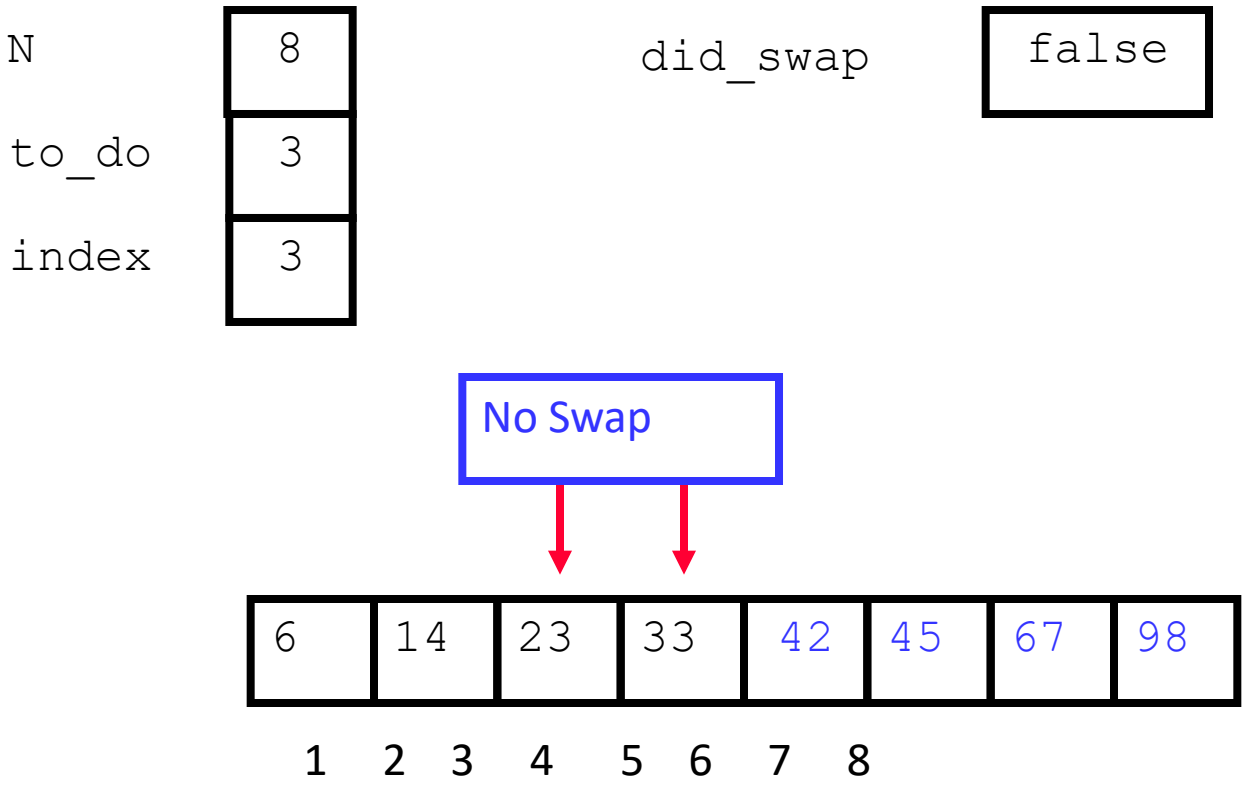
The Fifth “Bubble Up”



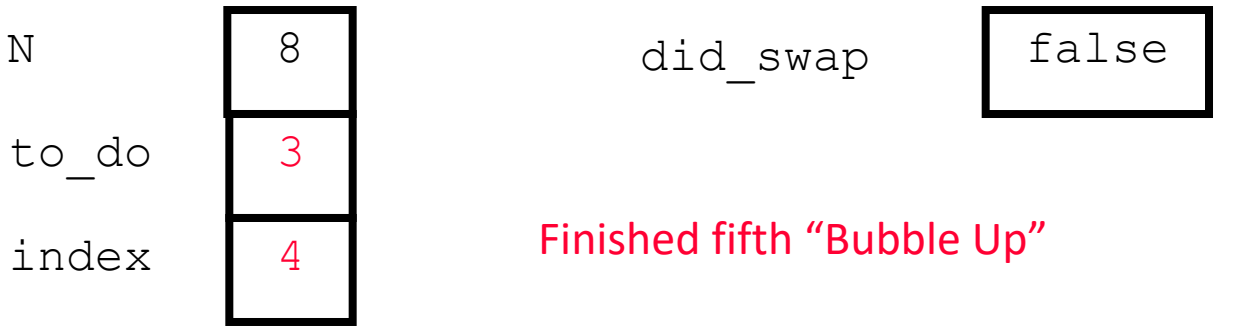
The Fifth “Bubble Up”

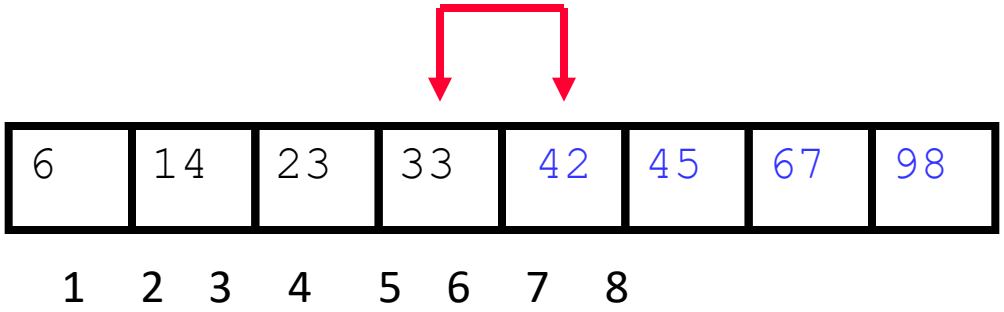


The Fifth “Bubble Up”

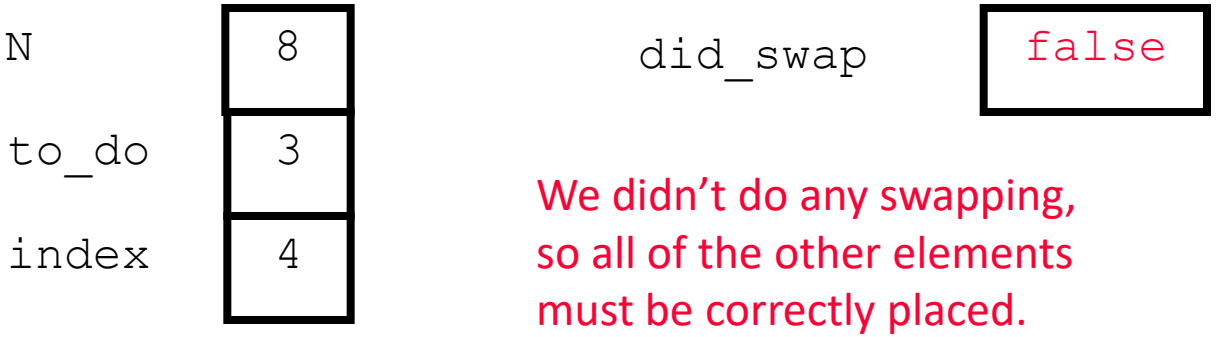


After Fifth Pass of Outer Loop



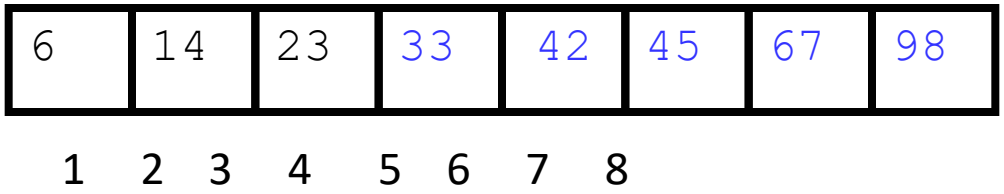


Finished “Early”



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

- **"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**

Bubble Sort

Pseudo-code Algorithm

```
public static void bubbleSort(Comparable a[]) {
   for (int p=a.length-1; p>0; p--) {
      for (int j=0; j<p; j++)
         if (a[j].compareTo(a[j+1])>0)
            swap(a,j,j+1);
   } // p
} // bubbleSort
```

Bubble Sort – Analysis

- ***Best-case:*** ➔ **O(n)**
  - Array is already sorted in ascending order.
  - The number of moves: 0 ➔ O(1)
  - The number of key comparisons: (n-1) ➔ O(n)
- ***Worst-case:*** ➔ **$O(n^2)$**
  - Array is in reverse order:
  - Outer loop is executed n-1 times,
  - The number of moves: 3*(1+2+...+n-1) = 3 * n*(n-1)/2 ➔ $O(n^2)$
  - The number of key comparisons: (1+2+...+n-1)= n*(n-1)/2 ➔ $O(n^2)$
- Average-case: ➔ **$O(n^2)$**
  - We have to look at all possible initial data organizations.
- **So, Bubble Sort is $O(n^2)$**

Quiz 1

What are the correct intermediate steps of the following data set when it is being sorted with the bubble sort? 15,20,10,18

A. 15,10,20,18 -- 15,10,18,20 -- 10,15,18,20

B. 10, 20,15,18 -- 10,15,20,18 -- 10,15,18,20

C. 15,20,10,18 -- 15,10,20,18 -- 10,15,20,18 -- 10,15,18,20

D. 15,18,10,20 -- 10,18,15,20 -- 10,15,18,20 -- 10,15,18,20

Quiz 2

- What is the maximum number of comparisons if there are 5 elements in array x?
- **A.** 10
- **B.** 2
- **C.** 5
- **D.** 25