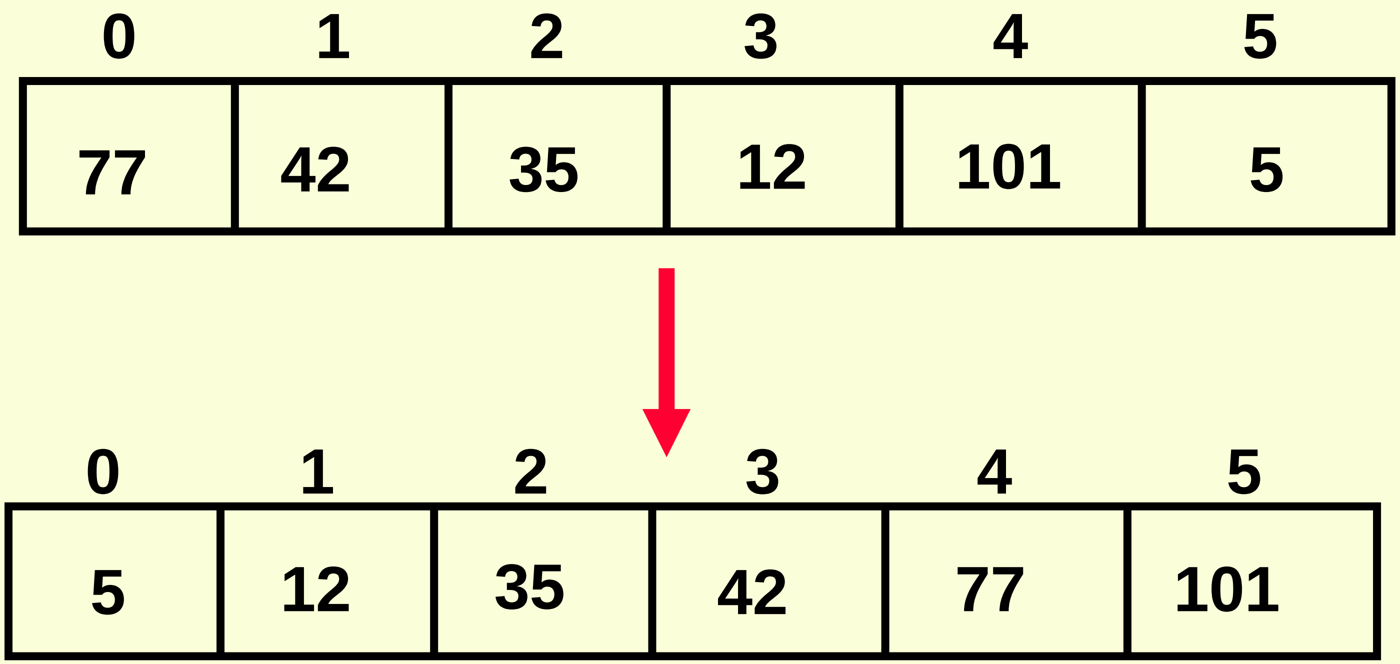
Bubble Sort

With thanks to Bill Leahy, GaTech for original slide deck

Sorting

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



Sorting

- Lots of algorithms for sorting
  - Selection sort (video lecture and textbook)
  - Insertion sort (textbook)
  - Bubble sort
  - Merge sort
  - Radix sort
  - ...

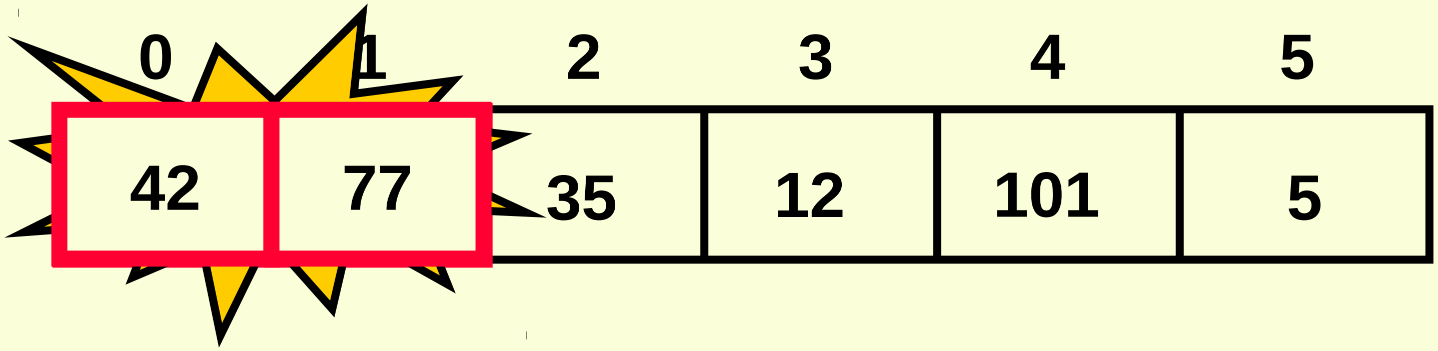
"Bubbling Up" the Largest Element

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

0	1	2	3	4	5
77	42	35	12	101	5

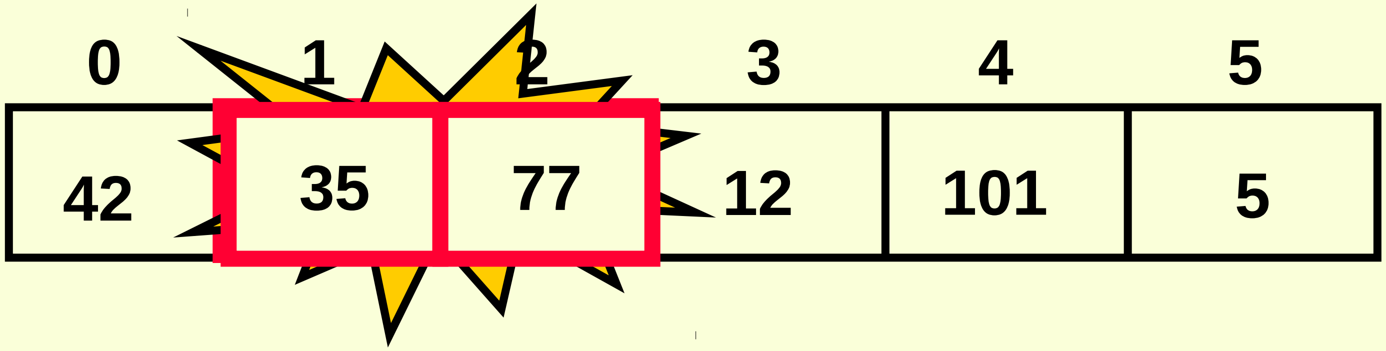
"Bubbling Up" the Largest Element

- 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



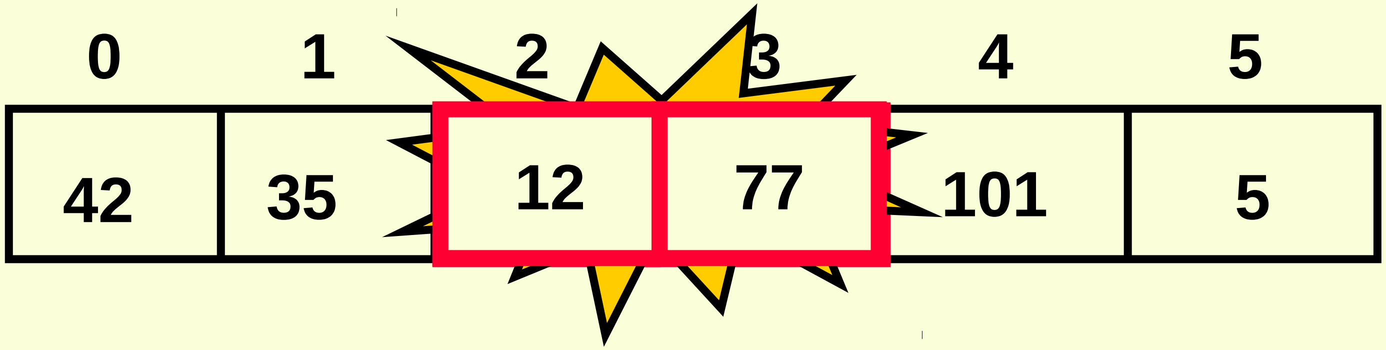
"Bubbling Up" the Largest Element

- 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



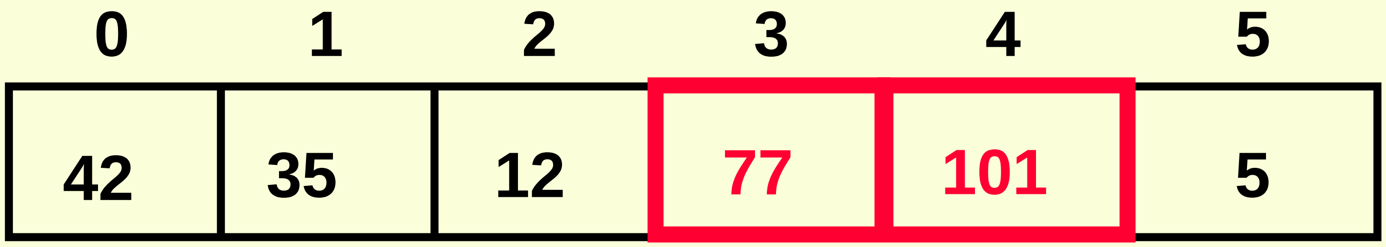
"Bubbling Up" the Largest Element

- 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

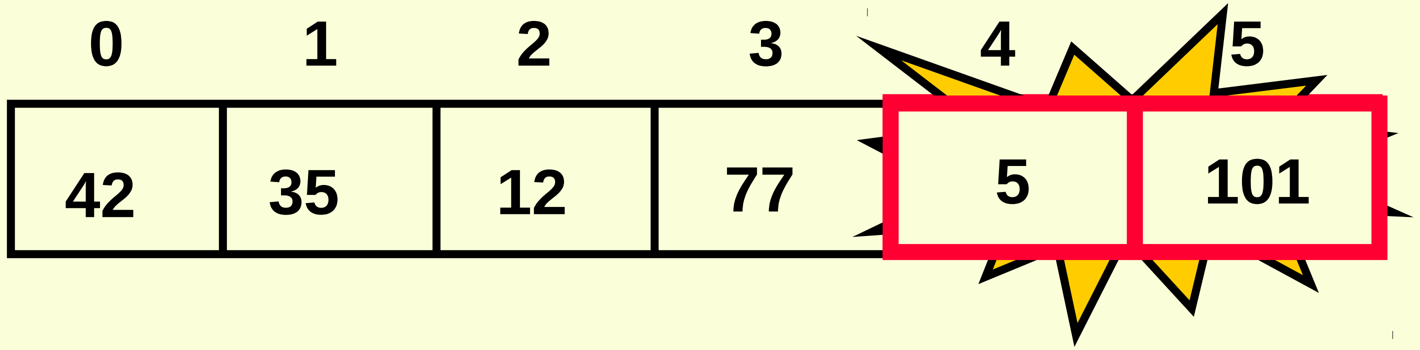
- Traverse a collection of elements
  - Move from the front to the end
  - “Bubble” the largest value to the end using pair-wise comparisons and swapping



No need to swap

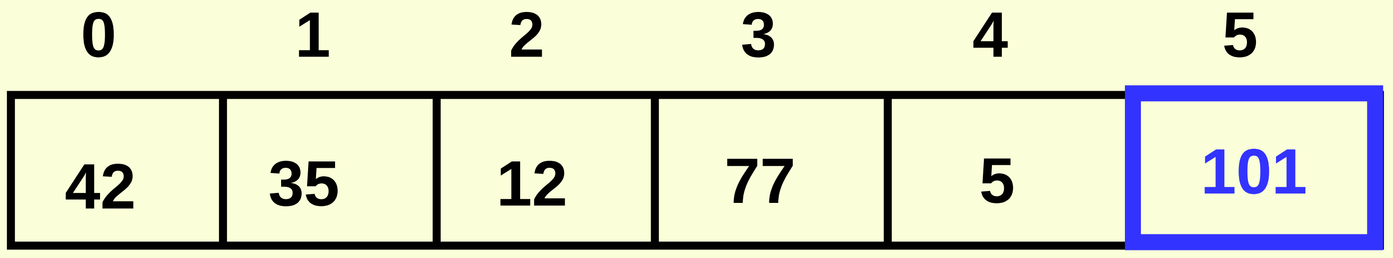
"Bubbling Up" the Largest Element

- 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

The “Bubble Up” Algorithm

```
Given some array, a

for(int i = 0; i < a.length-1; i++) {
  if(a[i] > a[i+1]) {
    int temp = a[i];
    a[i] = a[i+1];
    a[i+1] = temp;
  }
}
```

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

0	1	2	3	4	5
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

0	1	2	3	4	5
42	35	12	77	5	101

0	1	2	3	4	5
35	12	42	5	77	101

0	1	2	3	4	5
12	35	5	42	77	101

0	1	2	3	4	5
12	5	35	42	77	101

0	1	2	3	4	5
5	12	35	42	77	101

Reducing the Number of Comparisons

0	1	2	3	4	5
77	42	35	12	101	5

0	1	2	3	4	5
42	35	12	77	5	101

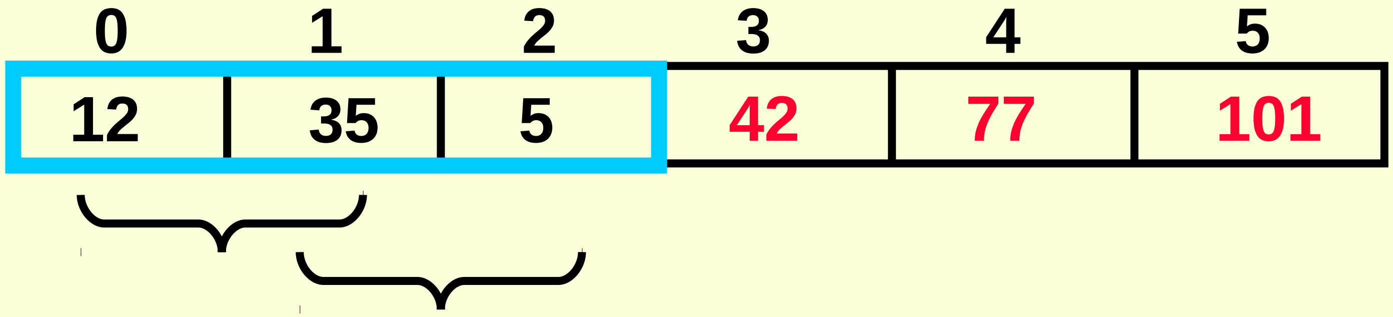
0	1	2	3	4	5
35	12	42	5	77	101

0	1	2	3	4	5
12	35	5	42	77	101

0	1	2	3	4	5
12	5	35	42	77	101

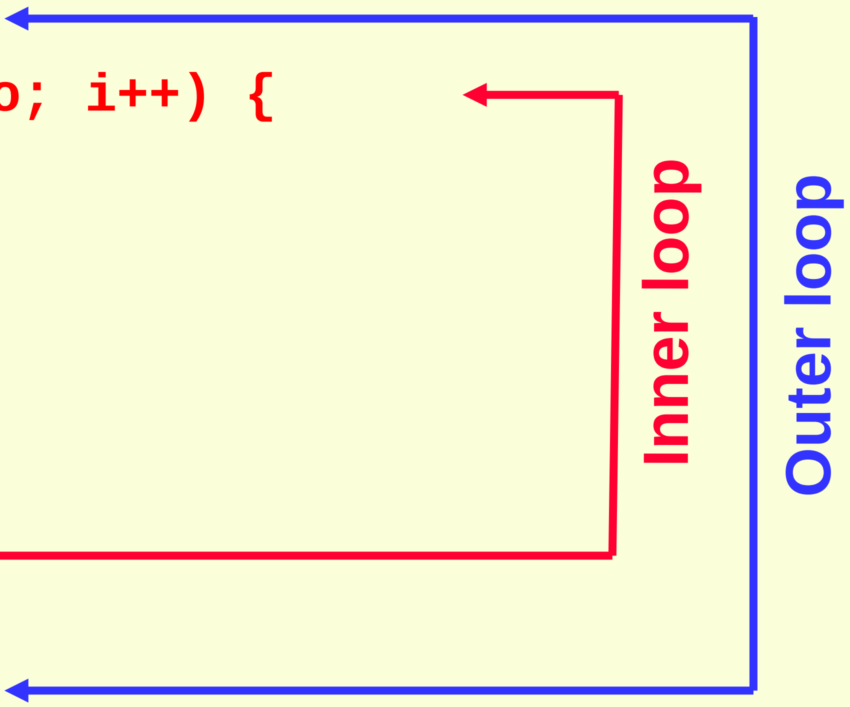
Reducing the Number of Comparisons

- On the Nth "bubble up", we only need to do MAX-N comparisons.
- For example:
  - This is the 4th "bubble up"
  - MAX is 6 (total number of elements in array)
  - Thus we have 2 comparisons to do



Putting It All Together

```
public static void Bubblesort(int[] a){
  int to_do = a.length-1;

  while (to_do != 0) {
    for(int i = 0; i < to_do; i++) {
      if(a[i] > a[i+1]) {
        int temp = a[i];
        a[i] = a[i+1];
        a[i+1] = temp;
      }
     }

    to_do = to_do - 1
} //end method
```



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”!

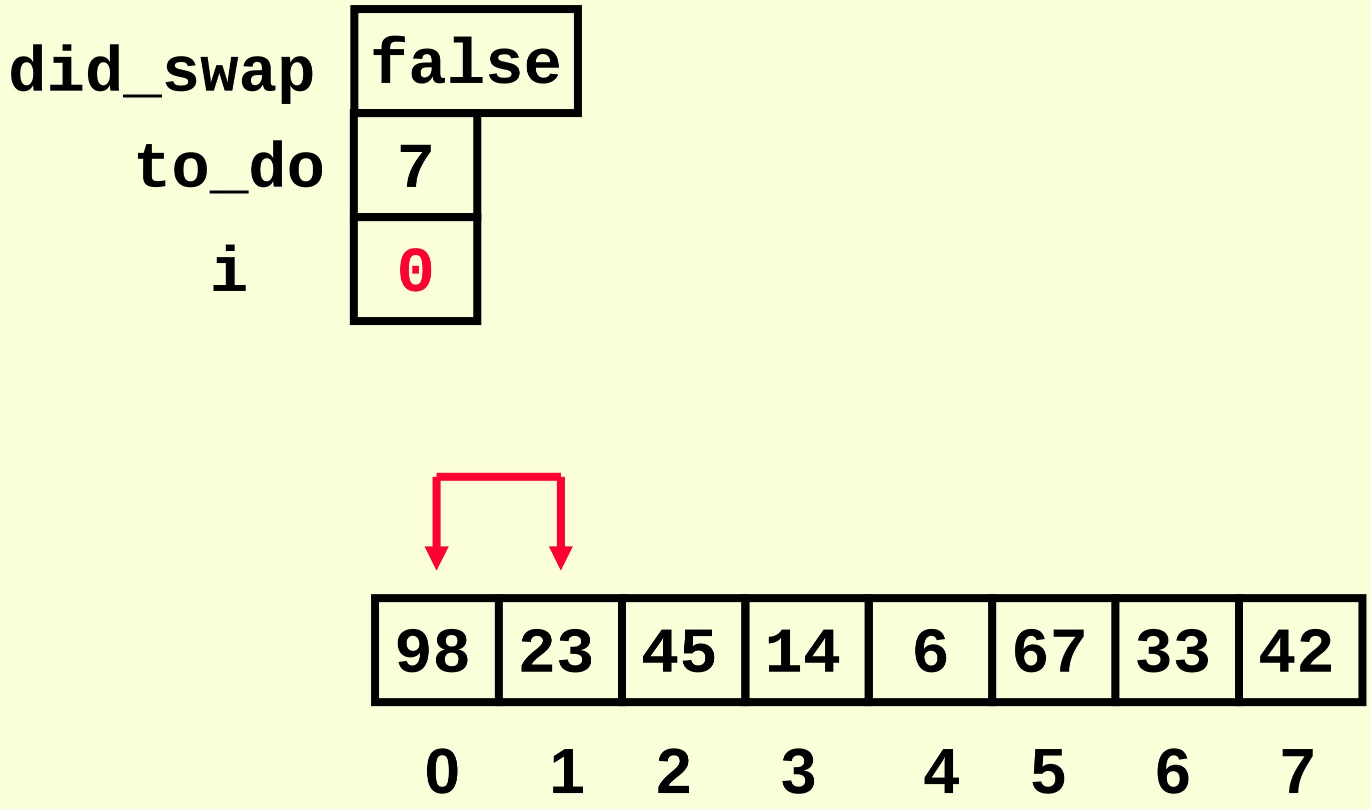
0	1	2	3	4	5
5	12	35	42	77	101

Using a Boolean “Flag”

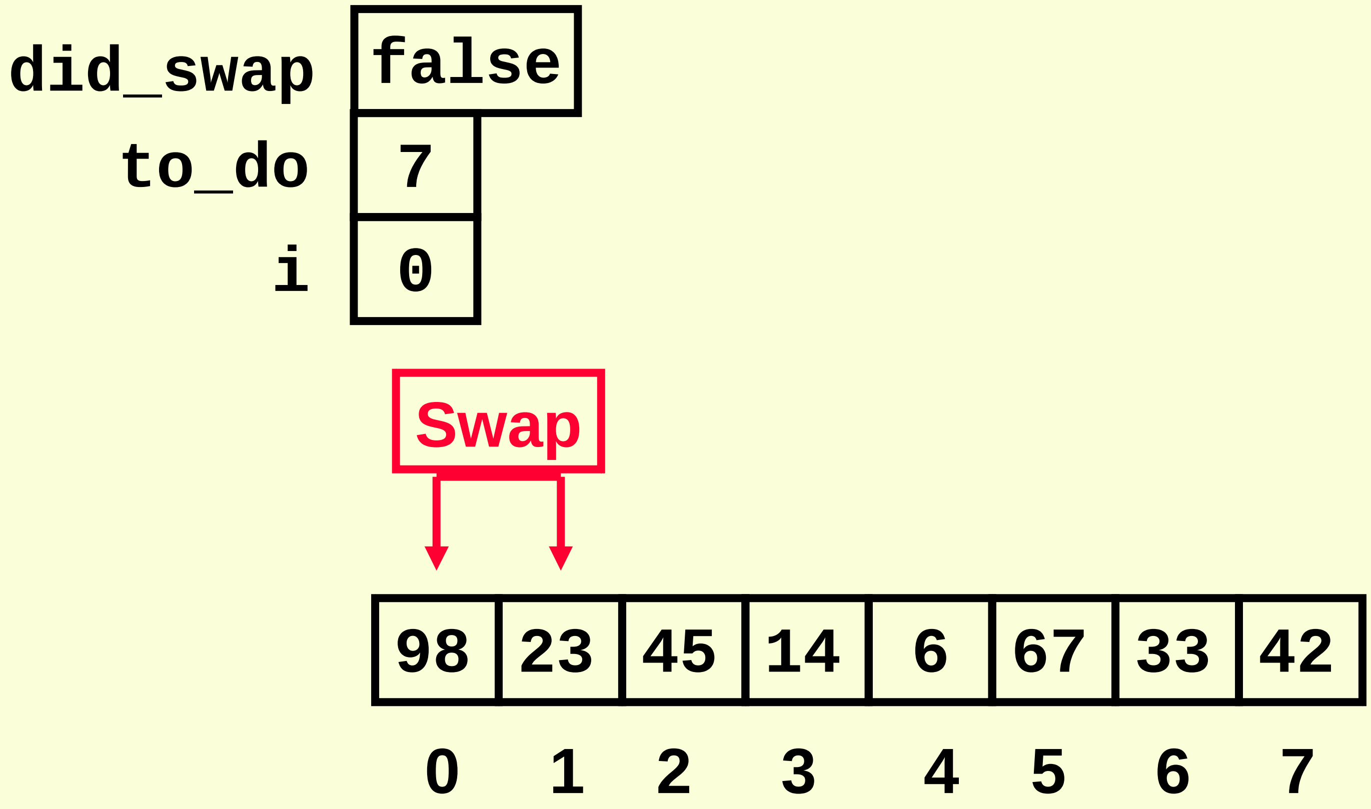
- We can use a boolean variable to determine if any swapping occurred during the “bubble up.”
- If no swapping occurred, then we know that the collection is already sorted!
- This boolean “flag” needs to be reset after each “bubble up.”

```
public static void Bubblesort(int[] a){
  int to_do = a.length-1;
  boolean did_swap = true;

  while (to_do != 0 && did_swap) {
    did_swap = false;
    for(int i = 0; i < to_do; i++) {
      if(a[i] > a[i+1]) {
        did_swap = true;
        int temp = a[i];
        a[i] = a[i+1];
        a[i+1] = temp;
      }
     }

    to_do = to_do - 1
} //end method
```

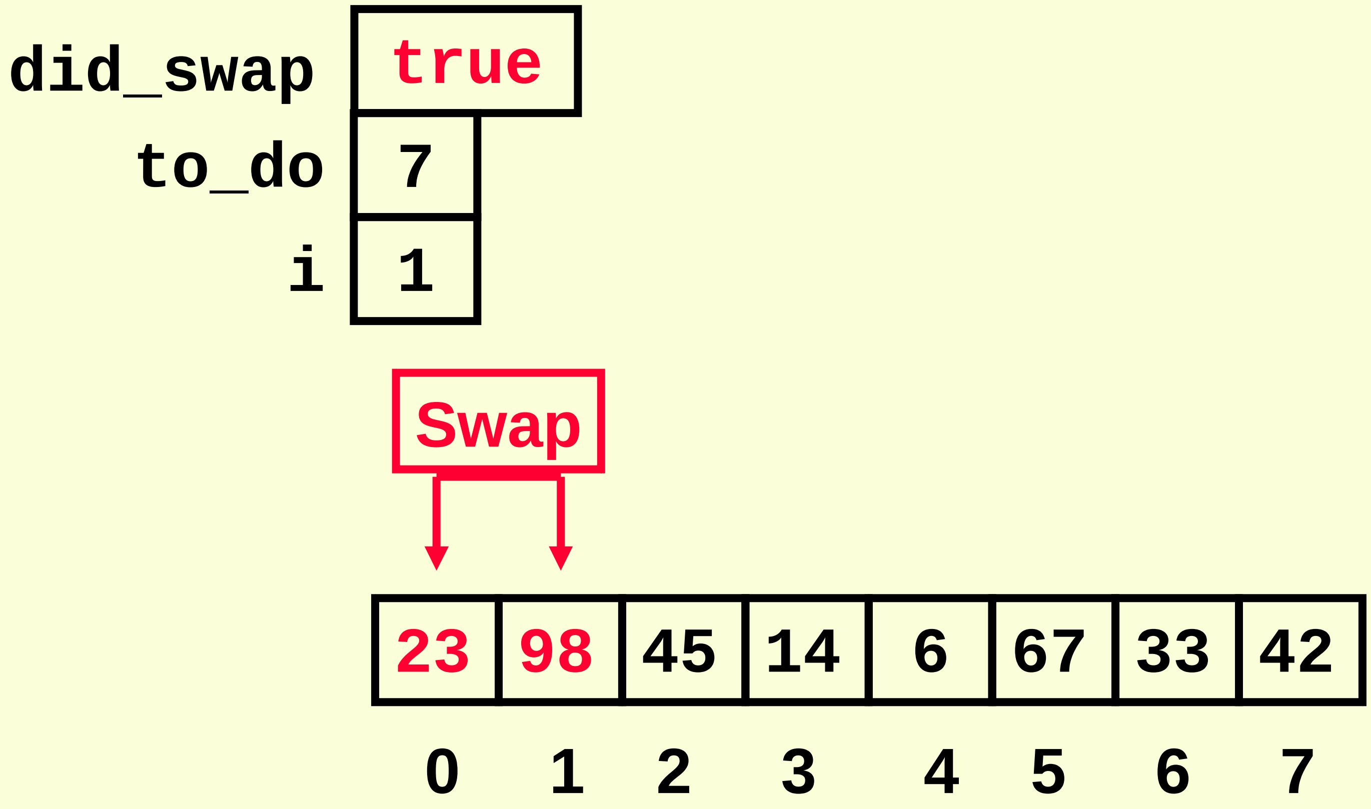
An Animated Example



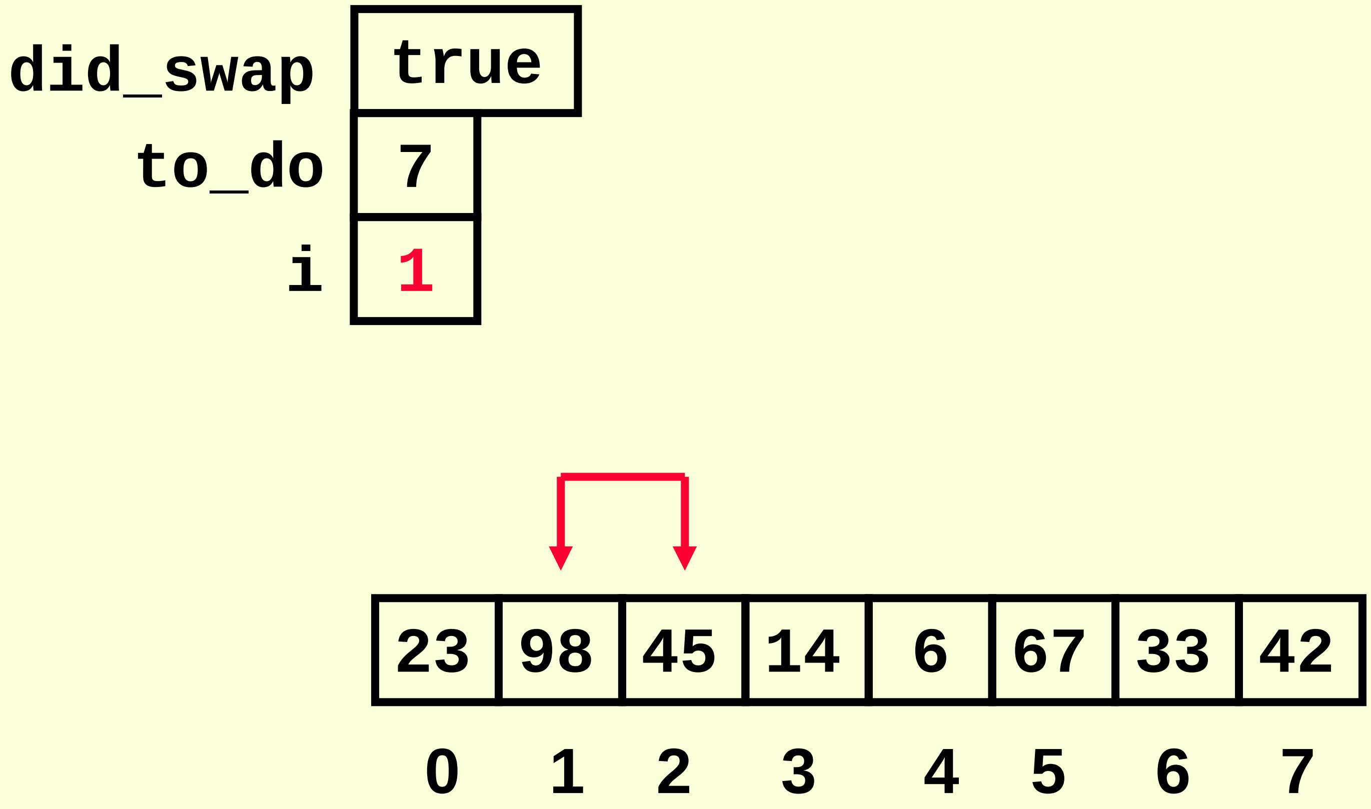
An Animated Example



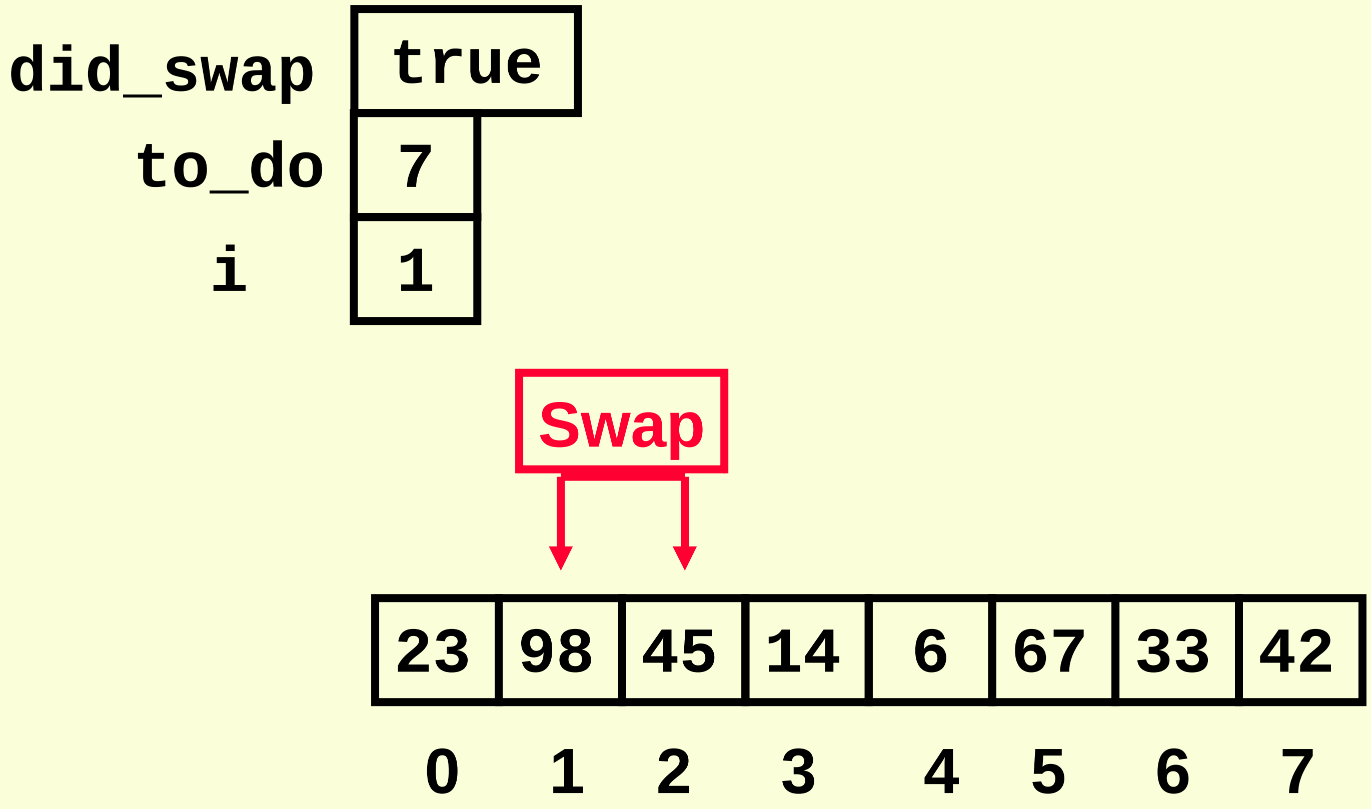
An Animated Example



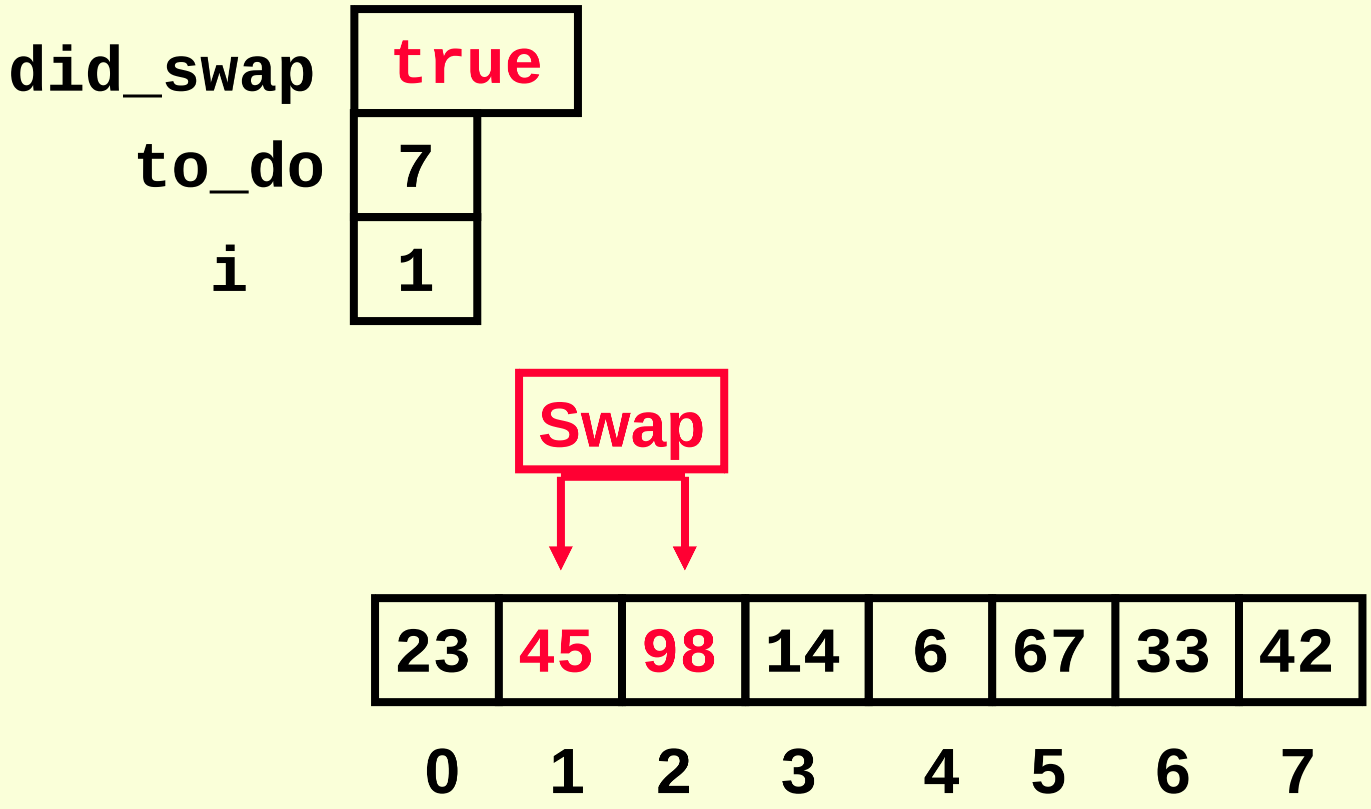
An Animated Example



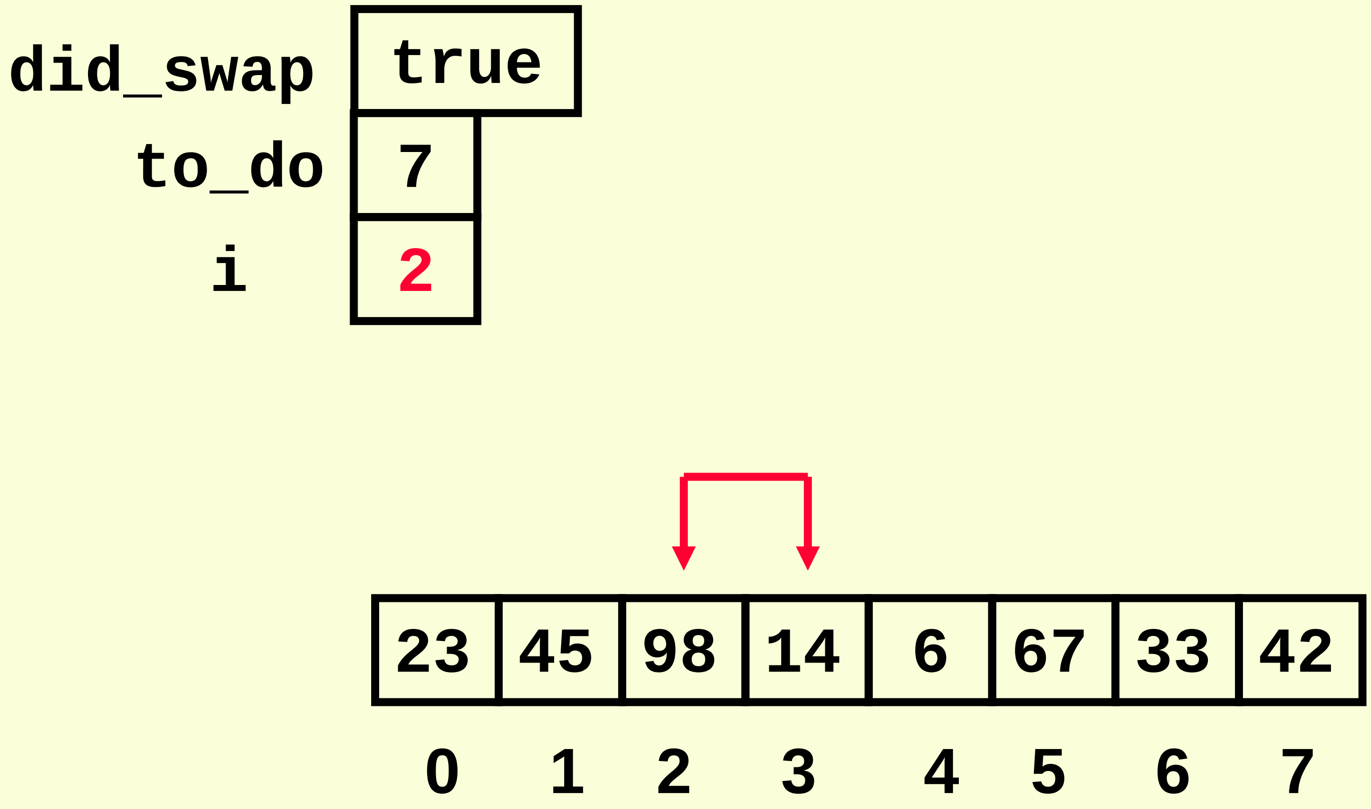
An Animated Example



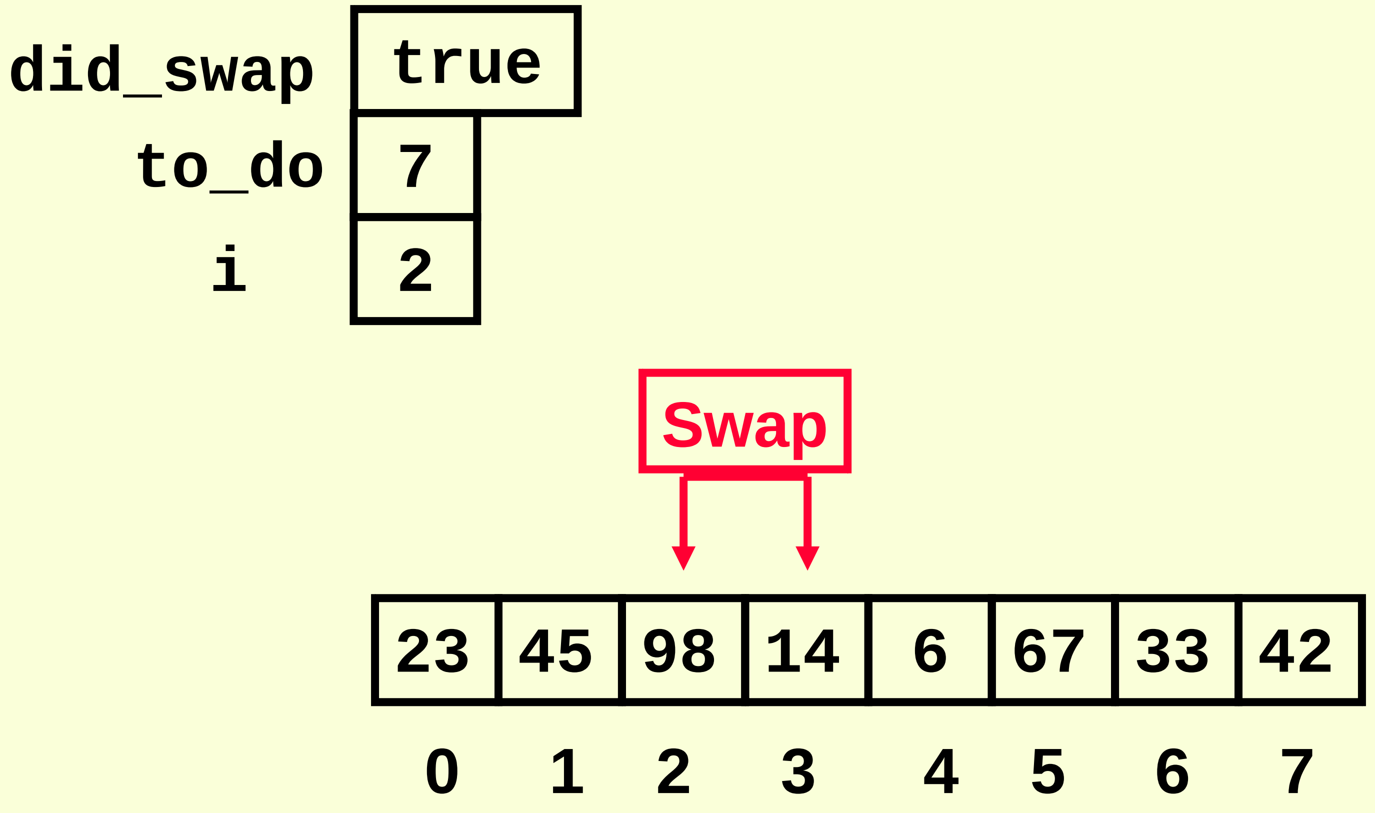
An Animated Example



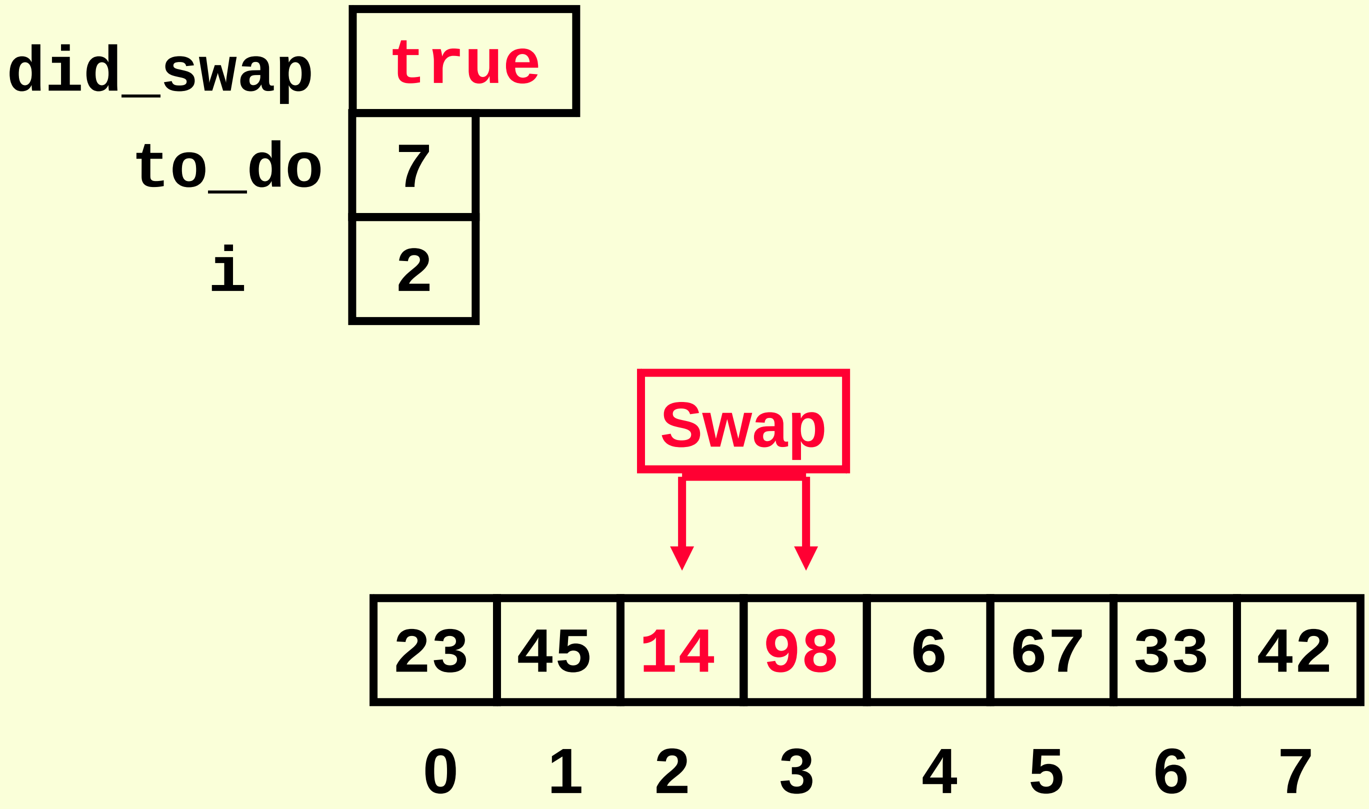
An Animated Example



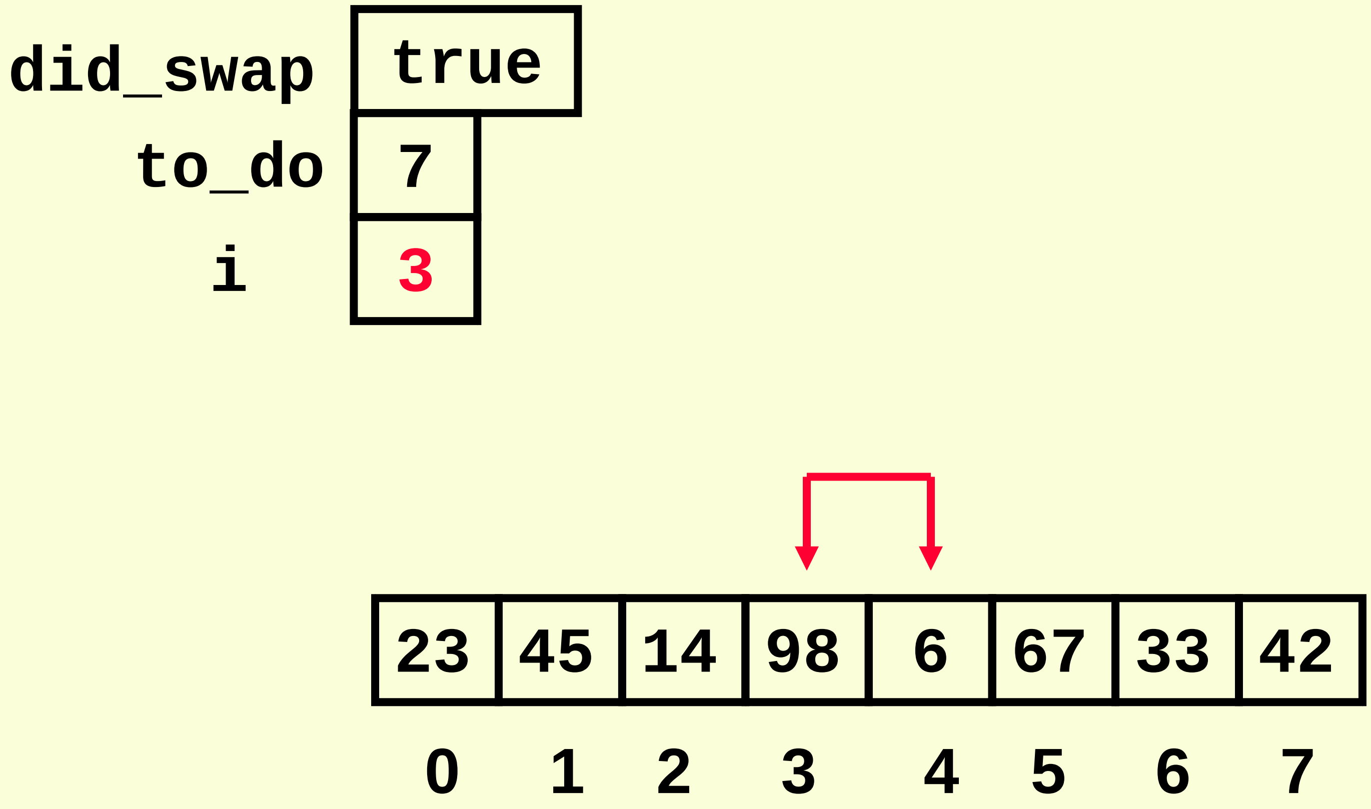
An Animated Example



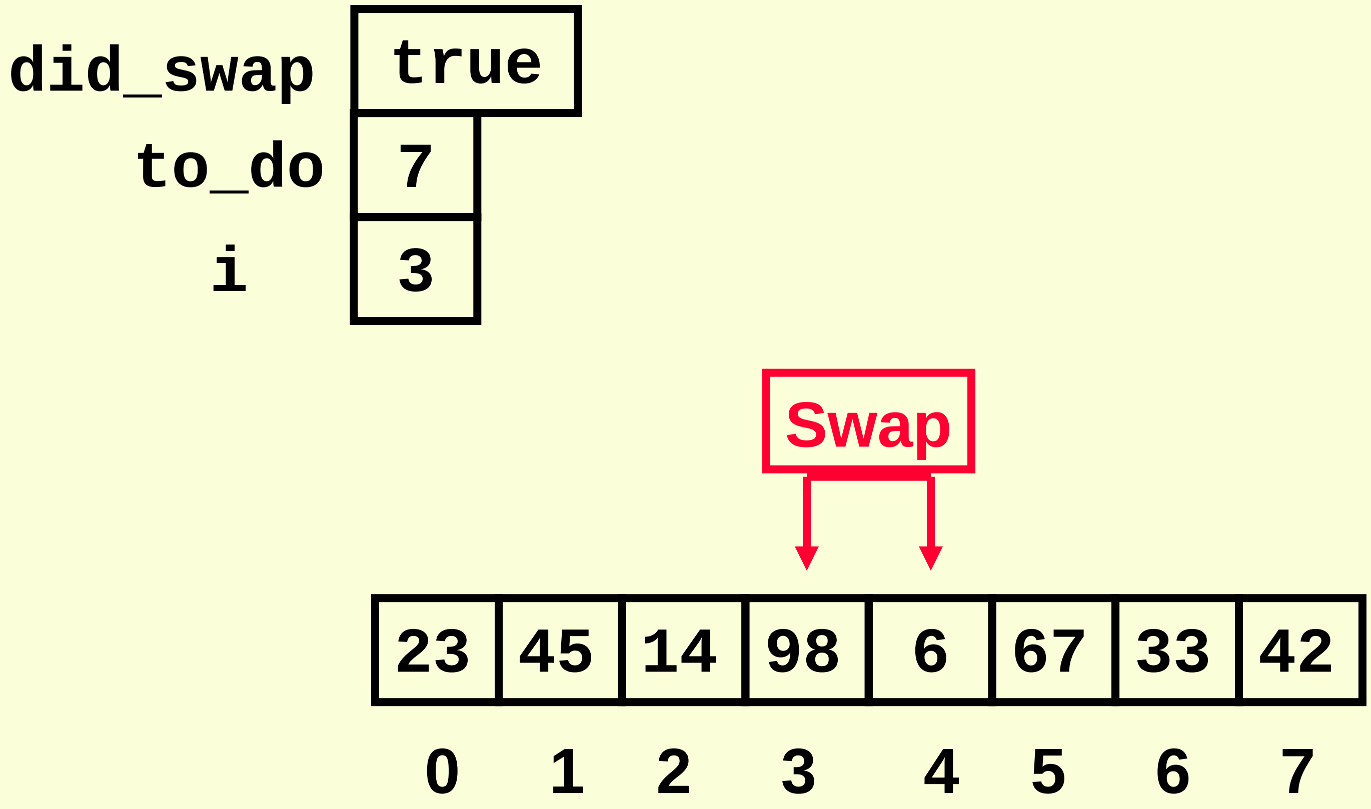
An Animated Example



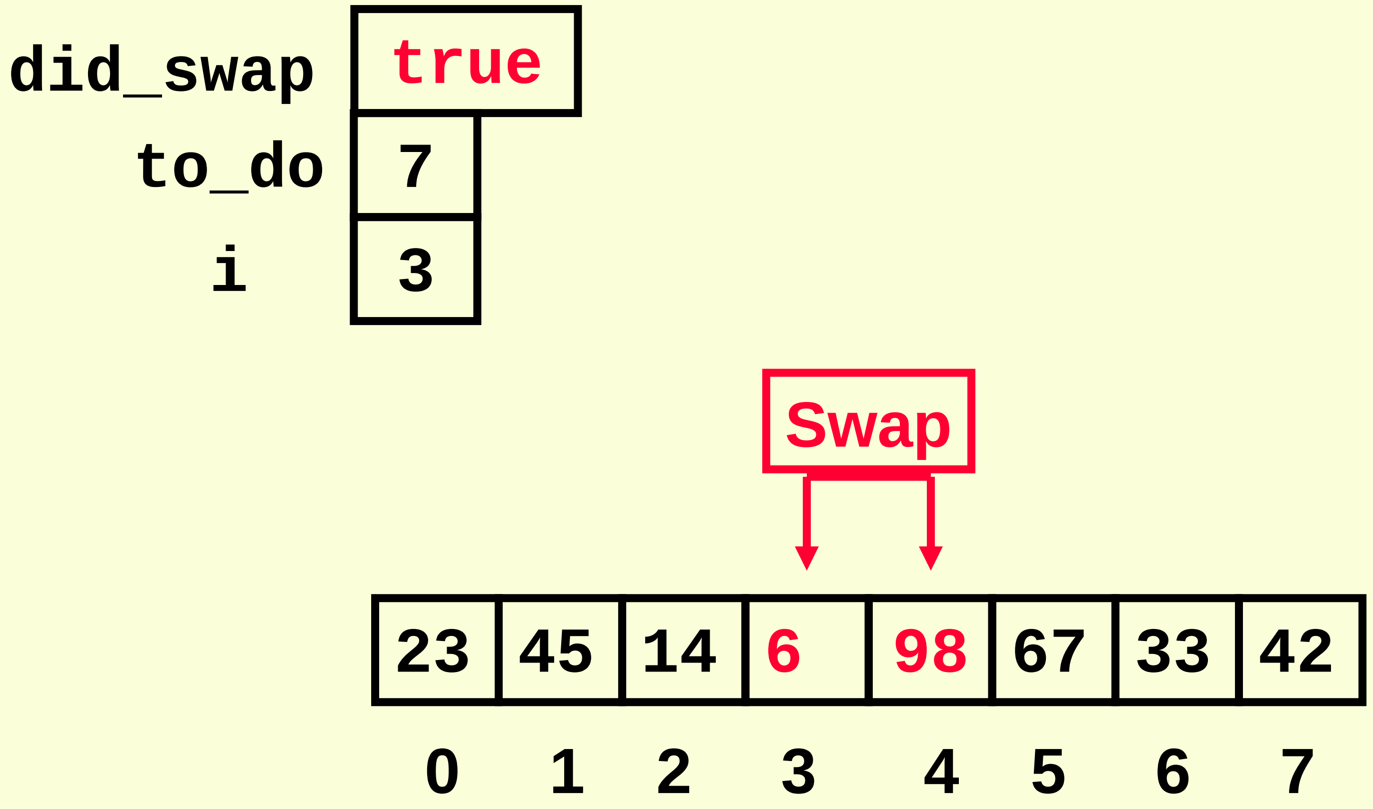
An Animated Example



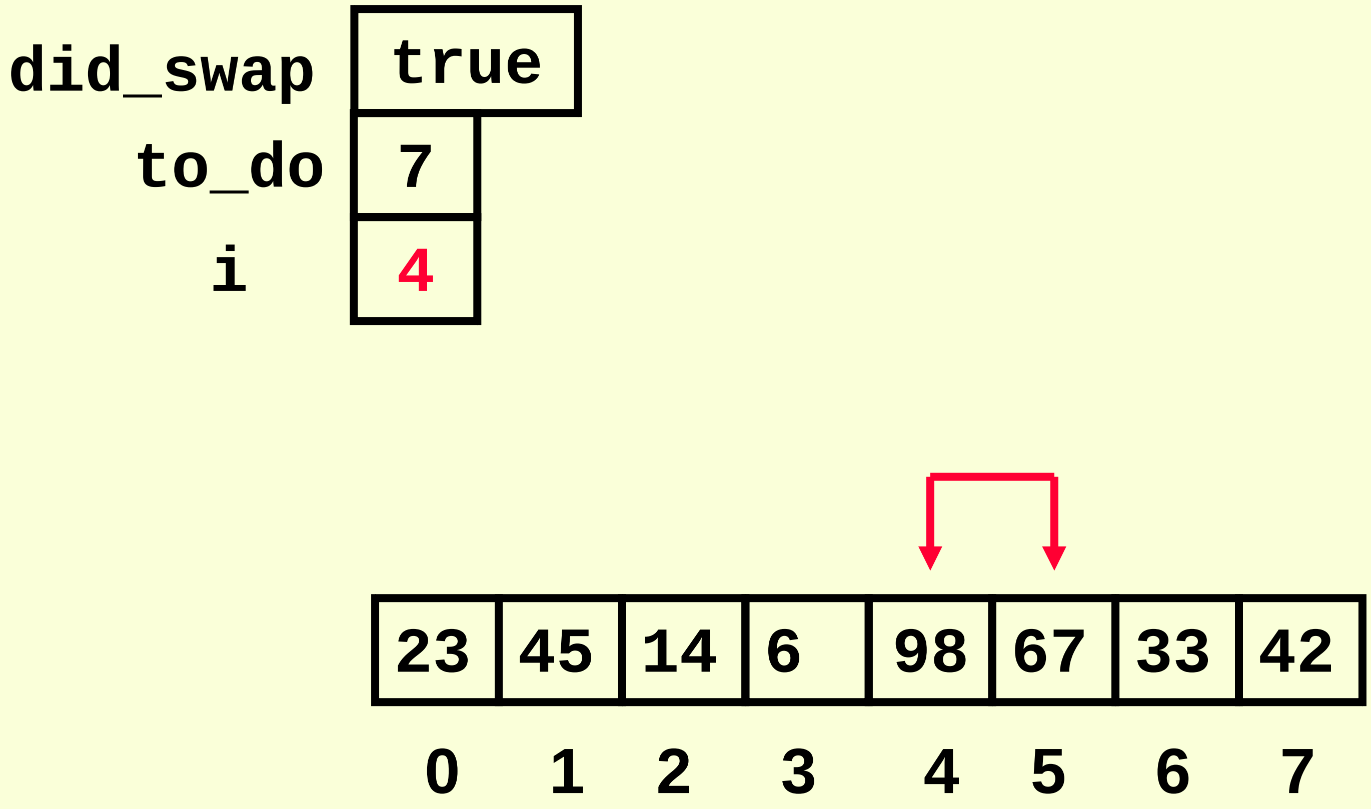
An Animated Example



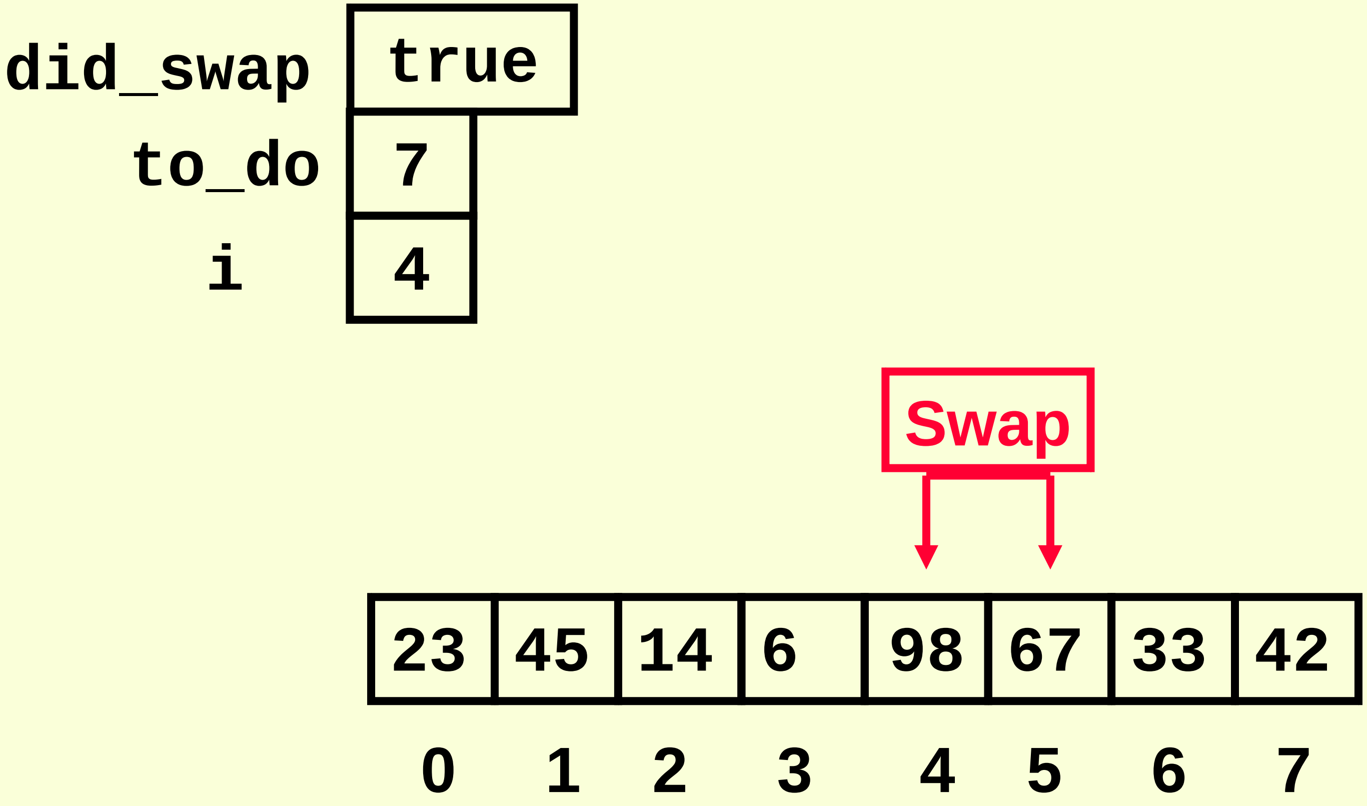
An Animated Example



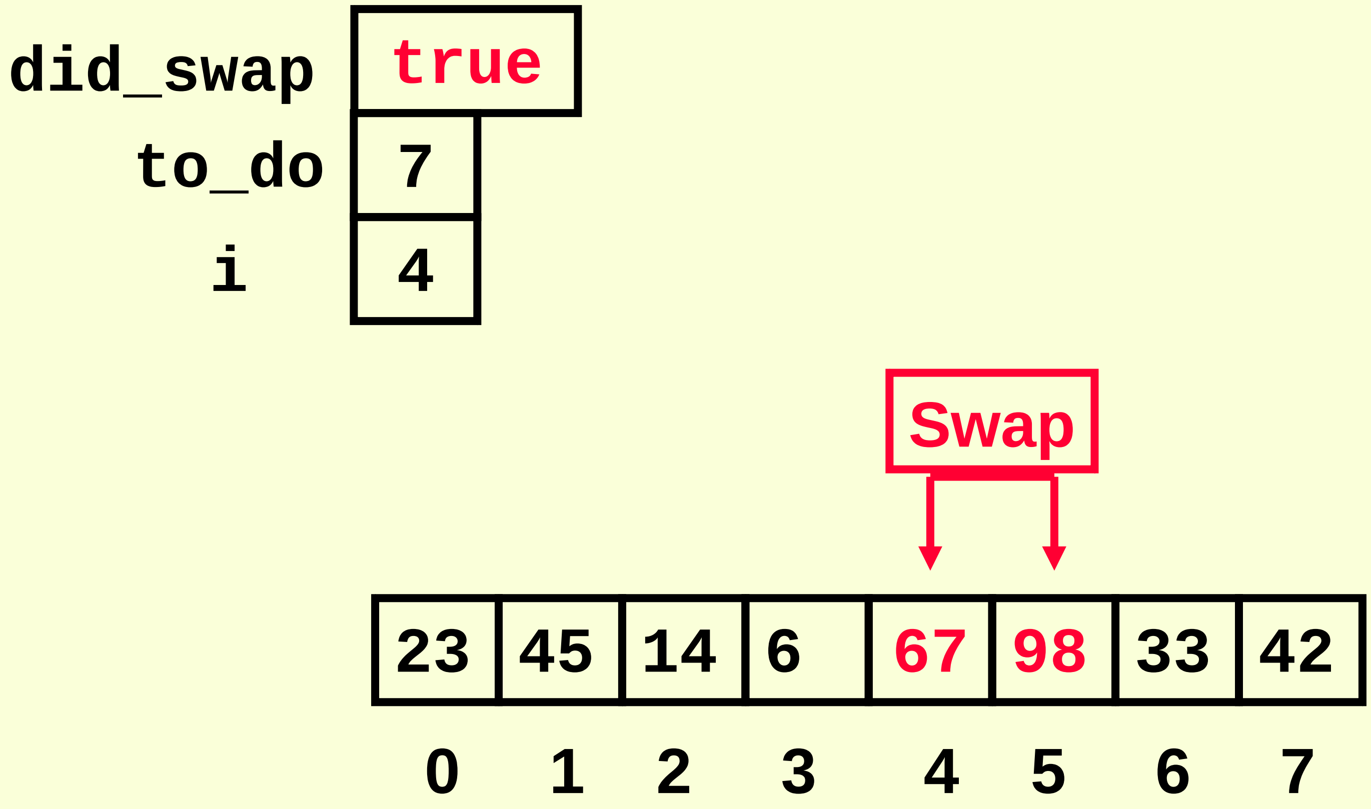
An Animated Example



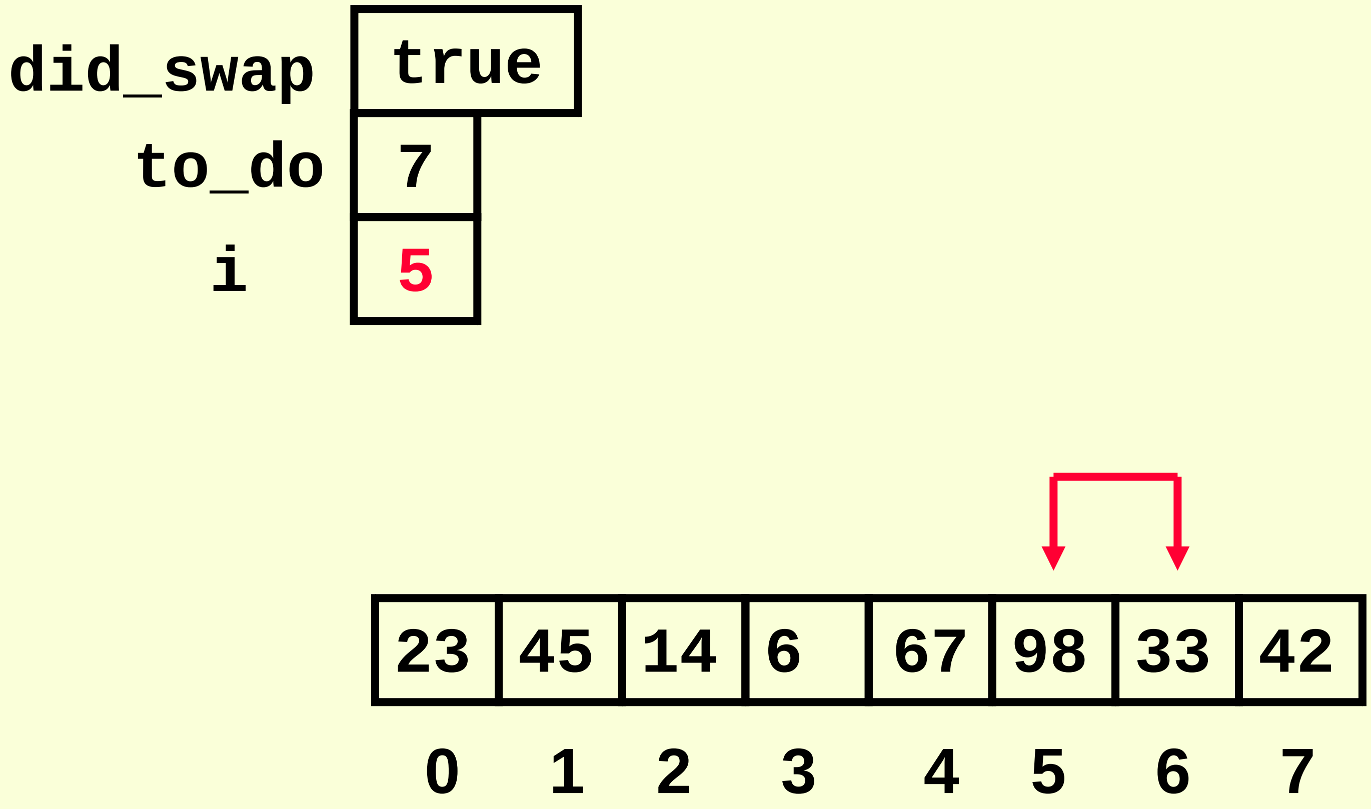
An Animated Example



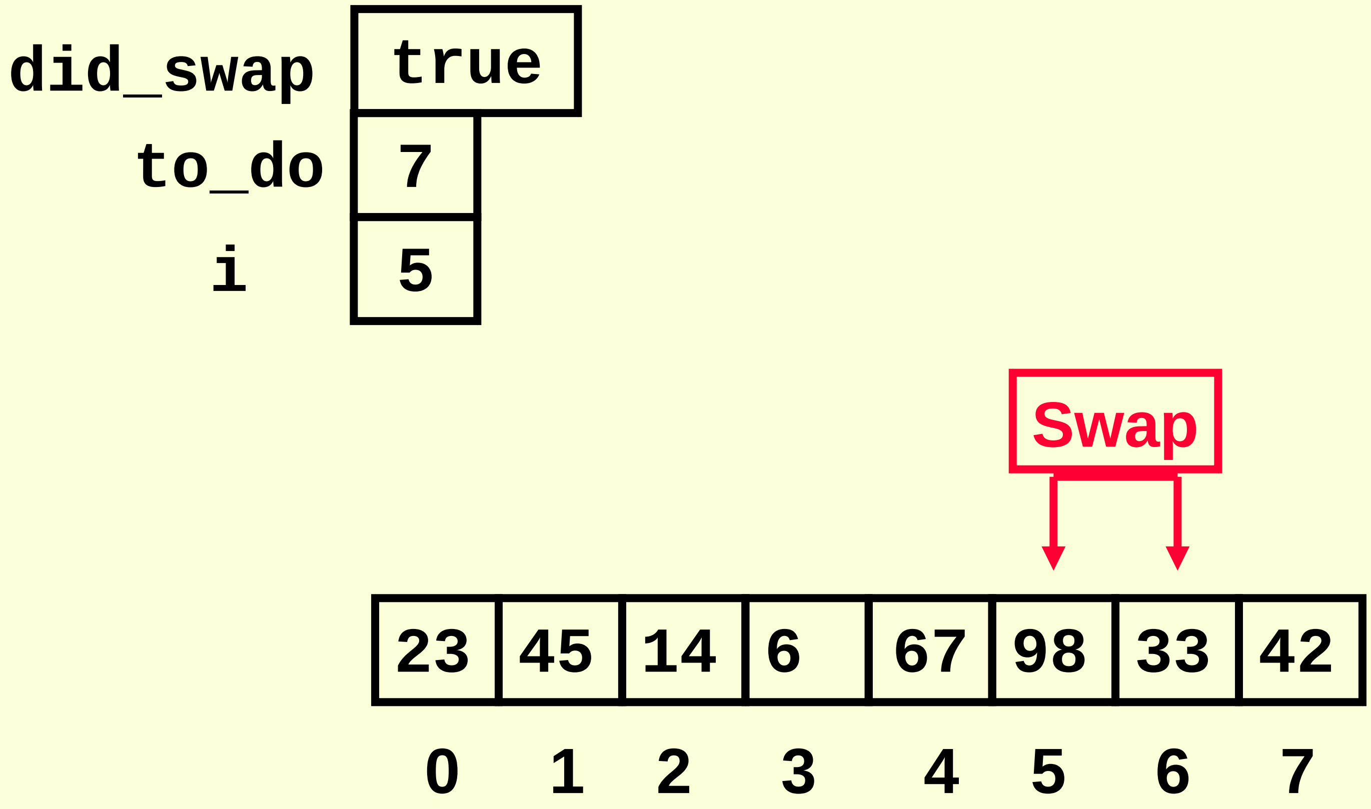
An Animated Example



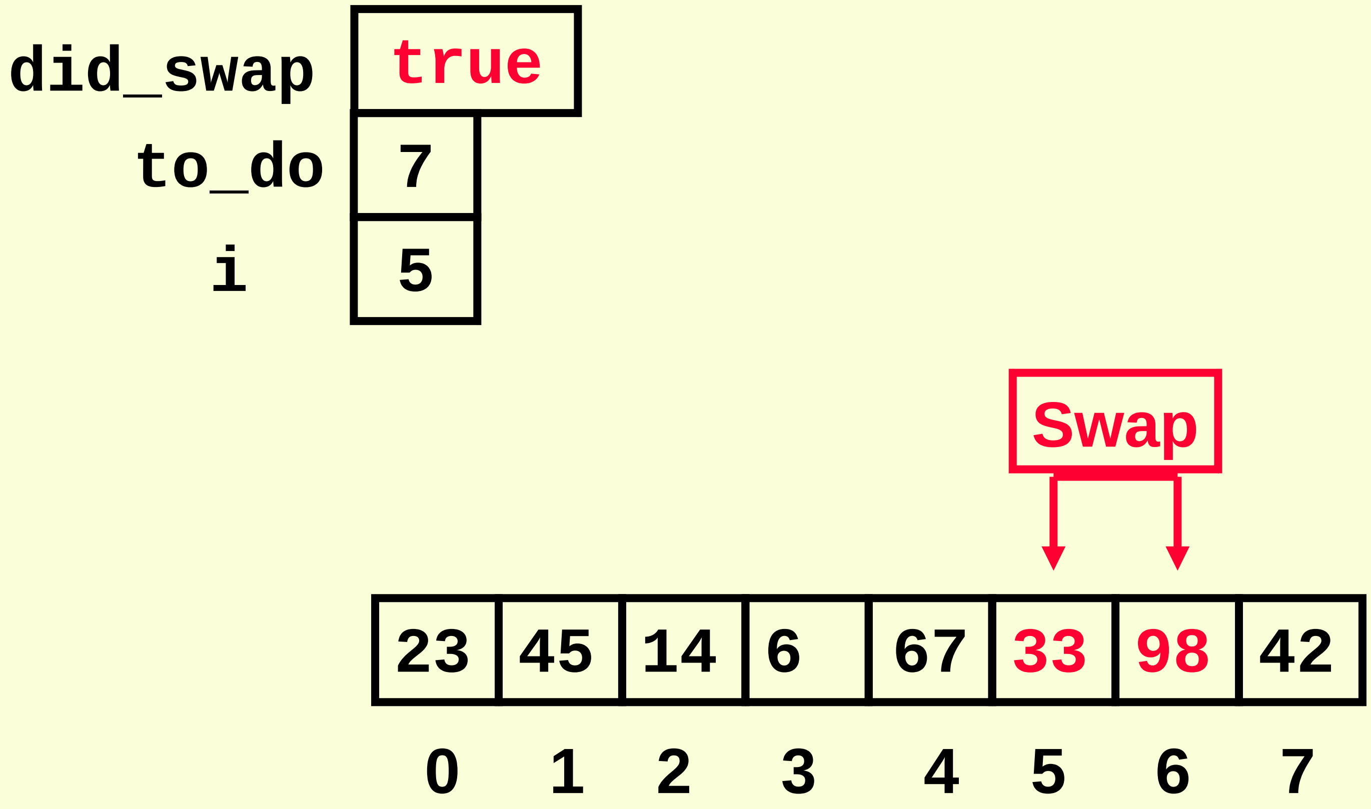
An Animated Example



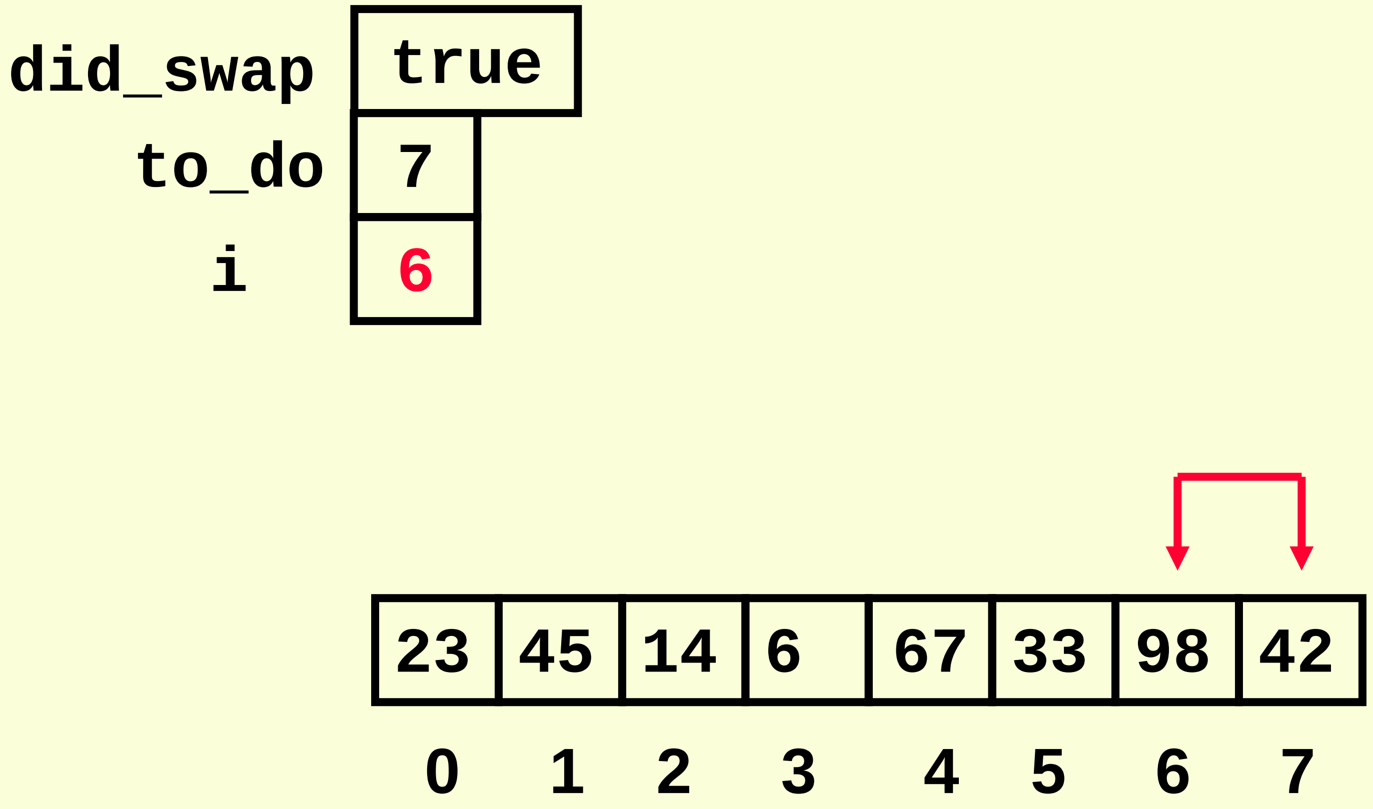
An Animated Example



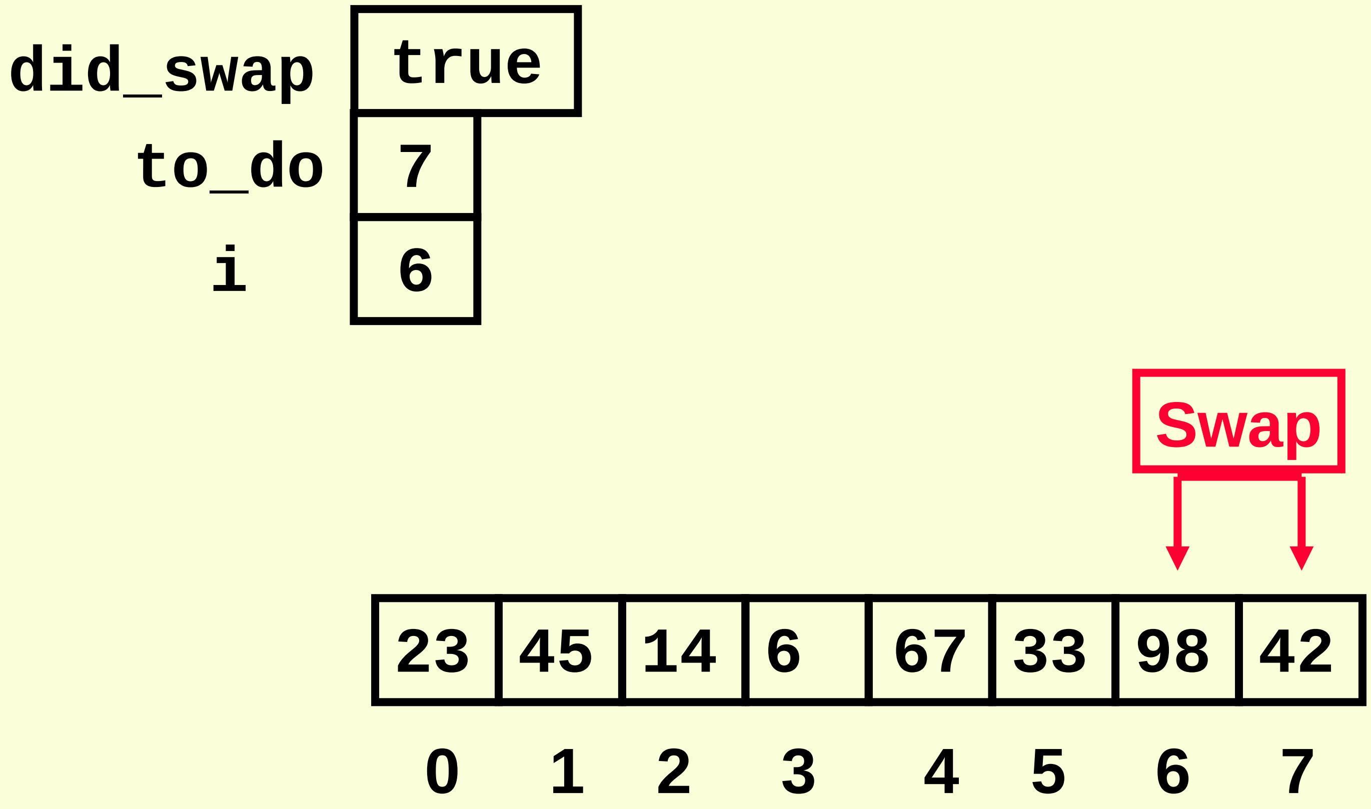
An Animated Example



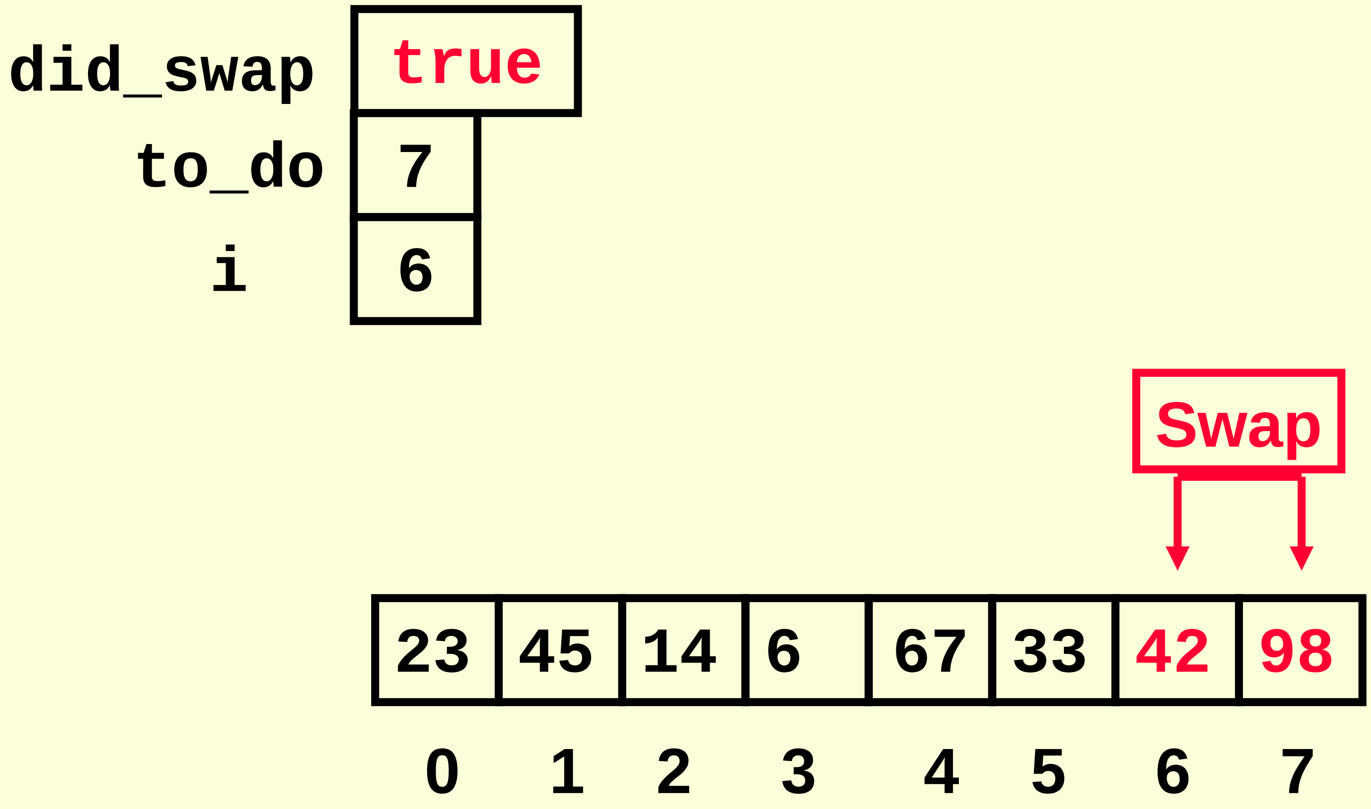
An Animated Example



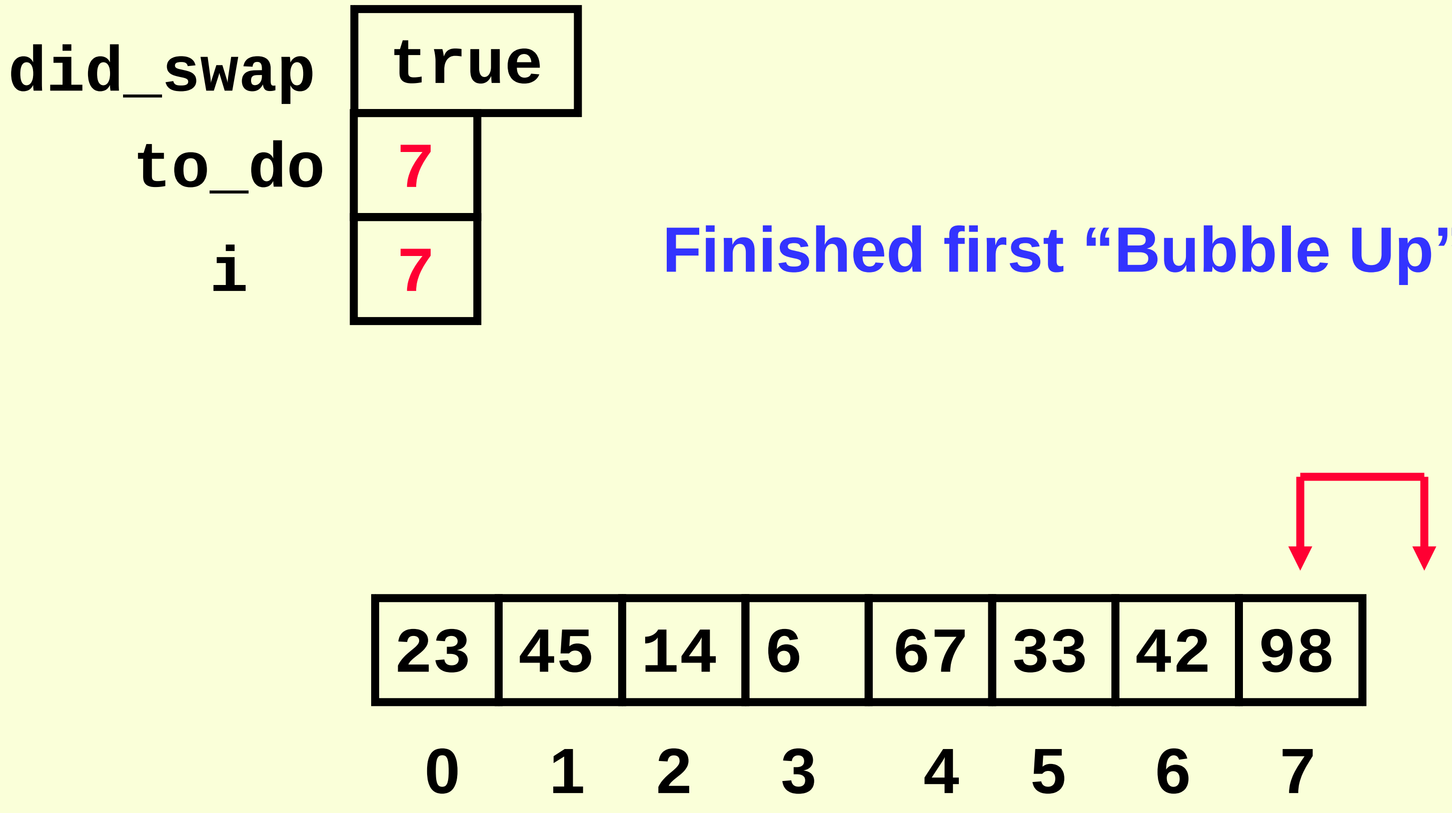
An Animated Example



An Animated Example

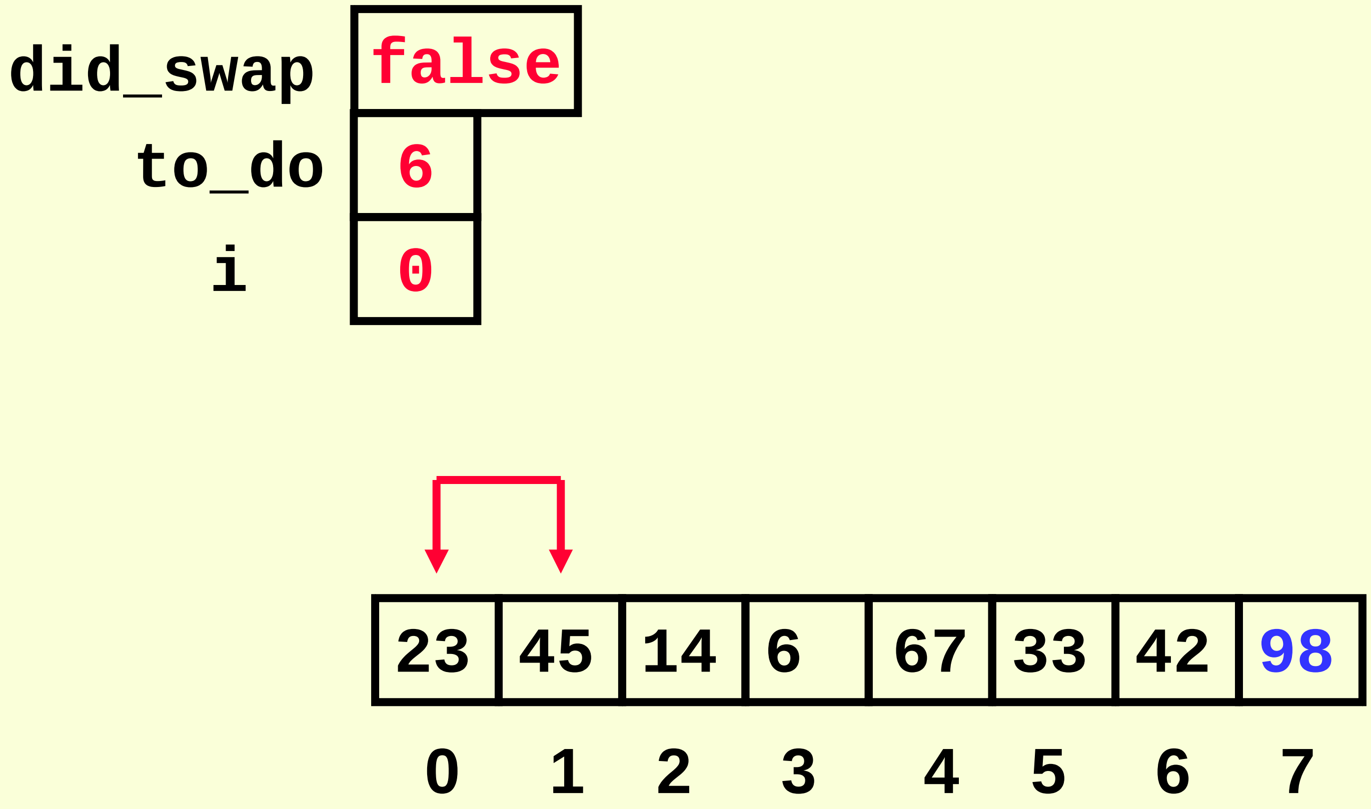


After First Pass of Outer Loop

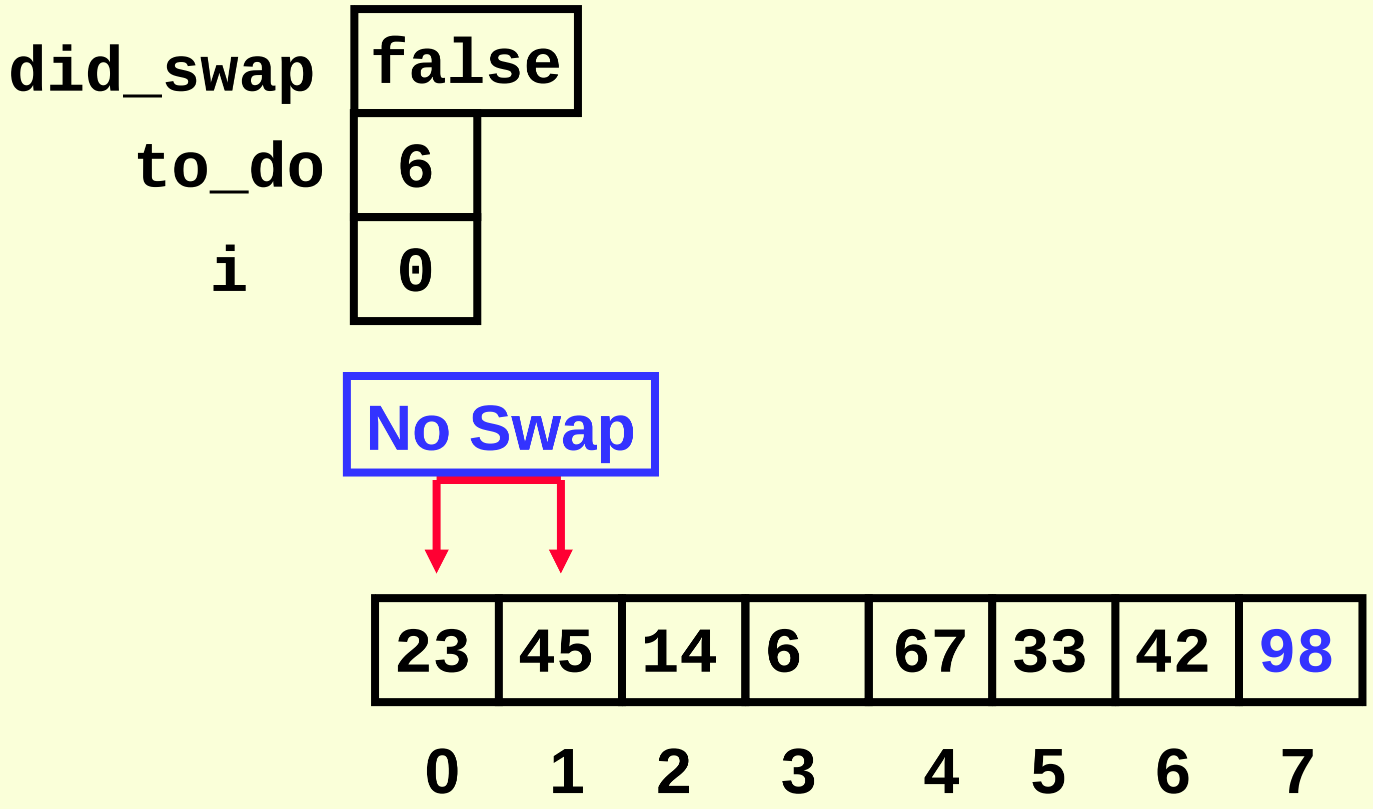


Finished first "Bubble Up"

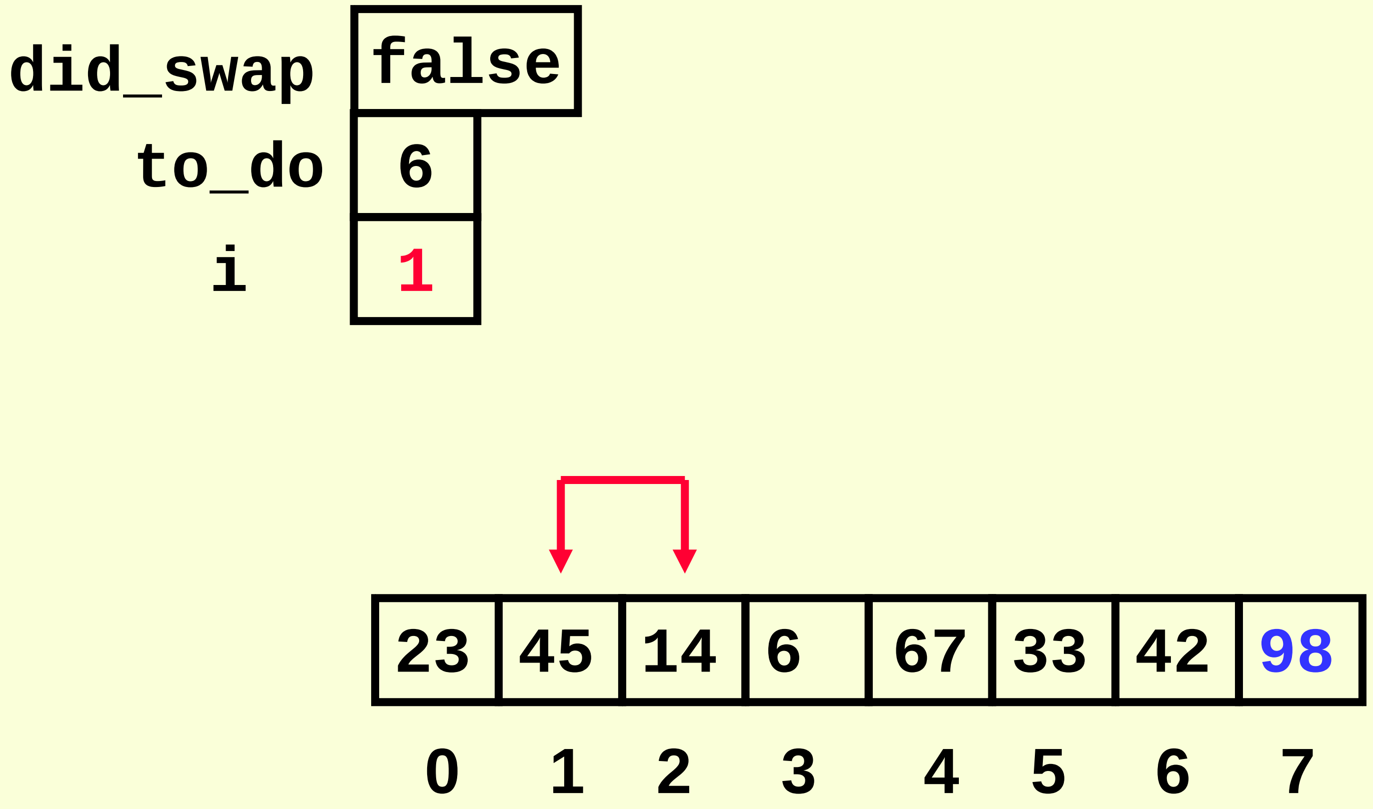
The Second “Bubble Up”



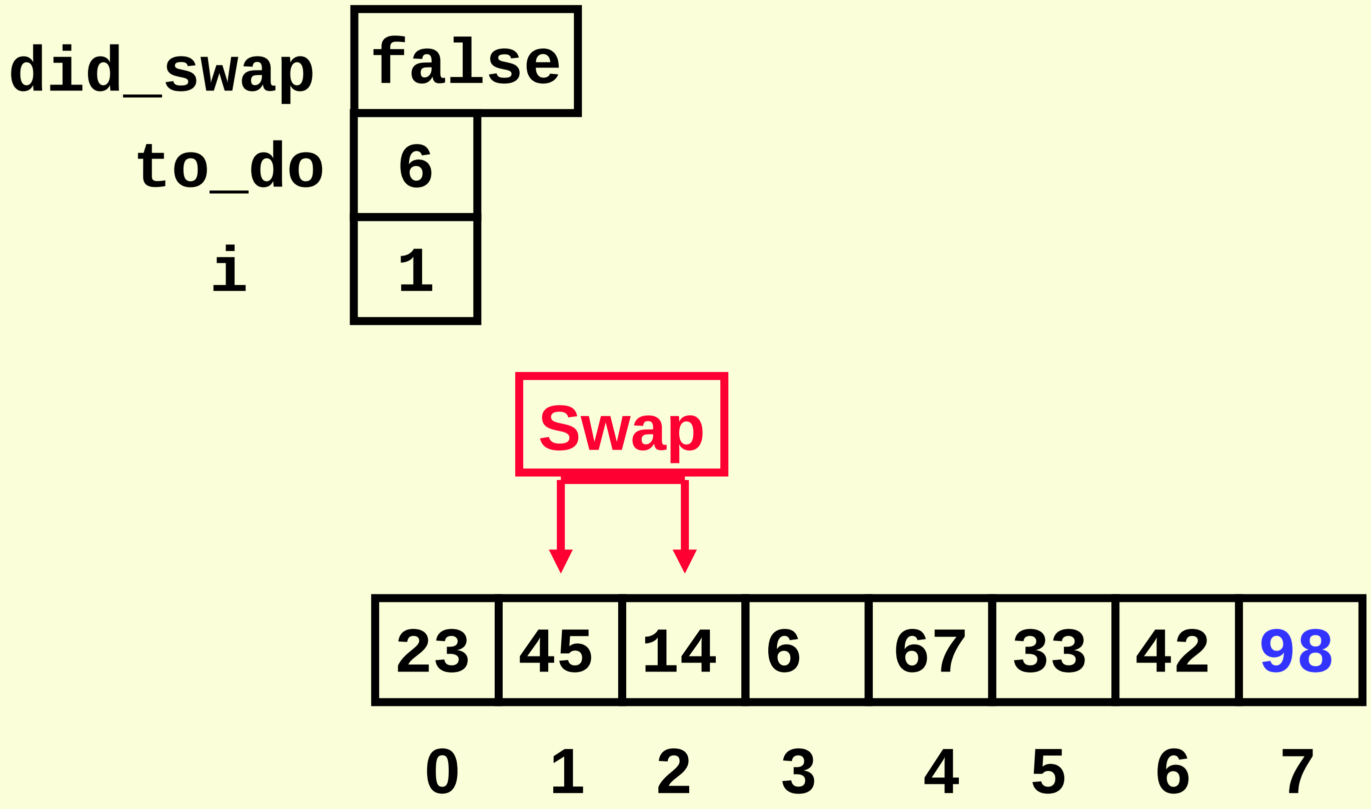
The Second “Bubble Up”



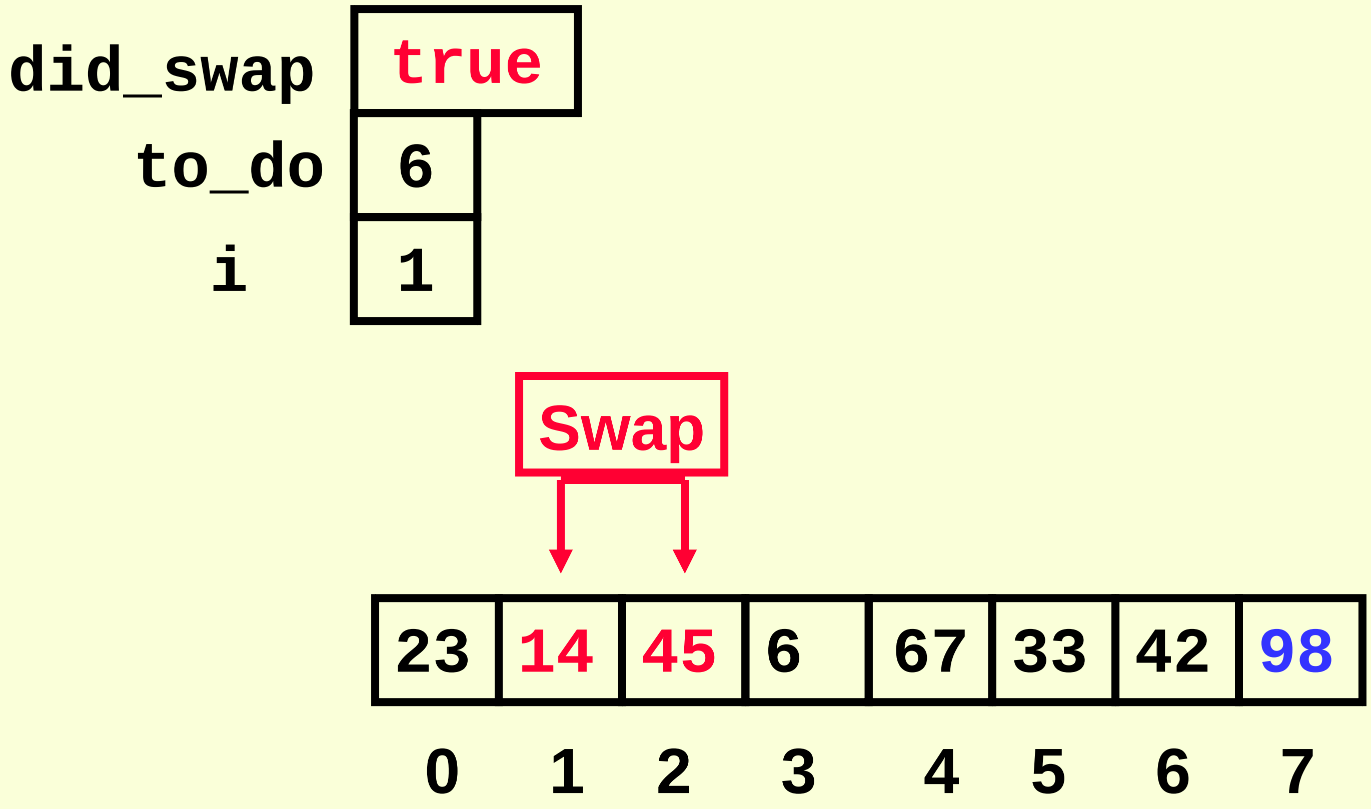
The Second “Bubble Up”



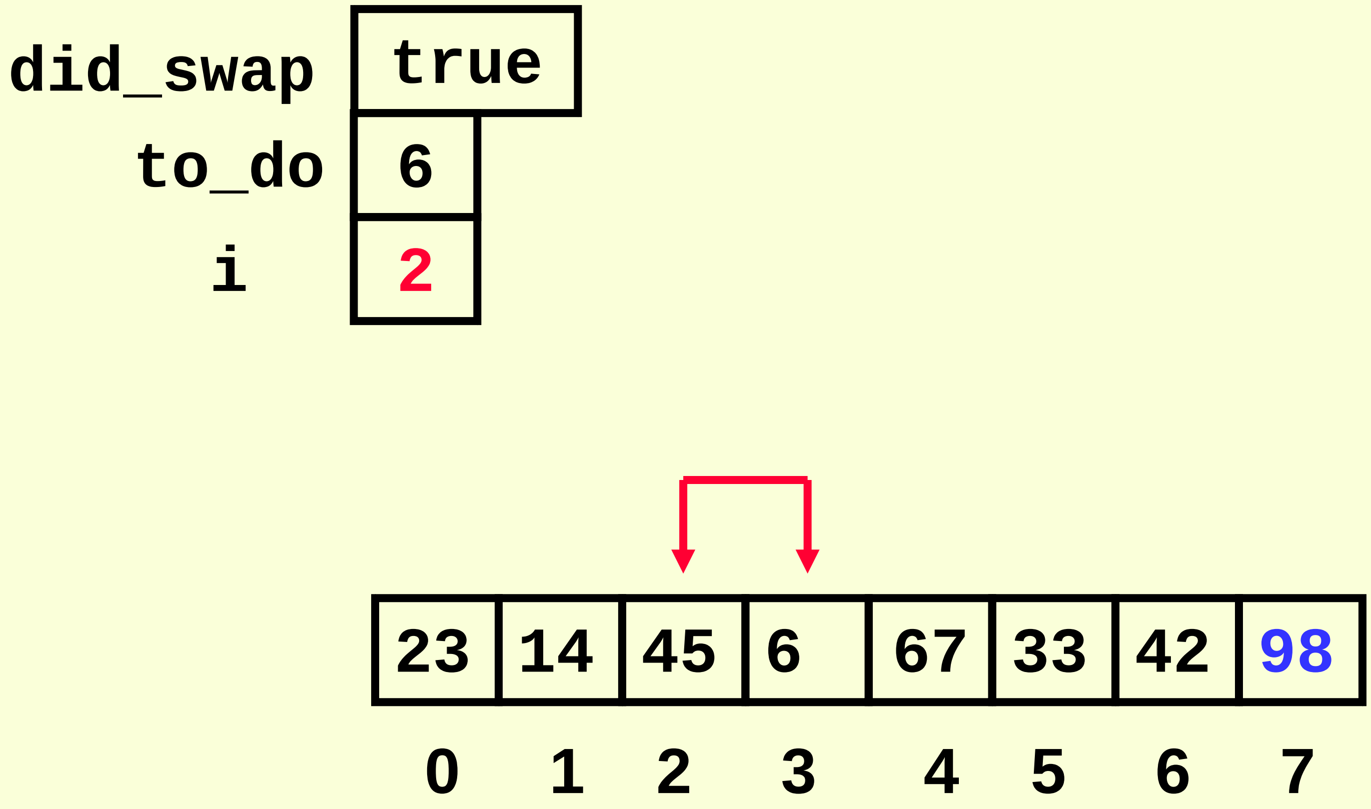
The Second "Bubble Up"



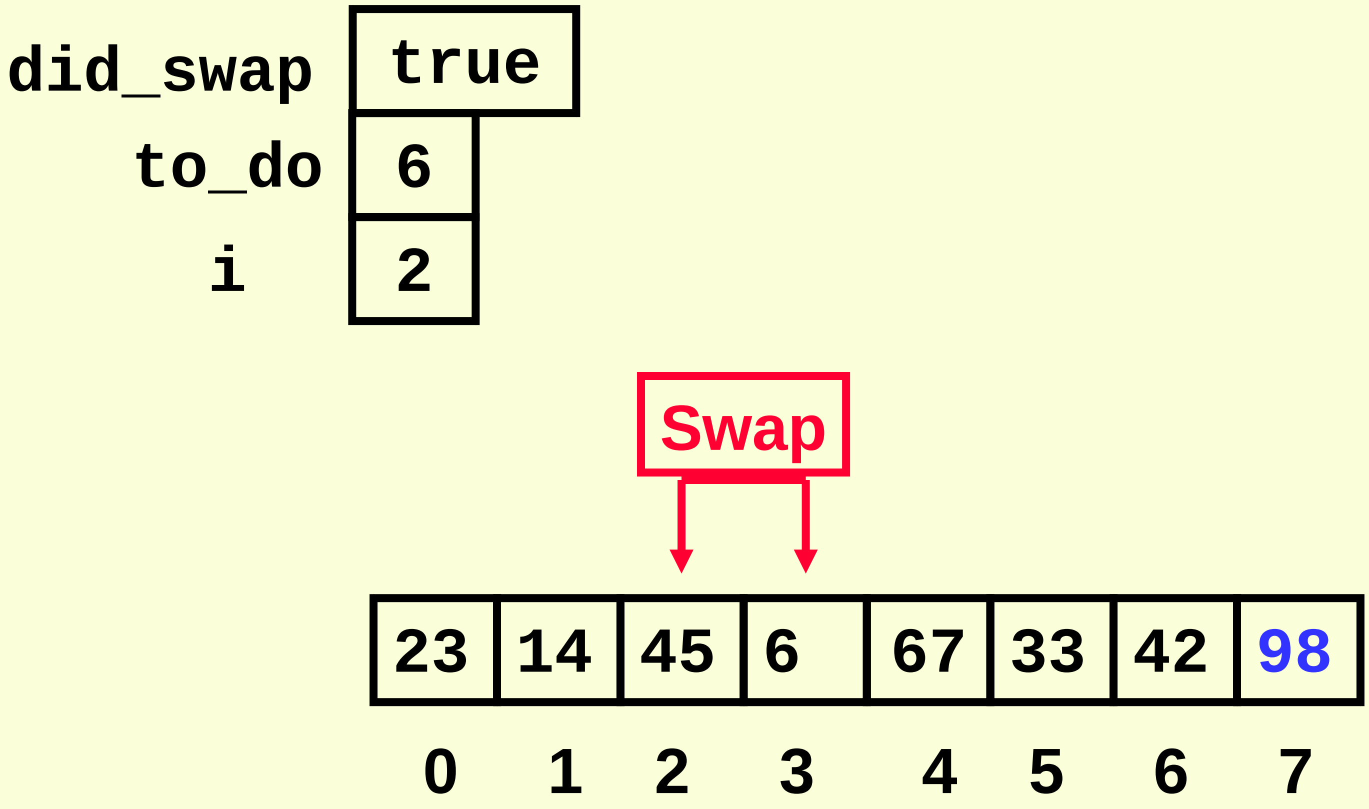
The Second “Bubble Up”



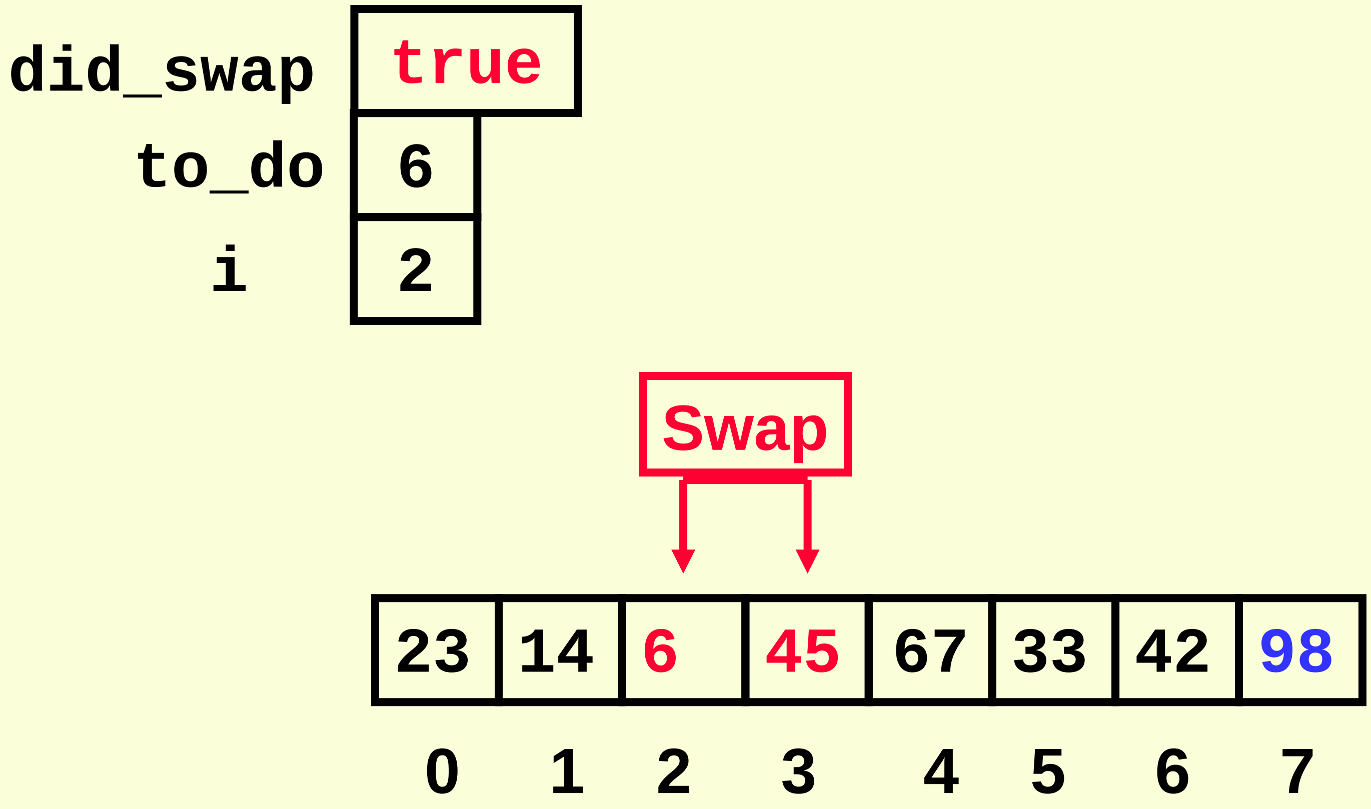
The Second “Bubble Up”



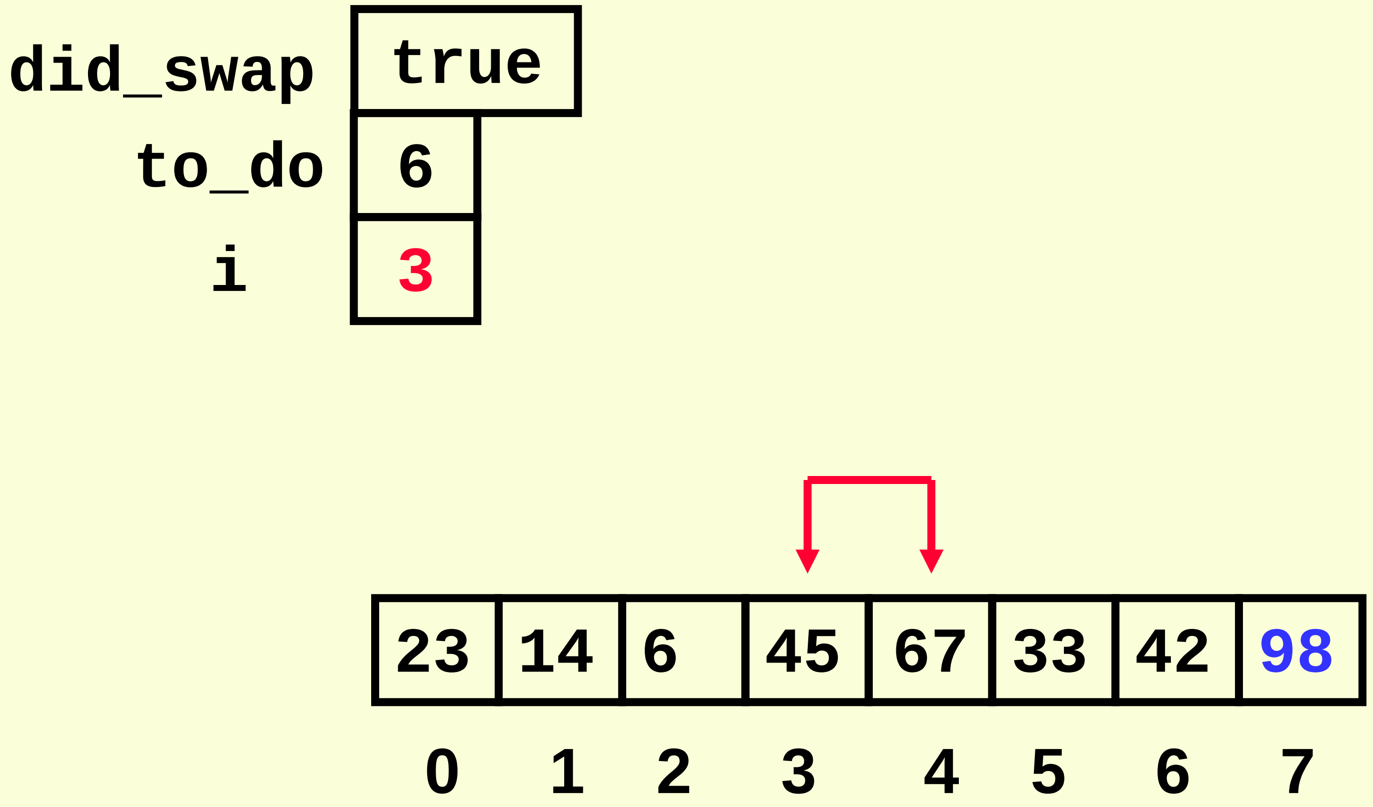
The Second "Bubble Up"



The Second “Bubble Up”

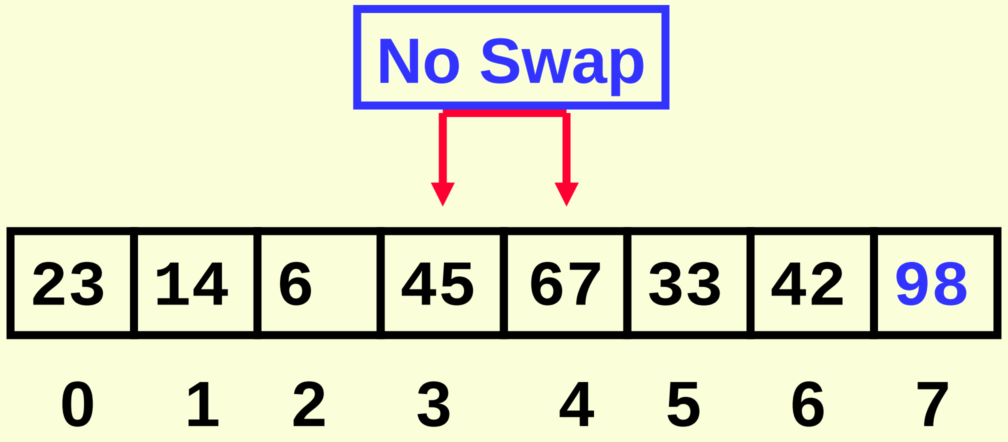


The Second “Bubble Up”

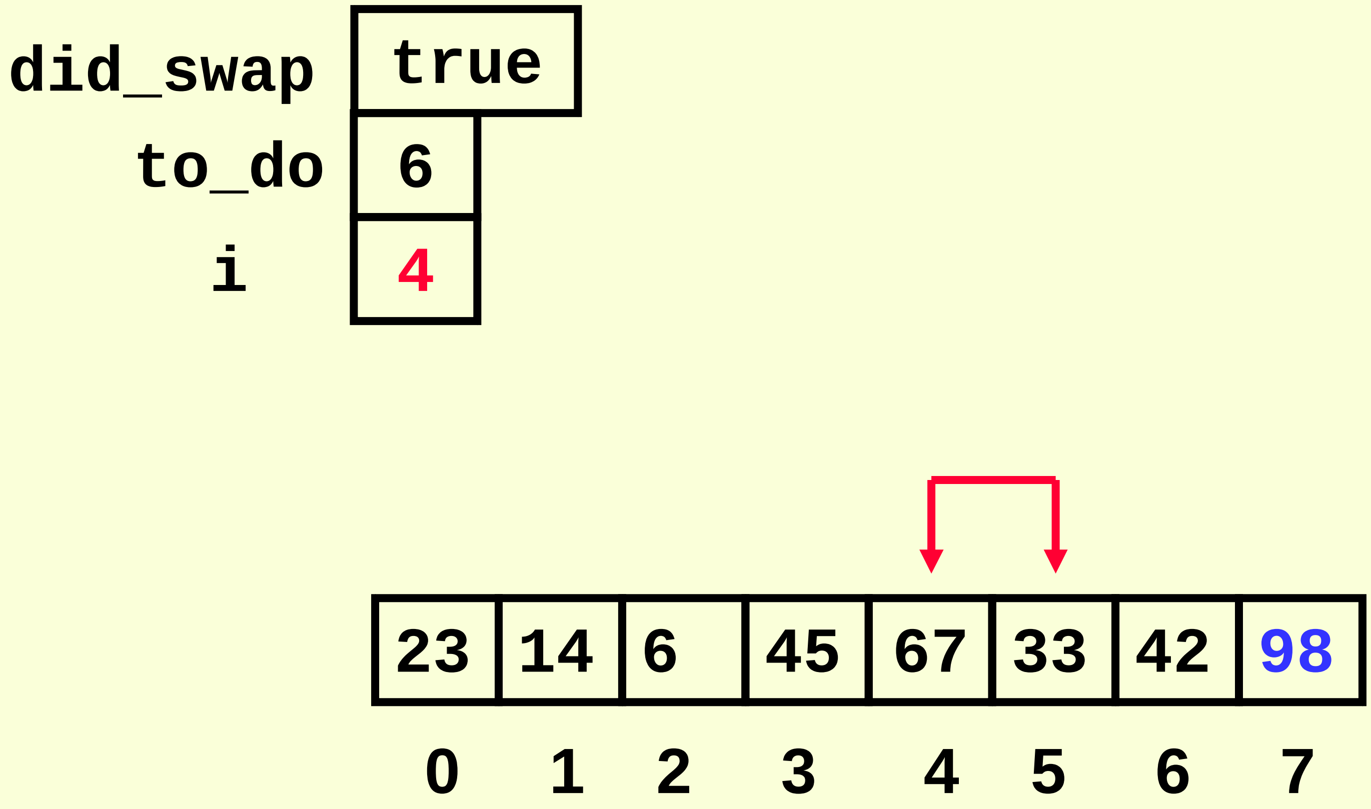


The Second “Bubble Up”

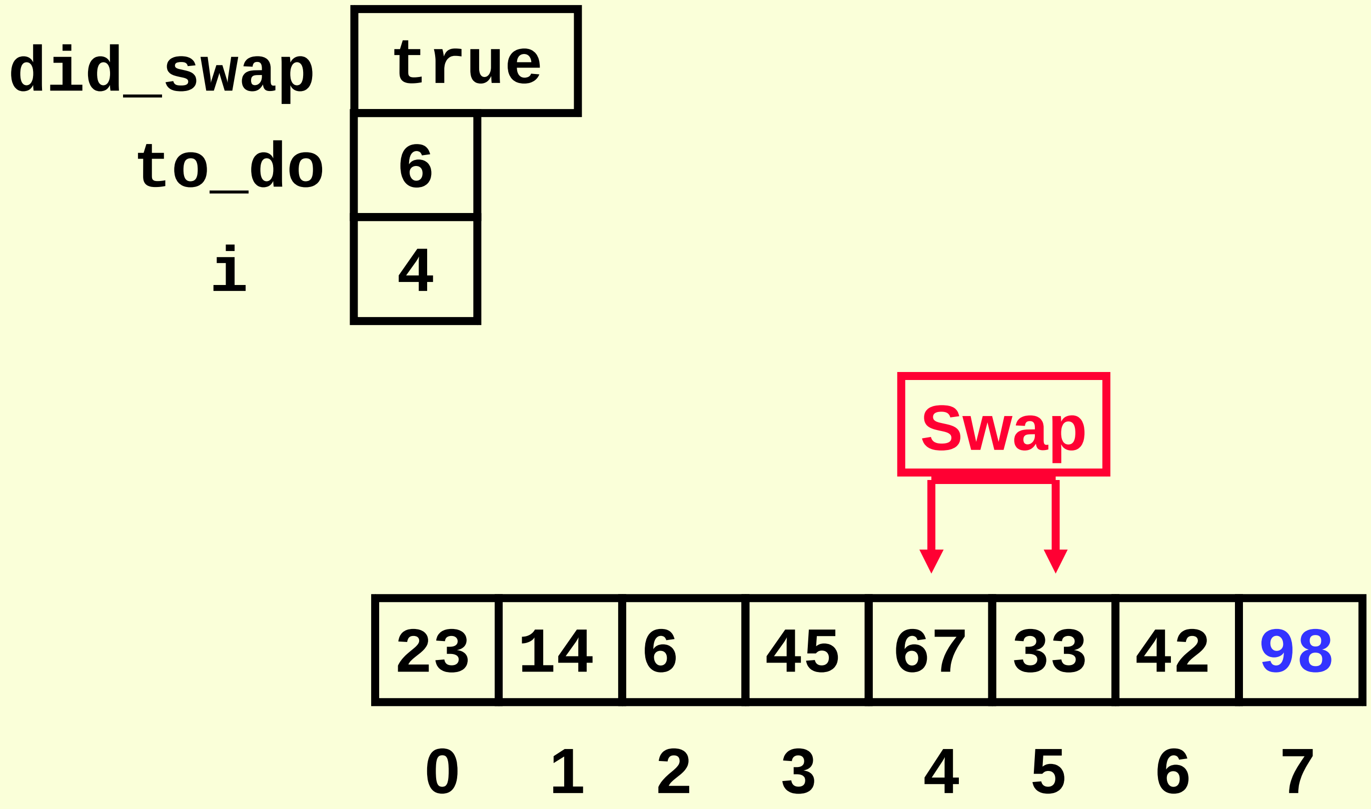
`did_swap`	`true`
`to_do`	`6`
`i`	`3`



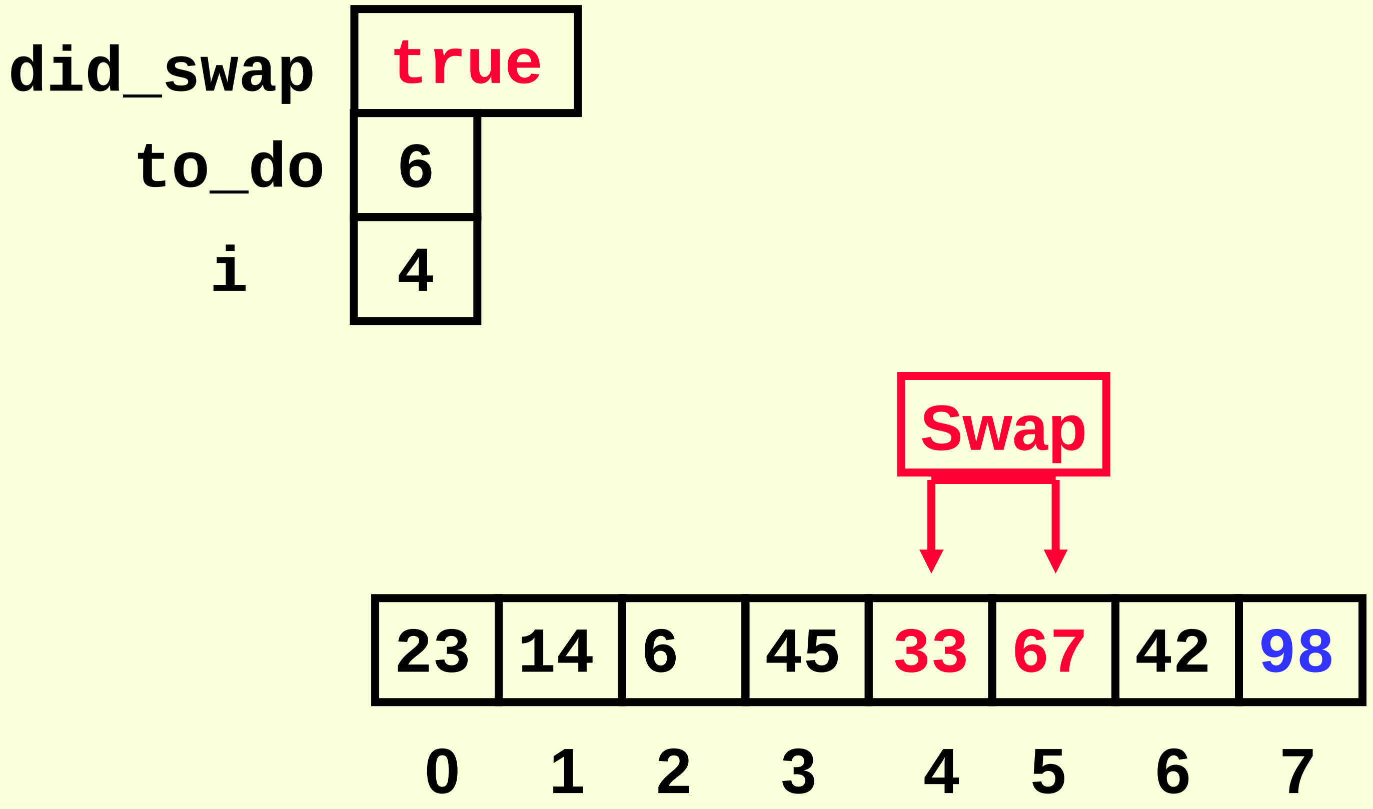
The Second "Bubble Up"



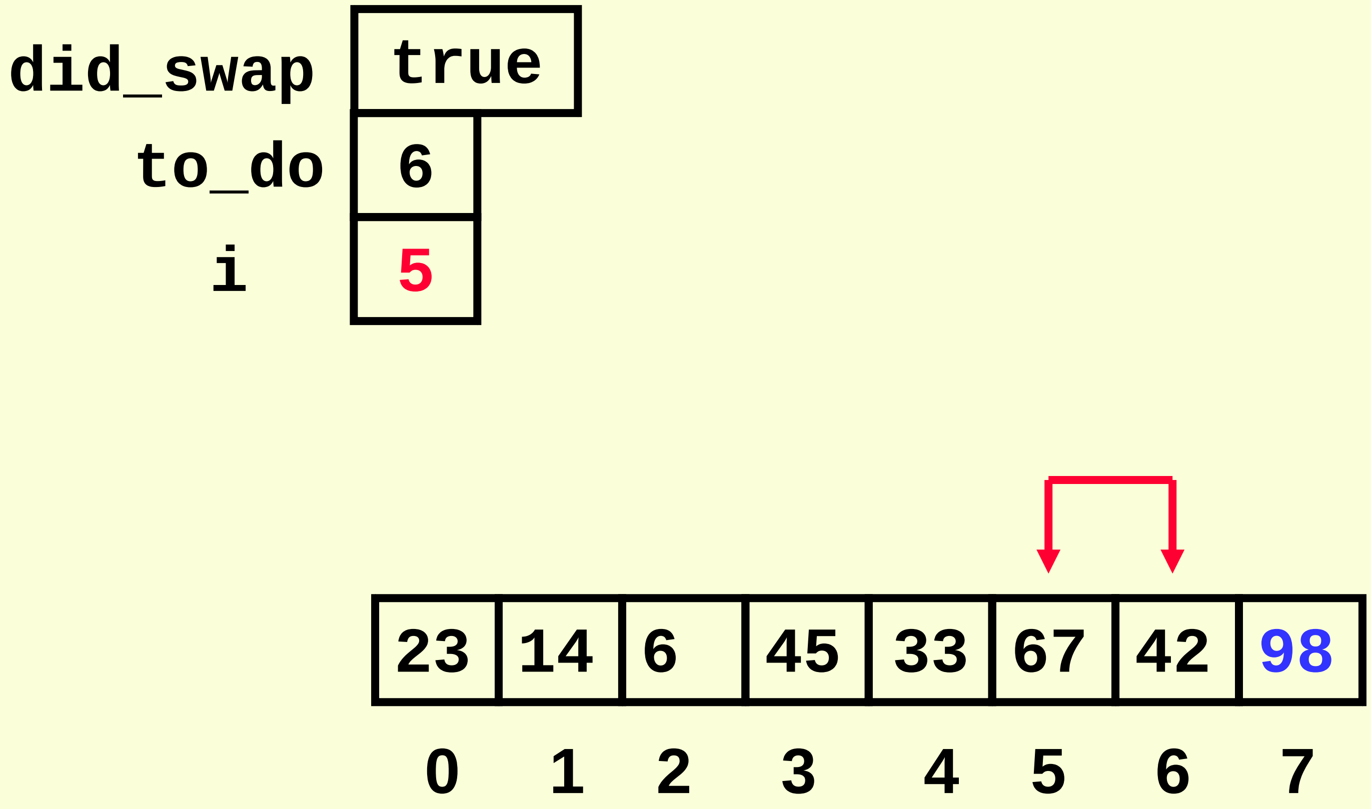
The Second “Bubble Up”



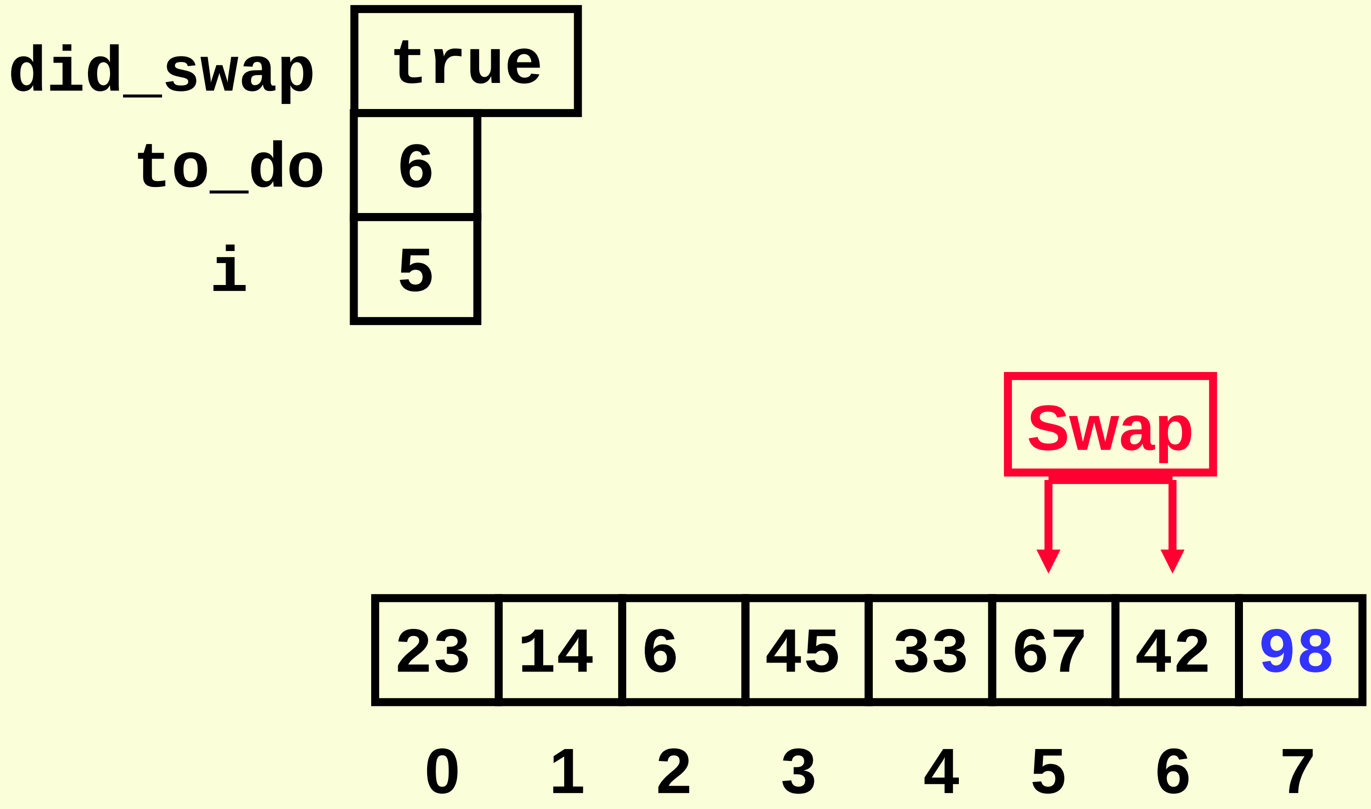
The Second "Bubble Up"



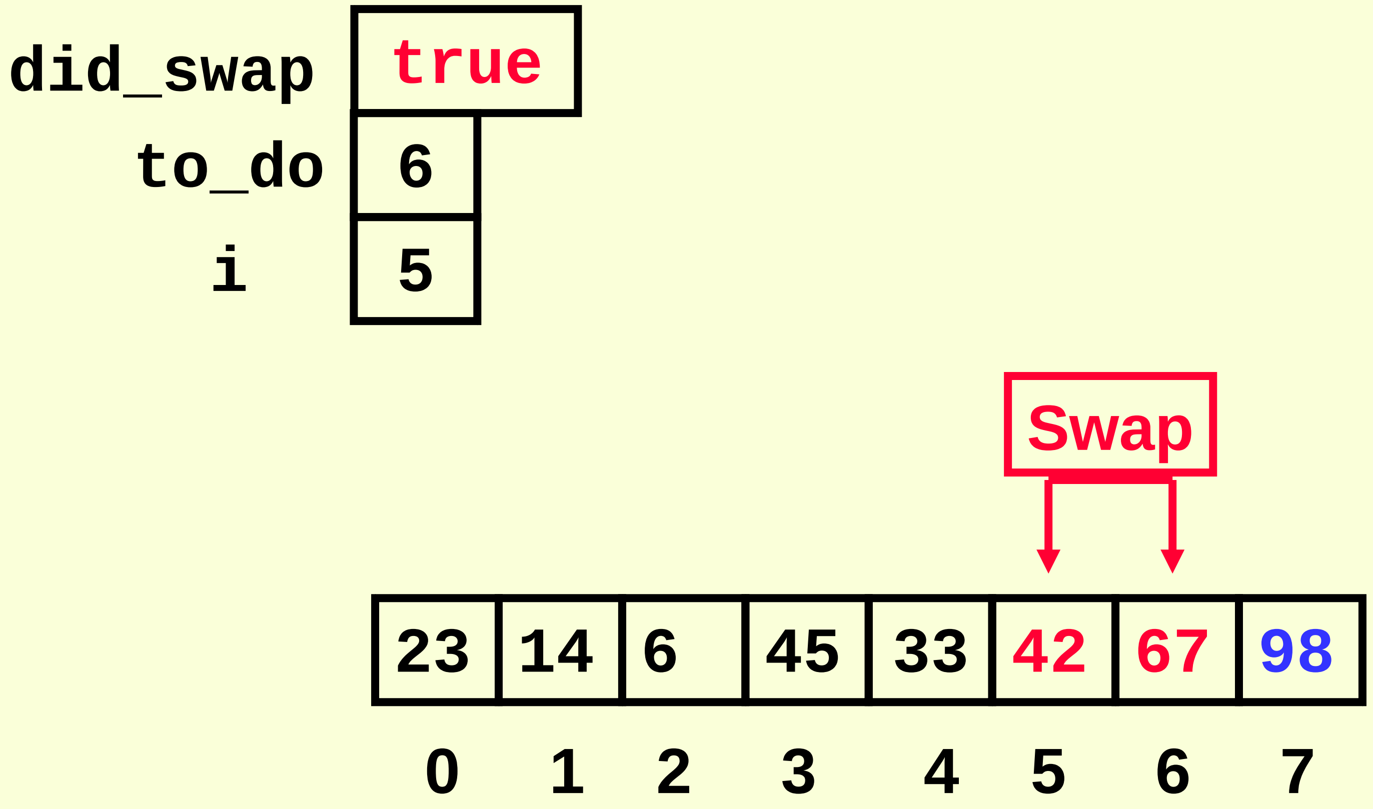
The Second “Bubble Up”



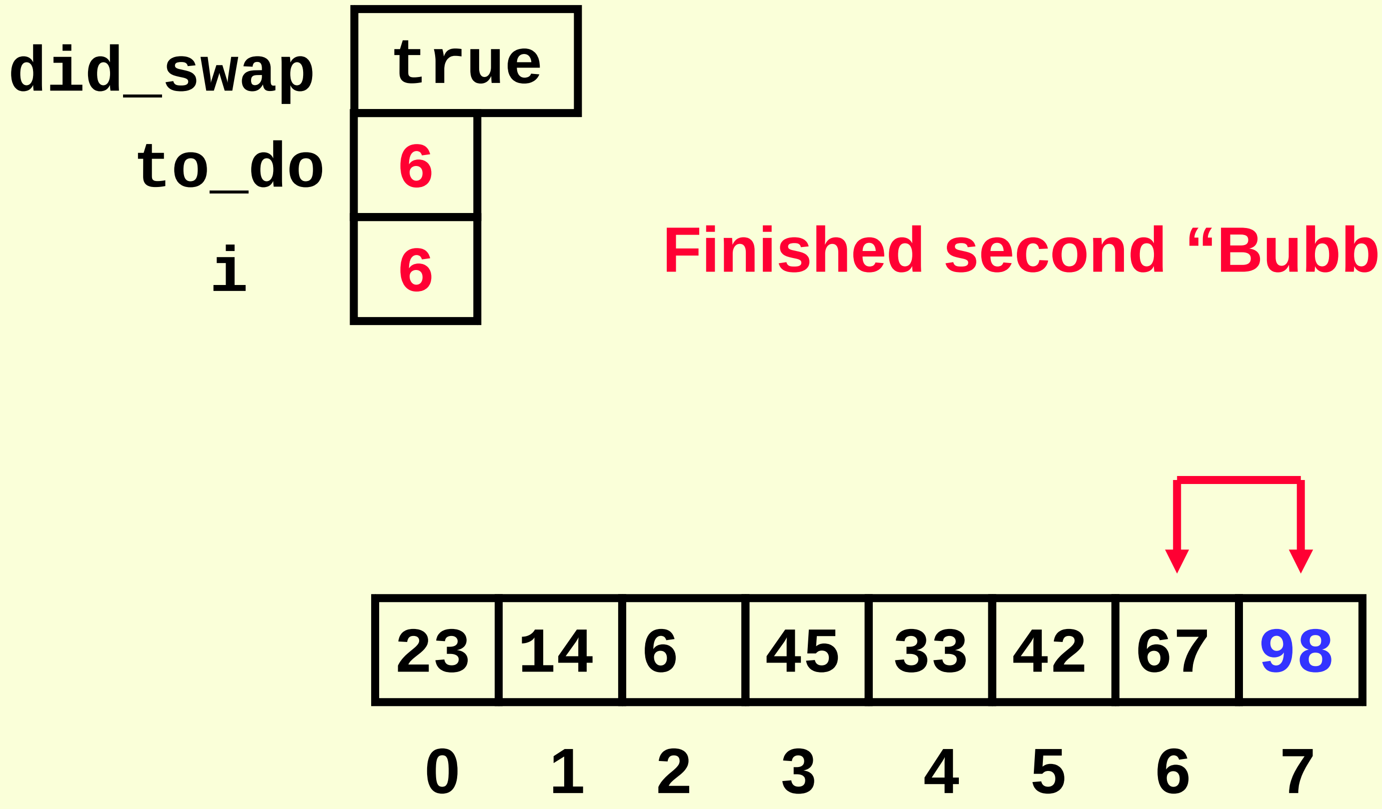
The Second "Bubble Up"



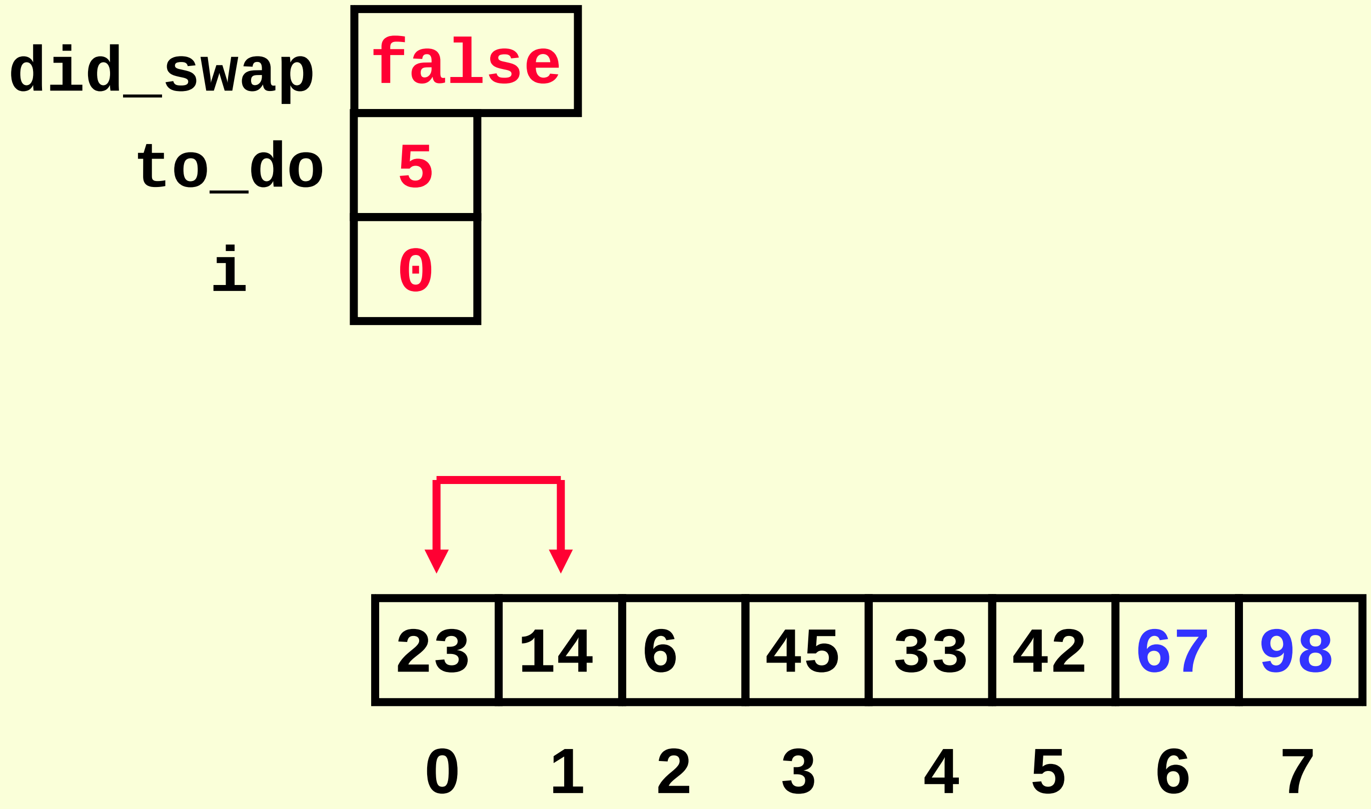
The Second "Bubble Up"



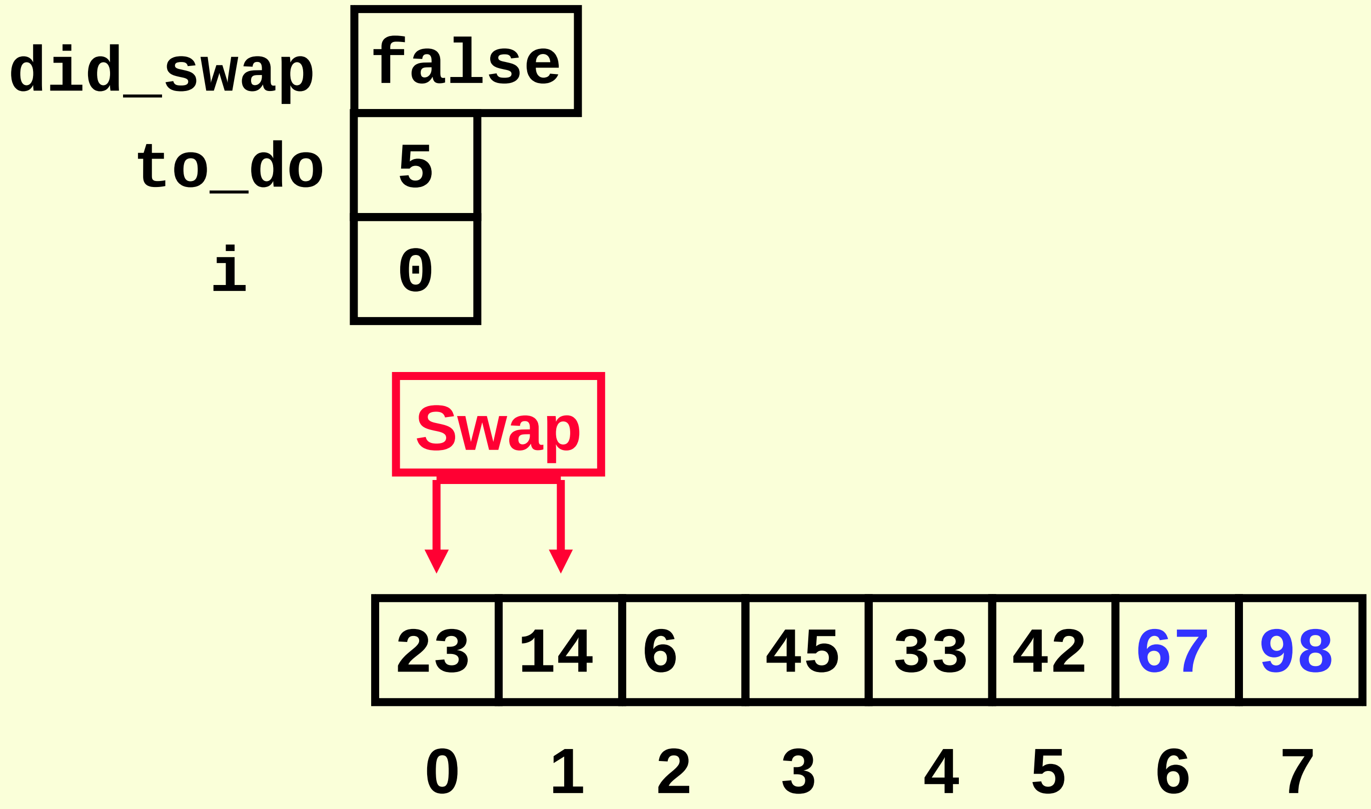
After Second Pass of Outer Loop



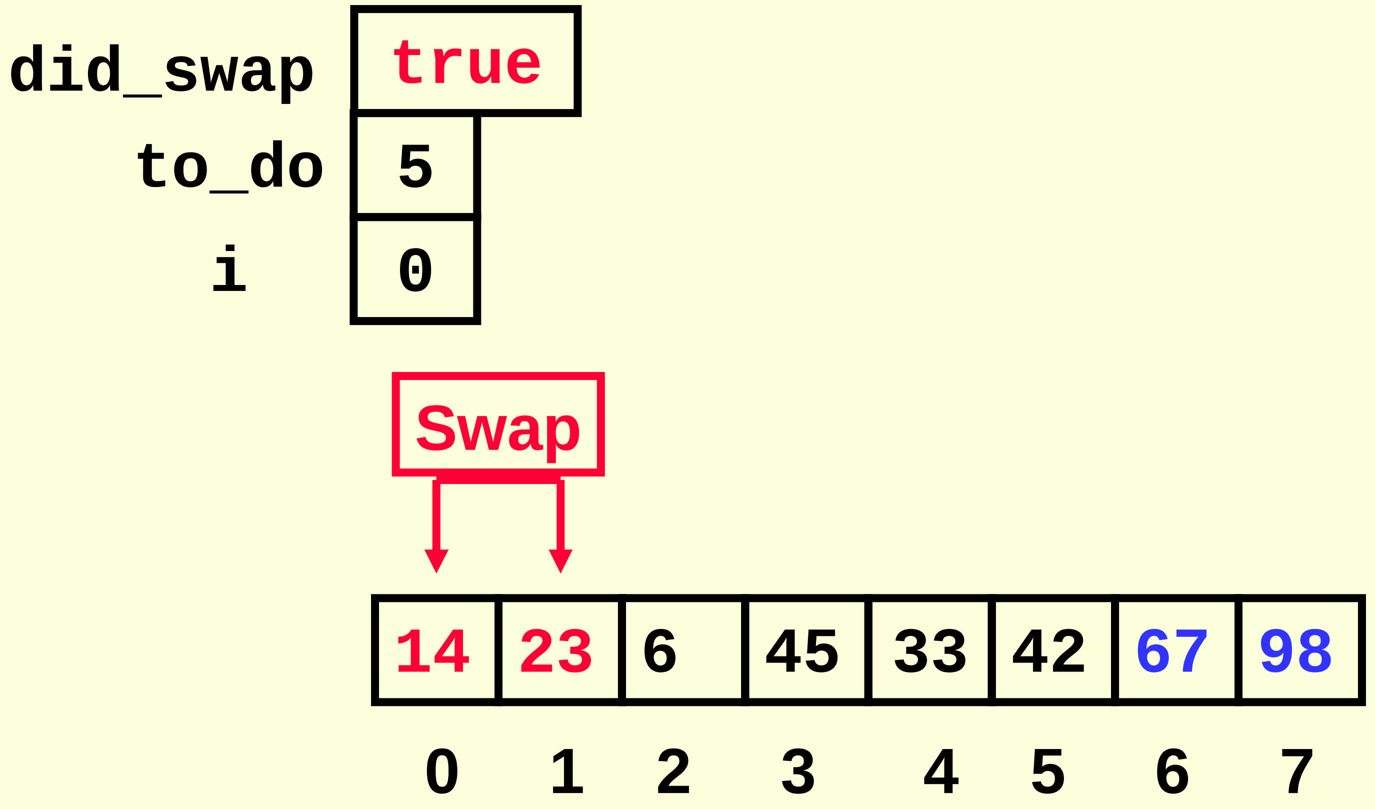
The Third “Bubble Up”



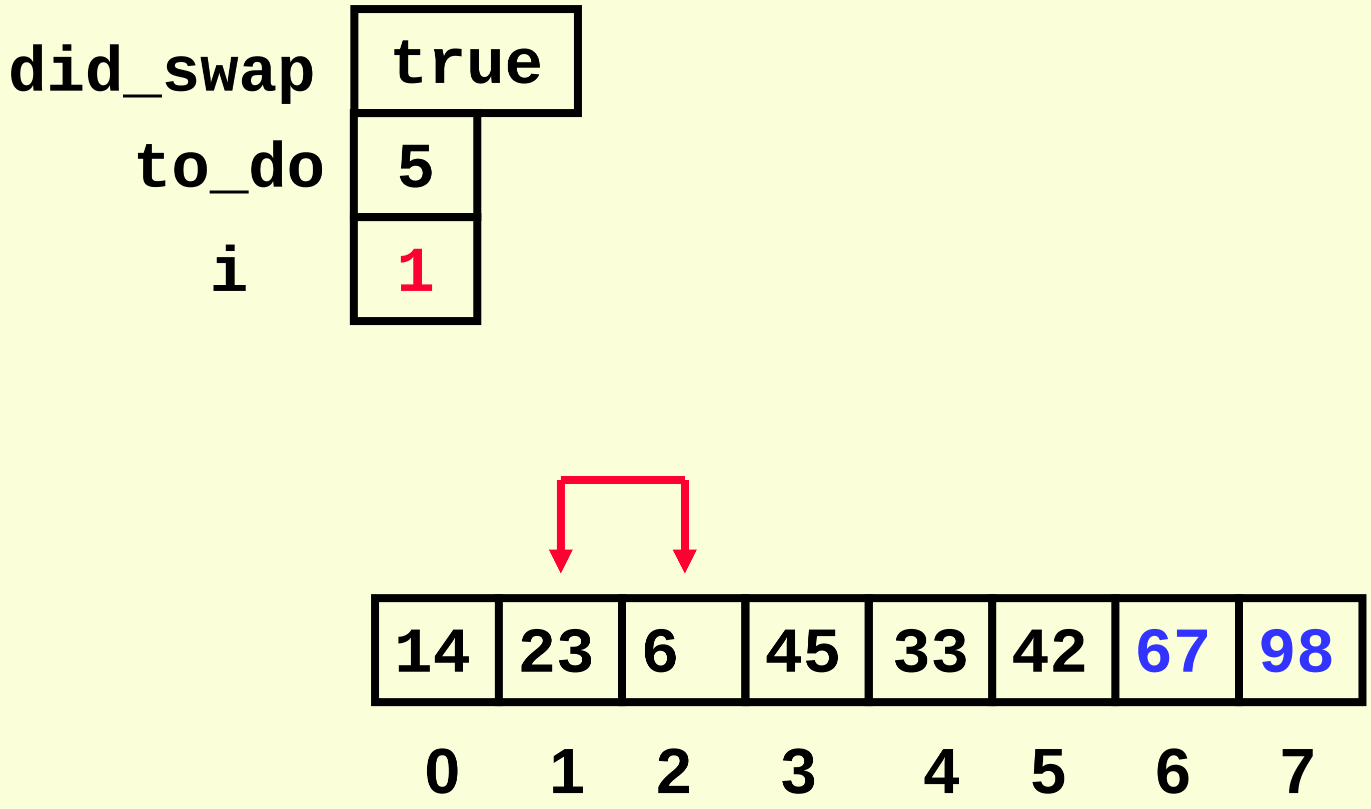
The Third “Bubble Up”



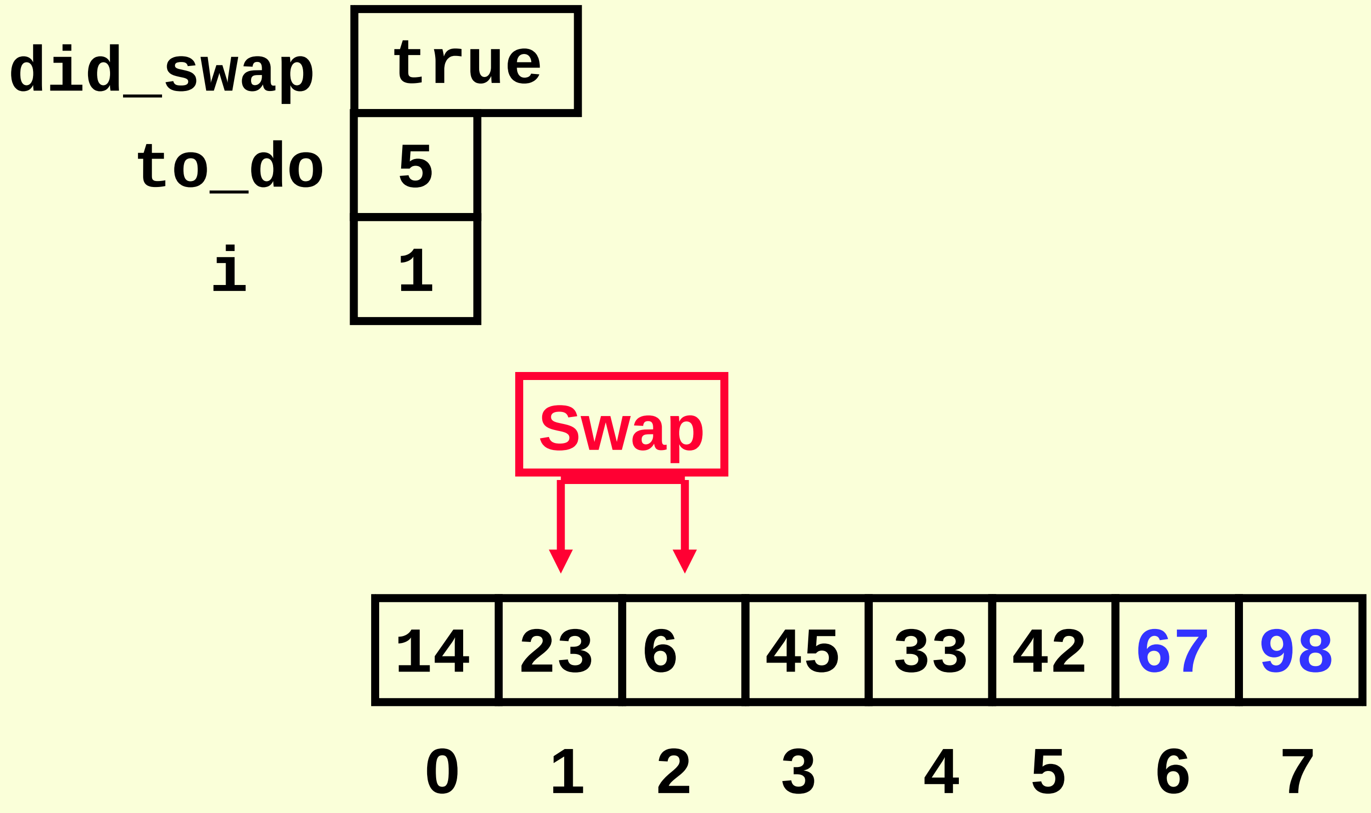
The Third “Bubble Up”



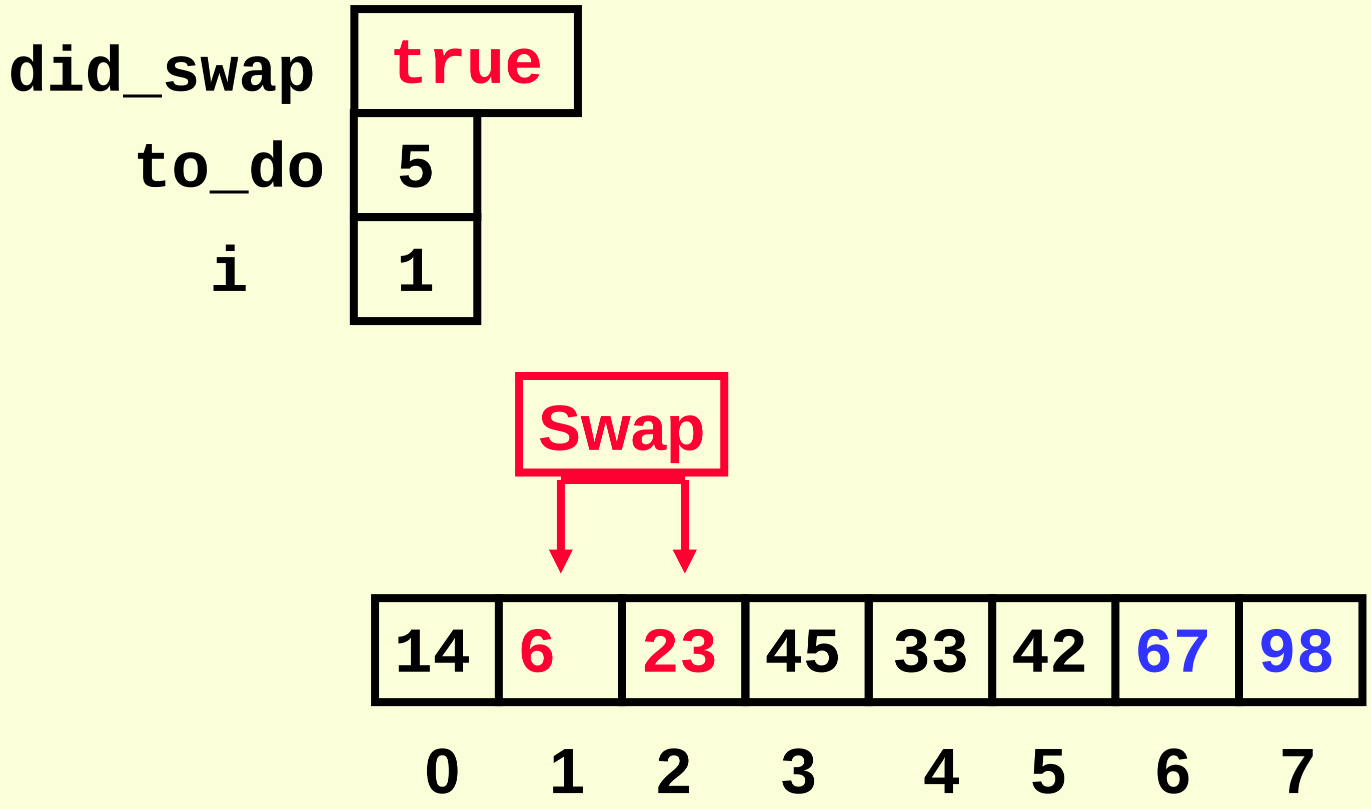
The Third "Bubble Up"



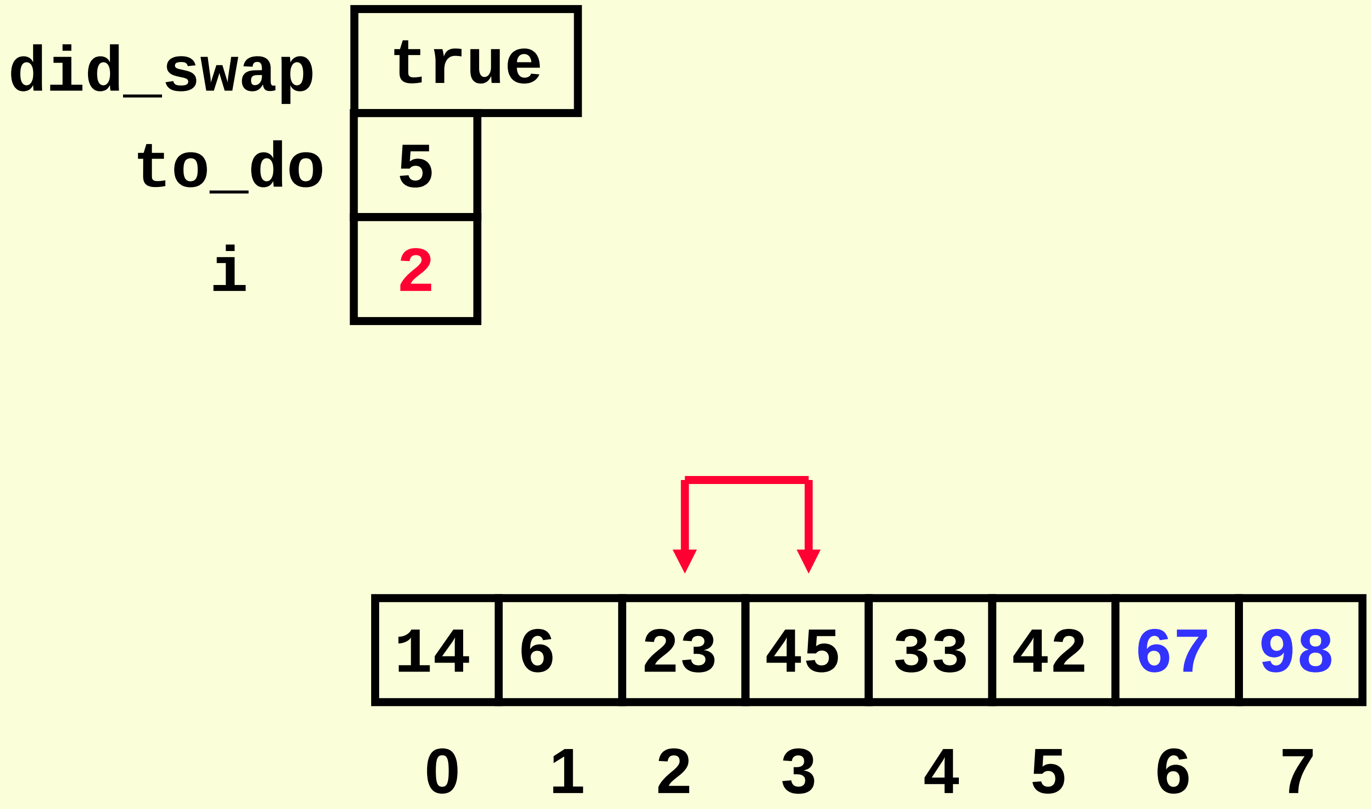
The Third “Bubble Up”



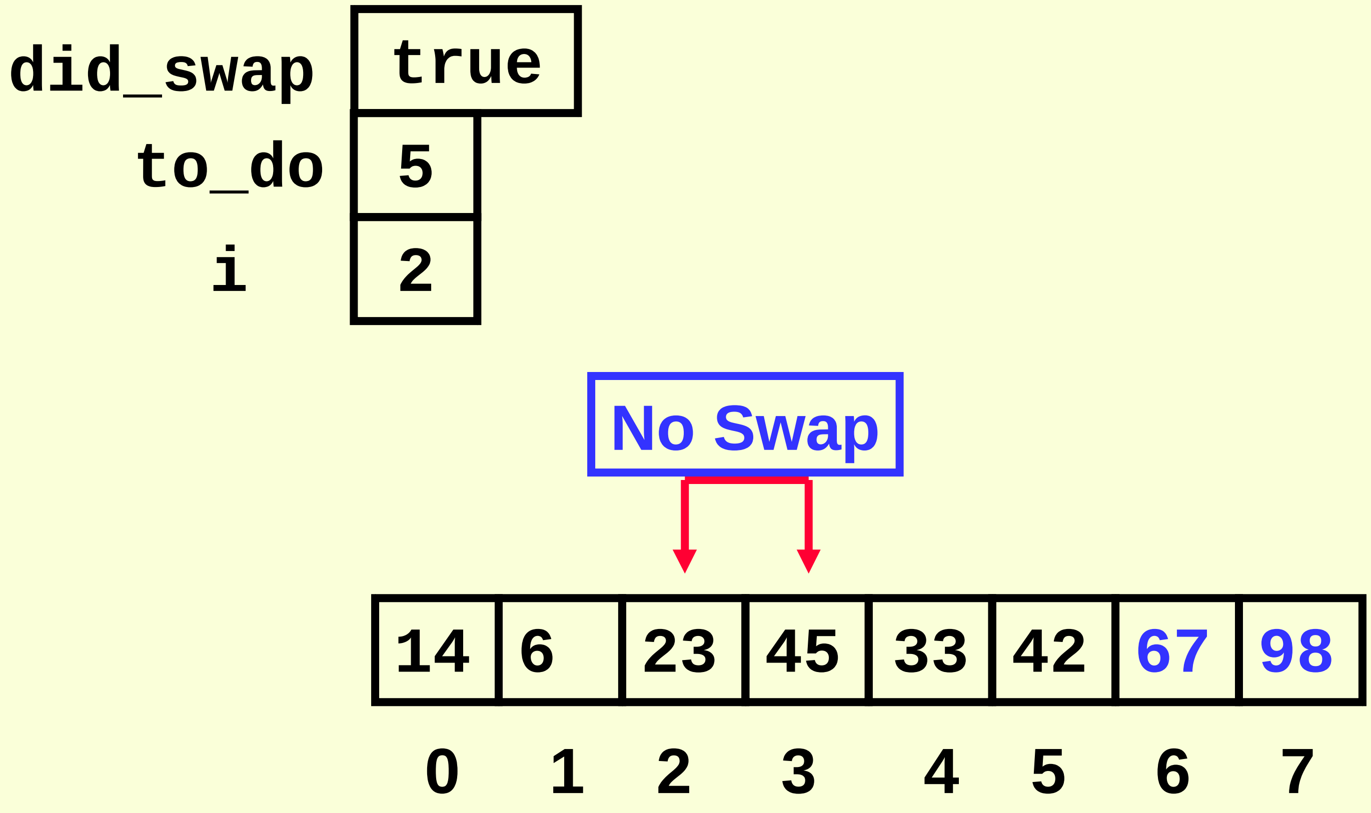
The Third “Bubble Up”



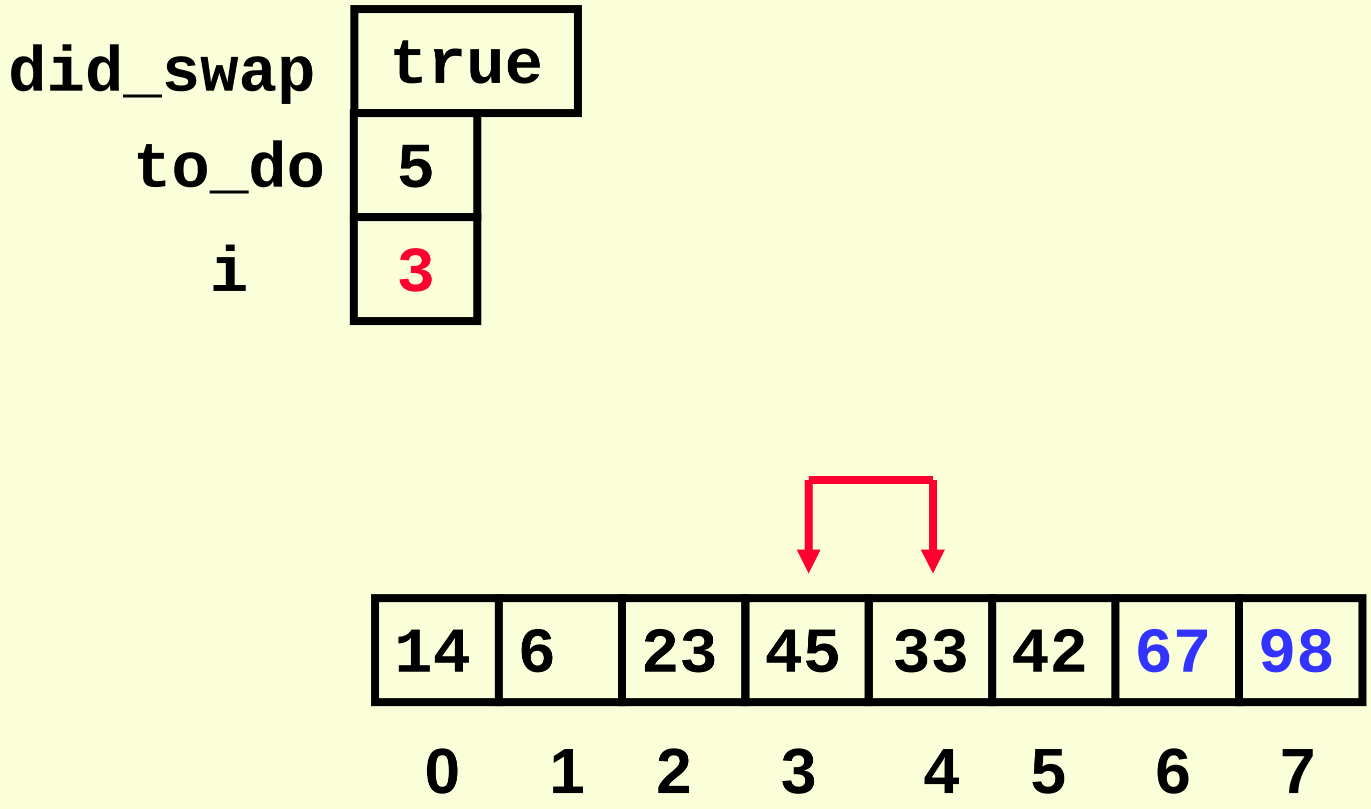
The Third "Bubble Up"



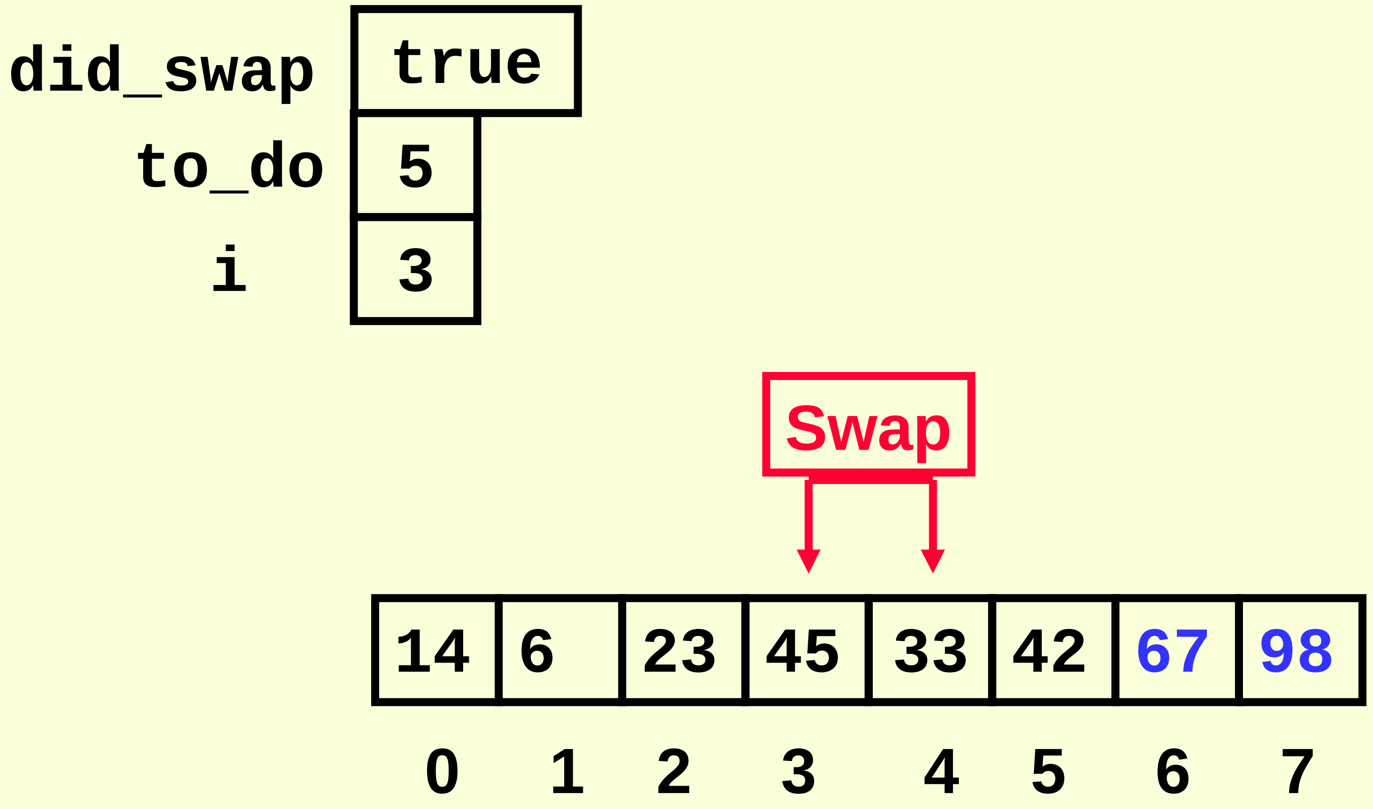
The Third “Bubble Up”



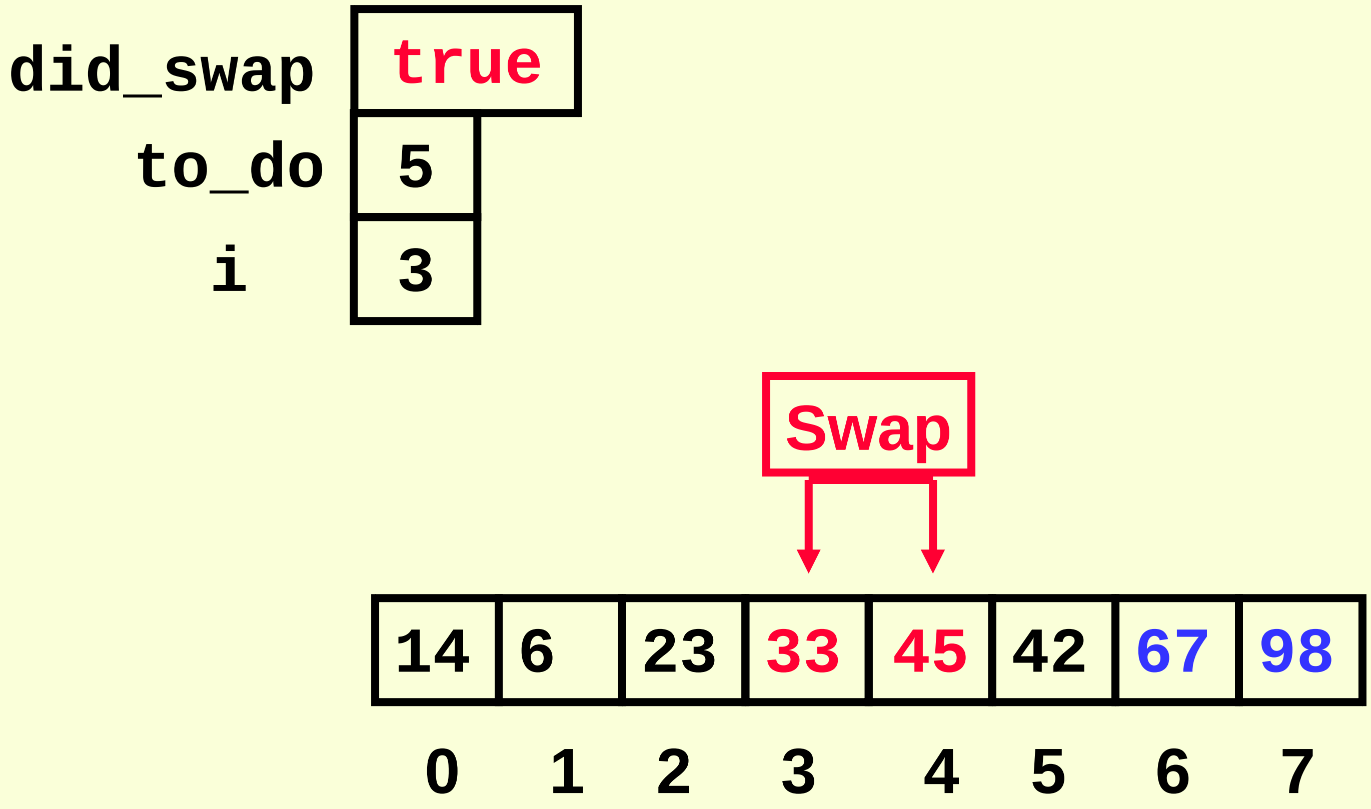
The Third "Bubble Up"



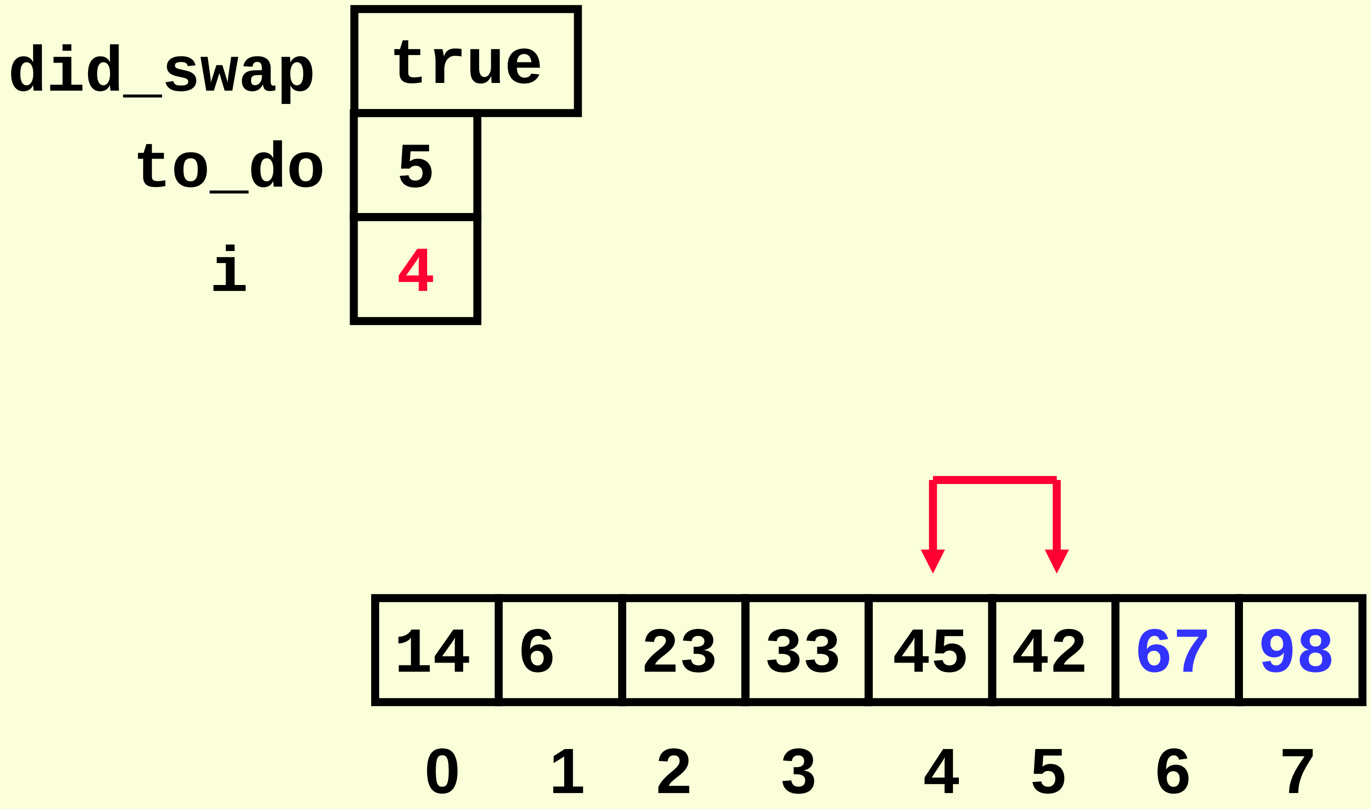
The Third "Bubble Up"



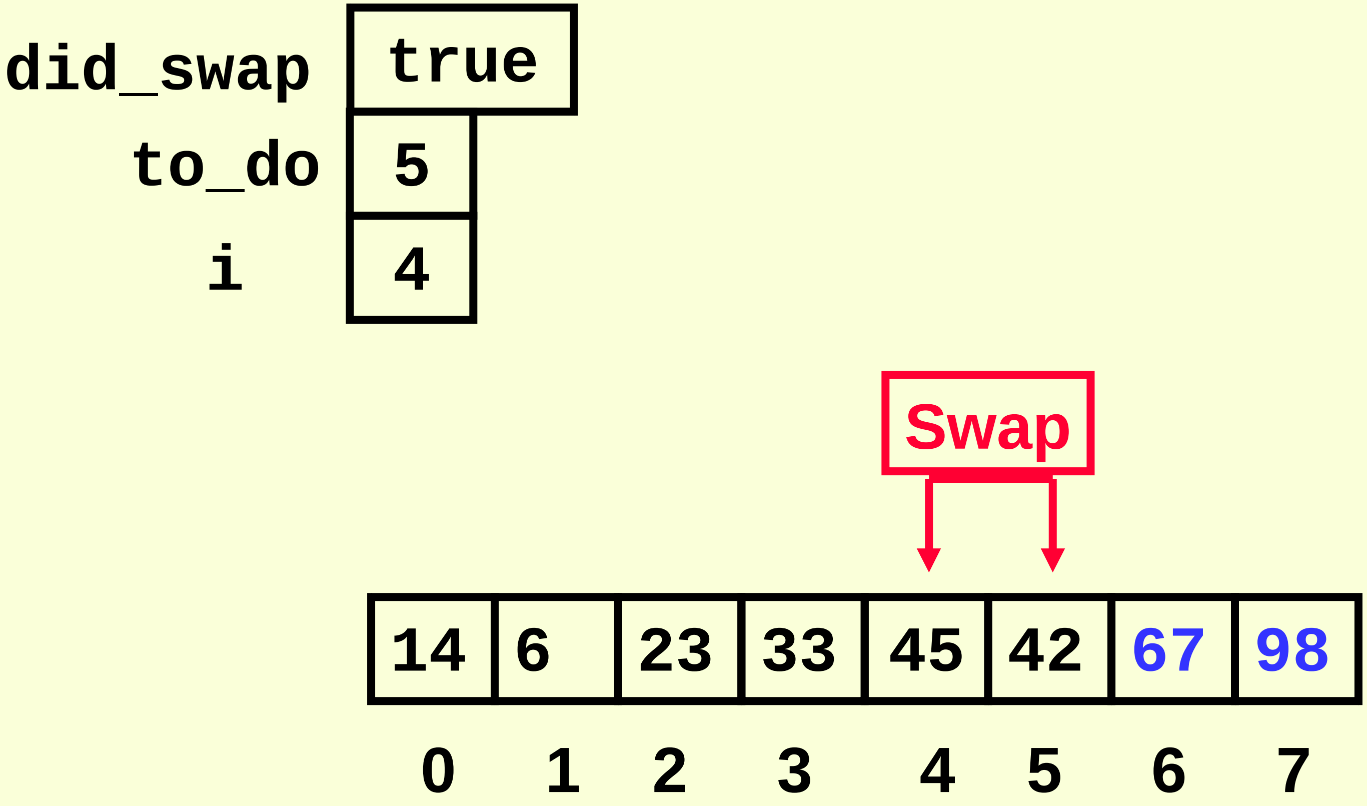
The Third “Bubble Up”



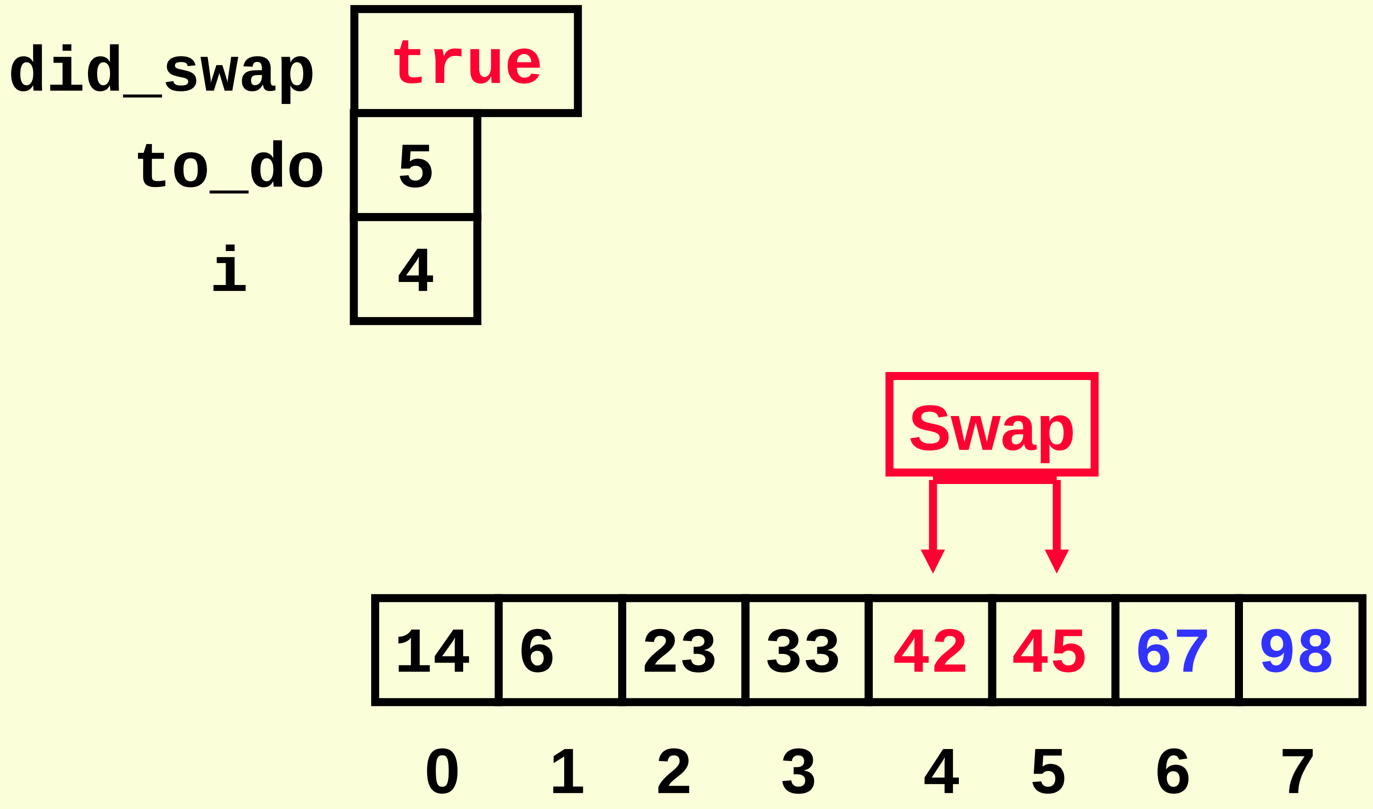
The Third "Bubble Up"



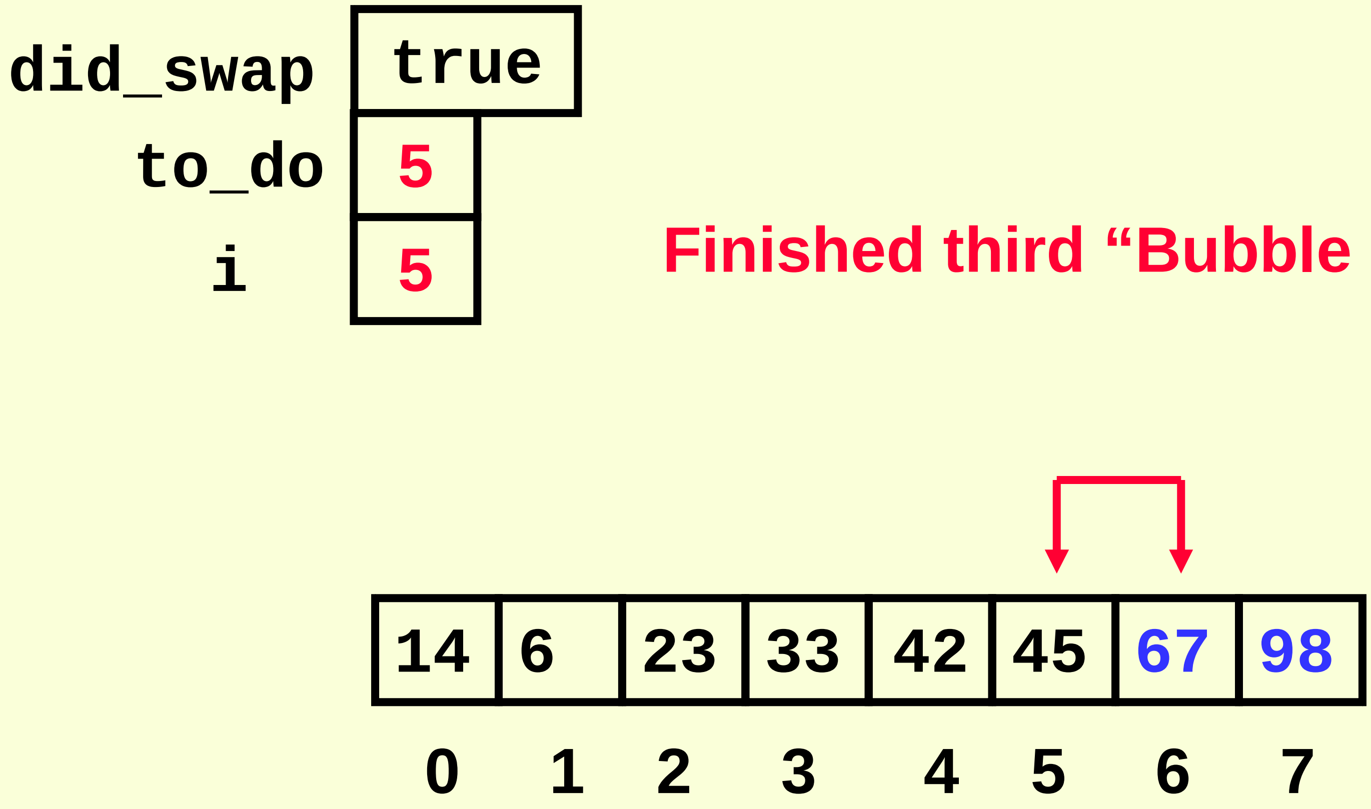
The Third "Bubble Up"



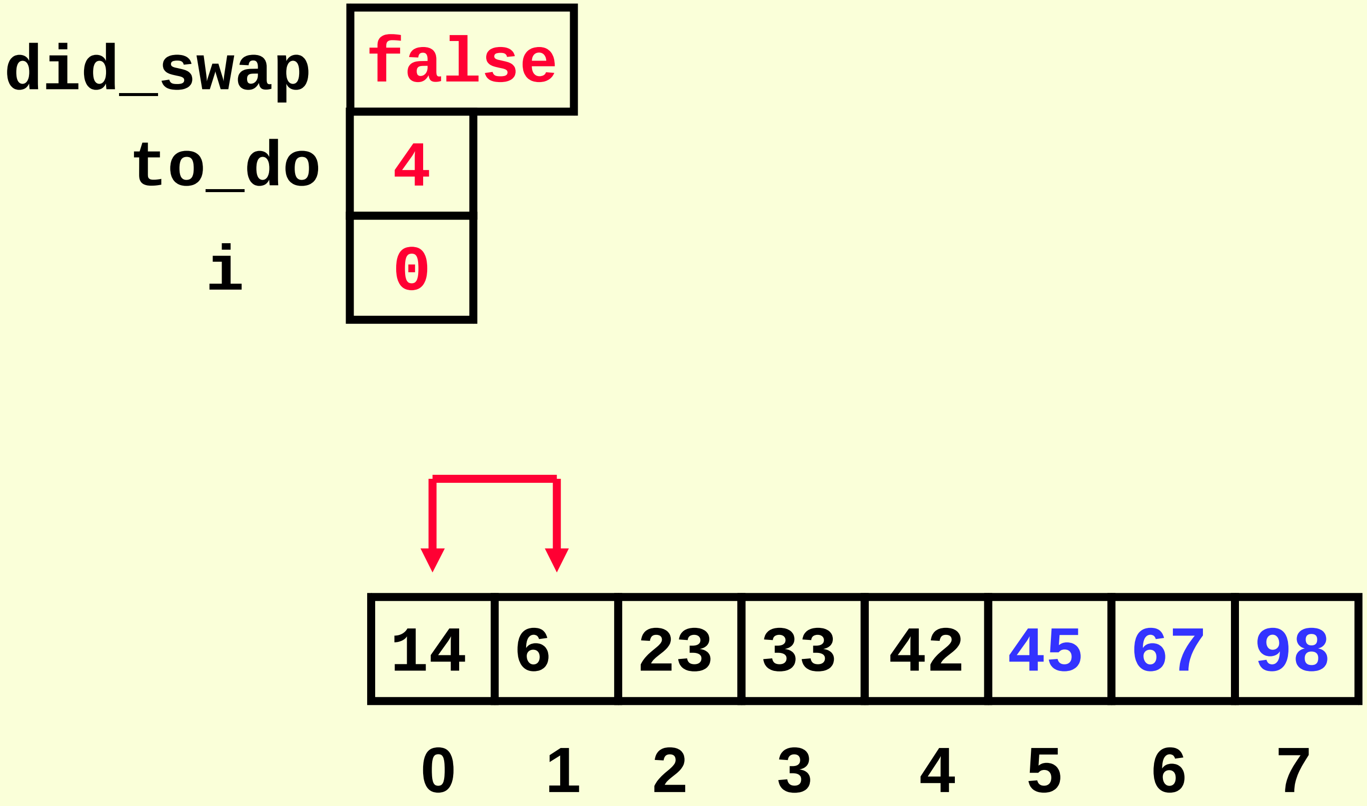
The Third “Bubble Up”



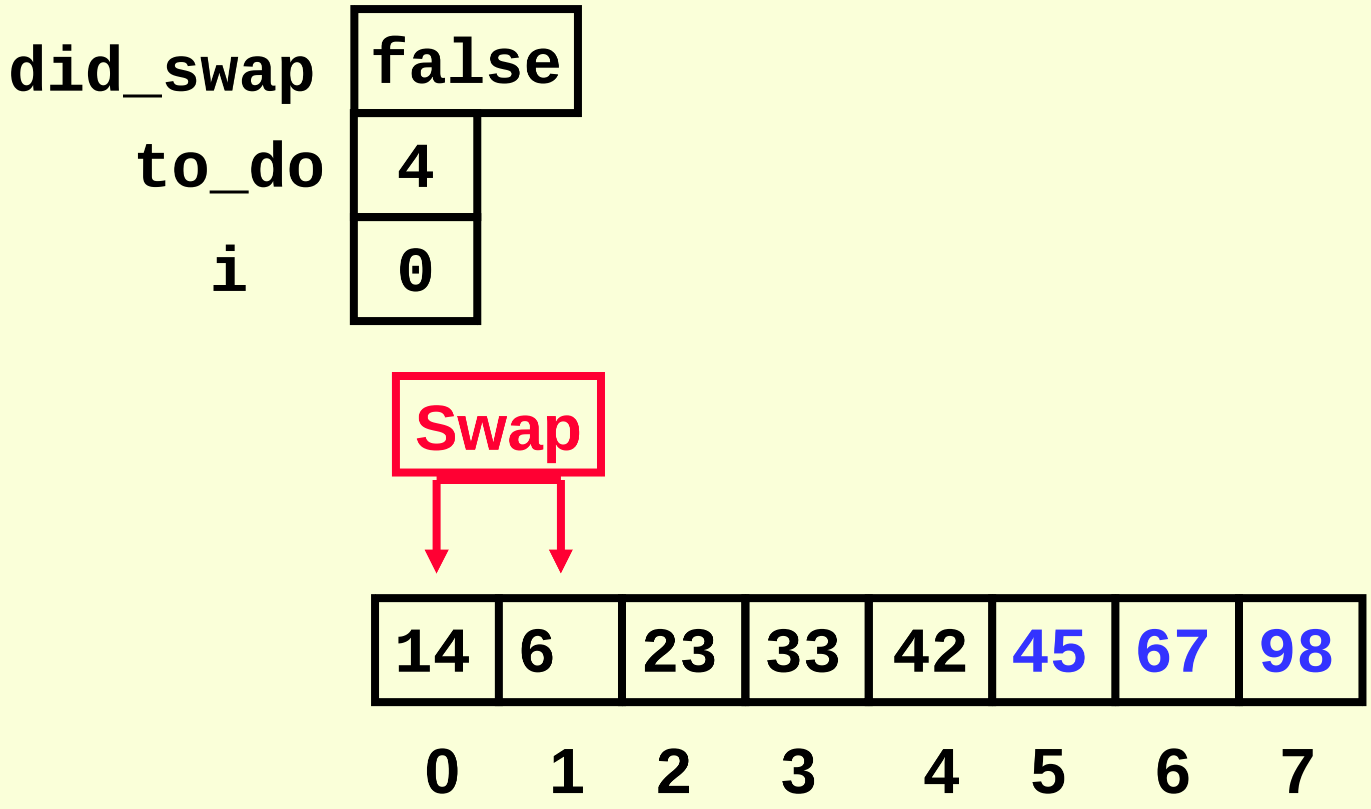
After Third Pass of Outer Loop



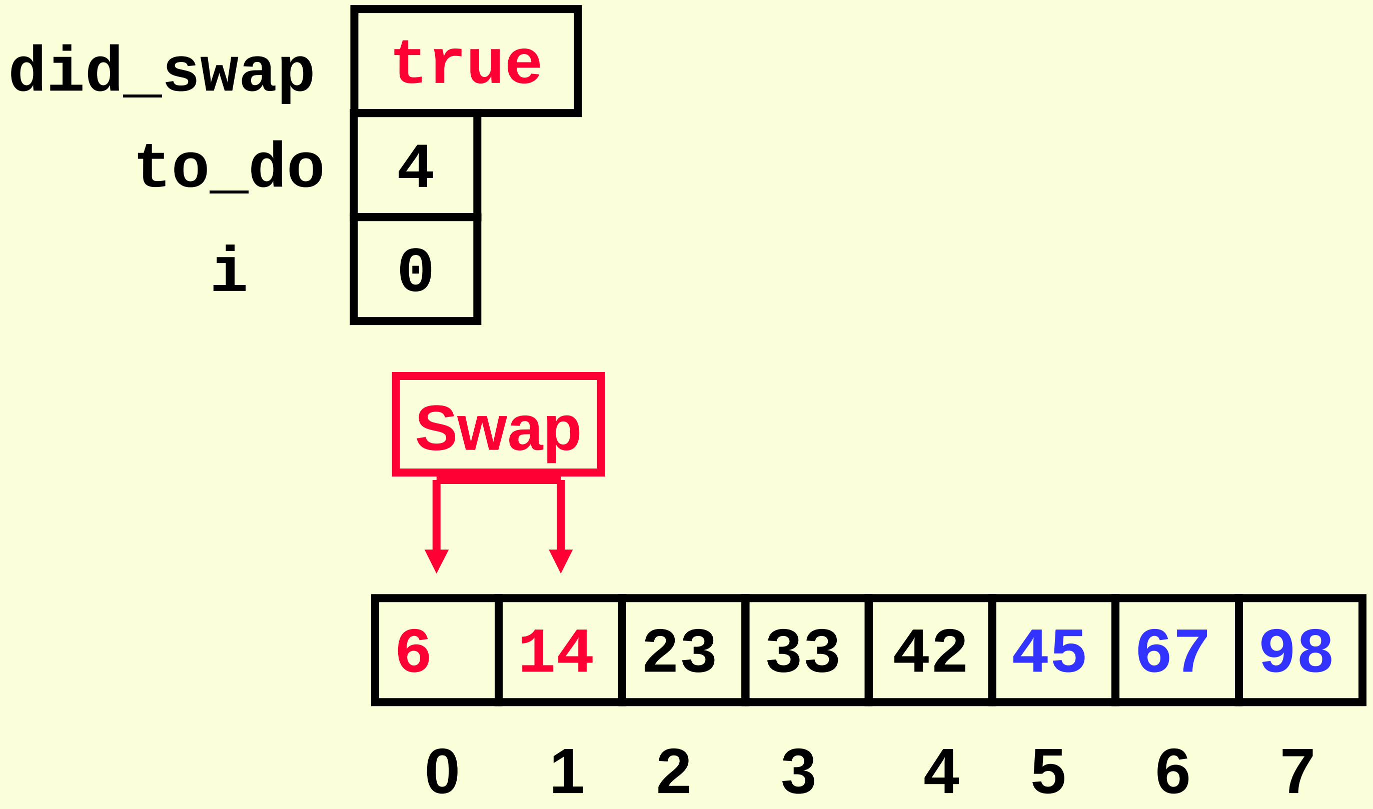
The Fourth "Bubble Up"



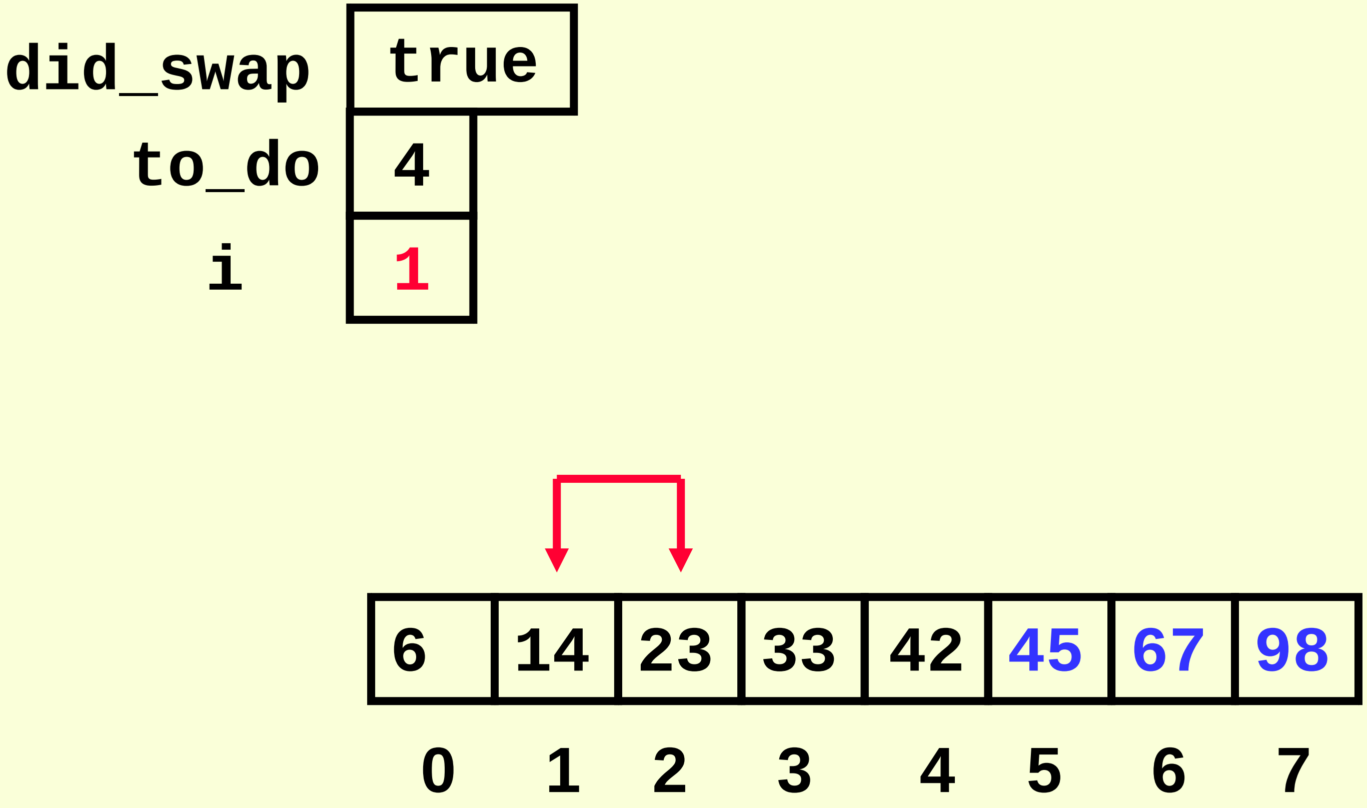
The Fourth "Bubble Up"



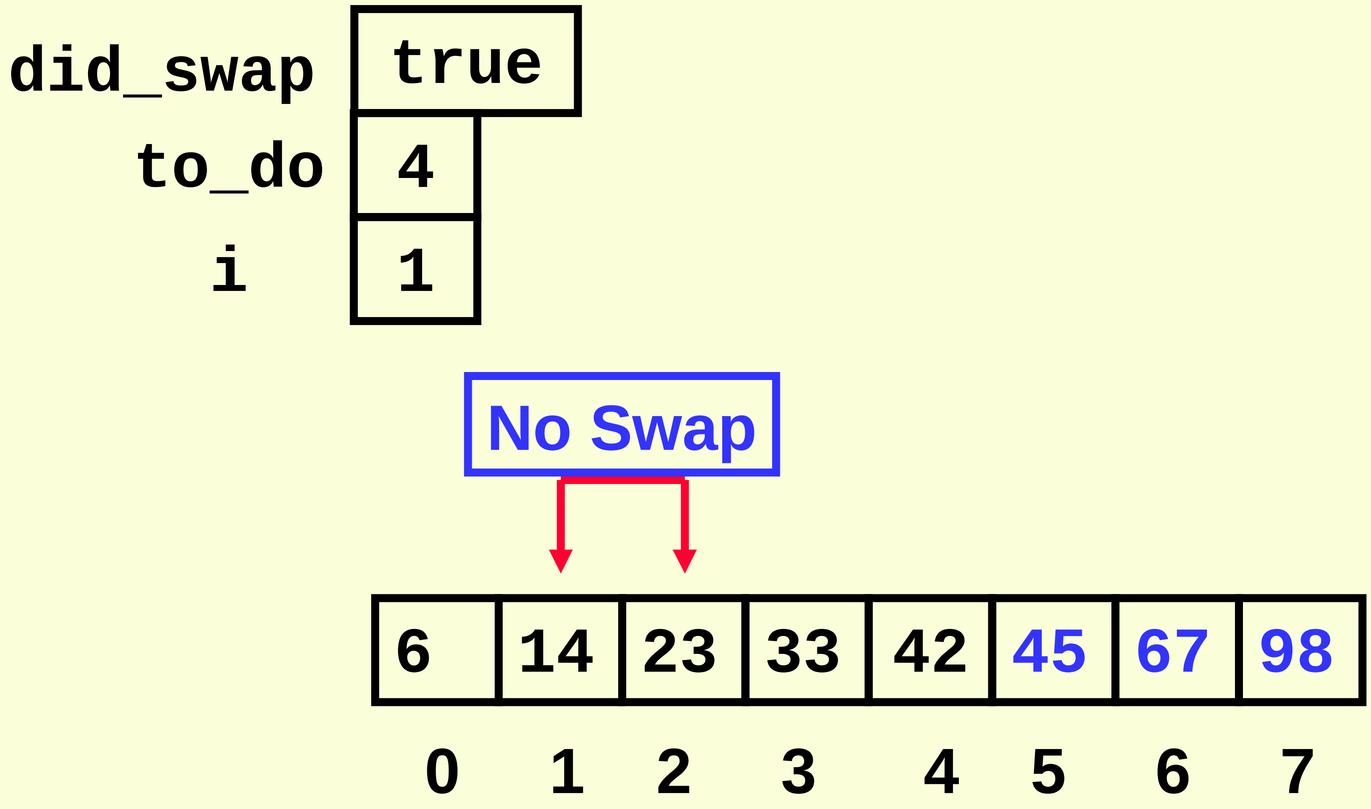
The Fourth "Bubble Up"



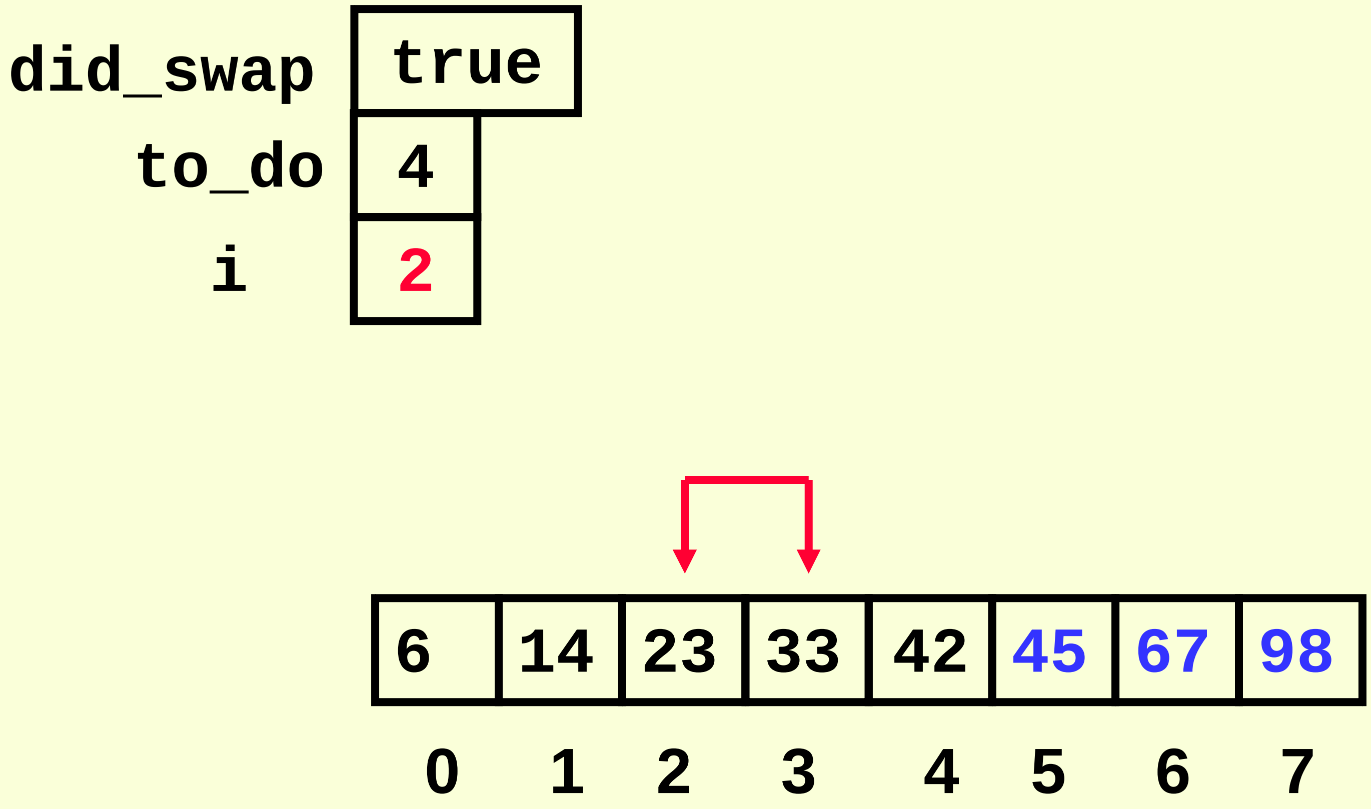
The Fourth “Bubble Up”



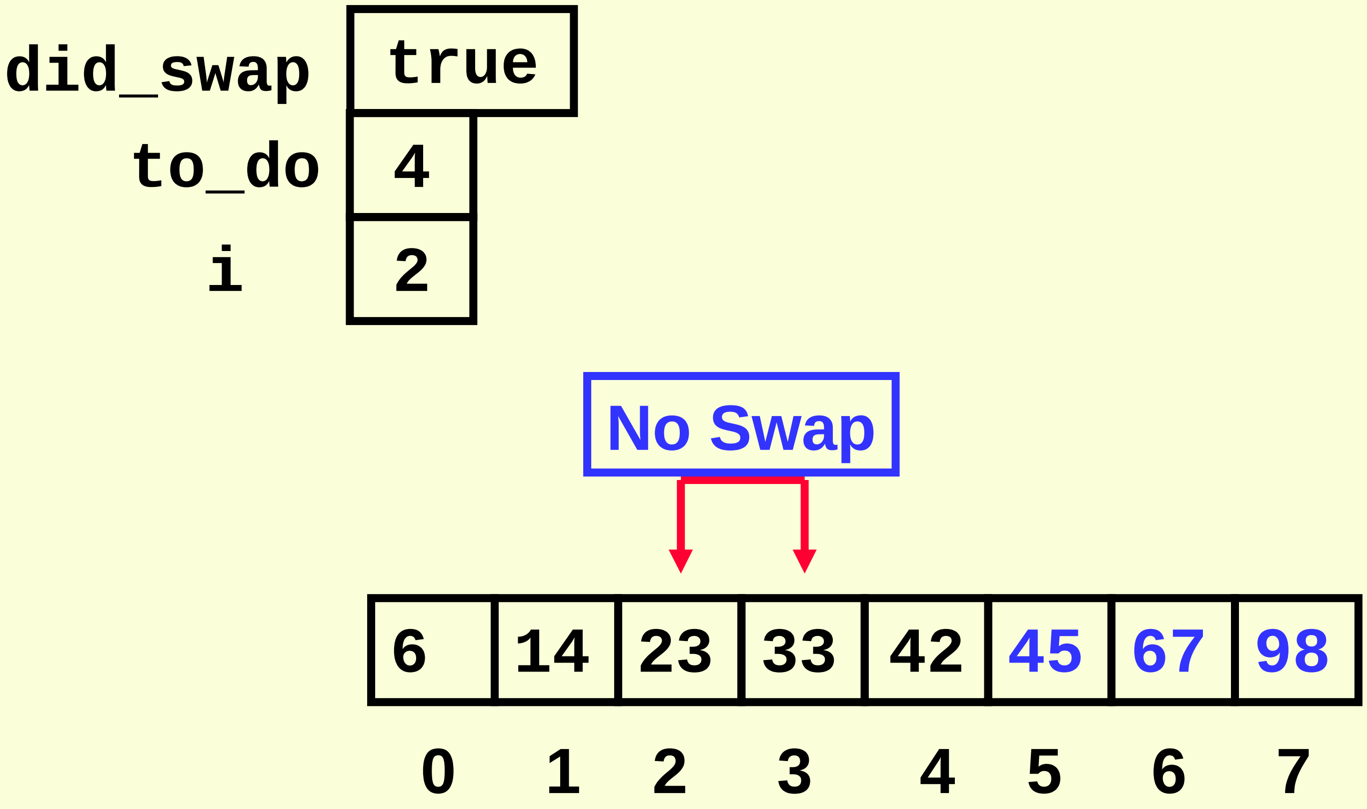
The Fourth “Bubble Up”



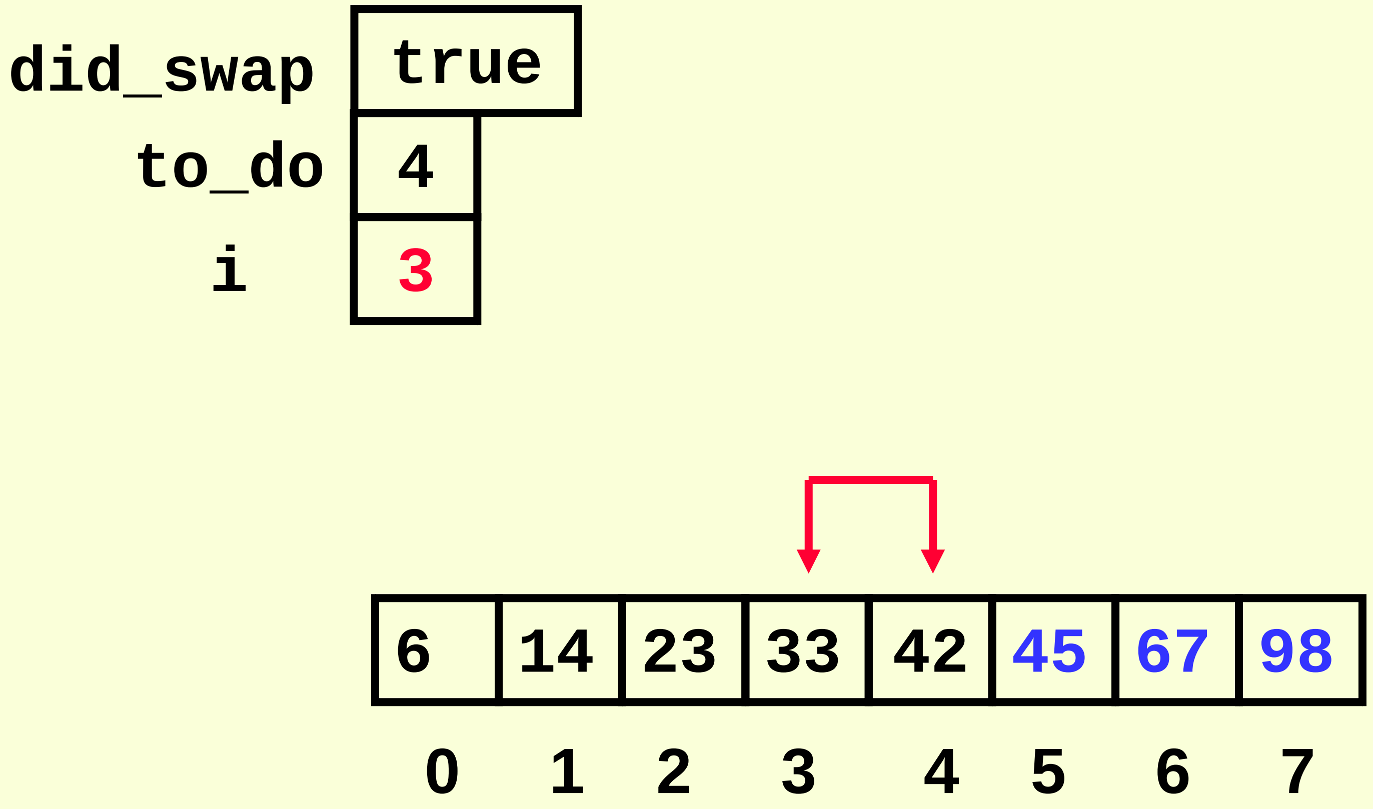
The Fourth “Bubble Up”



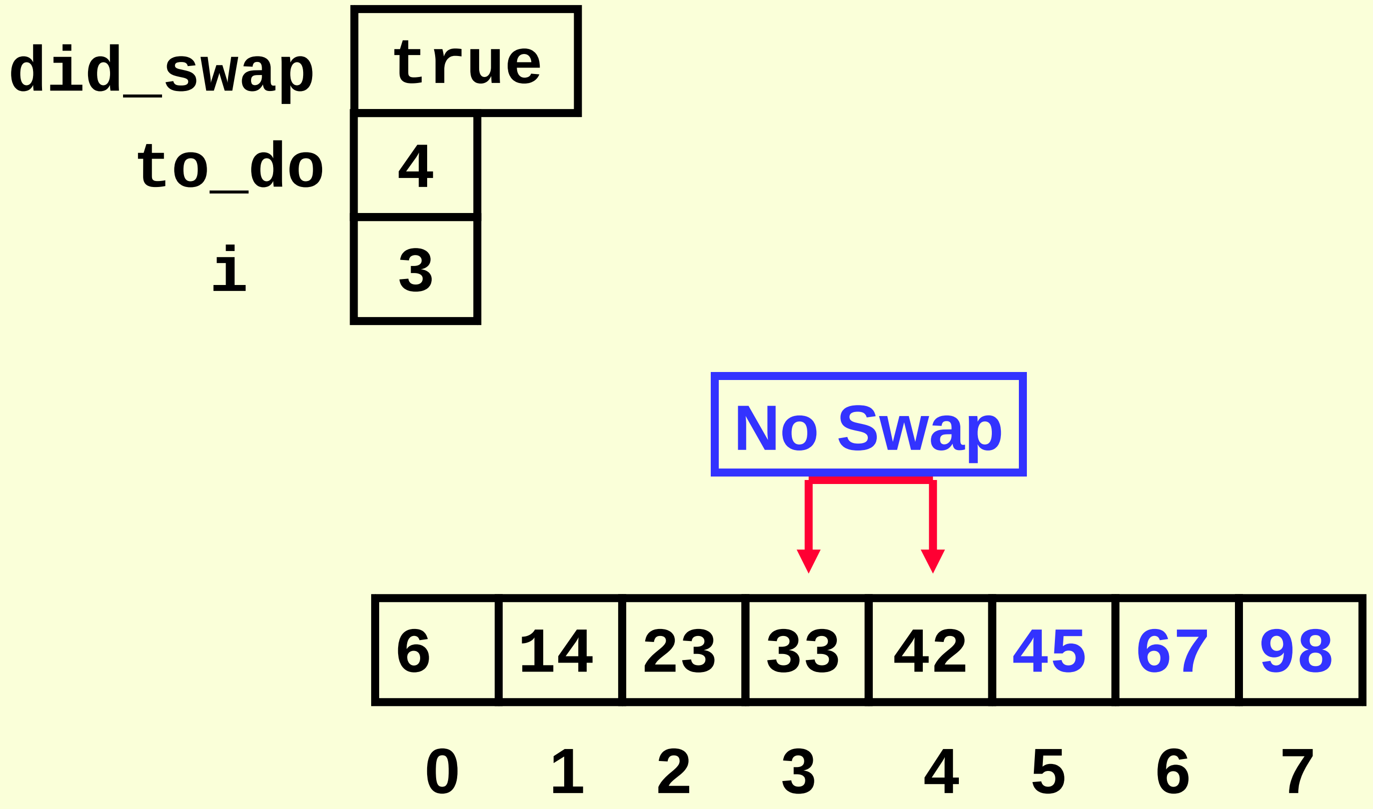
The Fourth "Bubble Up"



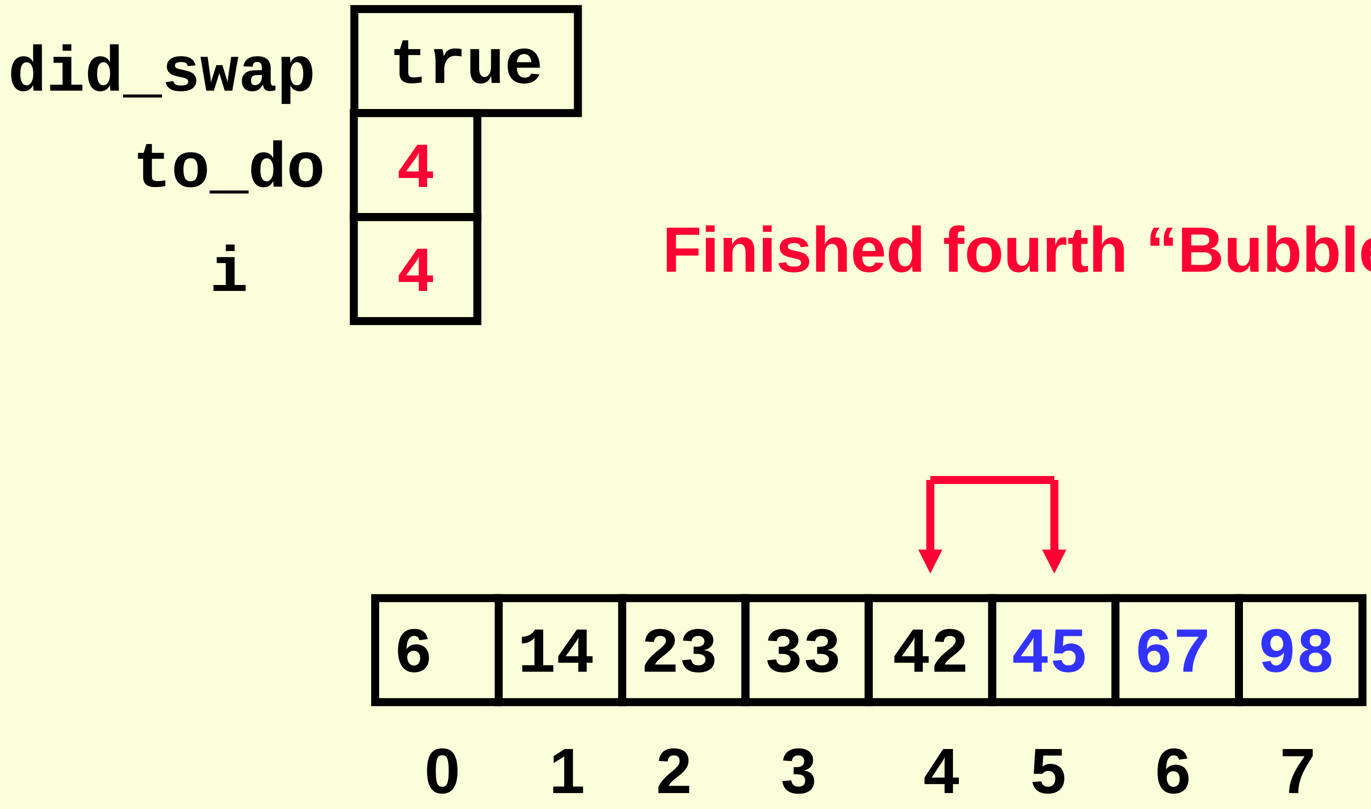
The Fourth “Bubble Up”



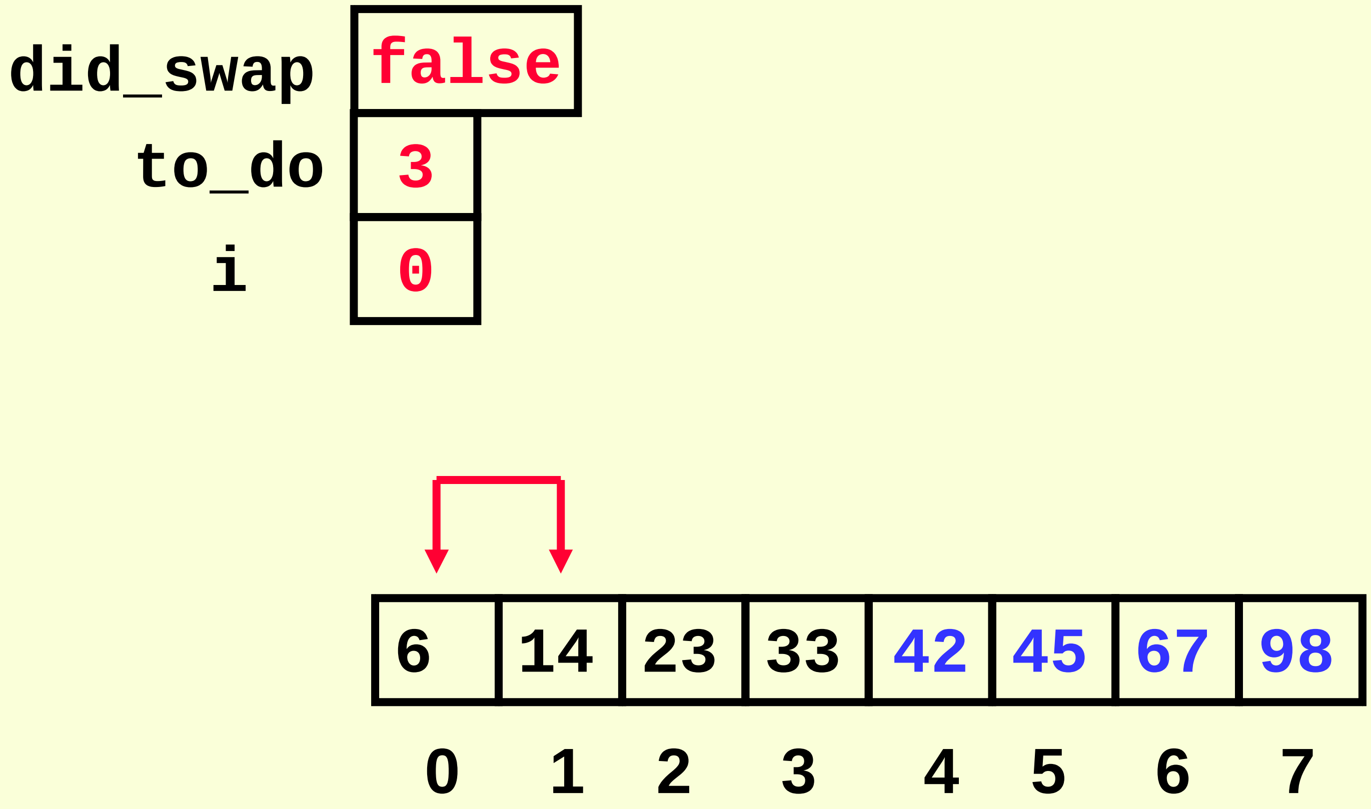
The Fourth "Bubble Up"



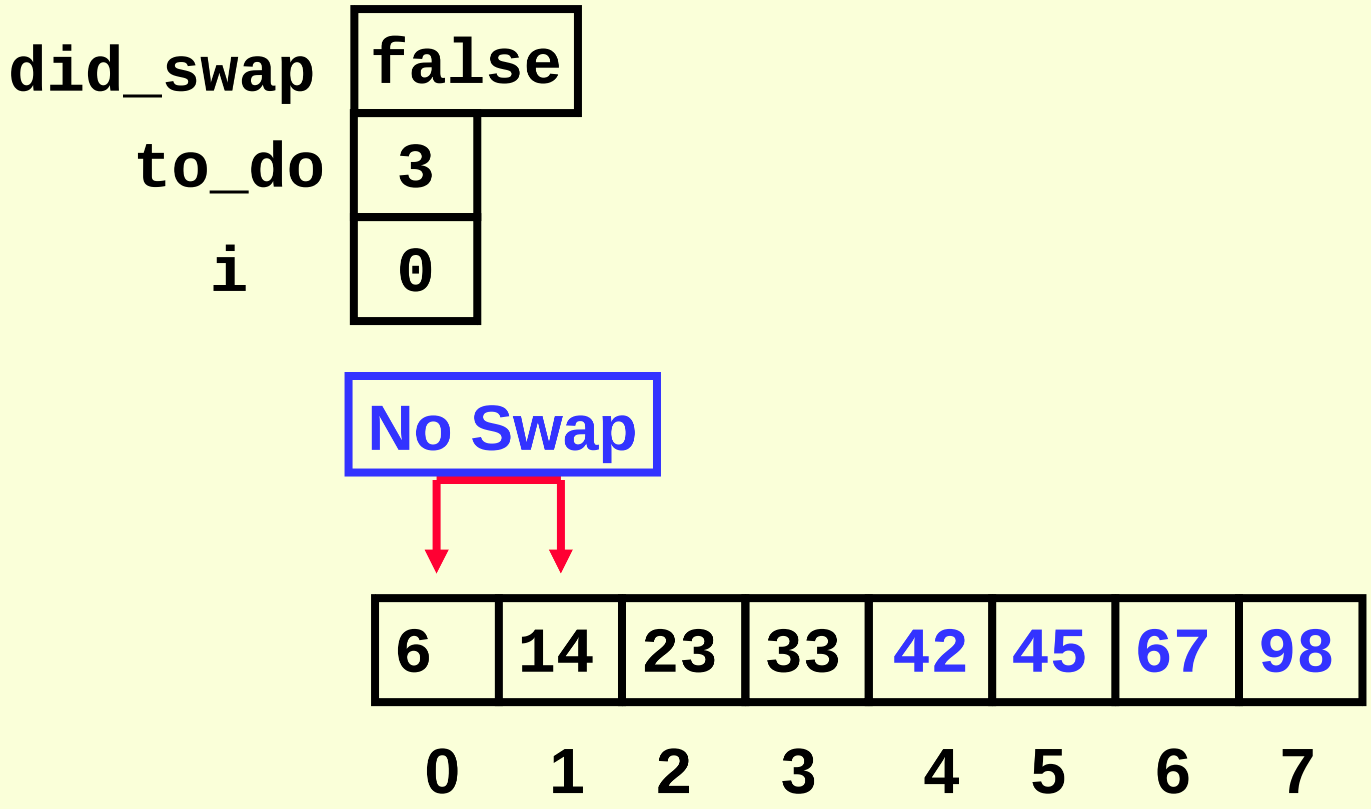
After Fourth Pass of Outer Loop



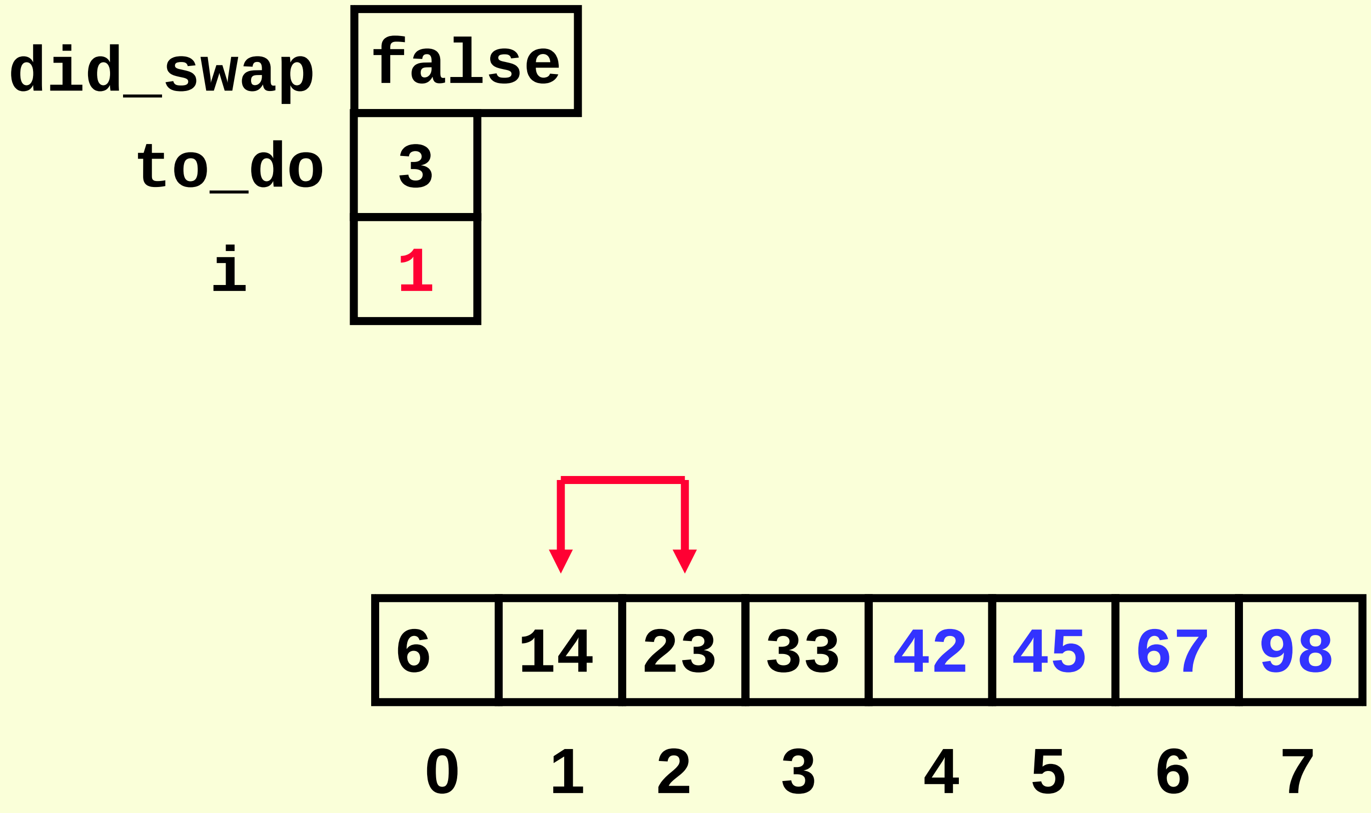
The Fifth “Bubble Up”



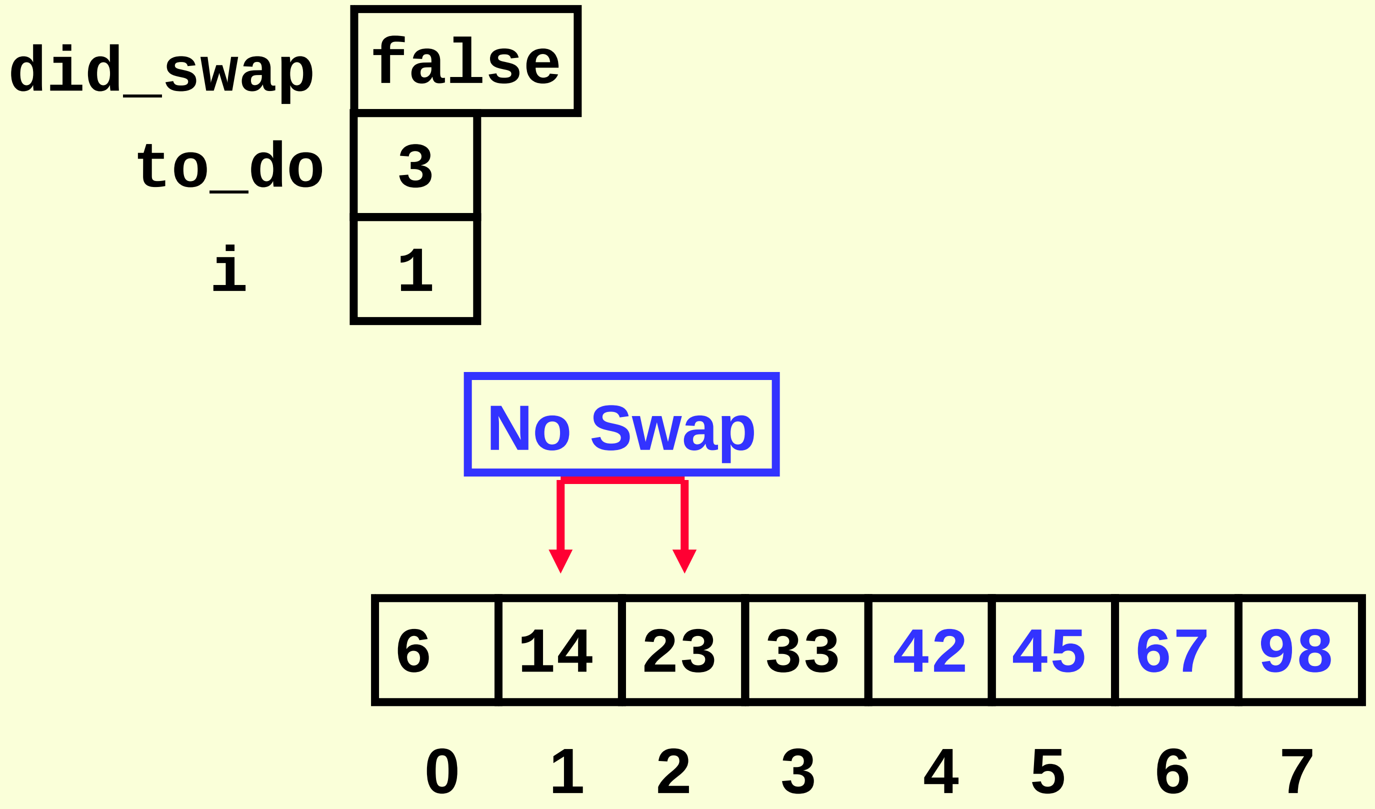
The Fifth "Bubble Up"



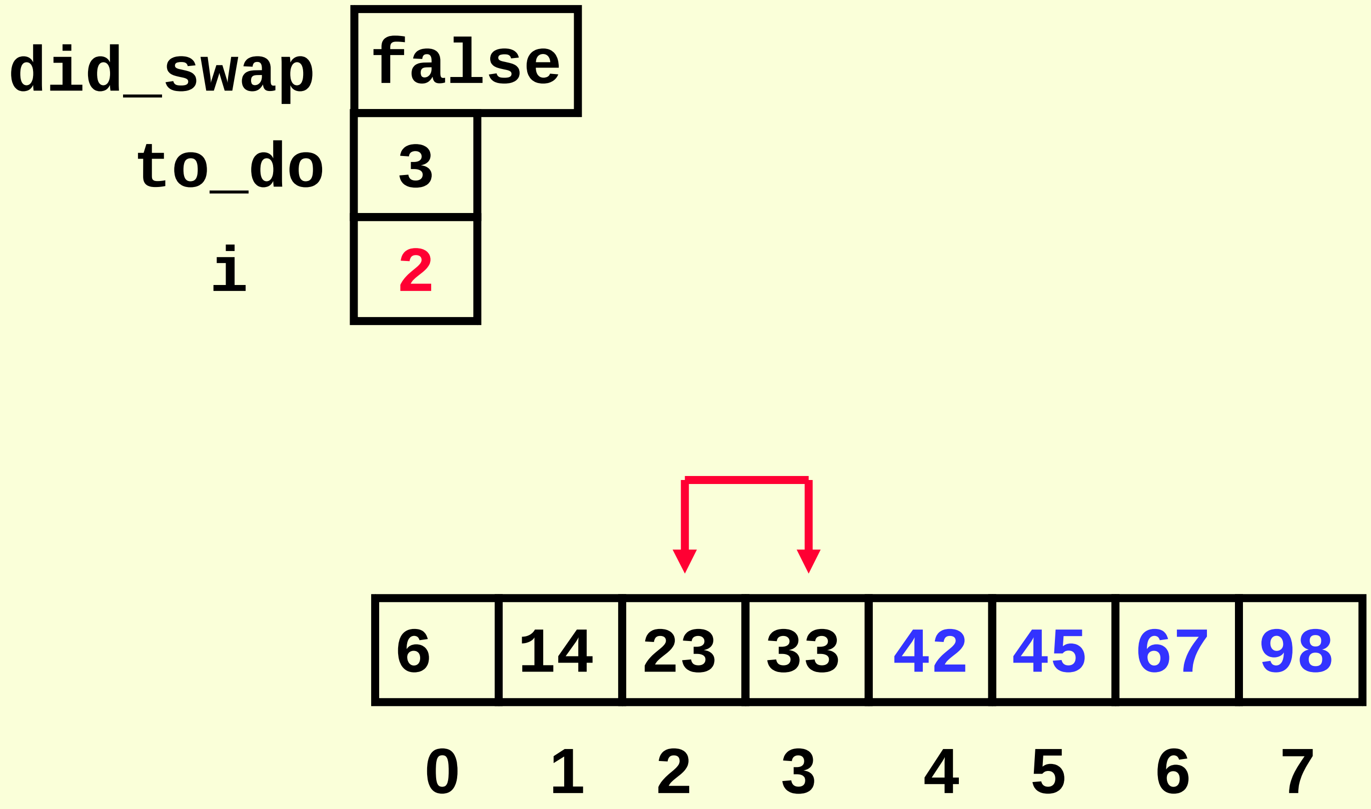
The Fifth “Bubble Up”



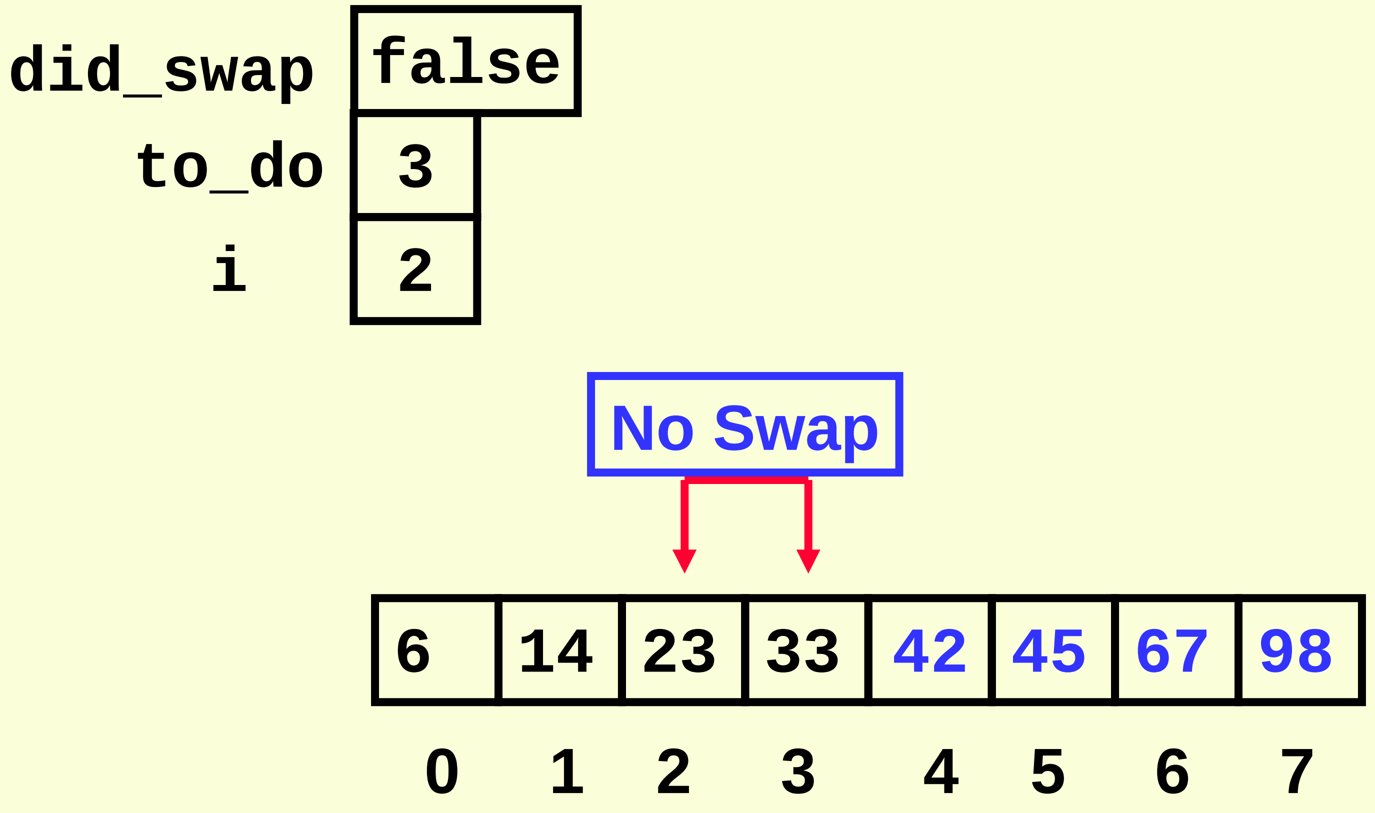
The Fifth "Bubble Up"



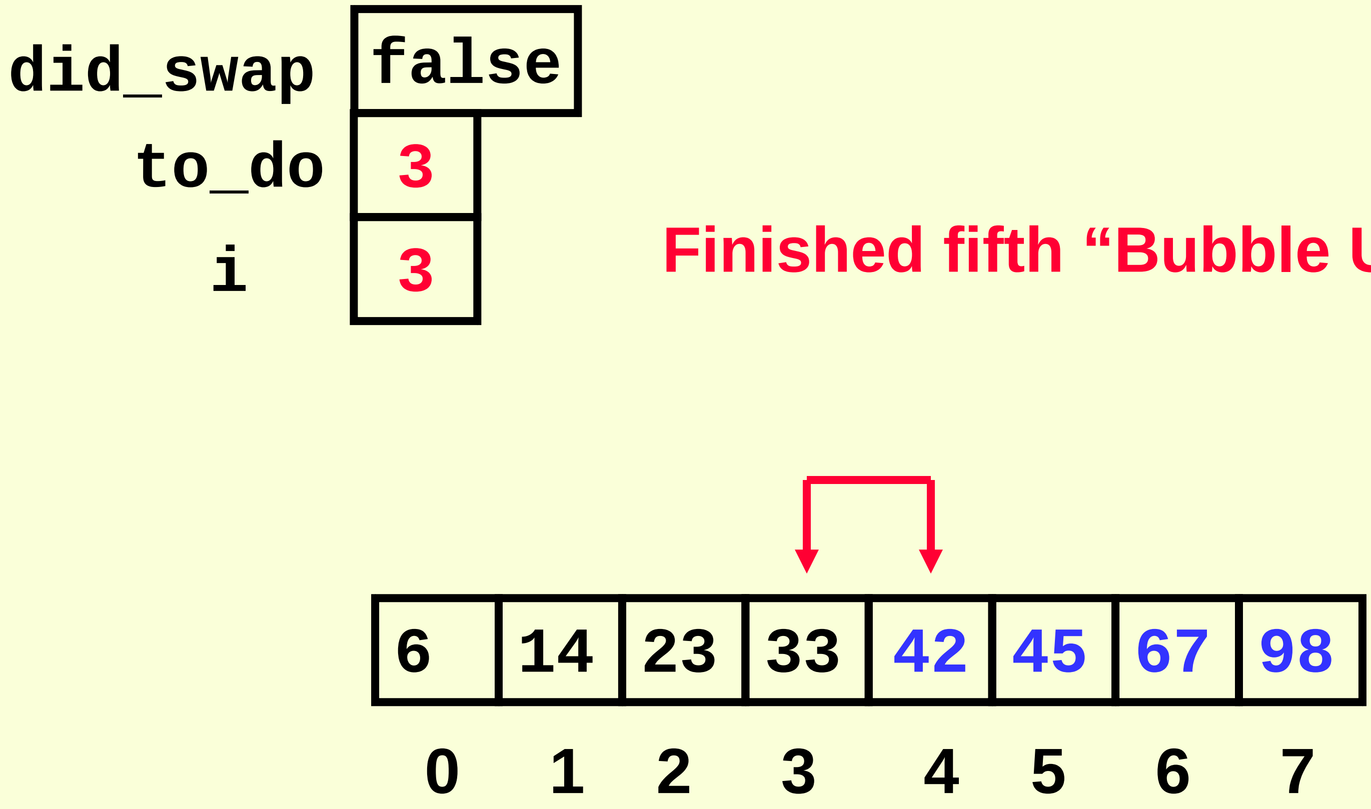
The Fifth “Bubble Up”



The Fifth “Bubble Up”



After Fifth Pass of Outer Loop



Finished fifth "Bubble Up"

Finished “Early”

did_swap	false
to_do	3
i	3

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
0	1	2	3	4	5	6	7

Summary

- “Bubble Up” algorithm will move largest value to its correct location (to the right/end of array)
- 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?