I & C SCI 46 Winter 2023
Lecture 21: Fundamentals of Sorting I

What is sorting?

- Input: sequence of n comparable values
- Reorder the input to be non-descending.
- Items we wish to sort are called "keys"

- Not here: retain associated information

Why discuss sorting?

- Standard library has sorting std::sort
- Why not use that and move on?

In this class, sorting is:

- a good intro for techniques
- a good intro to comparative algorithms

SelectionSort

Idea: Swap min into first spot, second-min to second, etc.
(This is hand-wavy on purpose)

85	24	63	45	17	31	96	50
17	24	63	45	85	31	96	50
17	24	63	45	85	31	96	50
17	24	31	45	85	63	96	50

SelectionSort

for $i \leftarrow 1$ to $n - 1$ **do**
  min $\leftarrow i$
  for $j \leftarrow i + 1$ to n **do**
    if $A[j] < A[\text{min}]$ **then**
      min $\leftarrow j$
  Swap $A[i]$ and $A[\text{min}]$

$O(n^2)$

What's nice about SelectionSort?

- Easy to program
- Easy to explain
- Does it waste memory? only $O(1)$ "in place" plus input
- Does it only work for numbers?
- What other info do we need?
- Are there inputs that are sorted faster?
- Is there a lot of data movement? $2(n-1)$? a lot ????

BubbleSort

Idea: Think globally act locally

Shortest

Tall

85	24	63	45	17	31	96	50
24	~~85~~ 63	45	17	31	85	50	96
24	45	17	31	63	50	85	96
24	17	31	45	50	63	85	96

BubbleSort

for $i \leftarrow 1$ to $n-1$ **do**
  for $j \leftarrow 1$ to $n-i$ **do**
    if $A[j+1] < A[j]$ **then**
      Swap $A[j]$ and $A[j+1]$

swap one adjacent inverted pair

If no swap inside inner loop, sorted already.

InsertionSort

Idea:

85	24	63	45	17	31	96	50
24	85	63	45	17	31	96	50
24	63	85	45	17	31	96	50
24	45	63	85	17	31	96	50

InsertionSort

for $j \leftarrow 2$ to n **do**
  key $\leftarrow A[j]$
  $i \leftarrow j - 1$
  while $i > 0$ and $A[i] >$key **do**
    $A[i+1] \leftarrow A[i]$
    $i = i - 1$
  $A[i+1] \leftarrow$ key

controls running time for fastest, this always false

slowest? Always true

- What is the running time of InsertionSort?

$O(n + \ell)$

inverted pairs in input

InsertionSort

for $j \leftarrow 2$ to n **do**
  key $\leftarrow A[j]$
  $i \leftarrow j - 1$
  while $i > 0$ and $A[i] >$key **do**
    $A[i+1] \leftarrow A[i]$
    $i = i - 1$
  $A[i+1] \leftarrow$ key

- Why is InsertionSort correct?
- What is true *every time* we check the **for** loop? (including the time we find $j > n$ and stop)

About that running time ...

- Why are we so concerned with worst case?
- Why not examine average case?