Sorting Algorithms

COMP2402
Carleton University
Winter 2018
MergeSort

- Sorting algorithm
  - Comparison-based
  - Recursive
  - Divide and conquer

- Repeatedly merge sorted sublists to create a sorted list

- Reading:
  - ODS Section 11.1.1
MergeSort

Theorem:

The **MergeSort** algorithm runs in $O(n \log n)$ time and performs at most $n \log n$ comparisons.
QuickSort

• Sorting algorithm
  – Comparison-based
  – Recursive
  – Divide and conquer

• Repeatedly partition a list around randomly selected pivot elements

• Reading:
  – ODS Section 11.1.2
QuickSort

Theorem:

The **QuickSort** algorithm runs in $O(n \log n)$ expected time and the expected number of comparisons it performs at most $2n \ln(n) + O(n)$. 
## Sorting Algorithms

<table>
<thead>
<tr>
<th>Algorithm</th>
<th>Comparisons</th>
<th>in-place</th>
<th>stable</th>
</tr>
</thead>
<tbody>
<tr>
<td>MergeSort</td>
<td>nlog(n) worst case</td>
<td>no</td>
<td>yes</td>
</tr>
<tr>
<td>QuickSort</td>
<td>1.38nlog(n) + O(n) expected</td>
<td>yes</td>
<td>no</td>
</tr>
<tr>
<td>HeapSort</td>
<td>2nlog(n) + O(n) worst case</td>
<td>yes</td>
<td>no</td>
</tr>
</tbody>
</table>
Lower bound on Comparison Sorting

Theorem:
For any comparison-based sorting algorithm A, the expected number of comparisons done by A when sorting a random permutation of n values is at least \( \log(n!) = \Omega(n \log n) \)
Non-Comparison-Based Sorting

• We can sort faster if we don't limit ourselves to using comparisons.

• Counting Sort and Radix Sort
  – Sort using array indexing

• Reading:
  – ODS Section 11.2
Non-Comparison-Based Sorting

Theorem:
The **CountingSort** algorithm can sort an array containing \( n \) integers from the set \( \{0, \ldots, k-1\} \) in \( O(n+k) \) time.

Theorem:
The **RadixSort** algorithm can sort an array containing \( n \) integers from the set \( \{0, \ldots, n^k\} \) in \( O(kn) \) time.