CS310: ALGORITHMS AND DATA STRUCTURES

Sorting Problem

- Given a randomly ordered sequence

  Input: <a1, a2, …, an>

- Sort them into non-decreasing order

  Output: <a1', a2', …, an'>

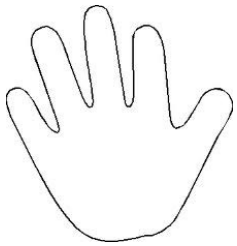
  $\exists$ a1' $\leq$ a2' $\leq$ a3' … $\leq$ an'

  (Duplicates are allowed)

- There exist a number of sorting algorithms with varying complexities involving different kinds of data structures like arrays, heaps

Insertion Sort

- A sorting method used by card players



- At each step, an element is taken from the unordered sequence and inserted into the ordered sequence
- Build the sorted sequence (in place) one element at a time

Pseudo Code

- A programming language with English
- Does not use braces but uses indentation
- Permitted Keywords:
  - Do While … EndDo; Do Until … EndDo; If … EndIf;
  - Case … EndCase; When; Return …; Return; For … End For
- Other Words:
  - Set; Reset; Increment; Compute; Calculate; Add; Sum; Multiple; Print; Display; Input; Output; Edit; Test

Insertion Sort *Cont…*

- Since our ordered sequence is maintained as an array, inserting a new element into it requires us to "make space" for it

To insert jth item

start from O[j-1] and scan left

while O[index] > x

shift the element to the right

index –

O[index+1] = x

Insertion Sort *Cont…*

```
For j = 2 to n
    // insert the jth element
    key = A[j]
    i = j – 1

        // Insert A[j] into the sorted sequence A[1 · · j-1]
        Do While i > 0 and A[i] > key
            A[i + 1] = A[i]
             i = i – 1
         End Do

    A[i + 1] = key
End For
```

Computational Time of Insertion Sort

- In general: $K_1(n-1) + K_2\, n(n-1)$
- Three notions of computational time
  - Worst case: Maximum time on any input of size n
  - Average case: Expected time over all inputs of size n
  - Best case: Minimum time on any input of size n
- Computational time equation for each case?

Big Idea … Next Time

- Not worry about the value of the constant but only look at the most dominant term in the computational time equation.