

# Sorting Algorithms

- Insertion Sort:  $\Theta(n^2)$
- Merge Sort:  $\Theta(n \log(n))$
- Heap Sort:  $\Theta(n \log(n))$
- We seem to be stuck at  $\Theta(n \log(n))$
- **Hypothesis:** Every sorting algorithm requires  $\Omega(n \log(n))$  time.

~~Egmr: There is no algorithm which can all inputs takes  $O(n \log n)$  time.~~

## Lower Bound Definitions

- Merge sort:  $\Omega(n \log(n))$  on *every* input
- Insertion sort:
  - $1, 2, 3, 4, \dots, n-1, n$       $O(n)$  time
  - $n, n-1, n-2, \dots, 2, 1$       $O(n^2)$  time
- **Hypothesis:** For every sorting algorithm A and every integer n, there is some input of length n on which A requires  $\Omega(n \log(n))$  time.

~~Maybe: every alg on every input is  $\Omega(n \log n)$  time~~

# Proving Lower Bounds

- What if there is some absurd  $O(n)$  algorithm? E.g.
  - Square every third element
  - For every prime  $j$ ,  $A[j] = 2^A[j]$
  - For every  $j$ , look up  $A[j]$ 'th word in the August 2013 New York Times
  - Etc.

# Comparison sorting

- Want algorithms that work on any input
- e.g. insertion/merge/heap sort
- Think about sorting: uses comparisons.
- Comparison based sorting algorithm only relies on comparisons + moves
- Running time =  $\Omega(\# \text{ of comparisons})$

Only access data using comparison

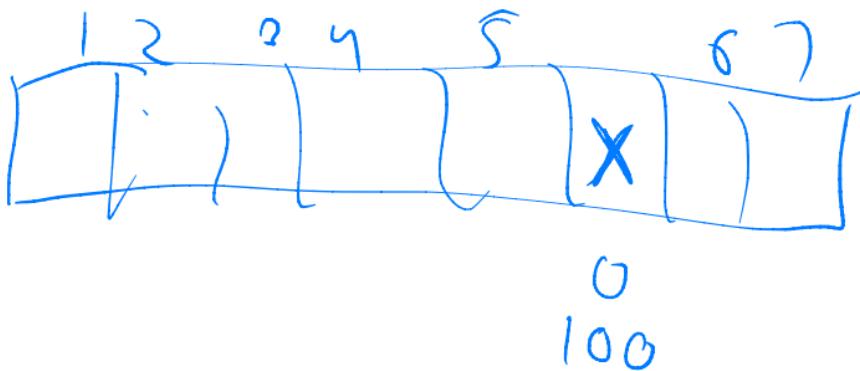
A	2	5	?	1	4	9	8
---	---	---	---	---	---	---	---

# New Hypothesis

Every comparison-based sorting algorithm requires  $\Omega(n \log(n))$  comparisons in the worst case.

# Required Comparisons

- $O(n \log(n))$  comparisons suffices
  - merge sort, heap sort
- Trivial: need  $\Omega(n/2)$  comparisons



## New Hypothesis

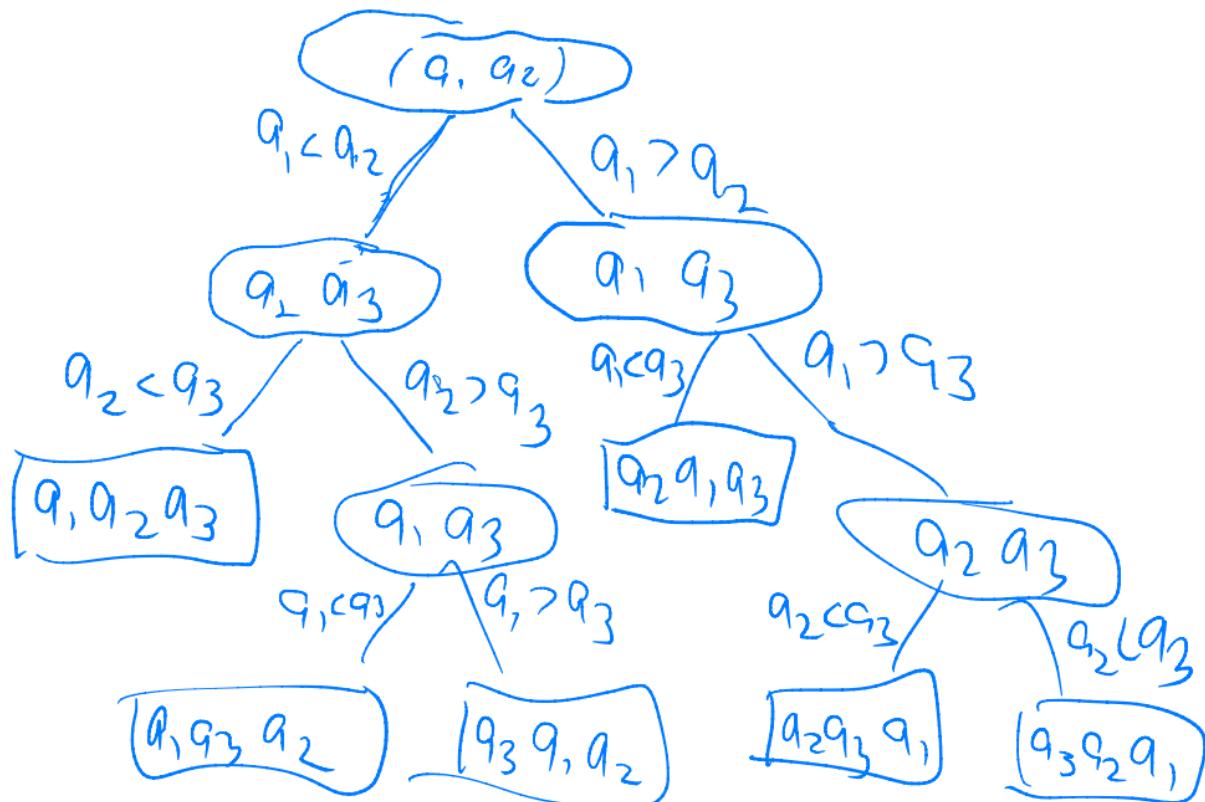
Every comparison-based sorting algorithm requires  $\Omega(n \log(n))$  comparisons in the worst case.

## Ideas for Proof

1. There are, a priori,  $n!$  possible permutations of the input
2. Each comparison can roughly halve the number of possible permutations
3. Therefore, we must make at least  $\log(n!)$  comparisons
4.  $\log(n!) = \Theta(n \log n)$



$a_1, a_2, a_3$   
Proof with decision trees



Every perm. must appear at a leaf

$\geq n!$  leaves

#comp-  
~~height~~  $\geq$  height of tree  $f_h$ )

#leaves  $\leq 2^h$

---

#comp  $\geq h \geq \lg(\#leaves)$   
 $\geq \lg(n!) = \Theta(n \lg n)$

# Average Case Analysis

- $\Omega(n \log(n))$  comparisons in worst case
- Merge sort: always  $\Theta(n \log(n))$
- Insertion sort: sometimes  $\Theta(n)$ , sometimes  $\Theta(n^2)$
- Can we get the best of both? Sometimes  $\Theta(n \log(n))$ , usually  $O(n)$ ?
- NO: need  $\Omega(n \log(n))$  comparisons *on average* among all possible inputs.

Stirling's approx:  $n! \approx \left(\frac{n}{e}\right)^n \sqrt{2\pi n} (1 + o(1))$