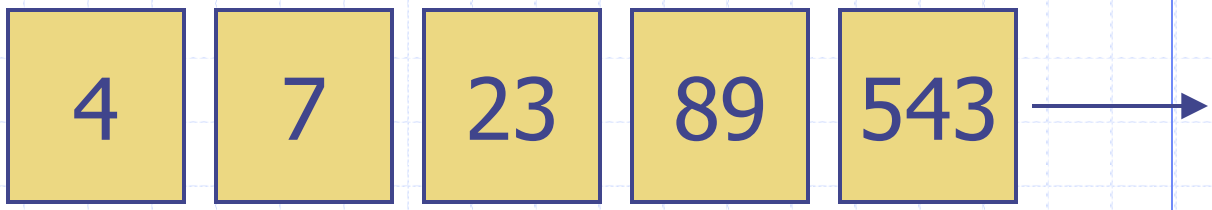
Sorting



Sorting

- A fundamental algorithm of significant importance to many applications
- There is no *single* best sorting algorithm – it is application specific
- Algorithms range from $O(n^2)$ to seemingly O(n) under certain conditions
- General sorting of arbitrary unorganized keys is at best O(nlogn)
  - If you can do better, please patent it, then let us know (me first)

How fast is it possible to sort?

- (see other slide deck)

Sorting 101

- What is the easiest sorting algorithm to program?
  - (using a standard built-in library is not a valid answer ☺)

Sorting 101

- Live demo

Bubble Sort

- "The easiest sorting algorithm to program"
- Algorithm:

```
repeat
    for x = 1 to N-1
      if (A[x] > A[x+1]) then swap
    end
until (no more swaps)
```

Bubble Sort

- What is the big-Oh?
  - $O(n^2)$
- Algorithm:

```
repeat
    for x = 1 to N-1
      if (A[x] > A[x+1]) then swap
    end
until (no more swaps)
```

Bubble Sort

- What is the worst-case input?
  - When array is in reverse order
- Algorithm:

```
repeat
    for x = 1 to N-1
      if (A[x] > A[x+1]) then swap
    end
until (no more swaps)
```

Bubble Sort

- Summary
  - $O(n^2)$
  - Worst case is when input is in reverse order
- When is the algorithm useful?
  - If need to implement something quick and dirty
  - If only have access to immediately neighboring array element and have a small memory cache

Improvements?

- How can you do better than standard bubble sort?
  - Lets remove the "restriction" of only swapping with the immediate neighbor
  - Instead and starting at the beginning of the array, lets find the smallest guy in the rest of the array and swap it with the current element

Selection Sort

- Algorithm:
  - 1. Find the minimum value in the array
  - 2. Swap it with the value in the first position
  - 3. Do again starting at the second position but only with later array elements
  - 4. Repeat until end of array
- What is the big-oh?
  - $O(n^2)$
- When/why is this better than bubble sort?
  - Application specific, but for instance if data can only be accessed "in a forward fashion" and previously processed array elements are not accessible anymore (e.g., tape? network?)

Improvements?

- How can you do better than bubble sort or selection sort?
  - Instead of "swapping" an element with a later element, lets "insert" the next element into place
  - This is the "natural sorting algorithm" often used by Homo Sapiens to sort items (e.g., exams, notes, bones)

Insertion Sort

- "Natural sorting algorithm"…live demo

Insertion Sort

- Algorithm:
  - Starting at the beginning of the array, find the next smallest element
  - Insert at the beginning (all other elements must be shifted)
  - Find the next smallest element and repeat
- What is the big-oh?
  - $O(n^2)$

Insertion Sort

- Why/when is it good?
  - Stable
    - does not change the order of equal keys
  - In-place
    - Only requires O(1) extra space
  - Online
    - Can sort an array as it receives it
  - (the combination of these characteristics makes insertion sort interesting)
- Note: if using linked lists, the "insert" operation is easy!

Insertion Sort

- What happens when array is almost sorted?
  - Not much "shifting is needed"
  - Performance becomes (near) O(n)
- Can you reduce the amount of shifting?
  - Yes, don't shift by 1-item but shift by k-items, then k/2, etc, until by 1-item
  - Average performance ranges from $O(n\log^2 n)$ to $O(n^2)$ – this is called "**Shell Sort**"

Further Improvements?

- Lots of other variants to $O(n^2)$ sorting algorithms exist, each with different pros/cons…