COMP 250

Lecture 7

Sorting a List:
bubble sort
selection sort
insertion sort

Sept. 22, 2017

Sorting

BEFORE

3
17
-5
-2
23
4

AFTER

-5
-2
3
4
17
23

Example: sorting exams by last name

Sorting Algorithms

- Bubble sort
- Selection sort
- Insertion sort

today $O(N^2)$

- Mergesort
- Heapsort
- Quicksort

later $O(N \log N)$

Sorting Algorithms

Today we are concerned with algorithms, not data structures.

The following algorithms are independent of whether we use an array list or a linked list.

Bubble Sort

Repeatedly loop (iterate) through the list.

For each iteration,

if two neighboring elements are in the wrong order,

then swap them.

Reminder from 202: swap(x, y)

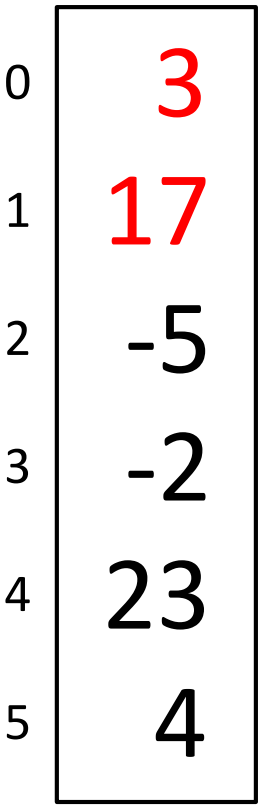
The following does not work:

```
x = y
y = x
```

Rather, you need to use a temporary variable:

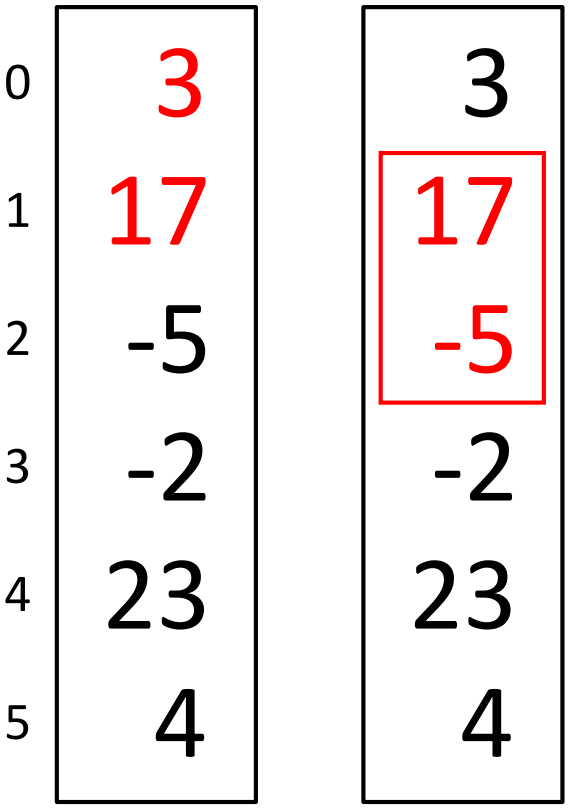
```
tmp = y
y = x
x = tmp
```

Example: first pass




```
if  list[ 0 ] > list[ 1 ]
        swap( list[ 0 ], list[1 ] )
```

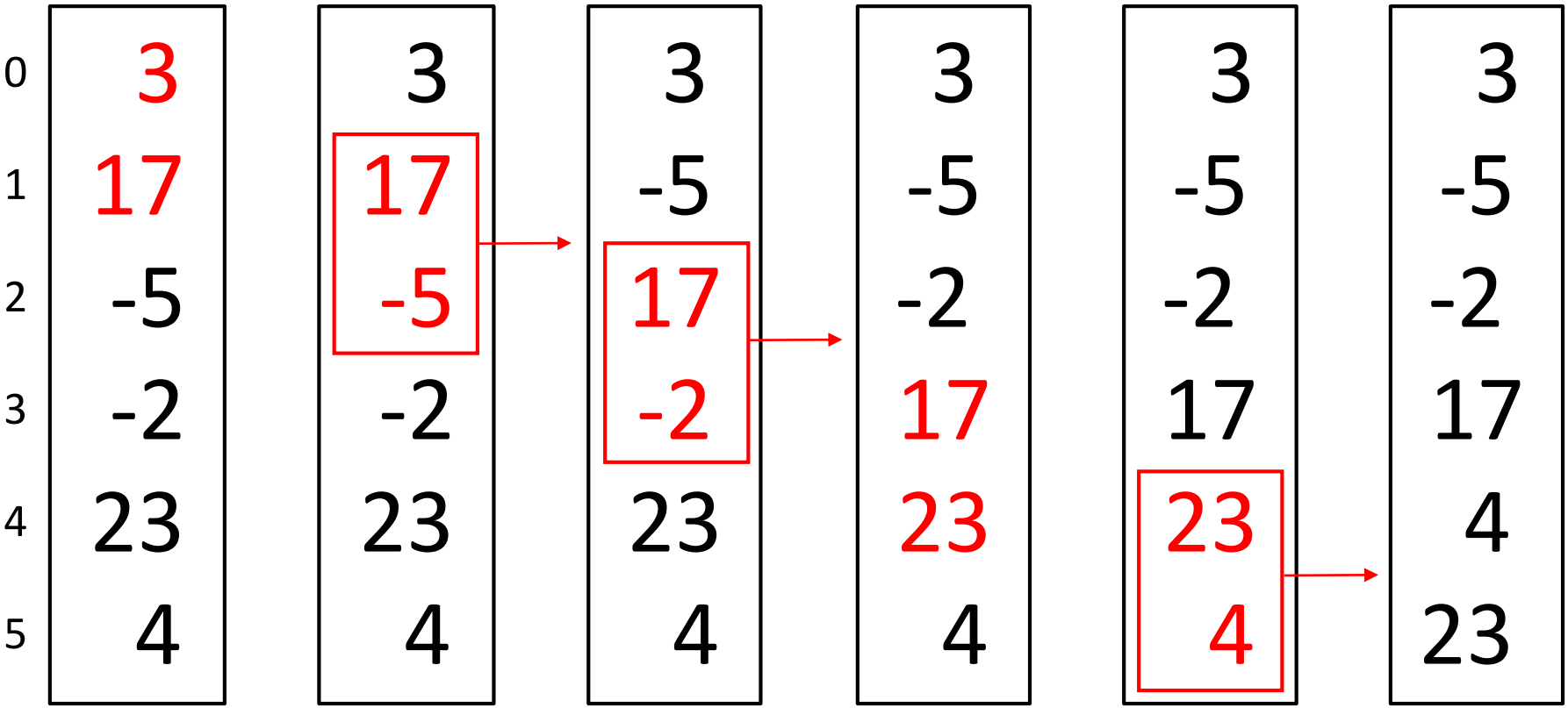
Example: first pass



Indicates elements get swapped

```
if  list[ 1 ] > list[ 2 ]
        swap( list[ 1 ], list[  2 ] )
```

Example: first pass



Indicates elements get swapped

What can we say at end of the first pass?

Q: Where is the largest element ?

A:

Q: Where is the smallest element?

A:

What can we say at end of the first pass?

Q: Where is the largest element ?

A: It must be at the end of the list (position N-1).

Q: Where is the smallest element ?

A: Anywhere (except position N-1).

Bubble Sort Algorithm

```
repeat {
    continue = false
    for i = 0 to N – 2                    //  N-1 is the last index
        if  list[ i ] > list[ i + 1 ]{
                swap( list[ i ], list[ i + 1 ] )
                continue = true
         }
}  until continue == false
```

Bubble Sort Algorithm

```
repeat {
    ct = 0
    continue = false
    for i = 0 to N – 2 – ct  {          //  N-1 is the last index
        if  list[ i ] > list[ i + 1 ]{
                swap( list[ i ], list[ i + 1 ] )
                continue = true
        }
        ct = ct + 1    //  now list[ N - ct, … N-1] is sorted
    }
}  until continue == false
```

Selection Sort

Partition the list into two parts: (1) a sorted part and (2) a “rest” part, as follows:

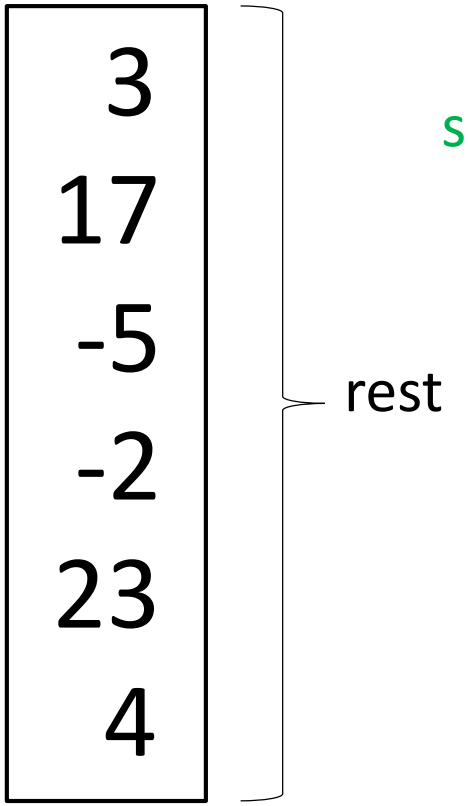
The sorted part is initially empty.

Repeat N times {

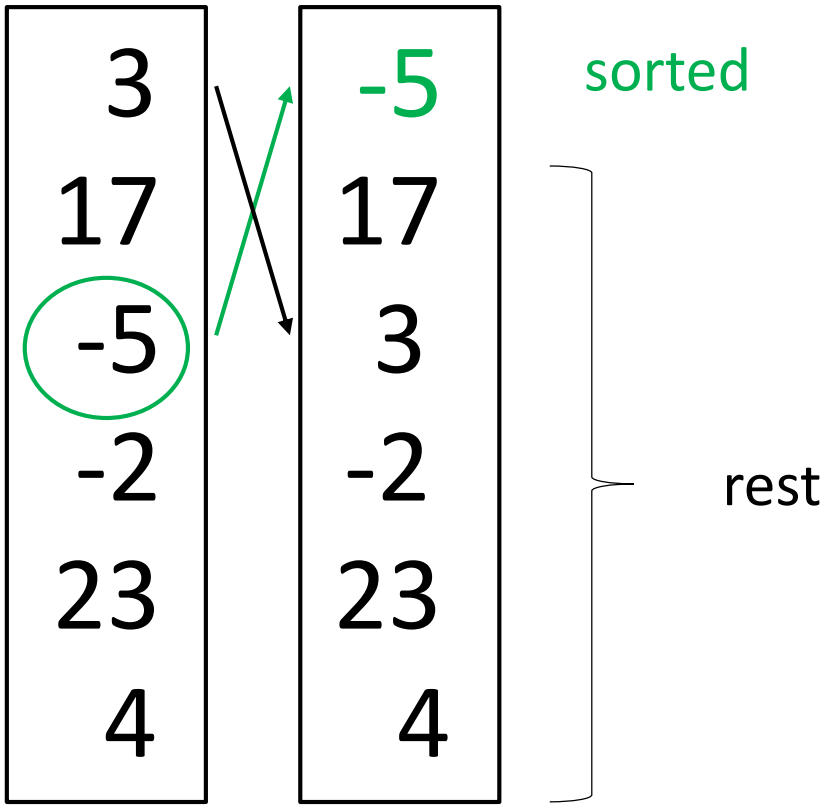
find the smallest element in the rest part and

swap it with the first element in the rest part

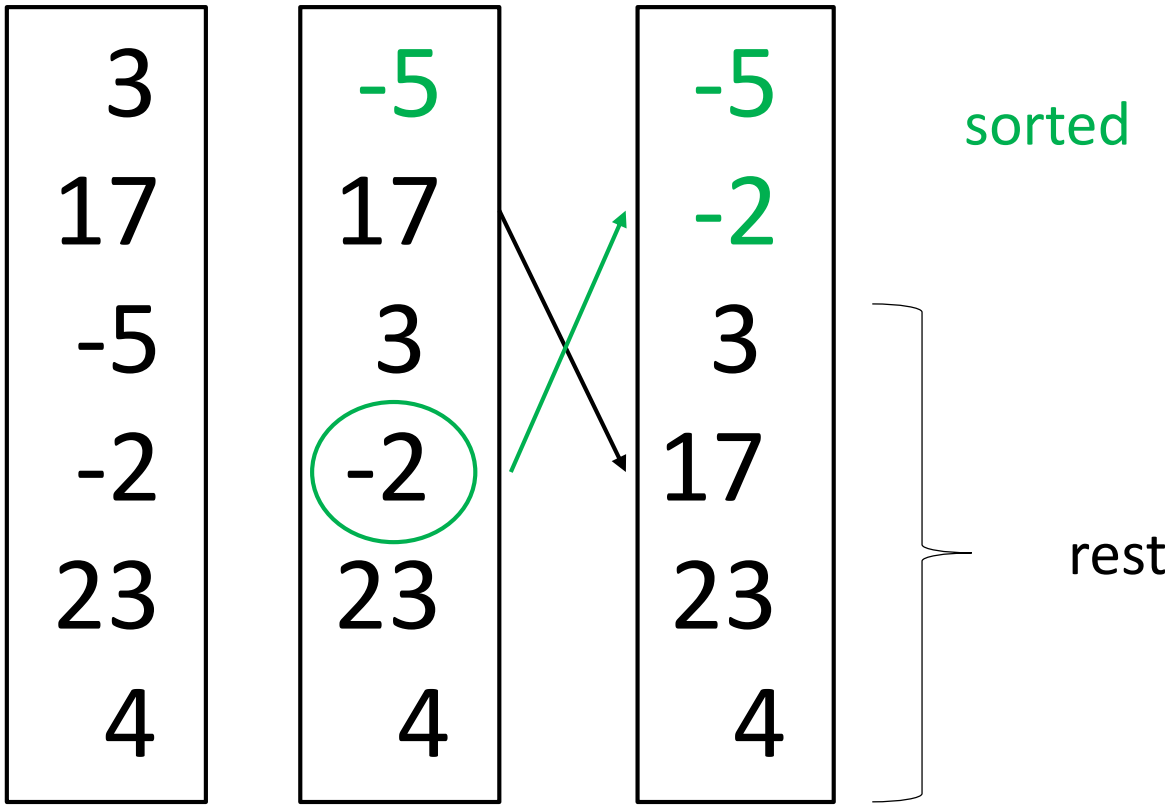
Example



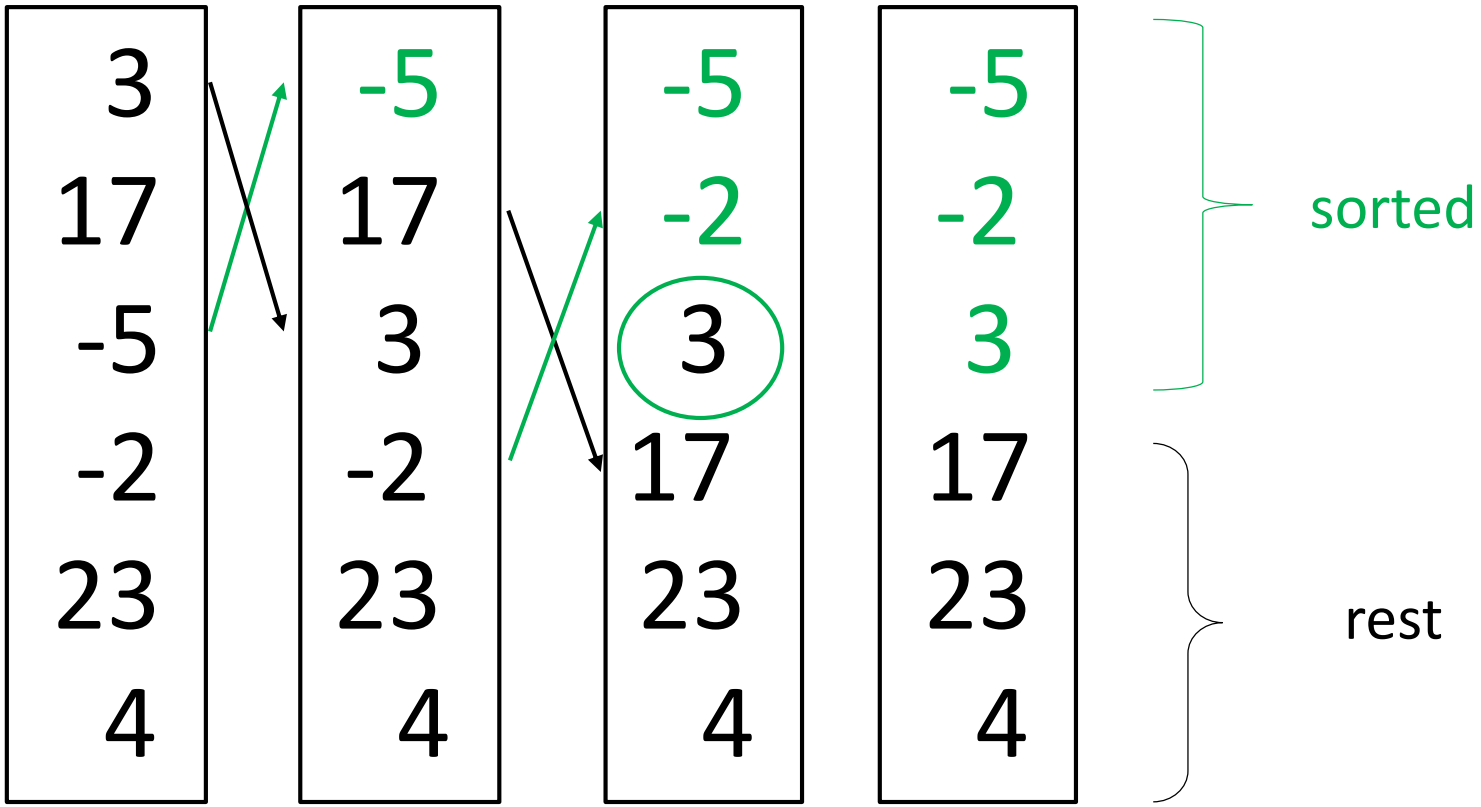
Example



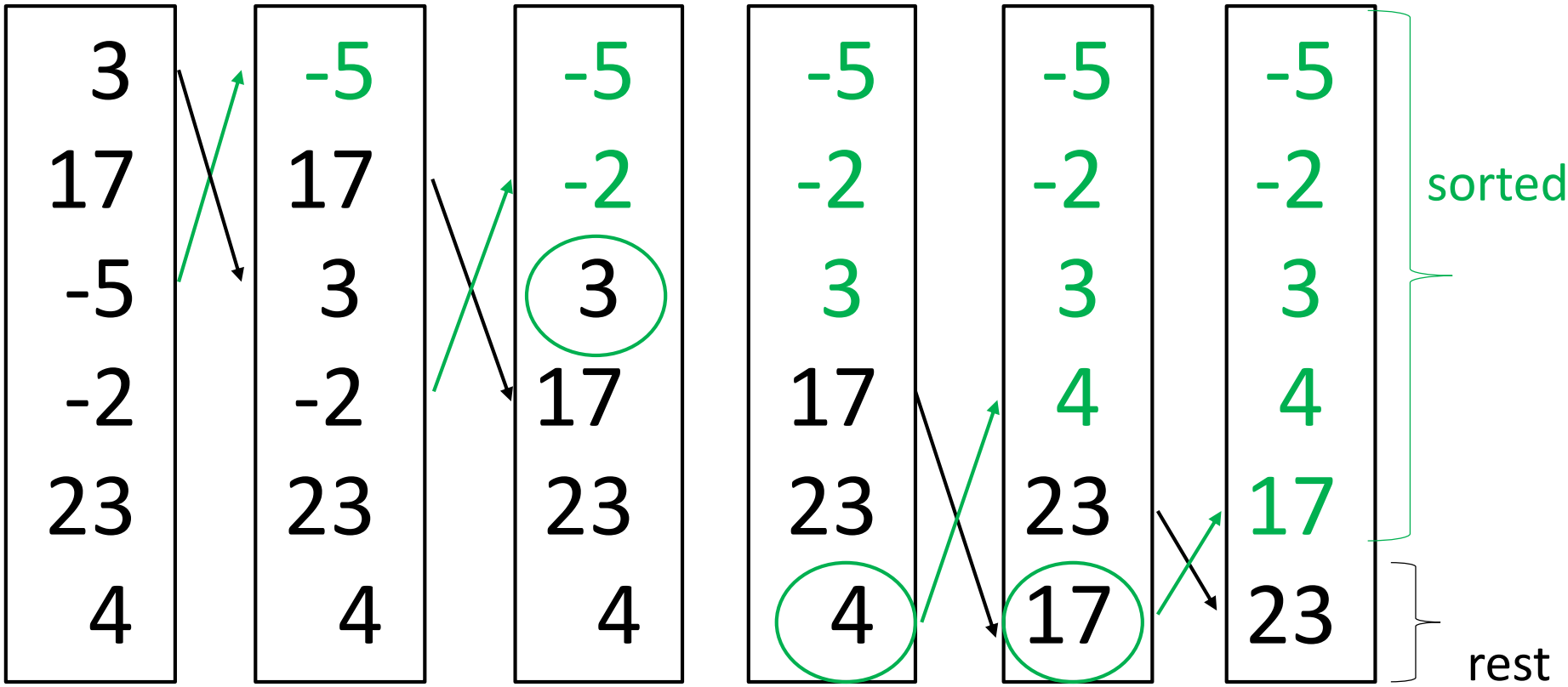
Example



Example



Example


3
17
-5
-2
23
4
-5
17
3
-2
23
4
-5
-2
3
17
23
4
-5
-2
3
17
23
4
-5
-2
3
4
23
17
-5
-2
3
4
17
23
sorted
rest

Selection Sort

```
for  i = 0  to N-2   {                  //   repeat  N times
    index = i                           //   Take the first element in the rest.
    minValue = list[ i ]                //    It has the min value so far.

    for k = i+1  to  N-1  {             //   For each other element in rest,
       if ( list[k] < minValue ){       //    if it is smaller than the min value,
            index = k                   //     then remember its index.
            minValue = list[k]          //     It is the new min value.
       }
     if ( index != i )                  //   Swap if necessary
          swap(  list[i], list[ index ] )
}
```

Selection Sort

```
for  i = 0  to N-2
        for k = i+1  to  N-1
            …….
```

Q: how many passes through inner loop?

Selection Sort

```
for  i = 0  to N-2
        for k = i+1  to  N-1
            …….
```

Q: how many passes through inner loop?

A: N-1 + N-2 + N-3 + …. + 2 + 1

Selection Sort

```
for  i = 0  to N-2
        for k = i+1  to  N-1
              …….
```

Q: how many passes through inner loop?

A: N-1 + N-2 + N-3 + …. + 2 + 1

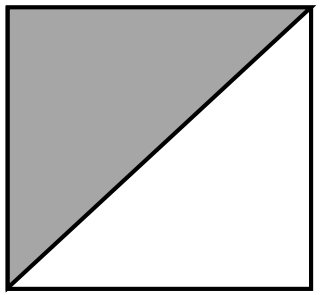
= N (N-1) / 2

Comparison

Bubblesort

```
repeat {
   for i = 0 to N – 2 – ct
until continue == false
```

We can terminate outer loop if there are no swaps during a pass.



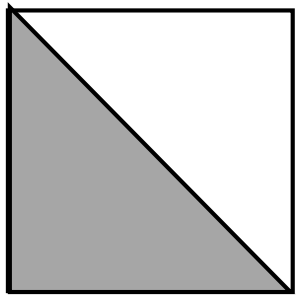
Outer loop

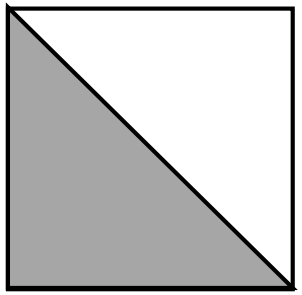
Best case

Worst case

Selection sort

```
for  i = 0  to N-2
    for k = i+1  to  N-1
```





Outer loop

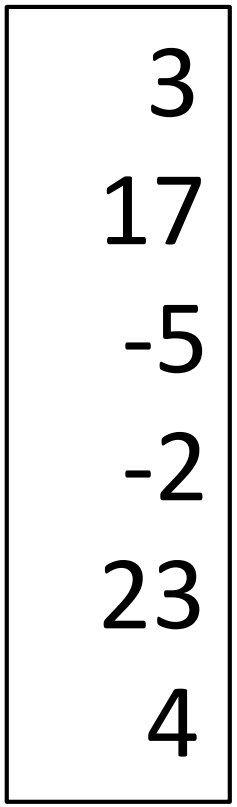
Insertion Sort

for k = 1 to N- 1 {

Insert list element at index k into its correct position with respect to the elements at indices 0 to k – 1

}

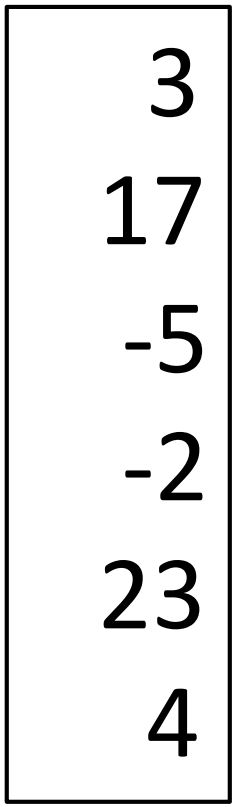
Initial list

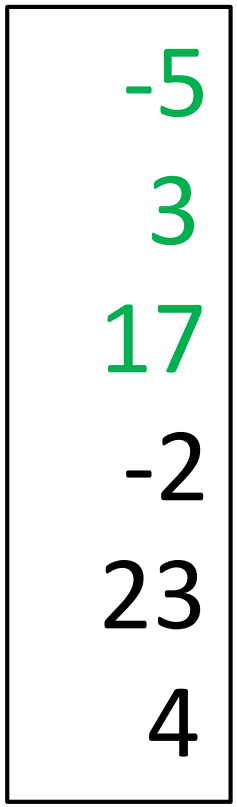


Initial list

Suppose we have sorted elements 0 to k-1

e.g. k = 3



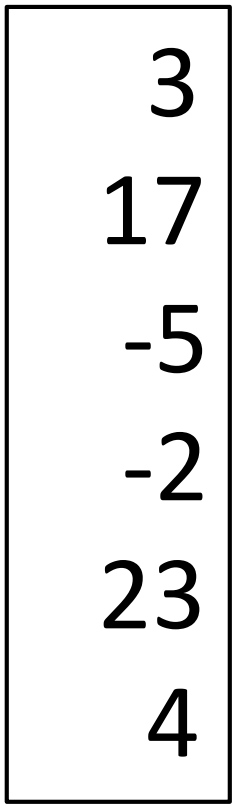


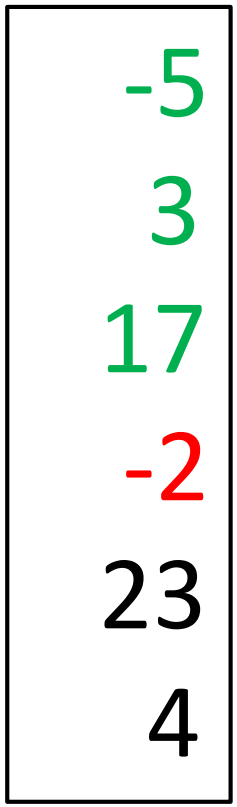
Initial list

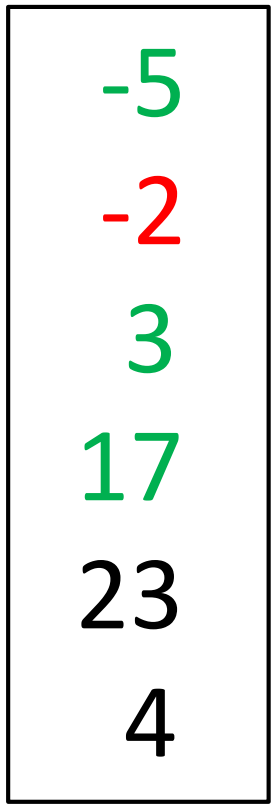
Suppose we have sorted elements 0 to k-1

e.g. k = 3

Insert element k into its correct position with respect to 0 to k-1







Mechanism is similar to inserting (adding) an element to an array list:

Shift all elements ahead by one position to make a hole, and then fill the hole.

Insertion Sort

```
for k = 1 to N - 1   {                  //  index of element to move
     elementK =  list[k]
      i = k
      while (i > 0) and ( elementK  < list[ i - 1]){
          list[i] =  list[i - 1]        // copy to next
          i  = i -1
       }
       list[i] = elementK               //  paste elementK
 }
```

Best case:

the list is already sorted, so it takes $O(N)$ time.
i.e. the while loop terminates immediately.

Worse case:

the list is sorted *in backwards order*.

$$1 + 2 + 3 + \ \ldots + \ \ N - 1 \ = \ \frac{N\ (\ N - 1)}{2}$$

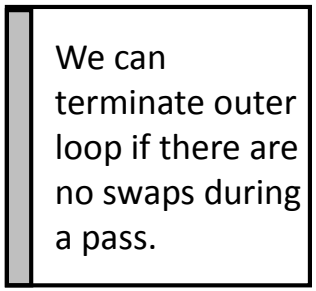
which takes time $O(\ N^2\ )$. Lots of shifts!

Comparison of 3 methods

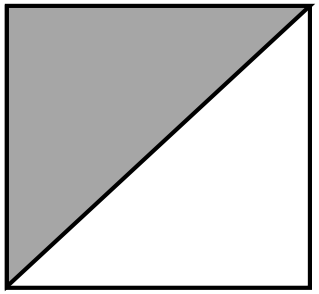
Bubblesort

```
repeat {
   for i = 0 to N – 2 – ct
until continue == false
```

Best case

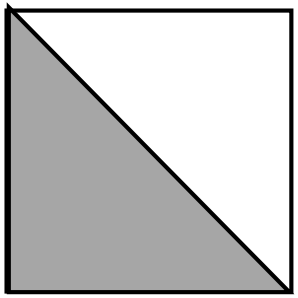


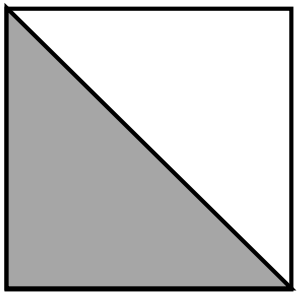
Worst case



Selection sort

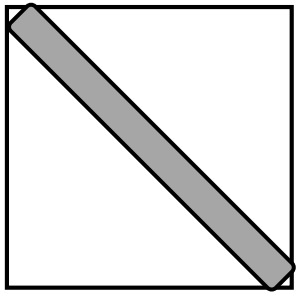
```
for  i = 0  to N-2
    for k = i+1  to  N-1
```

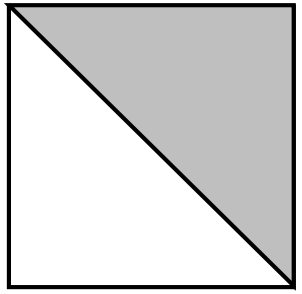




Insertion sort

```
for k = 1 to N - 1   {
     while ….
```





Performance depends highly on initial data. Also, it depends on implementation (array vs. linked list), e.g. what is cost of swap and 'shift'.

Assignment 1 division question: hint

```
      5 …
   ____________________
723 | 41672542996
      3615
      ----
       552  ...etc
```

You need to rethink what you are doing. Don't just try to blindly code what you learned in grade school.

Quiz 1 on Monday on mycourses

8 AM to 8 PM

No discussion during that time.

Email me if there is a problem.

Solutions, grades, feedback will be posted after.