Sorting

Insertion Sort:	$\Theta(n^2)$	worst case average case
Mergesort:	$\Theta(n \lg n)$	worst case average case
Heapsort:	$\Theta(n \lg n)$	worst case average case
Quicksort:	$\Theta(n^2)$ $\Theta(n \lg n)$	worst case average case

---

Is any sorting algorithm **faster**?

(i.e. $o(n \lg n)$)?

in worst case?

in average case?

Lower Bounds/Sorting Overview

Decision tree model

(used to establish lower bounds for comparison problems)

Lower bound for sorting:

$\Omega(n \lg n)$ for comparison-based sort.

Integers in fixed range $\{1, \ldots, k\}$:

Improvement possible!

Real Numbers in a bounded interval:

Improvement possible!

Lower Bounds

Function lower bound

"$g(n)$ is a lower bound for $f(n)$":

$f(n) \in \Omega(g(n))$

Algorithm lower bound

"$g(n)$ is a lower bound on the w.c. *time* required by algorithm A":

If $f(n)$ is the worst case time required by algorithm A on an input of size n, then $f(n) \in \Omega(g(n))$.

Problem lower bound

"$g(n)$ is a lower bound on the number of comparisons required to solve problem P in the worst case":

No matter what algorithm A one may devise to solve problem P, $g(n)$ is a lower bound on the w.c. number of comparisons required by algorithm A.

Binary Decision Tree

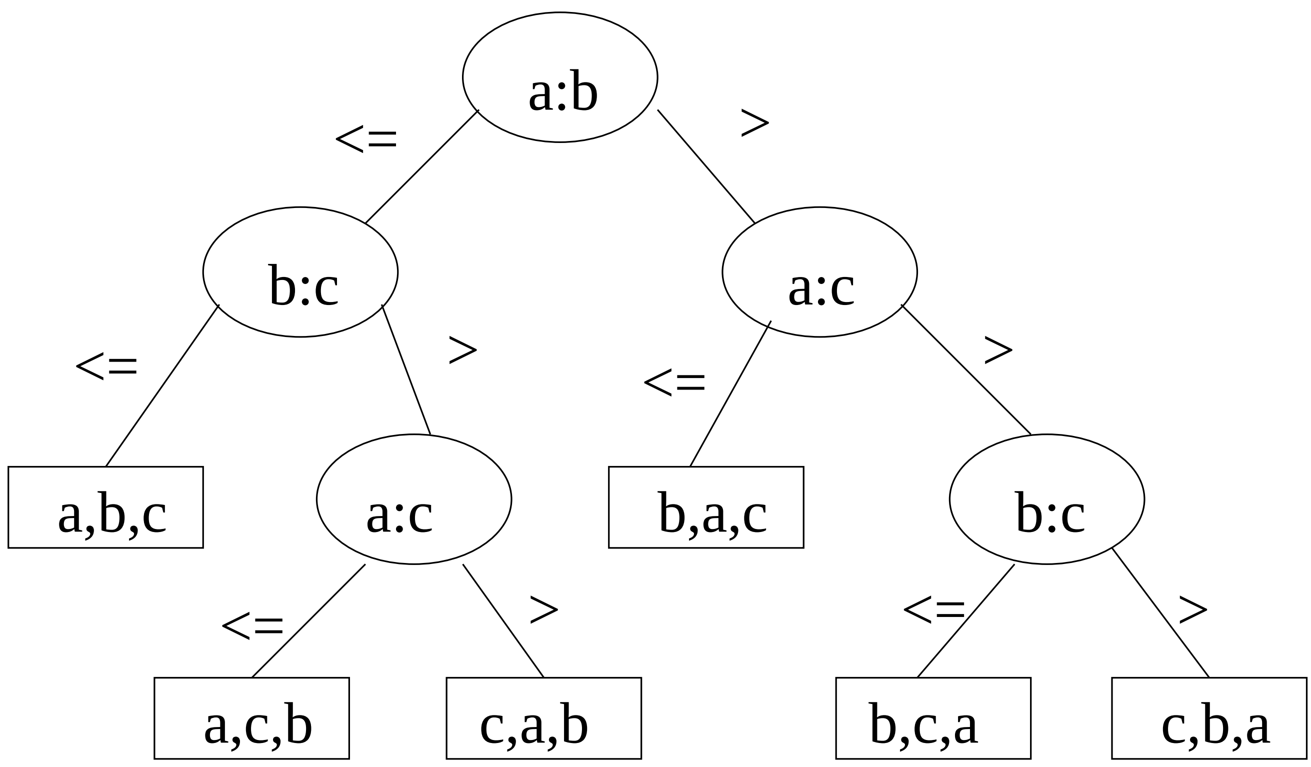
binary tree

internal node: comparison $(x : y)$

left branch: $x \leq y$**;** **right branch:** $x > y$

leaf: a possible final outcome (e.g. sorted order)

Ex. Decision tree for Insertion Sort on $[\boldsymbol{a}, \boldsymbol{b}, \boldsymbol{c}]$.



Any **comparison sort** can be modeled by a **decision tree**.

The **worst case** number of compares for a sorting algorithm is the **height** of its corresponding decision tree.

A decision tree must have at least one **leaf** for every possible **outcome**.

"Sort n elements" has $n!$ **possible outcomes**.

Decision tree for any algorithm which sorts n elements must have at least $n!$ **leaves**.

$2^{\textbf{height}} \geq \#$ **leaves** $\geq n!$

Decision tree **height** must be $\geq \lg(n!)$

Worst case number of compares must be $\geq \lg(n!)$

Any **comparison sort** can be modeled by a **decision tree**.

- A decision tree must have at least one **leaf** for every possible **outcome**.
- There are $n!$ **possible outcomes**.

$\longrightarrow$ A decision tree for any algorithm which sorts n elements must have at least $n!$ leaves.

- The **worst case** number of compares for a sorting algorithm is the **height** of the decision tree.
- A decision tree has less than $2^{\textbf{height}}$ leaves.

Altogether we get

$$\begin{aligned} 2^{\textbf{height}} &\geq \#\textbf{leaves} \geq n! \\ \textbf{height} &\geq \lg(n!) \end{aligned}$$

Conclusion: The worst case number of compares must be larger than $\lg(n!)$

Worst case number of compares $\geq \lg(n!)$

(Recall Stirling: $n! \geq (\frac{n}{e})^n$)

So, **worst case** number of compares is at least:

$$\begin{aligned}\lg(n!) &\geq \lg[(\frac{n}{e})^n] \\ &= n[\lg n - \lg e] \\ &\in \Omega(n \lg n)\end{aligned}$$

"Doing Better Than $n \lg n$"

Array $A[1..n]$ of elements in $\{0, \ldots, k\}$

Counting Sort

Phase I ▷ count

▷ use $C[0..k]$

for $i \leftarrow 0$ **to** k **do**

$C[i] \leftarrow 0$

for $j \leftarrow 1$ **to** n **do**

$C[A[j]] \leftarrow C[A[j]] + 1$

---

time?

Phase II

$\triangleright$ $C[0..k]$, $C[i] = \#$ of elts. $= i$

for $i \leftarrow 1$ **to** k **do**

$$C[i] \leftarrow C[i-1] + C[i]$$

$\triangleright$ Now $C[i] = \#$ of elts. $\leq i$

for $j \leftarrow n$ **downto** 1 **do**

$$B[C[A[j]]] \leftarrow A[j]$$

$$C[A[j]] \leftarrow C[A[j]] - 1$$

Phase I:

Phase II:

Total?

A sorting algorithm is

stable

if elements of equal key value remain in the same relative order.

Which are stable?

Counting sort?

Insertion sort?

Mergesort?

Heapsort?

Quicksort?