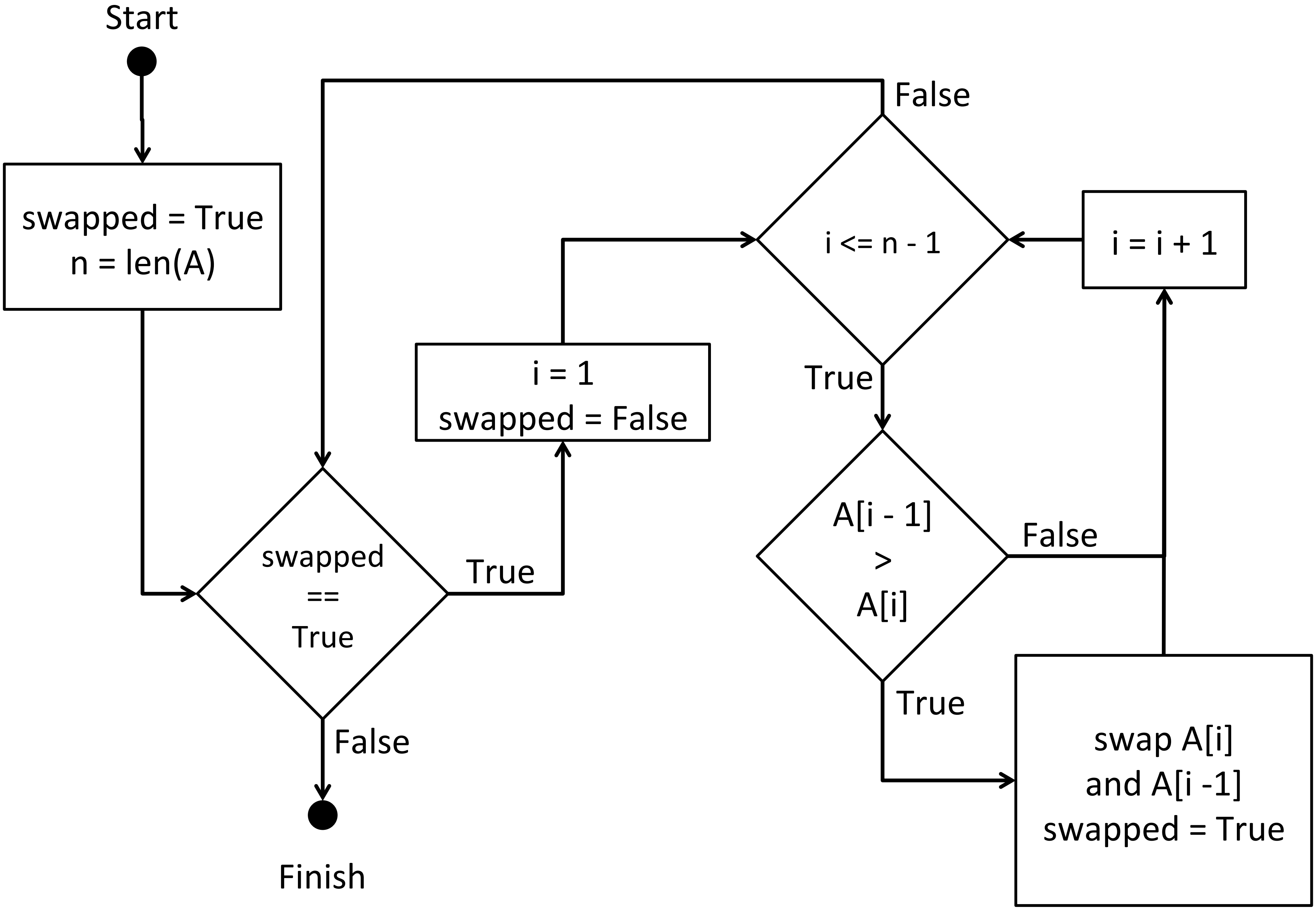
COMP10020
Introduction to Programming II
Simple Sorting

Dr. Brian Mac Namee
brian.macnamee@ucd.ie
School of Computer Science
University College Dublin

BUBBLE SORT

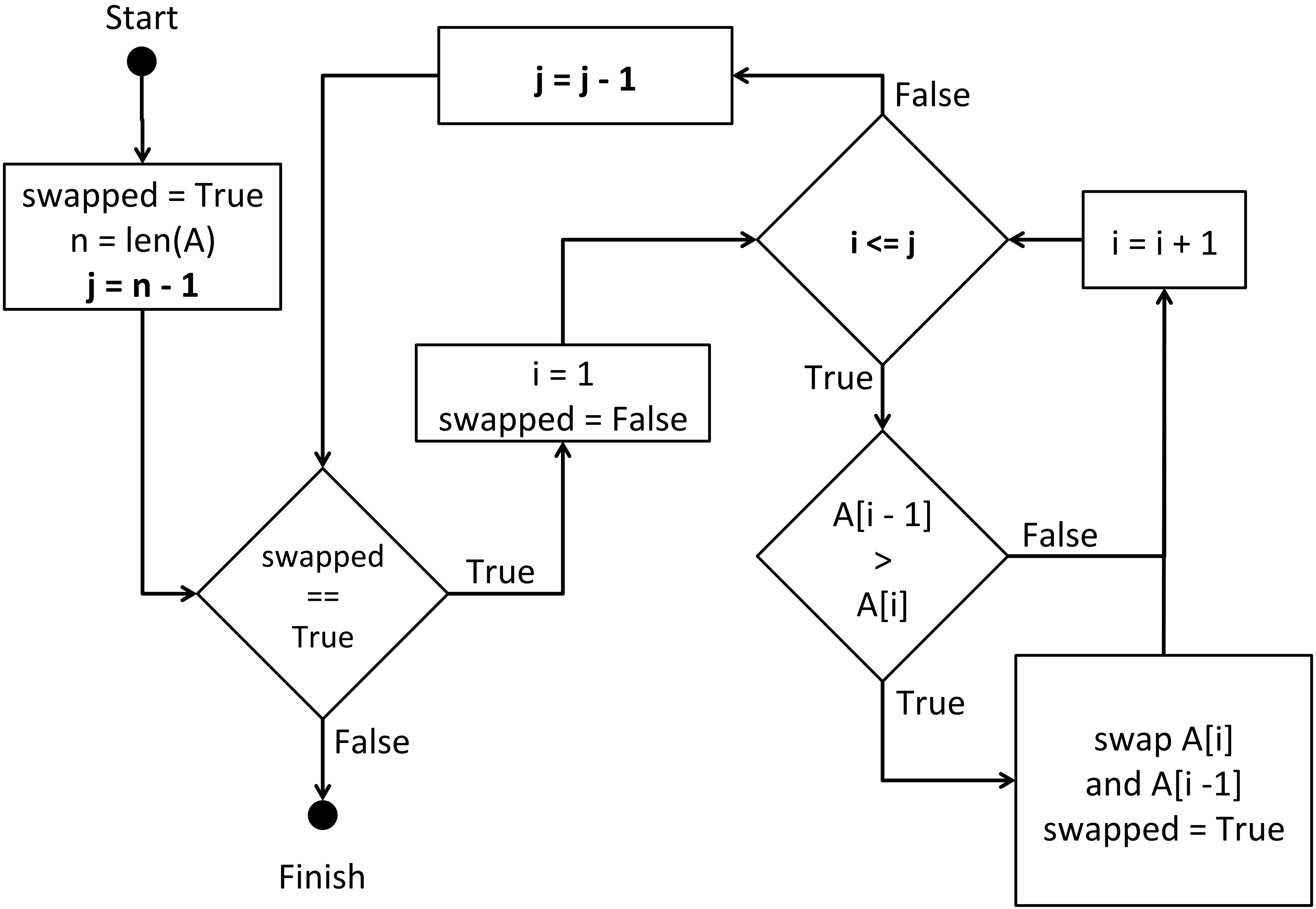
Bubble Sort Pseudocode

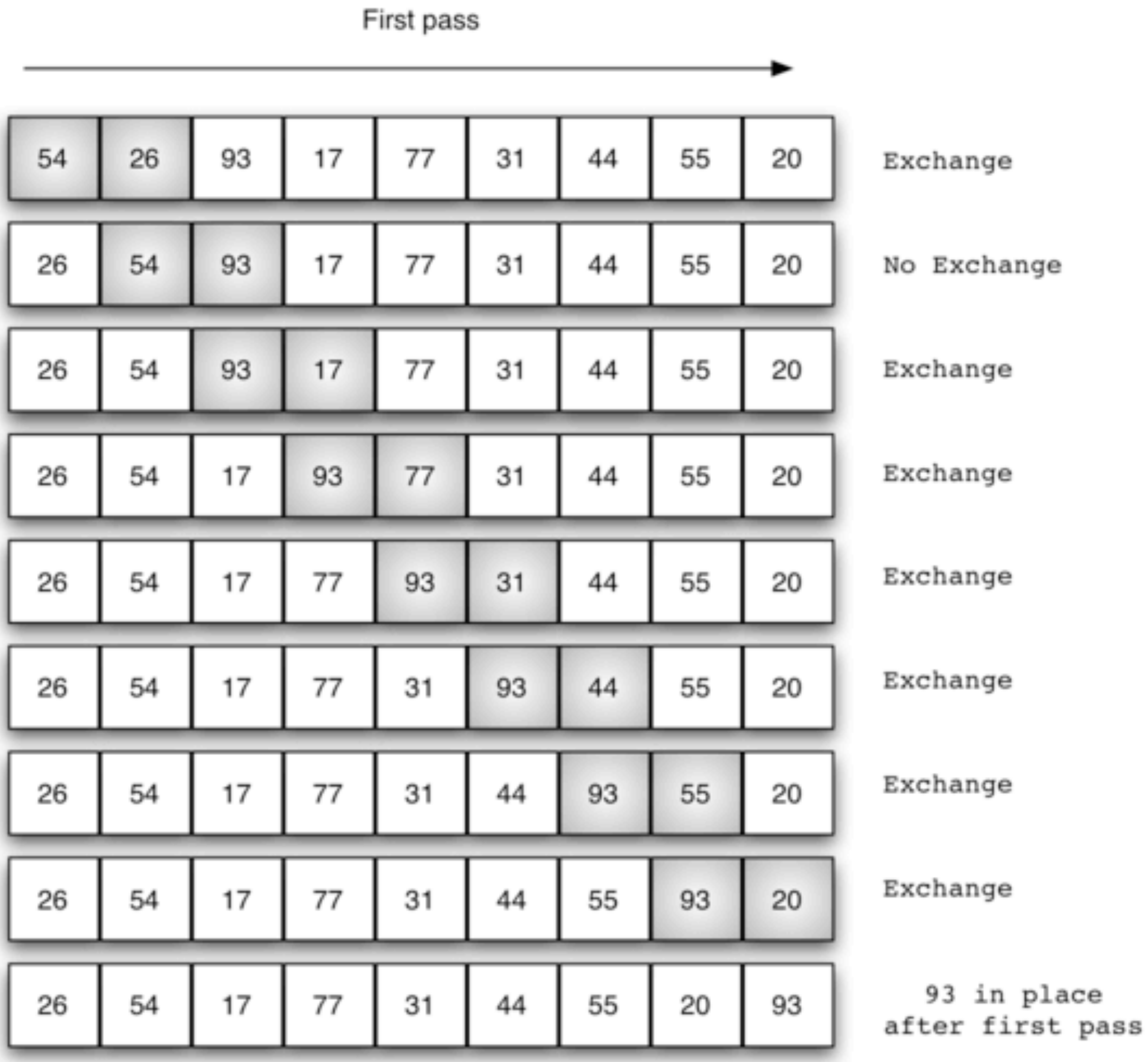
```
bubbleSort( A : list of sortable items )
    n = length(A)
    swapped = true
    while swapped == true
        swapped = false
        for i = 1 to n-1
            if A[i-1] > A[i]
                swap A[i-1] and A[i]
                swapped = true
```


Start
swapped = True
n = len(A)
swapped
==
True
False
Finish
True
i = 1
swapped = False
False
i <= n - 1
True
i = i + 1
A[i - 1]
>
A[i]
False
True
swap A[i]
and A[i -1]
swapped = True

Bubble Sort (Improved) Pseudocode

```
bubbleSort( A : list of sortable items )
    n = length(A)
    swapped = true
    j = n - 1
    while swapped == true
        swapped = false
        for i = 1 to j
            if A[i-1] > A[i]
                swap A[i-1] and A[i]
                swapped = true
        j = j - 1
```


Start
swapped = True
n = len(A)
j = n - 1
swapped
==
True
False
Finish
True
i = 1
swapped = False
j = j - 1
False
i <= j
True
i = i + 1
A[i - 1]
>
A[i]
False
True
swap A[i]
and A[i -1]
swapped = True

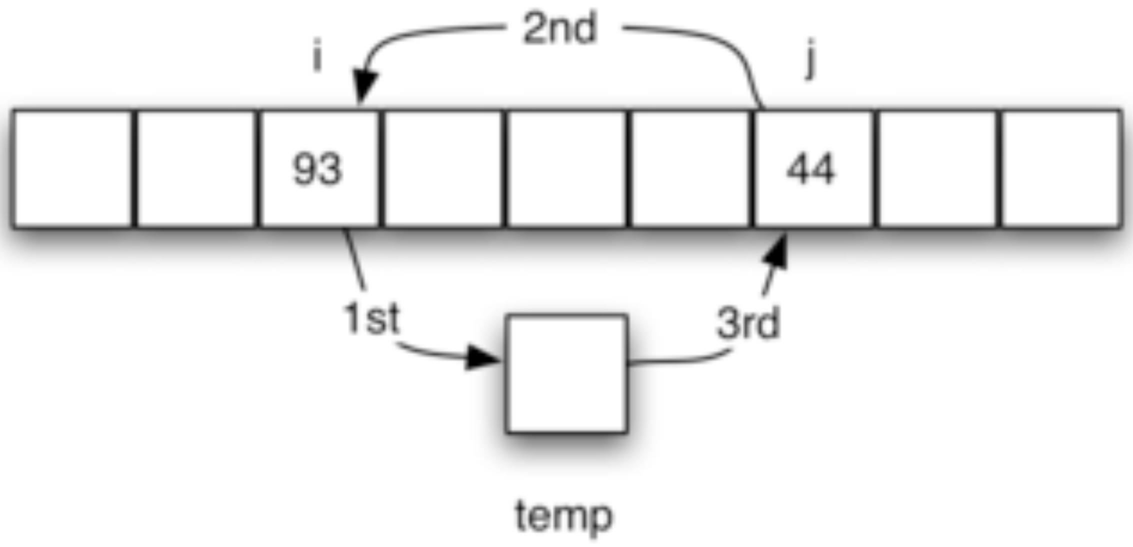


First pass

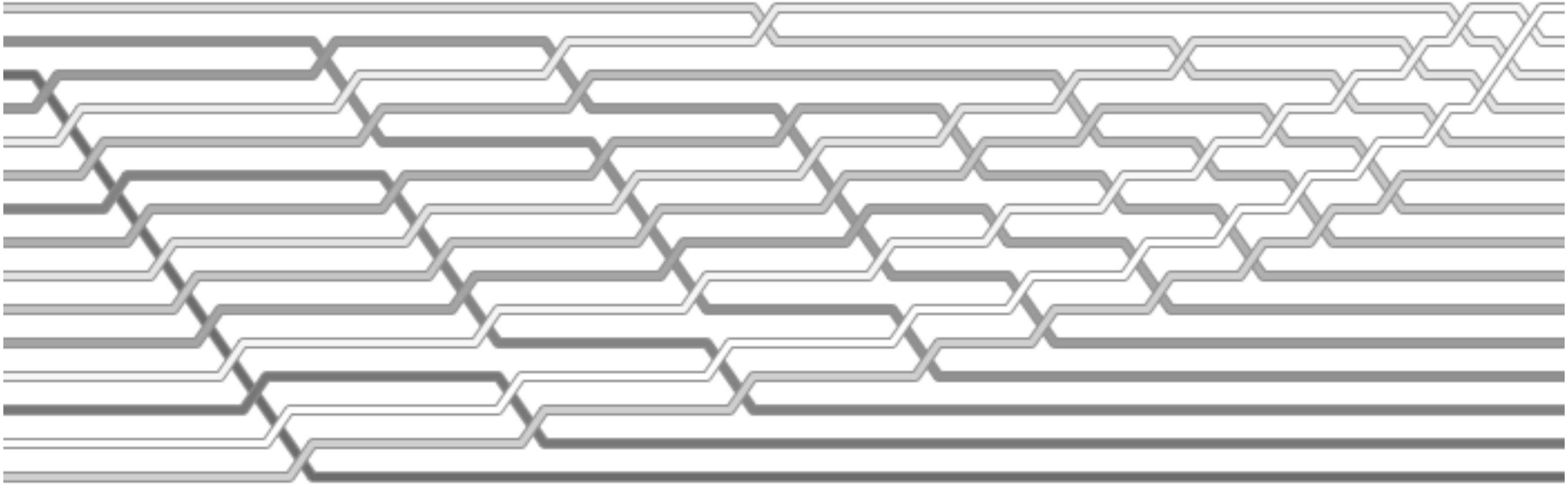
54	26	93	17	77	31	44	55	20	Exchange
26	54	93	17	77	31	44	55	20	No Exchange
26	54	93	17	77	31	44	55	20	Exchange
26	54	17	93	77	31	44	55	20	Exchange
26	54	17	77	93	31	44	55	20	Exchange
26	54	17	77	31	93	44	55	20	Exchange
26	54	17	77	31	44	93	55	20	Exchange
26	54	17	77	31	44	55	93	20	Exchange
26	54	17	77	31	44	55	20	93	93 in place after first pass

http://interactivepython.org/runestone/static/pythonds/SortSearch/TheInsertionSort.html

Most programming languages require a 3-step process with an extra storage location.



http://interactivepython.org/runestone/static/pythonds/SortSearch/TheInsertionSort.html

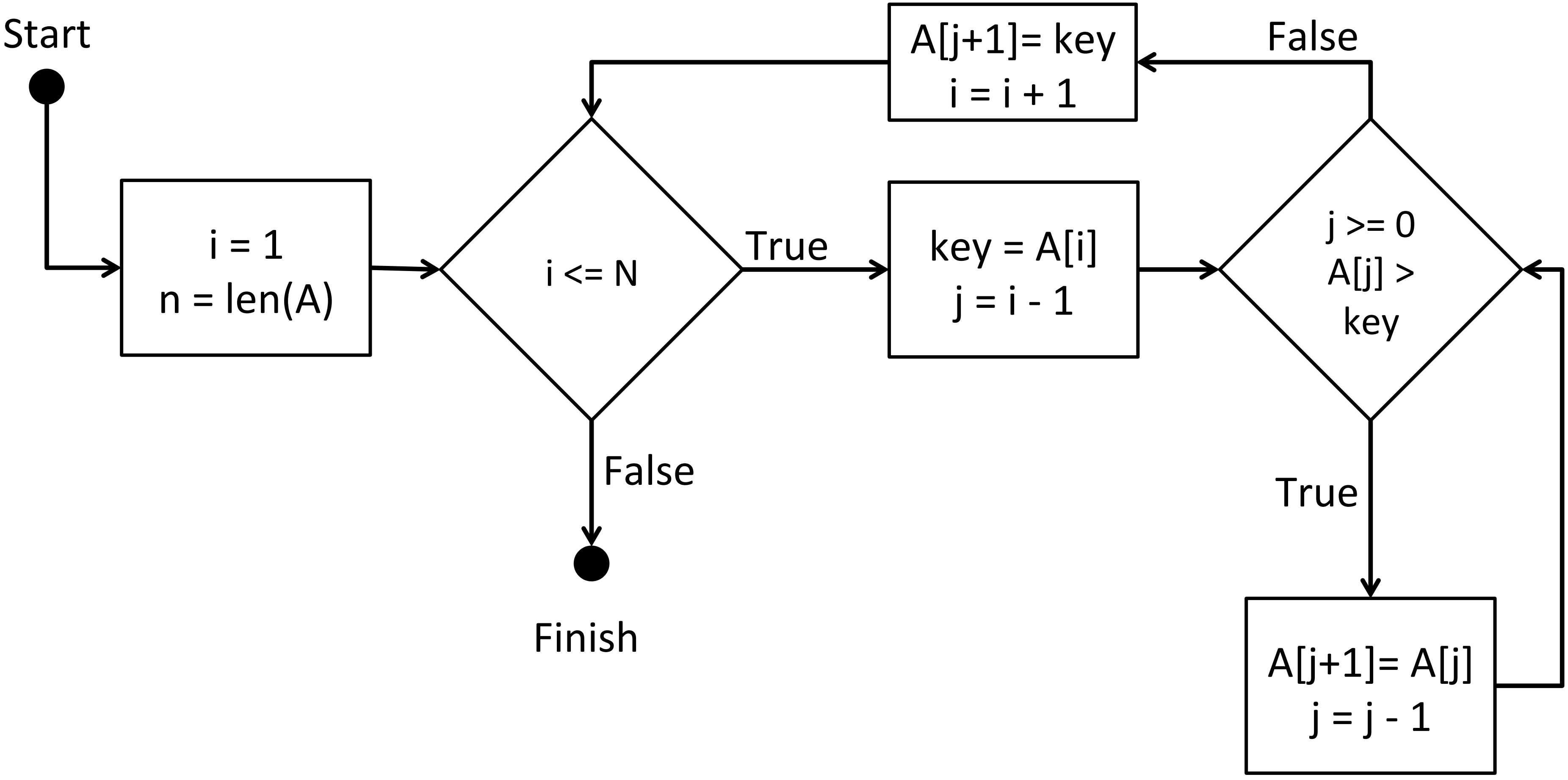


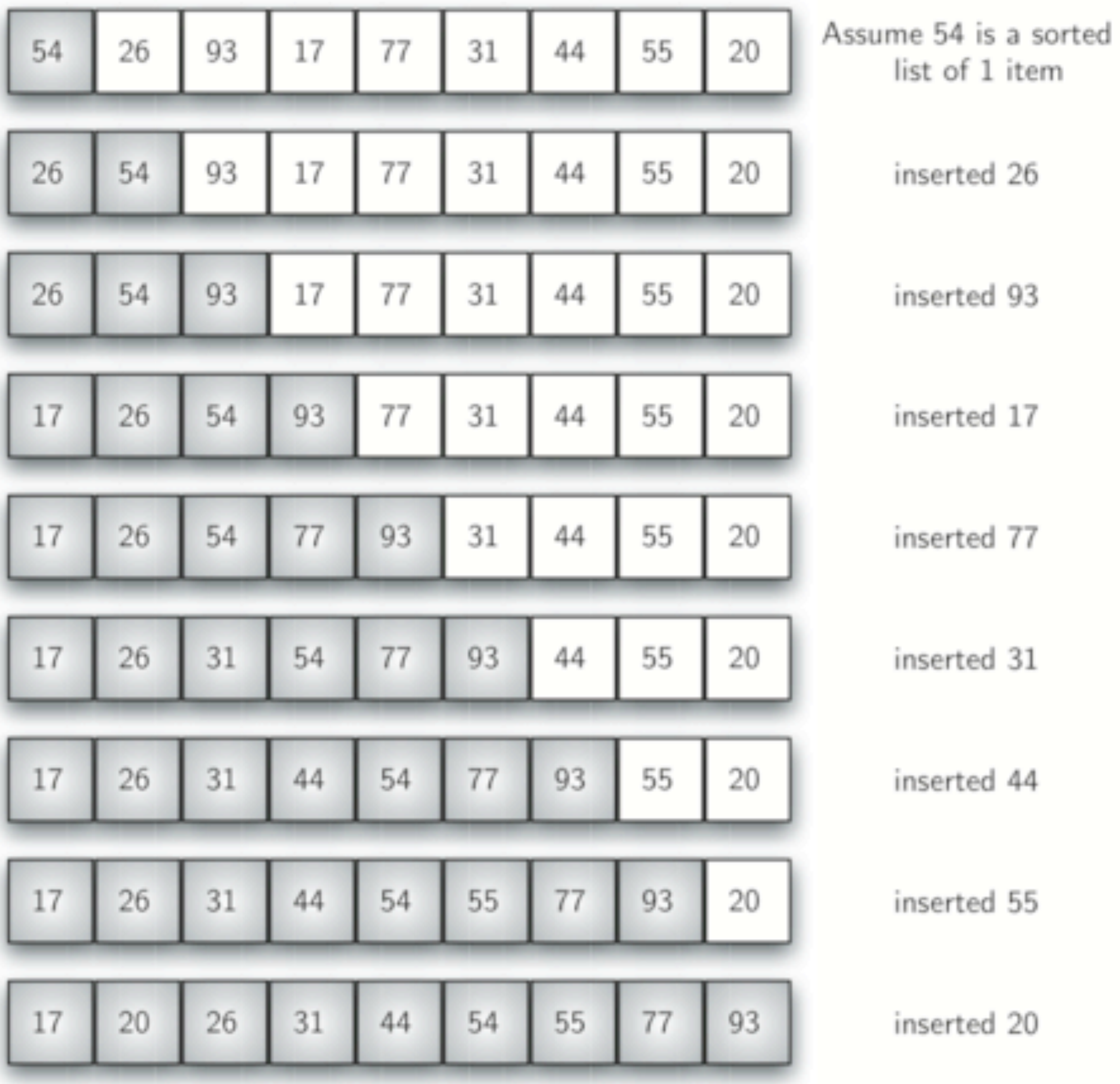
http://corte.si/posts/code/visualisingsorting/index.html

INSERTION SORT

Insertion Sort Pseudocode

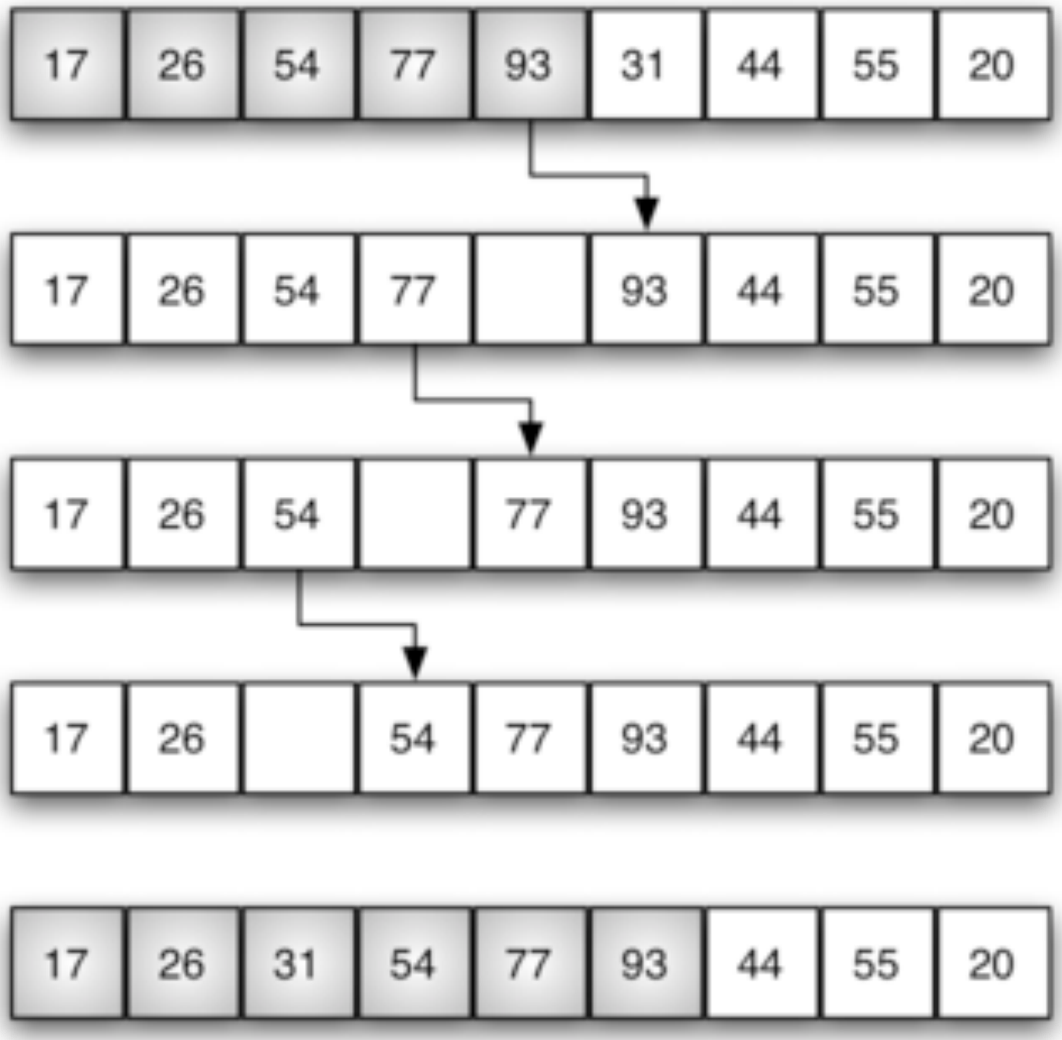
```
insertionSort( A : list of sortable items )

n = length(A)

    for i from 1 to n

        key = A[ i ]
        j = i - 1

        while j >= 0 and A[ j ] > key
            A[ j+1 ] = A[ j ]
            j = j - 1

        A[j+1] = key
```


Start
i = 1
n = len(A)
i <= N
True
key = A[i]
j = i - 1
j >= 0
A[j] >
key
False
A[j+1]= key
i = i + 1
True
A[j+1]= A[j]
j = j - 1
False
Finish



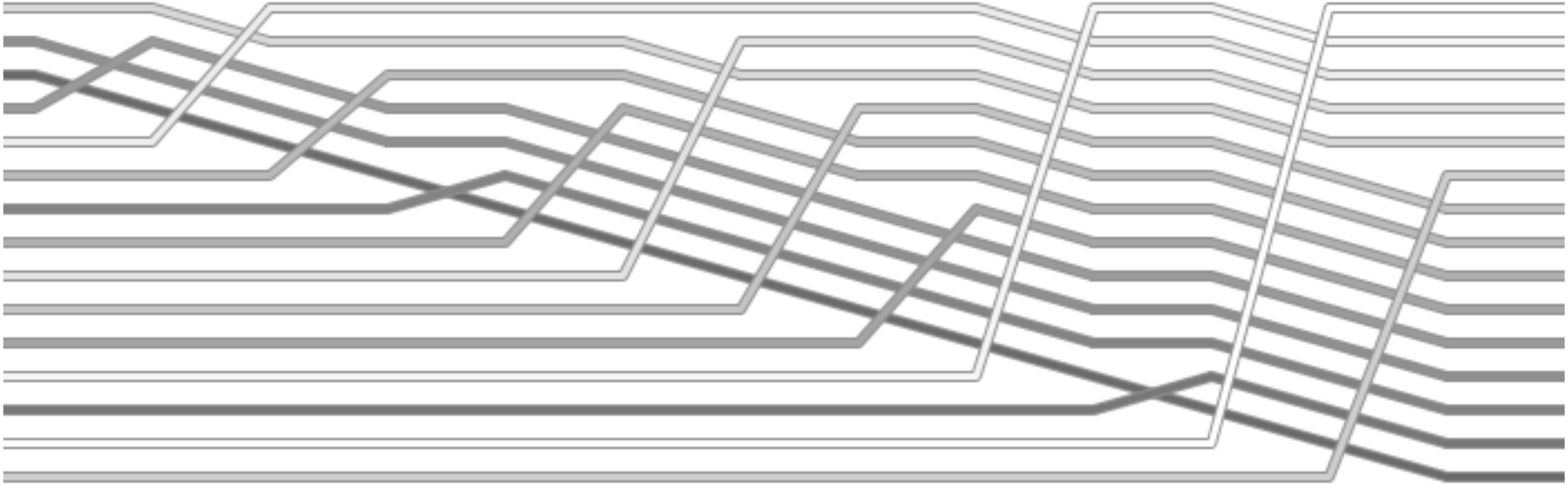
54	26	93	17	77	31	44	55	20	Assume 54 is a sorted list of 1 item
26	54	93	17	77	31	44	55	20	inserted 26
26	54	93	17	77	31	44	55	20	inserted 93
17	26	54	93	77	31	44	55	20	inserted 17
17	26	54	77	93	31	44	55	20	inserted 77
17	26	31	54	77	93	44	55	20	inserted 31
17	26	31	44	54	77	93	55	20	inserted 44
17	26	31	44	54	55	77	93	20	inserted 55
17	20	26	31	44	54	55	77	93	inserted 20

http://interactivepython.org/runestone/static/pythonds/SortSearch/TheInsertionSort.html



17	26	54	77	93	31	44	55	20	Need to insert 31 back into the sorted list
17	26	54	77		93	44	55	20	93>31 so shift it to the right
17	26	54		77	93	44	55	20	77>31 so shift it to the right
17	26		54	77	93	44	55	20	54>31 so shift it to the right
17	26	31	54	77	93	44	55	20	26<31 so insert 31 in this position

http://interactivepython.org/runestone/static/pythonds/SortSearch/TheInsertionSort.html



http://corte.si/posts/code/visualisingsorting/index.html

SUMMARY

Summary

Sorting algorithms are a great way to start thinking about moving from simply **writing code** to **solving problems with code**