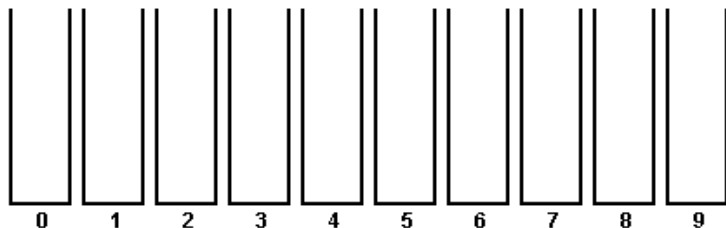What is the min height of a
tree with X external nodes?

# CSSE 230

## Is there a limit to how fast you can sort?
## Is there a way to break that limit?

Radix sort to the rescue … sort of…

(2805) (1081) (7556) (3684) (6428) (6321) (1456) (8850) (8022) (7919)

After today, you should be able to…
…explain why comparison-based sorts
need at least O(n log n) time
… explain bucket sort
… explain radix sort
… explain the situations in which radix sort
is faster than O(n log n)

0    1    2    3    4    5    6    7    8    9

http://www.cs.auckland.ac.nz/software/AlgAnim/radixsort.html

# Announcements

▸ SortingRaces is due Friday.
Day 29's class time will be SortingRaces work time.

▸ The sounds of sorting. Radix sort later.
  ◦ https://www.youtube.com/watch?v=kPRA0W1kECg

# A Lower-Bound on Sorting Time

We can't do much better than what we already know how to do.

# What's the best best case?

▸ Lower bound for best case?

▸ A particular algorithm that achieves this?

# What's the best worst case?

▸ Want a function $f(N)$
such that the **worst case running time**
for **all sorting algorithms** is $\Omega(f(N))$

▸ How do we get a handle on
"all sorting algorithms"?

Tricky!

# What are "all sorting algorithms"?

▸ We can't list all sorting algorithms and analyze all of them
  ◦ Why not?

▸ But we can find a **uniform representation** of any sorting algorithm that is based on **comparing** elements of the array to each other

# First of all…

- The problem of sorting N elements is at least as hard as determining their ordering
  - e.g., determining that $a_3 < a_4 < a_1 < a_0 < a_2$

| 0 | 1 | 2 | 3 | 4 |
|---|---|---|---|---|
| 58 | 55 | 73 | 5 | 10 |

  - sorting = determining order, then movement

- So any lower bound on all "order-determination" algorithms is also a lower bound on "all sorting algorithms"

# Sort Decision Trees

▸ Let A be any **comparison-based algorithm** for sorting an array of distinct elements

▸ We can draw an EBT that corresponds to the comparisons that will be used by A to sort an array of N elements
  - This is called a **sort decision tree**
  - Internal nodes are comparisons
  - External nodes are orderings

  - Different algorithms will have different trees
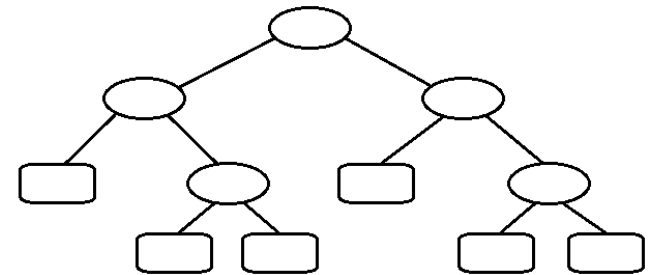
# Insertion Sort

▸ Basic idea:
  ◦ Think of the array as having a sorted part (at the beginning) and an unsorted part (the rest)

| 0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 |
|---|---|---|---|---|---|---|---|---|---|
| 38 | 44 | 87 | 2033 | 99 | 1500 | 100 | 90 | 239 | 748 |

  ◦ Get the first value in the unsorted part
  ◦ Insert it into the correct location in the sorted part, moving larger values up to make room

Repeat until unsorted part is empty

# So what?

- Minimum number of external nodes in a sort decision tree?  (As a function of N)

- Is this number dependent on the algorithm?

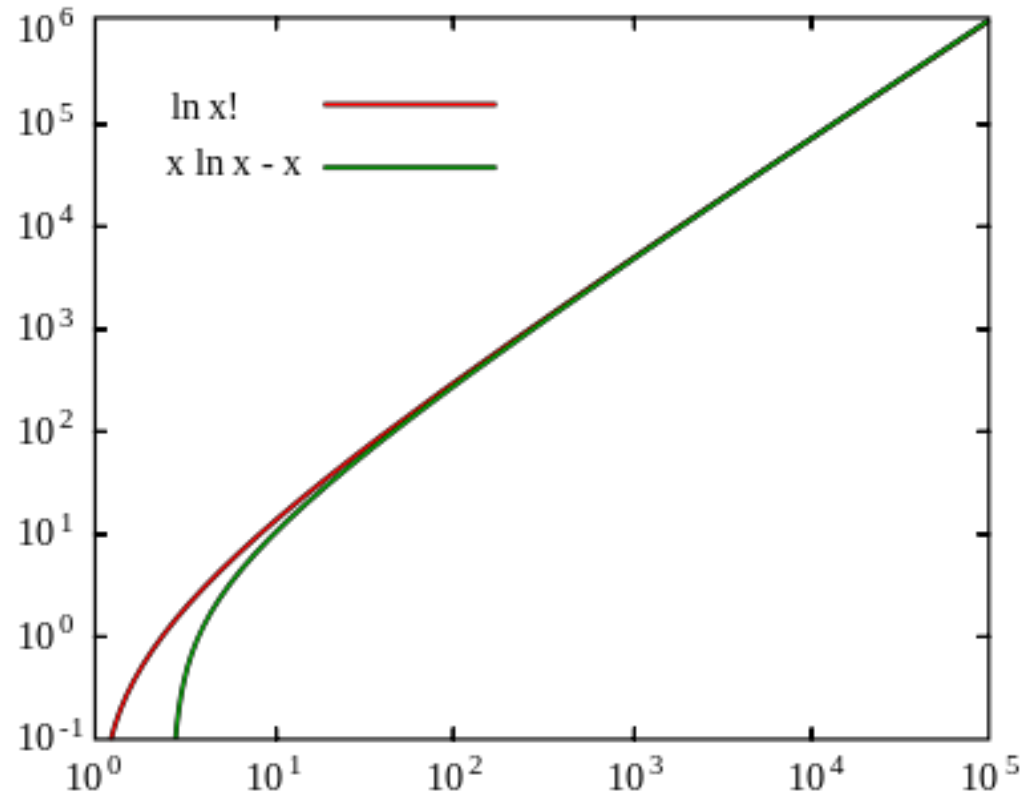- What's the height of the shortest EBT with that many external nodes?

$$\lceil \log N! \rceil \approx N \log N - 1.44N = \Omega(N \log N)$$

No comparison-based sorting algorithm, known or not yet discovered, can **ever** do better than this!

# An approximation for log (n!)

- Use **Stirling's approximation**:

$$\ln n! = n \ln n - n + O(\ln(n))$$

# Can we do better than *N* log *N*?

- $\Omega(N \log N)$ is the best we can do if we compare items

- Can we sort without comparing items?

# Yes, we can! We can avoid comparing items and still sort. This is fast if the range of data is small. <span style="color:orange">Q5</span>

- ▸ Observation:
  - ◦ For N integers, if the range of data is less than N, then we have duplicates

- ▸ O(N) sort: Bucket sort
  - ◦ Efficient if possible values come from limited range and have a uniform distribution over the range
  - ◦ Example: Exam grades histogram
  - ◦ Put N items in b "buckets" to split them.
    - • Can recurse until they are sorted.
    - • If b =2, then recursive bucket sort is just _____sort.

# A variation:  Radix sort

▸ A picture is worth $10^3$ words, but an animation is worth $2^{10}$ pictures, so we will look at one.

▸ https://yongdanielliang.github.io/animation/web/RadixSort.html

▸ http://www.algostructure.com/sorting/radixsort.php

▸ http://www.cs.auckland.ac.nz/software/AlgAnim/radixsort.html
▸  (good but blocked)

▸ https://www.youtube.com/watch?v=xuU-DS_5Z4g&src_vid=4S1L-pyQm7Y&feature=iv&annotation_id=annotation_133993417
(video, good basic idea, distracting zooms)

▸ http://www.cs.usfca.edu/~galles/visualization/RadixSort.html
(good, uses single array)

Summarize what you see on your quiz.

# RadixSort and BucketSort can be O(n)

- It takes $\log_b$(data range) passes if b buckets.
  - So runtime is $\Theta(N\log_b(range))$

- Assume b=2 and integers in range [1,max]
  - max is just the largest in the set.

- Runtime is $\Theta(N\log_2 max)$

1. If all max=N, perhaps ints are unique:

2. If max=c≪N so many duplicates:

3. If max≫N so data is sparse: