Sorting

… and Insertion Sort

Sorting Techniques

- Ideally, when we set up a new data collection, we ensure that it is (somehow) ordered:
  - Initially easy if it is empty or contains one item
  - Each addition ensures that the ordering is preserved (e.g. by inserting in the correct place)
- We also need a sorting technique to put jumbled data into order
- The data may be completely randomly ordered, or we may know that some order already exists
  - Different sorting algorithms may perform better in different circumstances

Sorting Algorithms

- Problem: Given a sequence of N values, rearrange them so that they are in non-decreasing order.
  - E.g. ascending numerical order, or alphabetical order
  - 'non-decreasing' allows for repeat/duplicate values
  - For our examples, we restrict ourselves to arrays of numbers
- Algorithms for sorting lists:
  - "Naïve": Bubble sort, Selection sort
  - Cleverer: Quick sort
  - There are many others: Insert sort, Merge sort,...
- We'll look at a few of these,
  - and understand their complexity analysis

Things to keep in mind…

- The data to be sorted may vary in type and be simple or more complex objects
  - The sorting techniques remain the same.
- The result of sorting is simply the rearranged list
  - No value is "returned", and no exception can be thrown
- As before we will assume:
  - The data is `size` integers, in elements indexed `0` to `(size-1)` of array `numbers`

Insertion Sort

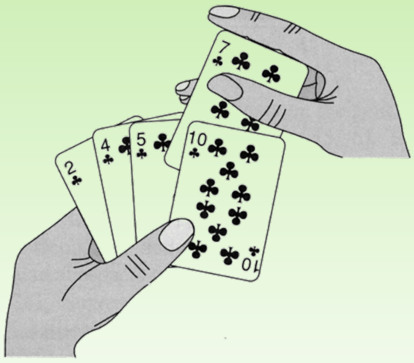


Figure 2.1 Sorting a hand of cards using insertion sort.

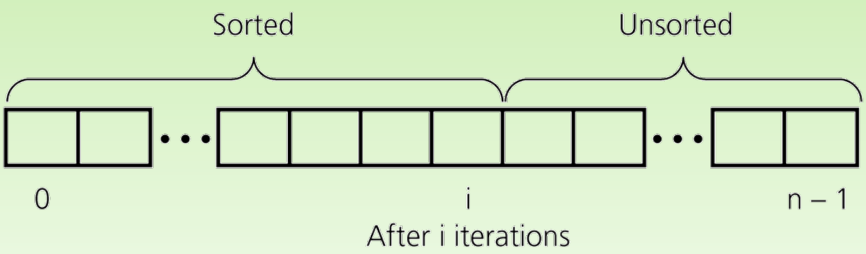
Insertion Sort

- while some elements unsorted:
  - Using linear search, find the location in the sorted portion where the 1st element of the unsorted portion should be inserted
  - Move all the elements after the insertion location up one position to make space for the new element

			45												
38	60	66		79	47	13	74	36	21	94	22	57	16	29	81

`the fourth iteration of this loop is shown here`

38	45	60	66	79	47	13	74	36	21	94	22	57	16	29	81



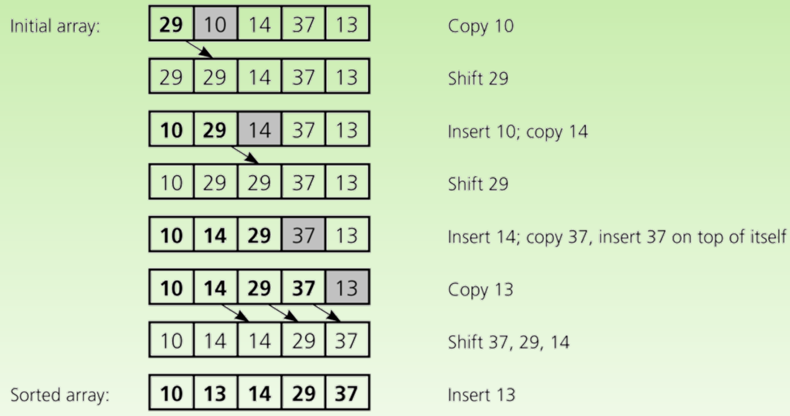
An insertion sort partitions the array into two regions; variable i referred to as pivot.

Algorithm

```
Input: An array ‘A’ of n comparable items
Output: The array ‘A’ with elements in non-decreasing
order
InsertionSort(A)
   for i←1 to n-1 do
       //Insert A[i] at is proper location in A[0]…A[i-1].
       pivot ← A[i]
       j ← i-1

       While j >= 0 and A[j] > pivot do
           A[j+1] ← A[j]
           j ← j - 1

       A[j+1] ← pivot
```



An insertion sort of an array of five integers

Insertion Sort Demo: Another perspective

Sorting problem (recall):

- Given an array of N values, rearrange them so that they are in increasing order.

Insertion sort (general idea)

- Brute-force sorting solution.
- Move left-to-right through array.
- Exchange next element with larger elements to its left, one-by-one.

B	R	U	T	E	F	O	R	C	E

Insertion Sort Demo

Sorting problem:

- Given an array of N values, rearrange them so that they are in increasing order.

Insertion sort

- Brute-force sorting solution.
- Move left-to-right through array.
- Exchange next element with larger elements to its left, one-by-one.

B	R	U	T	E	F	O	R	C	E

unsorted active sorted

Insertion Sort Demo

Sorting problem:

- Given an array of N values, rearrange them so that they are in increasing order.

Insertion sort

- Brute-force sorting solution.
- Move left-to-right through array.
- Exchange next element with larger elements to its left, one-by-one.

B	R	U	T	E	F	O	R	C	E

unsorted active sorted

Insertion Sort Demo

Sorting problem:

- Given an array of N values, rearrange them so that they are in increasing order.

Insertion sort

- Brute-force sorting solution.
- Move left-to-right through array.
- Exchange next element with larger elements to its left, one-by-one.

B	R	U	T	E	F	O	R	C	E

unsorted active sorted

Insertion Sort Demo

Sorting problem:

- Given an array of N values, rearrange them so that they are in increasing order.

Insertion sort

- Brute-force sorting solution.
- Move left-to-right through array.
- Exchange next element with larger elements to its left, one-by-one.

B	R	U	T	E	F	O	R	C	E

unsorted active sorted

Insertion Sort Demo

Sorting problem:

- Given an array of N values, rearrange them so that they are in increasing order.

Insertion sort

- Brute-force sorting solution.
- Move left-to-right through array.
- Exchange next element with larger elements to its left, one-by-one.

B	R	U	T	E	F	O	R	C	E

unsorted active sorted

Insertion Sort Demo

Sorting problem:

- Given an array of N values, rearrange them so that they are in increasing order.

Insertion sort

- Brute-force sorting solution.
- Move left-to-right through array.
- Exchange next element with larger elements to its left, one-by-one.

B	R	U	T	E	F	O	R	C	E

unsorted active sorted

Insertion Sort Demo

Sorting problem:

- Given an array of N values, rearrange them so that they are in increasing order.

Insertion sort

- Brute-force sorting solution.
- Move left-to-right through array.
- Exchange next element with larger elements to its left, one-by-one.

B	R	U	T	E	F	O	R	C	E

unsorted active sorted

Insertion Sort Demo

Sorting problem:

- Given an array of N values, rearrange them so that they are in increasing order.

Insertion sort

- Brute-force sorting solution.
- Move left-to-right through array.
- Exchange next element with larger elements to its left, one-by-one.

B	R	T	U	E	F	O	R	C	E

unsorted active sorted

Insertion Sort Demo

Sorting problem:

- Given an array of N values, rearrange them so that they are in increasing order.

Insertion sort

- Brute-force sorting solution.
- Move left-to-right through array.
- Exchange next element with larger elements to its left, one-by-one.

B	R	T	U	E	F	O	R	C	E

unsorted active sorted

Insertion Sort Demo

Sorting problem:

- Given an array of N values, rearrange them so that they are in increasing order.

Insertion sort

- Brute-force sorting solution.
- Move left-to-right through array.
- Exchange next element with larger elements to its left, one-by-one.

B	R	T	U	E	F	O	R	C	E

unsorted active sorted

Insertion Sort Demo

Sorting problem:

- Given an array of N values, rearrange them so that they are in increasing order.

Insertion sort

- Brute-force sorting solution.
- Move left-to-right through array.
- Exchange next element with larger elements to its left, one-by-one.

B	R	T	E	U	F	O	R	C	E

unsorted active sorted

Insertion Sort Demo

Sorting problem:

- Given an array of N values, rearrange them so that they are in increasing order.

Insertion sort

- Brute-force sorting solution.
- Move left-to-right through array.
- Exchange next element with larger elements to its left, one-by-one.

B	R	E	T	U	F	O	R	C	E

unsorted active sorted

Insertion Sort Demo

Sorting problem:

- Given an array of N values, rearrange them so that they are in increasing order.

Insertion sort

- Brute-force sorting solution.
- Move left-to-right through array.
- Exchange next element with larger elements to its left, one-by-one.

B	E	R	T	U	F	O	R	C	E

unsorted active sorted

Insertion Sort Demo

Sorting problem:

- Given an array of N values, rearrange them so that they are in increasing order.

Insertion sort

- Brute-force sorting solution.
- Move left-to-right through array.
- Exchange next element with larger elements to its left, one-by-one.

B	E	R	T	U	F	O	R	C	E

unsorted active sorted

Insertion Sort Demo

Sorting problem:

- Given an array of N values, rearrange them so that they are in increasing order.

Insertion sort

- Brute-force sorting solution.
- Move left-to-right through array.
- Exchange next element with larger elements to its left, one-by-one.

B	E	R	T	U	F	O	R	C	E

unsorted active sorted

Insertion Sort Demo

Sorting problem:

- Given an array of N values, rearrange them so that they are in increasing order.

Insertion sort

- Brute-force sorting solution.
- Move left-to-right through array.
- Exchange next element with larger elements to its left, one-by-one.

B	E	R	T	F	U	O	R	C	E

unsorted active sorted

Insertion Sort Demo

Sorting problem:

- Given an array of N values, rearrange them so that they are in increasing order.

Insertion sort

- Brute-force sorting solution.
- Move left-to-right through array.
- Exchange next element with larger elements to its left, one-by-one.

B	E	R	F	T	U	O	R	C	E

unsorted active sorted

Insertion Sort Demo

Sorting problem:

- Given an array of N values, rearrange them so that they are in increasing order.

Insertion sort

- Brute-force sorting solution.
- Move left-to-right through array.
- Exchange next element with larger elements to its left, one-by-one.

B	E	F	R	T	U	O	R	C	E

unsorted active sorted

Insertion Sort Demo

Sorting problem:

- **Given an array of N values, rearrange them so that they are in increasing order.**

Insertion sort

- **Brute-force sorting solution.**
- **Move left-to-right through array.**
- **Exchange next element with larger elements to its left, one-by-one.**

B	E	F	R	T	U	O	R	C	E

unsorted active sorted

Insertion Sort Demo

Sorting problem:

- **Given an array of N values, rearrange them so that they are in increasing order.**

Insertion sort

- **Brute-force sorting solution.**
- **Move left-to-right through array.**
- **Exchange next element with larger elements to its left, one-by-one.**

B	E	F	R	T	U	O	R	C	E

unsorted active sorted

Insertion Sort Demo

Sorting problem:

- Given an array of N values, rearrange them so that they are in increasing order.

Insertion sort

- Brute-force sorting solution.
- Move left-to-right through array.
- Exchange next element with larger elements to its left, one-by-one.

B	E	F	R	T	O	U	R	C	E

unsorted active sorted

Insertion Sort Demo

Sorting problem:

- Given an array of N values, rearrange them so that they are in increasing order.

Insertion sort

- Brute-force sorting solution.
- Move left-to-right through array.
- Exchange next element with larger elements to its left, one-by-one.

B	E	F	R	O	T	U	R	C	E

unsorted active sorted

Insertion Sort Demo

Sorting problem:

- Given an array of N values, rearrange them so that they are in increasing order.

Insertion sort

- Brute-force sorting solution.
- Move left-to-right through array.
- Exchange next element with larger elements to its left, one-by-one.

B	E	F	O	R	T	U	R	C	E

unsorted active sorted

Insertion Sort Demo

Sorting problem:

- Given an array of N values, rearrange them so that they are in increasing order.

Insertion sort

- Brute-force sorting solution.
- Move left-to-right through array.
- Exchange next element with larger elements to its left, one-by-one.

B	E	F	O	R	T	U	R	C	E

unsorted active sorted

Insertion Sort Demo

Sorting problem:

- **Given an array of N values, rearrange them so that they are in increasing order.**

Insertion sort

- **Brute-force sorting solution.**
- **Move left-to-right through array.**
- **Exchange next element with larger elements to its left, one-by-one.**

B	E	F	O	R	T	U	R	C	E

unsorted active sorted

Insertion Sort Demo

Sorting problem:

- **Given an array of N values, rearrange them so that they are in increasing order.**

Insertion sort

- **Brute-force sorting solution.**
- **Move left-to-right through array.**
- **Exchange next element with larger elements to its left, one-by-one.**

B	E	F	O	R	T	R	U	C	E

unsorted active sorted

Insertion Sort Demo

Sorting problem:

- Given an array of N values, rearrange them so that they are in increasing order.

Insertion sort

- Brute-force sorting solution.
- Move left-to-right through array.
- Exchange next element with larger elements to its left, one-by-one.

B	E	F	O	R	R	T	U	C	E

unsorted active sorted

Insertion Sort Demo

Sorting problem:

- Given an array of N values, rearrange them so that they are in increasing order.

Insertion sort

- Brute-force sorting solution.
- Move left-to-right through array.
- Exchange next element with larger elements to its left, one-by-one.

B	E	F	O	R	R	T	U	C	E

unsorted active sorted

Insertion Sort Demo

Sorting problem:
- Given an array of N values, rearrange them so that they are in increasing order.

Insertion sort
- Brute-force sorting solution.
- Move left-to-right through array.
- Exchange next element with larger elements to its left, one-by-one.

B	E	F	O	R	R	T	U	C	E

unsorted active sorted

Insertion Sort Demo

Sorting problem:
- Given an array of N values, rearrange them so that they are in increasing order.

Insertion sort
- Brute-force sorting solution.
- Move left-to-right through array.
- Exchange next element with larger elements to its left, one-by-one.

B	E	F	O	R	R	T	C	U	E

unsorted active sorted

Insertion Sort Demo

Sorting problem:

- **Given an array of N values, rearrange them so that they are in increasing order.**

Insertion sort

- **Brute-force sorting solution.**
- **Move left-to-right through array.**
- **Exchange next element with larger elements to its left, one-by-one.**

B	E	F	O	R	R	C	T	U	E

unsorted active sorted

Insertion Sort Demo

Sorting problem:

- **Given an array of N values, rearrange them so that they are in increasing order.**

Insertion sort

- **Brute-force sorting solution.**
- **Move left-to-right through array.**
- **Exchange next element with larger elements to its left, one-by-one.**

B	E	F	O	R	C	R	T	U	E

unsorted active sorted

Insertion Sort Demo

Sorting problem:

- Given an array of N values, rearrange them so that they are in increasing order.

Insertion sort

- Brute-force sorting solution.
- Move left-to-right through array.
- Exchange next element with larger elements to its left, one-by-one.

B	E	F	O	C	R	R	T	U	E

unsorted   active   sorted

Insertion Sort Demo

Sorting problem:

- Given an array of N values, rearrange them so that they are in increasing order.

Insertion sort

- Brute-force sorting solution.
- Move left-to-right through array.
- Exchange next element with larger elements to its left, one-by-one.

B	E	F	C	O	R	R	T	U	E

unsorted   active   sorted

Insertion Sort Demo

Sorting problem:

- Given an array of N values, rearrange them so that they are in increasing order.

Insertion sort

- Brute-force sorting solution.
- Move left-to-right through array.
- Exchange next element with larger elements to its left, one-by-one.

B	E	C	F	O	R	R	T	U	E

unsorted active sorted

Insertion Sort Demo

Sorting problem:

- Given an array of N values, rearrange them so that they are in increasing order.

Insertion sort

- Brute-force sorting solution.
- Move left-to-right through array.
- Exchange next element with larger elements to its left, one-by-one.

B	C	E	F	O	R	R	T	U	E

unsorted active sorted

Insertion Sort Demo

Sorting problem:

- Given an array of N values, rearrange them so that they are in increasing order.

Insertion sort

- Brute-force sorting solution.
- Move left-to-right through array.
- Exchange next element with larger elements to its left, one-by-one.

B	C	E	F	O	R	R	T	U	E

unsorted active sorted

Insertion Sort Demo

Sorting problem:

- Given an array of N values, rearrange them so that they are in increasing order.

Insertion sort

- Brute-force sorting solution.
- Move left-to-right through array.
- Exchange next element with larger elements to its left, one-by-one.

B	C	E	F	O	R	R	T	U	E

unsorted active sorted

Insertion Sort Demo

Sorting problem:

- **Given an array of N values, rearrange them so that they are in increasing order.**

Insertion sort

- **Brute-force sorting solution.**
- **Move left-to-right through array.**
- **Exchange next element with larger elements to its left, one-by-one.**

B	C	E	F	O	R	R	T	E	U

unsorted active sorted

Insertion Sort Demo

Sorting problem:

- **Given an array of N values, rearrange them so that they are in increasing order.**

Insertion sort

- **Brute-force sorting solution.**
- **Move left-to-right through array.**
- **Exchange next element with larger elements to its left, one-by-one.**

B	C	E	F	O	R	R	E	T	U

unsorted active sorted

Insertion Sort Demo

Sorting problem:

- Given an array of N values, rearrange them so that they are in increasing order.

Insertion sort

- Brute-force sorting solution.
- Move left-to-right through array.
- Exchange next element with larger elements to its left, one-by-one.

B	C	E	F	O	R	E	R	T	U

unsorted active sorted

Insertion Sort Demo

Sorting problem:

- Given an array of N values, rearrange them so that they are in increasing order.

Insertion sort

- Brute-force sorting solution.
- Move left-to-right through array.
- Exchange next element with larger elements to its left, one-by-one.

B	C	E	F	O	E	R	R	T	U

unsorted active sorted

Insertion Sort Demo

Sorting problem:

- Given an array of N values, rearrange them so that they are in increasing order.

Insertion sort

- Brute-force sorting solution.
- Move left-to-right through array.
- Exchange next element with larger elements to its left, one-by-one.

B	C	E	F	E	O	R	R	T	U

unsorted active sorted

Insertion Sort Demo

Sorting problem:

- Given an array of N values, rearrange them so that they are in increasing order.

Insertion sort

- Brute-force sorting solution.
- Move left-to-right through array.
- Exchange next element with larger elements to its left, one-by-one.

B	C	E	E	F	O	R	R	T	U

unsorted active sorted

Insertion Sort Demo

Sorting problem:

- **Given an array of N values, rearrange them so that they are in increasing order.**

Insertion sort

- **Brute-force sorting solution.**
- **Move left-to-right through array.**
- **Exchange next element with larger elements to its left, one-by-one.**

B	C	E	E	F	O	R	R	T	U

unsorted active sorted

Insertion Sort: Number of Comparisons

# of Sorted Elements	Best case	Worst case
0	0	0
1	1	1
2	1	2
…	…	…
n-1	1	n-1
	n-1	**n(n-1)/2**

Remark: we only count comparisons of elements in the array.

Count the steps

```
Input: An array 'A' of n comparable items
Output: The array 'A' with elements in non-decreasing order

InsertionSort(A)
for i←1 to n-1 do                                  ← outer loop
    //Insert A[i] at is proper location in A[0]…A[i-1].

    pivot ← A[i]                                   ← outer steps
    j ← i-1

    While j >= 0 and A[j] > pivot do               ← inner loop
        A[j+1] ← A[j]
        j ← j - 1                                  ← inner steps

    A[j+1] ← pivot                                 ← outer step
```

CSCU9A3 2017 Sorting

Insertion Sort: Cost Function

- 1 operation to initialize the outer loop
- The outer loop is evaluated *n-1* times
  - 5 instructions (including outer loop comparison and increment)
  - Total cost of the outer loop: 5(n-1)
- How many times the inner loop is evaluated is affected by the state of the array to be sorted
- Best case: the array is already completely sorted, so no "shifting" of array elements is required.
  - We only test the condition of the inner loop once (2 operations = 1 comparison + 1 element comparison), and the body is never executed
  - Requires 2(n-1) operations, ie. **O(n)**.

CSCU9A3 2017 Sorting

Insertion Sort: Cost Function

- Worst case: the array is sorted in reverse order (so each item has to be moved to the front of the array)
  - In the *i*-th iteration of the outer loop, the inner loop will perform *4i+1* operations
  - Therefore, the total cost of the inner loop will be *2n(n-1)+n-1*, ***ie.*** ***O(n²)***
- Time cost:
  - Best case: *7(n-1)*
  - Worst case: *5(n-1)+2n(n-1)+n-1*
- What about the number of moves?
  - Best case: *no moves*
  - Worst case: *2(n-1)+n(n-1)/2*
- Aside: Where are the dominant terms, above?

Insertion Sort: Average Case

- Is it closer to the best case (*n* comparisons)?
- Is it closer to the worst case (*n* * (*n*-1) / 2) comparisons?
- It turns out that when random data is sorted, insertion sort is usually closer to the worst case
  - Around *n* * (*n*-1) / 4 comparisons
  - Calculating the average number of comparisons more exactly would require us to state assumptions about what the "average" input data set looked like
  - This would, for example, necessitate discussion of how items were distributed over the array
- Exact calculation of the number of operations required to perform even simple algorithms can be challenging!