Sorting based on Comparisons

- Basic operation: *compare* two items

- Consider sorting three items \((x, y, z)\)
  - How many comparisons are needed (at least)?

- Consider sorting \(n\) items
  - Is there a *lower bound*?

Decision Tree (sorting \(x, y, z\))

- What is the worst-case number of comparisons?
  - *Height* of the decision tree (length of longest path from root to a leaf)

- Consider sorting \(n\) distinct items
  - What is the height?
  - … Use the number of leaves

  - Number of leaves *at least* \(n!\) # of all permutations
  - Number of leaves *at most* \(2^h\) perfect binary tree
What is the height?

\[
2^h \geq \# \text{ leaves} \geq n! \\
2^h \geq n! \\
\log 2^h \geq \log n! \\
h \geq n \log n
\]  ... by Stirling’s formula

Cost of Sorting

- What is a lower bound for the cost of sorting algorithms based on comparisons?

\[\Omega(n \log n)\]

- What is the cost of sorting algorithms considered optimal?

\[\Theta(n \log n)\]

Problem

- Sort a flight departures table
  - by time then by location

Stability

http://algs4.cs.princeton.edu/25applications/
Stability

A sorting algorithm is **stable** if it preserves the order of **equal** elements

In the example:

Original: 5 3 2 4 1 4 7 5

After sorting: 1 2 3 4 4 5 5 7

- Is selection sort stable? 👎
  - long distance swaps

- Is insertion sort stable? 👍
  - equal items never pass each other (depends on correct implementation)

In-place?

A sorting algorithm is **in-place** if it uses \(O(\log n)\) extra memory

- Selection and Insertion sorts are **in-place**
## Summary

<table>
<thead>
<tr>
<th></th>
<th>Best-Case</th>
<th>Average-Case</th>
<th>Worst-Case</th>
<th>Stable?</th>
<th>In-place?</th>
</tr>
</thead>
<tbody>
<tr>
<td>Selection Sort</td>
<td>$\Theta(n^2)$</td>
<td>$\Theta(n^2)$</td>
<td>$\Theta(n^2)$</td>
<td>No</td>
<td>Yes</td>
</tr>
<tr>
<td>Insertion Sort</td>
<td>$\Theta(n)$</td>
<td>$\Theta(n^2)$</td>
<td>$\Theta(n^2)$</td>
<td>Yes</td>
<td>Yes</td>
</tr>
</tbody>
</table>