I & C SCI 46 Spring 2022
Lecture 23: Fundamentals of Sorting II

2

BubbleSort

Idea: Think globally act locally

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

3 # 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 $A[j+1]$

4 # 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
17	24	45	63	85	31	96	50

InsertionSort $O(n + ɛl)$

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

best: false n
already ascending

worst: true
descending n^2

- What is the running time of InsertionSort?

avg? $\cancel{\frac{n+n^2}{2}}$ no

Possible Inverted Pairs? $\frac{n(n-1)}{2} \times \frac{1}{2}$

Avg time $O(n^2)$ $= \frac{n(n-1)}{4}$

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)

6 ## About that running time ...

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

7 ## HeapSort

Idea: Use a max heap.

- Find max, put max at end
- Then second-max, etc.
- Use the yet-to-be-sorted array as max heap

Heapify: make array into max heap

- Idea 1: insert each into growing heap

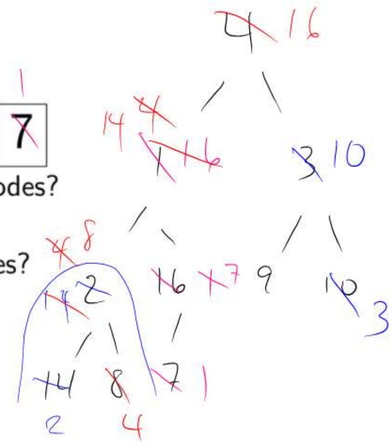
$O(n \log n)$ time, $O(n)$ add'tl space

Phase 2: $O(n \log n)$ time

8

Heapify: Better way

16	14	10	8	7	9	3	2	4	1
~~4~~	~~1~~	~~3~~	~~2~~	~~16~~	9	~~10~~	~~14~~	8	~~7~~



- Treat array as heap. Where are leaf nodes?

  last half

- What should we do with non-leaf nodes?
- In which order?

9

How long to heapify?

- The cost to insert varies by height.
- Node at height h costs $\mathcal{O}(h)$.
  How many nodes at height h? $\frac{n}{2^{h+1}}$
- How many different height values are there?
- Cost for total is:

$$\sum_{h=0}^{\lfloor \log n \rfloor} \lceil \frac{n}{2^{h+1}} \rceil \mathcal{O}(h) = \mathcal{O}\Big(n \underbrace{\cancel{\sum_{h=0}^{\lfloor \log n \rfloor} \frac{h}{2^h}}}_{\to\, O(1)}\Big) \to O(n)$$

Using formula for infinite geometric series:

$$\sum_{h=0}^{\infty} \frac{h}{2^h} = \frac{1/2}{(1-1/2)^2} = O(1)$$

10 And how, HeapSort

16	14	10	8	7	9	3	2	4	1