Searching and Sorting

It is very common to store large amounts of data, and to want to search the data for

- a particular data item, identified by its Key field
- the data item that has smallest or largest key value
- all the items that have a particular characteristic

If the data is sorted then many of these operations can be done more efficiently

We have seen

- a sorting algorithm, insertion sort
- binary search, an algorithm that quickly finds a given key value in a sorted array.
- We also looked at linear search, which does not need the array to be sorted.

- let us now look at Bubble Sort

Bubble Sort

Idea: make a pass through the array
- at each index up to sz-2
check if it should be swapped
with its right neighbour - if so,
swap it.

[Do an example.]

After a single pass, is it guaranteed to be sorted?

Bubble Sort

Idea: make a pass through the array
- at each index up to sz-2
check if it should be swapped
with its right neighbour - if so,
swap it.

[Do an example.]

After a single pass, is it guaranteed to be sorted?

No.

Keep making passes [ ]
until [ ],
at which time it is sorted.

Bubble Sort

Idea: make a pass through the array
- at each index up to sz-2
check if it should be swapped with its right neighbour - if so, swap it.

[Do an example.]

After a single pass, is it guaranteed to be sorted?

No.

Keep making passes [up to sz-3
up to sz-4
⋮]
until [no swap is made in a pass],
at which time it is sorted.

Is Bubble Sort a good sort?

Number of comparisons is (worst case)

$$n-1 + n-2 + \cdots + 3 + 2 + 1$$

$$= \sum_{i=1}^{n-1} i$$

$$= \frac{n(n-1)}{2} \quad \text{or approx} \quad \frac{1}{2}n^2$$

(There are more efficient sorts.)