$n^2$ Sorts

- Lots of $\Theta(n^2)$ sort algorithms
  - Bubble sort
  - Insertion sort
  - many, many others...

- NEVER use these for large datasets
  - Use $\Theta(n \log n)$ sorts instead

- But very useful for small datasets
Insertion Sort

- Keep a list of sorted values **so far**
- Select a new value
- Insert the new value into the sorted list – $O(n)$
- Repeat until all are inserted
Insertion Sort

EXAMPLE:

- 3 values in the sorted array already (1,3,8)
- New value is 2
- How to find the location (1\textsuperscript{st} \texttt{for}() loop)
- How to find the shift over (2\textsuperscript{nd} \texttt{for}() loop)
  - Alternate: combine the two loops
• Time Cost?
  - $O(n)$ operations each time we insert
    • On average, only need to scan $\frac{1}{2}$ the sorted array
    • But time grows as inserted list gets longer
  - $O(n^2)$ operations overall
    • $1 + 2 + 3 + 4 + \ldots + n = O(n^2)$
Why Insertion Sort?

- Often the fastest choice for small arrays (≤4)
  - Never use this for large arrays!

- Reasonably easy to write
INPUT: arr[]
for (ins=1; ins<arr.length; ins++)
{
    move = arr[ins];
    for (cmp=ins-1; cmp>=0; cmp--)
        if (arr[cmp] <= move)
            break;
    for (shift=cmp+1; shift<=ins; shift++)
    {
        tmp = arr[shift];
        arr[shift] = move;
        move = tmp;
    }
}
Bubble Sort

- The famous “worst sorting algorithm”
- But it's very, very simple to implement:
  - “Bubble up” elements through the array
  - Repeat enough times so that everything eventually is in the right place.
Bubble Sort

- Compare 3, 8
  - Proper order, so no action required
Bubble Sort

- Compare 8, 1
  - Out of order, so swap
Bubble Sort

- Compare 8, 2
  - Swap

- See how the 8 is “bubbling up”?
Bubble Sort

repeat n times
  scan through the array (n steps)
  compare adjacent elements
  swap if out of order
Why Bubble Sort?

• Trivial to get right

• Never a good idea for performance-sensitive code!