Quadratic Sorting

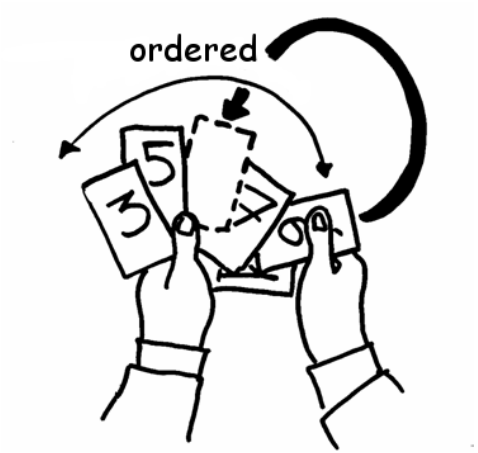
Review

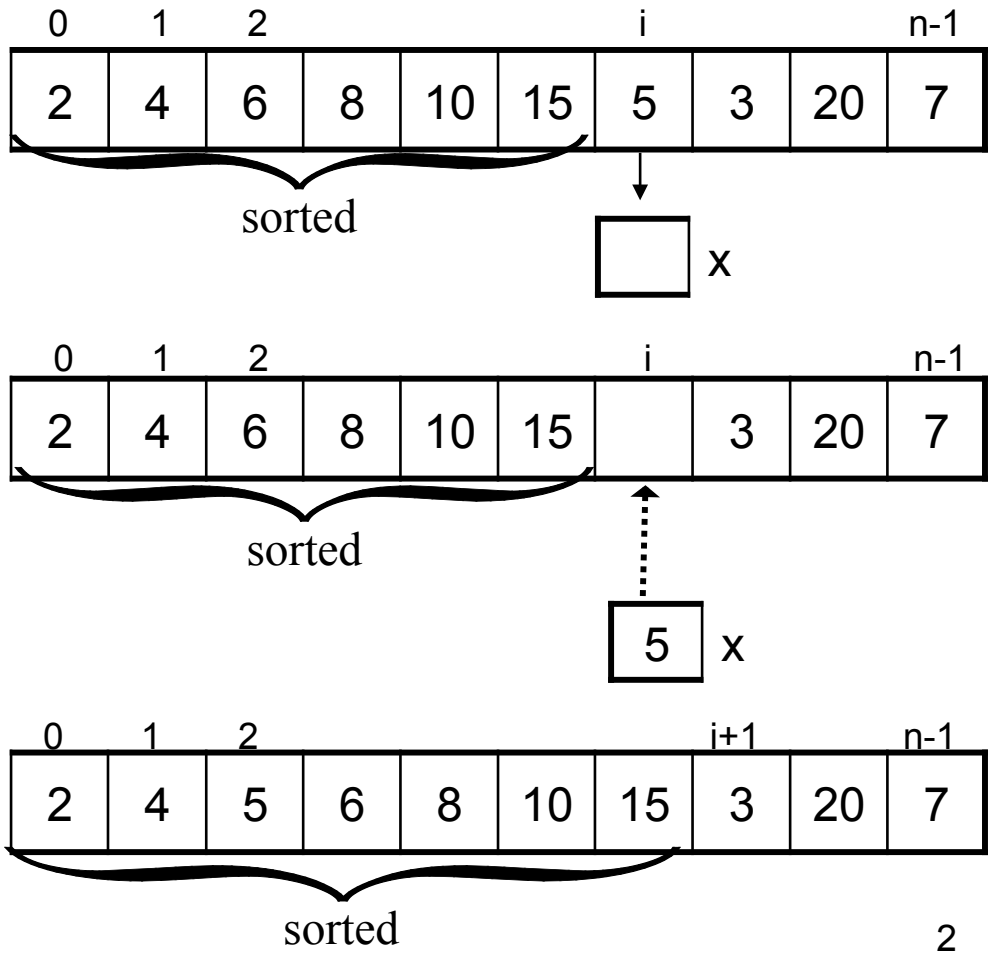
- Insertion Sort with an array

Review

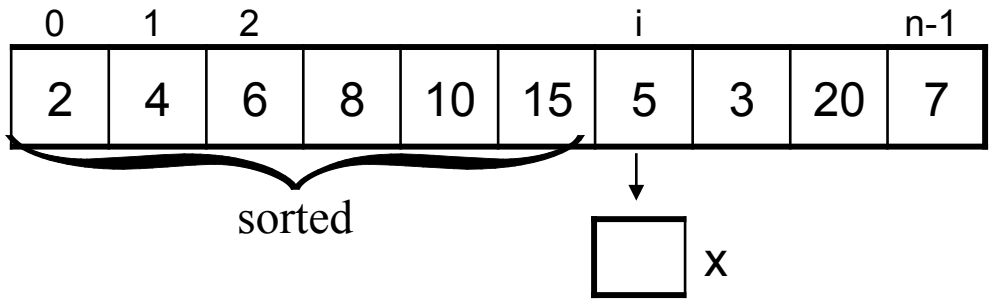
- Selection Sort with an array
- Bubblesort with an array

Insertion Sort (array)





```
for i=1 to n-1
  x ← A[i]
  j ← i-1
  while x.key < A[j].key
        and j >= 0
     A[j+1] ← A[j]
     j ← j-1
  A[j+1] ← x
```



Complexity — Insertion sort for arrays

Number of comparisons:

MIN

$(C_i)_{min} = 1$, i=1..n-1 (already in order) $\longrightarrow$ $C_{min} = n-1 = O(n)$

MAX

$(C_i)_{max} = i$, i=1..n-1 (in reverse order) $\longrightarrow$ $C_{max} = \sum_{i=1}^{n-1} i$

$= n(n-1)/2$

$= O(n^2)$

And with Sequence implemented with linked list ?

Complexity

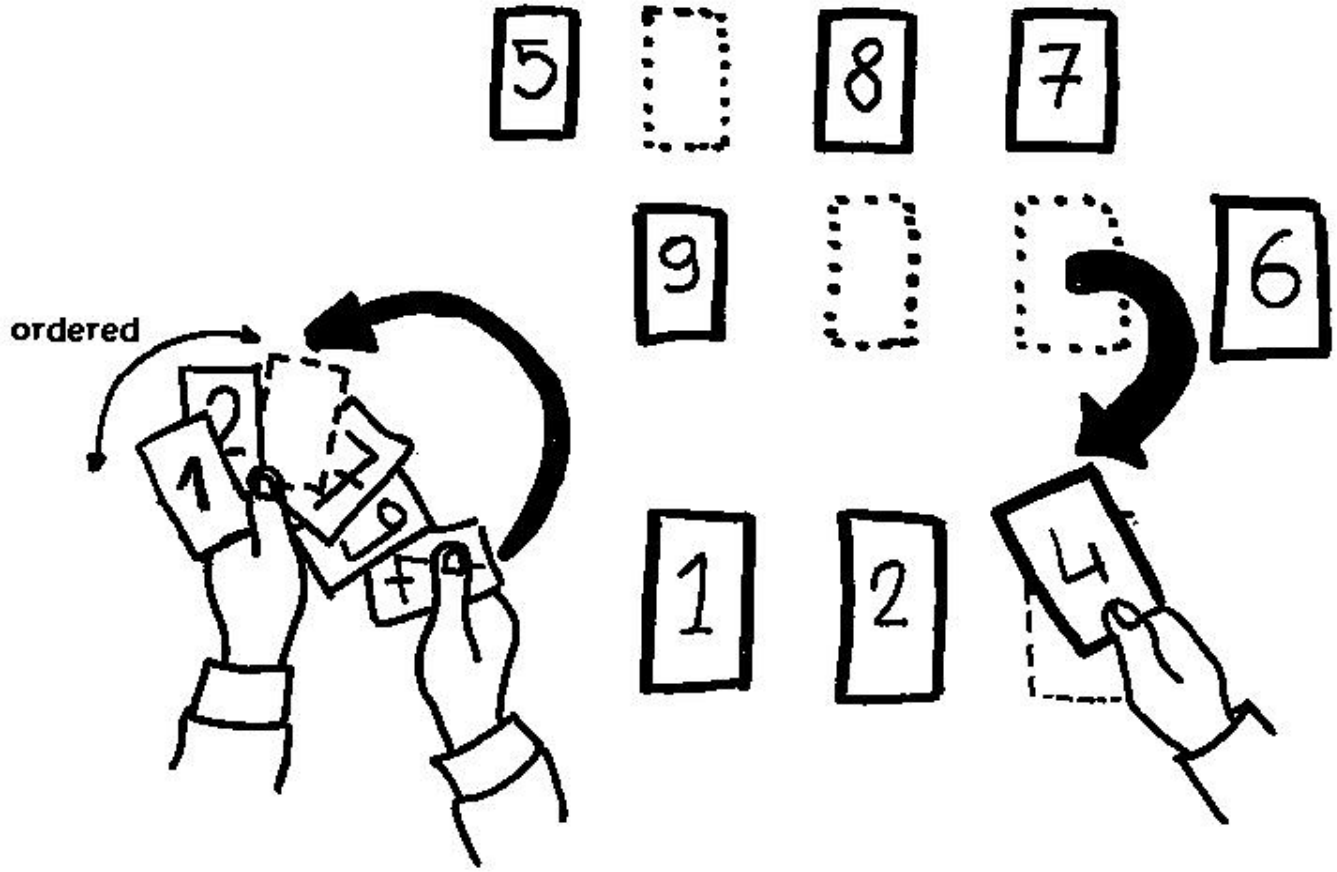
Insertion sort for arrays

Number of movements:

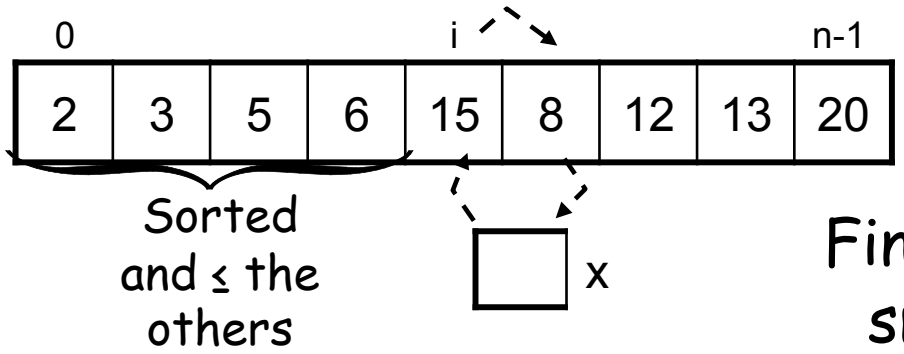
Same as comparisons

And with Sequence implemented with linked list ?

Selection Sort (array)



Selection Sort (array)



Finding the smallest element

```
for i=0 to n-2
  k ← i
  x ← A[i]
  for j=i+1 to n-1
    if A[j].key < x.key
          k ← j
          x ← A[j]
  A[k] ← A[i]
  A[i] ← x
```

Complexity of Selection Sort (array)

Comparisons (Does not depend on the initial order of the elements)

$$c = \sum_{i=0}^{n-2} \sum_{j=i+1}^{n-1} 1 = \sum_{i=0}^{n-2} (n - 1 - i) = \sum_{i=1}^{n-1} (n - i) = \sum_{i=1}^{n-1} n - \sum_{i=1}^{n-1} i$$

$$= n(n - 1) - n(n - 1) / 2$$

$$= n(n - 1) / 2$$

$$= O(n^2)$$

Complexity of Selection Sort (array)

Movements

MIN (already in order) = O(n)

MAX (in reverse order)
$(D_i)_{max} = O(n^2)$

And with Sequence implemented with linked list ?

Bubble Sort

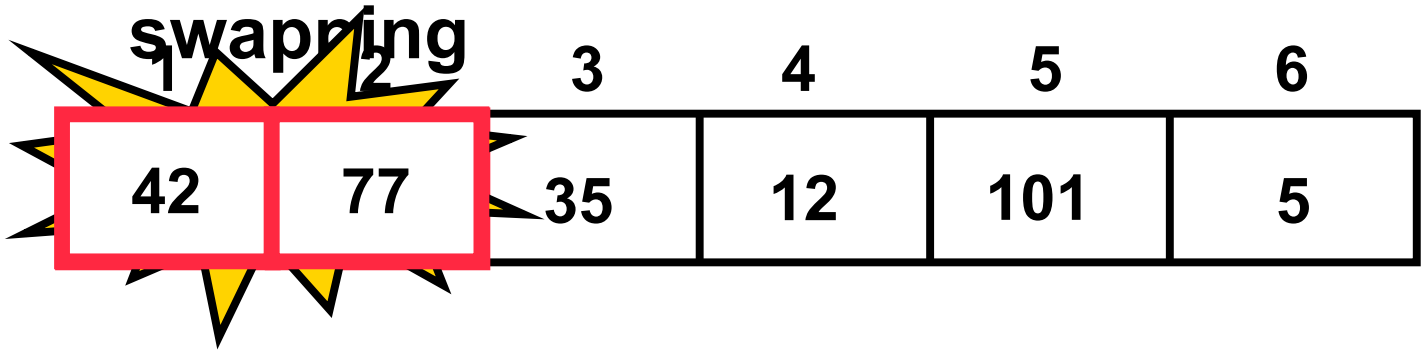
"Bubbling Up" the Largest Element

- **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**

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

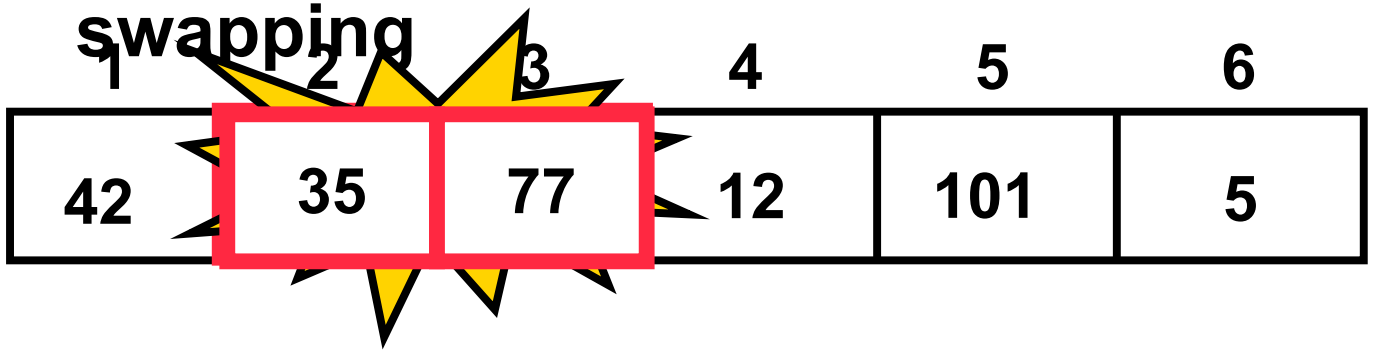
"Bubbling Up" the Largest Element

- **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**



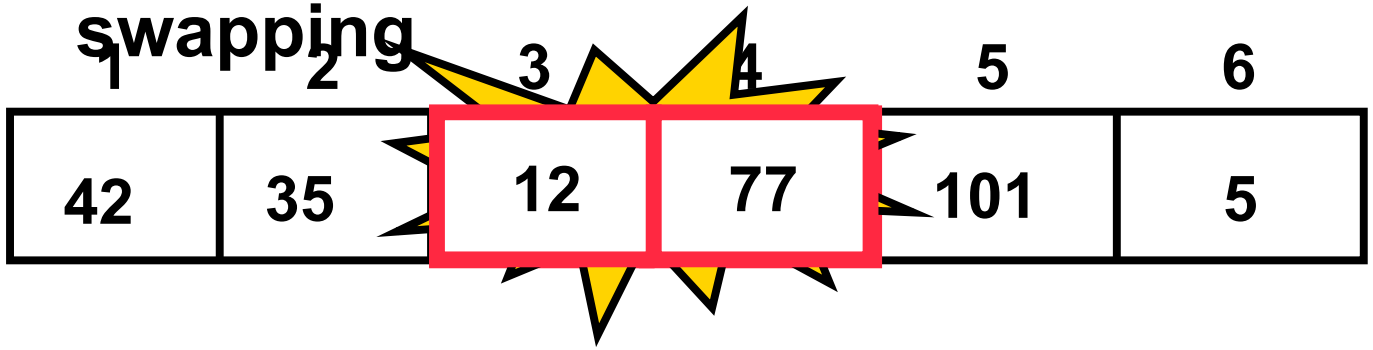
"Bubbling Up" the Largest Element

- **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**



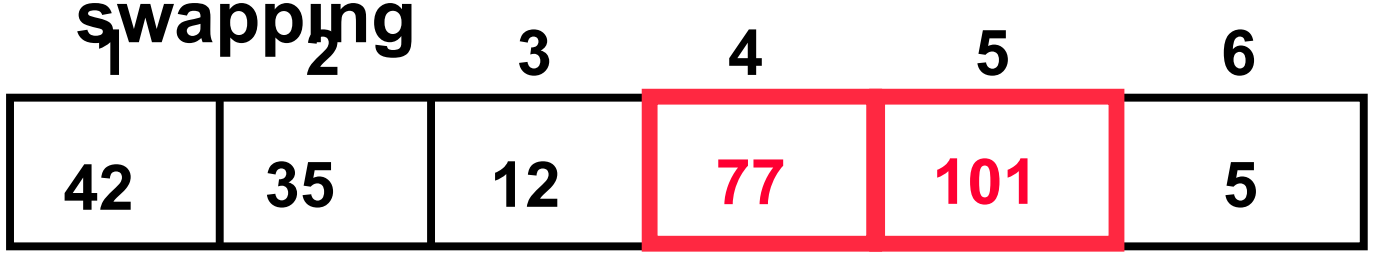
"Bubbling Up" the Largest Element

- **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

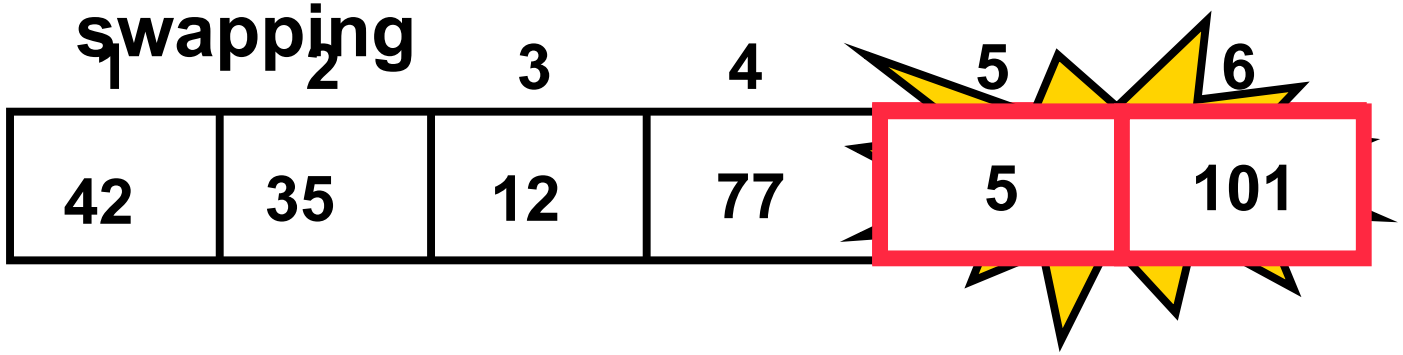
- **Traverse a collection of elements**
  - **Move from the front to the end**
  - **"Bubble" the largest value to the end using pair-wise comparisons and swapping**



No need to swap

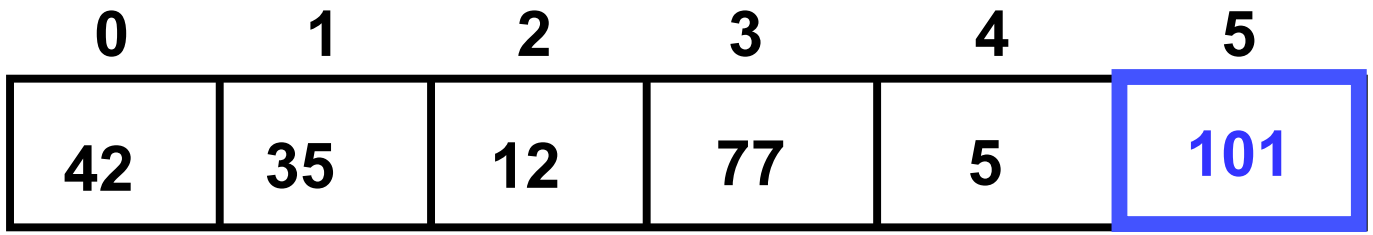
"Bubbling Up" the Largest Element

- **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

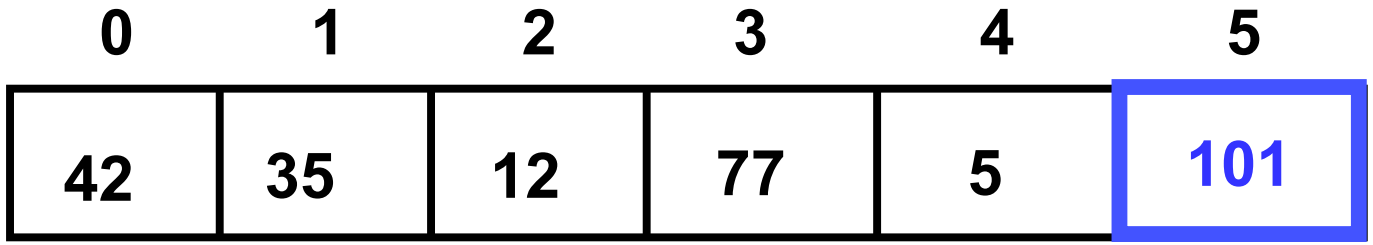
- **Traverse a collection of elements**
  - **Move from the front to the end**
  - **“Bubble” the largest value to the end using pair-wise comparisons and swapping**



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**

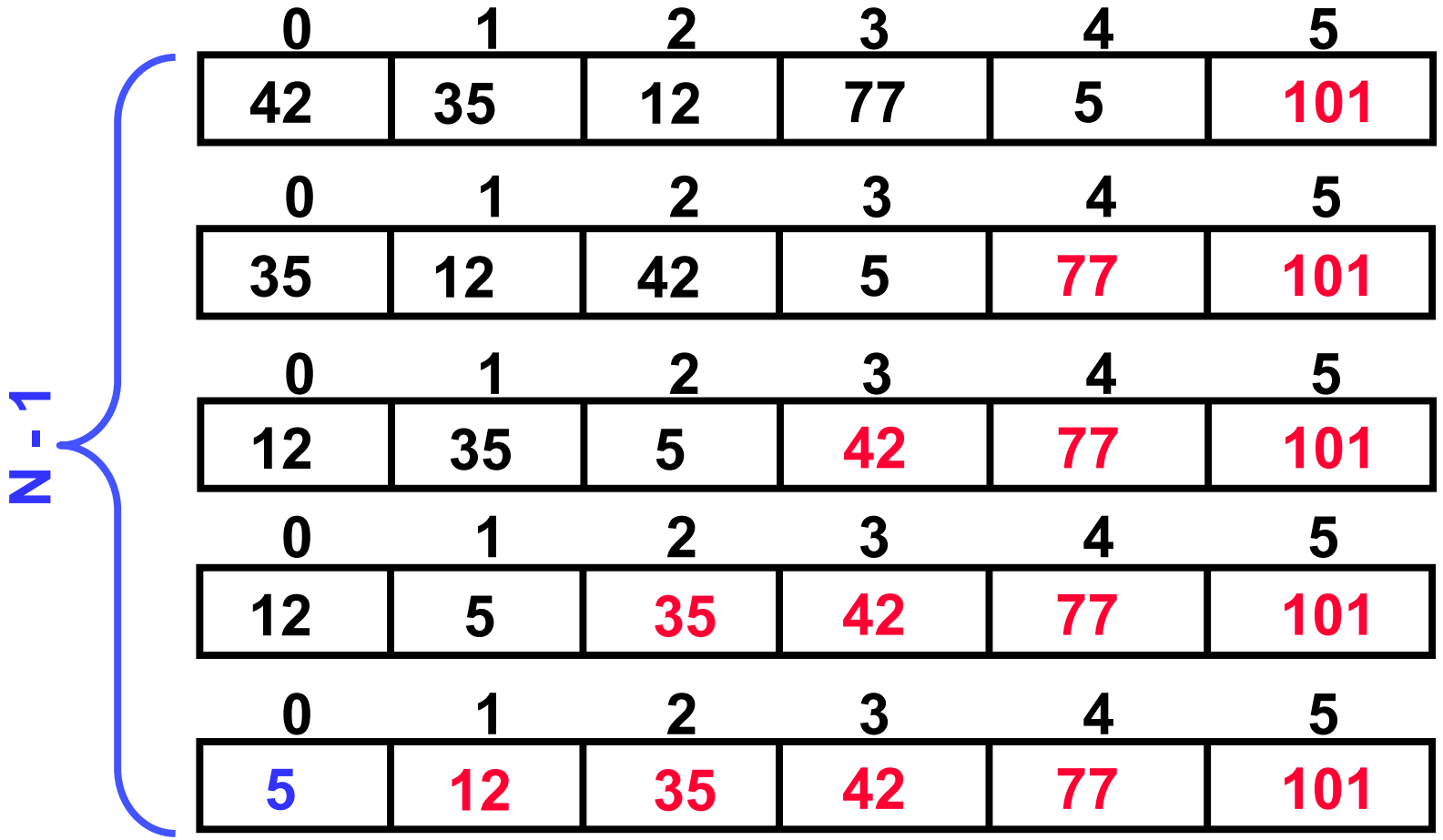


Largest value correctly placed

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



Complexity of Bubblesort (arrays)

And with Sequence implemented with linked list ?

Comparisons

$$c = \sum_{i=1}^{n-1} (n - i) = \frac{n}{2}(n - 1) = O(n^2)$$

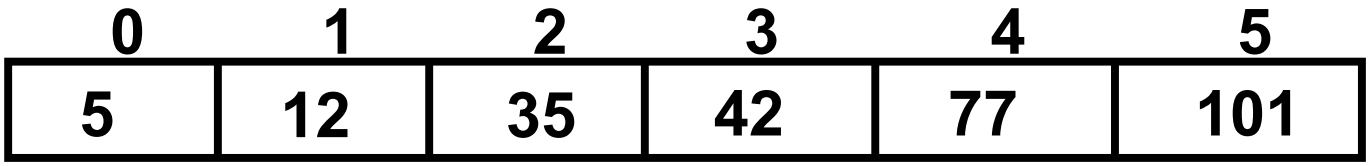
Movements

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

$D_{max} = 3 \bullet C = O(n^2)$ (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"!**



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.”**

Bubblesort algorithm

```
j ← 0
swapped ← true
while (swapped)
  swapped ← false
  j ← j+1
  for i=0 to n-j-1
     if A[i].key > A[i+1].key
        tmp= A[i]
        A[i]= A[i+1]
        A[i+1]= tmp
        swapped ← true
```