Limits to Comparison Sorting

Models of Computation

- If we know something about the structure of the file being sorted, then special case sorts can perform miracles.

- What model of complexity are we using? *Spaghetti sort*, for example, has a fast worst-case run-time, but would be difficult to implement in Java.
How Good a Sort Can One Expect?

<table>
<thead>
<tr>
<th>Algorithm</th>
<th>Worst</th>
<th>Comments</th>
</tr>
</thead>
<tbody>
<tr>
<td>Insertion Sort</td>
<td>$n^2$</td>
<td>Best for nearly sorted files</td>
</tr>
<tr>
<td>Quicksort</td>
<td>$n^2$</td>
<td>Very fast on average</td>
</tr>
<tr>
<td>Mergesort</td>
<td>$n \log n$</td>
<td>Best worst case</td>
</tr>
</tbody>
</table>

Decision Tree Model

- We consider algorithms in which the only operation used to gain information about the sequence is comparison of two elements.
Recall: Insertion Sort

```
InsertionSort(A)
for j = 2 to A.length
    key = A[j]
    // Insert A[j] into the sorted sequence A[1..j-1]
    i = j-1
    while i > 0 and A[i] > key
        A[i+1] = A[i]
        i = i-1
    A[i+1] = key
```

Leaves and Branches

**Lemma**
A binary tree of height $h$ has at most $2^h$ leaves.

**Corollary**
A binary tree with $l$ leaves must be at least $\log_2 n$ high.
Limits to Comparison Sorting

Theorem 9.1
Any decision tree that sorts $n$ elements has height $\Omega(n \lg n)$.

Corollary 9.2
Heapsort and merge sort are asymptotically optimal comparison sorts.