COMP 250

Lecture 6

bubble sort
selection sort
insertion sort

Sept. 19, 2016

Sorting

BEFORE	AFTER
3	-5
17	-2
-5	3
-2	4
23	17
4	23

Example: sorting exams by last name

Sorting Algorithms

- Bubble sort
- Selection sort
- Insertion sort

today $O(N^2)$

- Mergesort
- Heapsort
- Quicksort

later $O(N \log N)$

Bubble Sort

Loop (iterate) through the list many times.

For each iteration through the list,

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
```

Bubble Sort

```
for  ct = 1 to N-1 {   //  a counter, not an index

    for i = 0 to N-2-ct        //  ask  why is there "ct" here?
        if  list[ i ] > list[ i + 1 ]
            swap( list[ i ], list[ i + 1 ] )
}
```

Example: first pass (counter = 1)

3
17
-5
-2
23
4

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

Example: first pass (counter = 1)

3
17
-5
-2
23
4

3
17
-5
-2
23
4

Indicates elements get swapped

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

Example: first pass (counter = 1)

3	3	3	3	3	3
17	17	-5	-5	-5	-5
-5	-5	17	-2	-2	-2
-2	-2	-2	17	17	17
23	23	23	23	23	4
4	4	4	4	4	23

Indicates elements get swapped

```
for  ct = 1 to N-1 {   //  a counter, not an index

    for i = 0 to N-2 - ct
        if  list[ i ] > list[ i + 1 ]
            swap( list[ i ], list[ i + 1 ] )
}
```

Q: How many times is the inner loop executed in total ?

Q: How to improve the algorithm ?

Q: Does the algorithm require that an array list, or could it be (efficiently) implemented with a (singly or doubly) linked list ?

Selection Sort

Partition the list into two parts: (1) a sorted list and (2) the rest of the elements, as follows:

The sorted list is initially empty. So all elements are in the rest.

Repeat N times {

find the smallest element in the rest list, and

add it to the end i.e. tail of the sorted list}

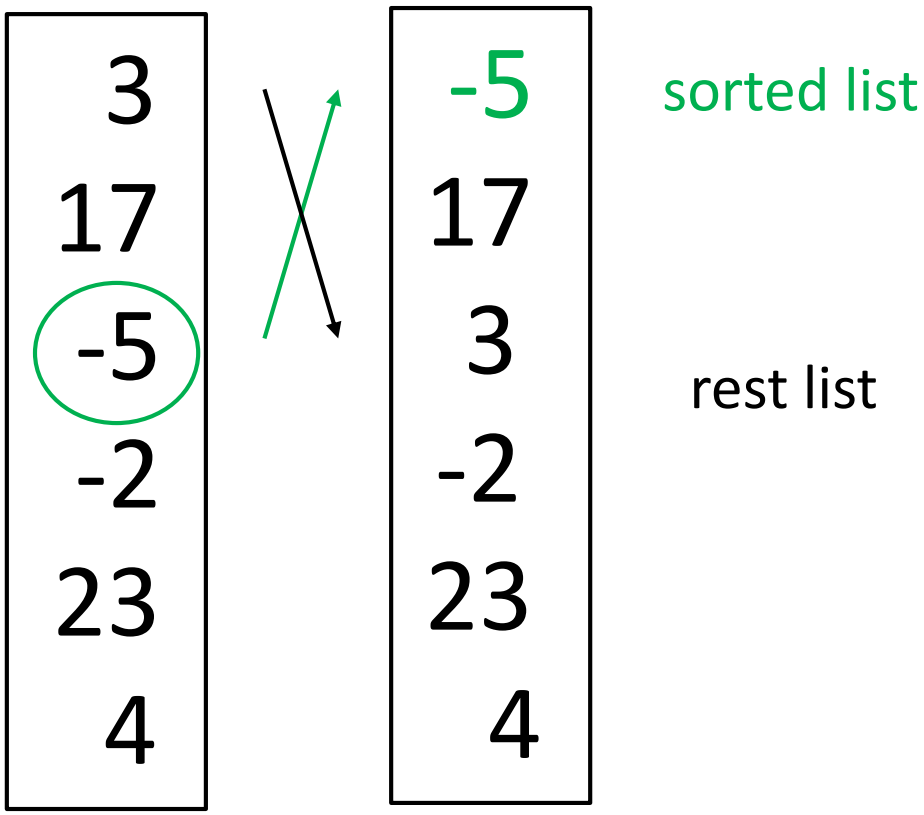
Example

3
17
-5
-2
23
4

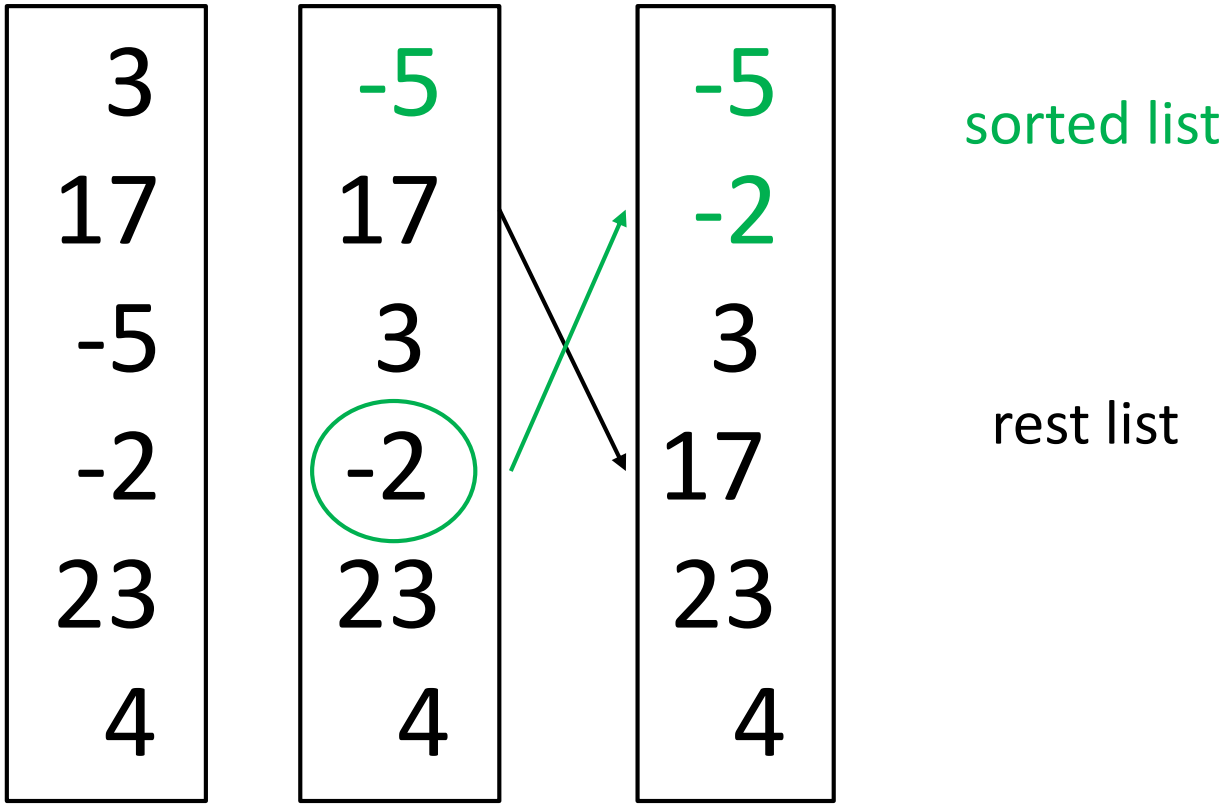
sorted list (empty)

rest list

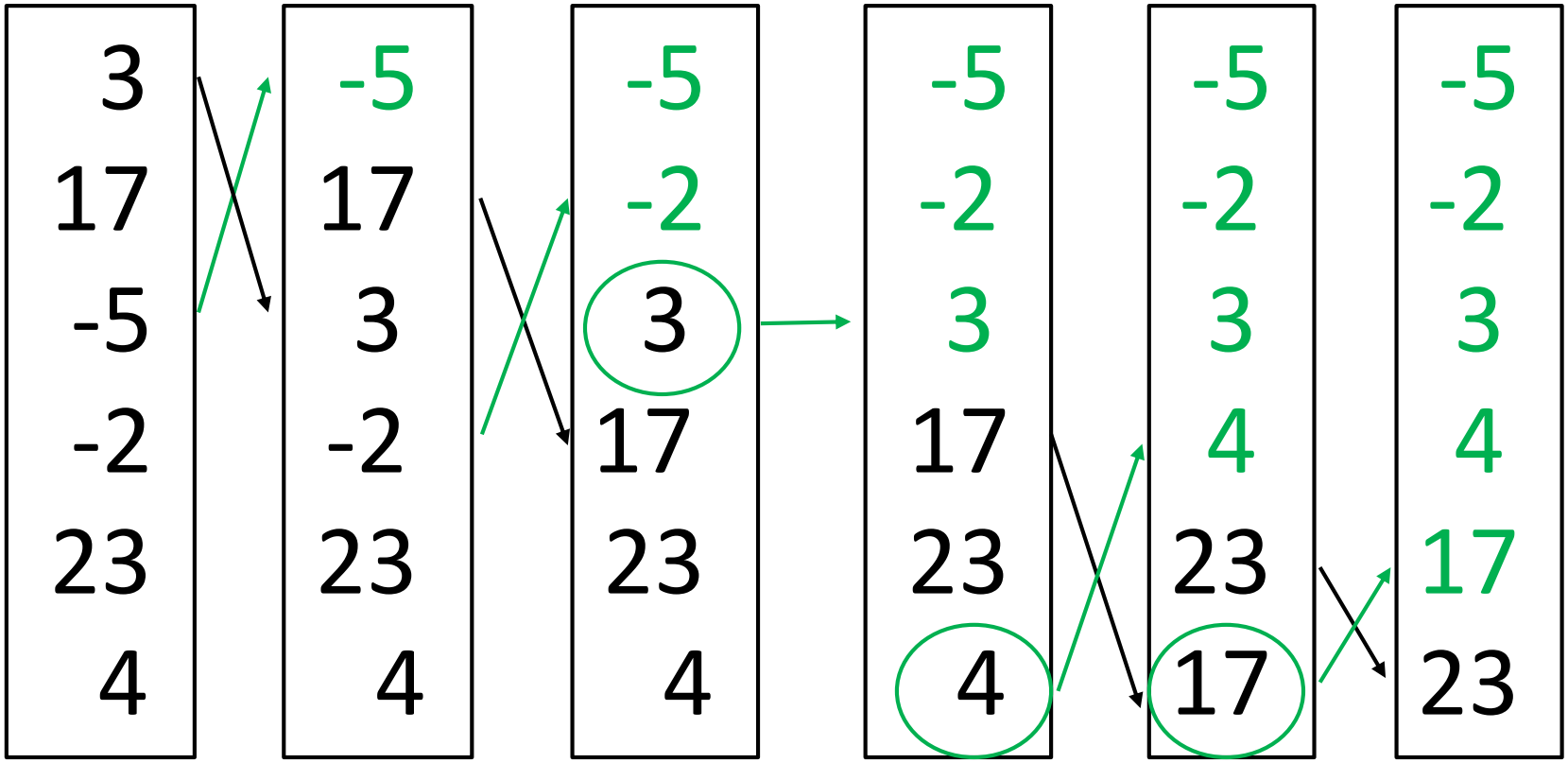
Example



Example



Example



Selection Sort

```
for  i = 0  to N-2   {                          //   repeat  N-1 times
    tmpIndex = i                                //   Take the first element in the rest.
    tmpMinValue = list[ i ]                     //    Let it be the tmp minimum.

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

Selection Sort

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

Q: how many passes through inner loop?

A: $N-1 + N-2 + N-3 + \ldots + 2 + 1$

$= N(N-1)/2$

Comparison

Bubblesort

```
for  ct = 1 to N-1
   for  i = 0 to N-2-ct
```

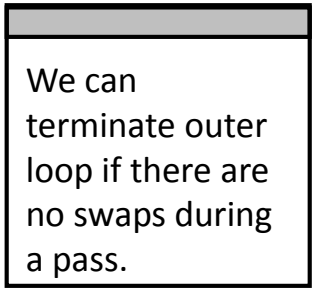
Selection sort

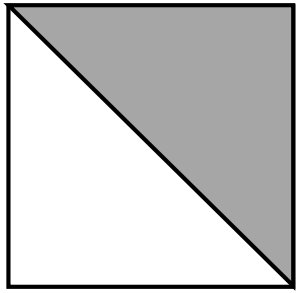
```
for  i = 0  to N-2
    for k = i+1  to  N-1
```

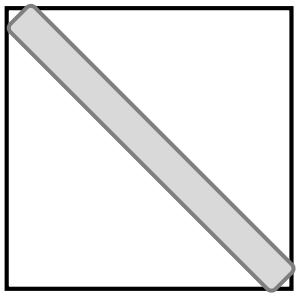
Insertion sort

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

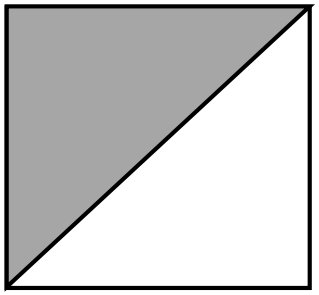
Best case

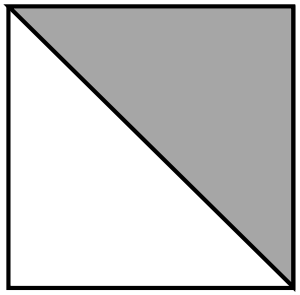


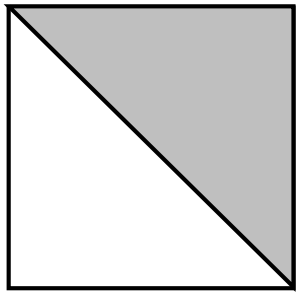




Worst case







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

3
17
-5
-2
23
4

Suppose we have sorted elements 0 to k-1

e.g. k = 3

-5
3
17
-2
23
4

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

-5
-2
3
17
23
4

Insertion Sort

```
for k = 1 to N - 1   {
    elementK =  list[k]
     i = k
     while (i > 0) and (list[ i - 1] > elementK ){
         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

```
for  ct = 1 to N-1
      for  i = 0 to N-2
```

Selection sort

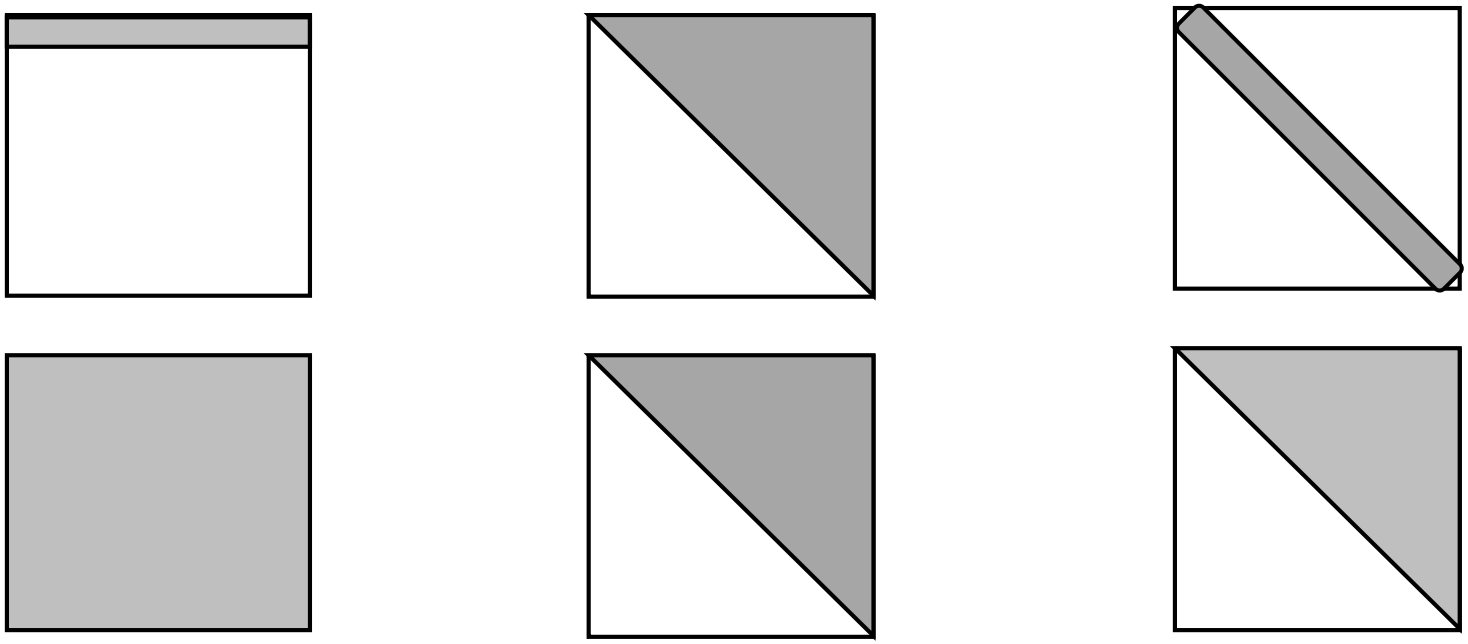
```
for  i = 0  to N-2
    for k = i+1  to  N-1
```

Insertion sort

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

Best case

Worst case



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

Eclipse Tutorials

Monday Sept 19 2:00-3:00 (Pierre & Victor)

Tuesday Sept 20 3:00-4:00 (Ben & Rohit)

Assignment 1 TA Office hours

Monday 3:00-5:00 in Trottier 3104 (Pierre)

Tuesday 4:00-6:00 in Trottier 3104 (Rohit)

Thursday 10:30-12:30 in Trottier 3104 (Victor)

Friday 2:30-4:30 in Trottier 3104 (Ben)

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.