Sorting - part two

Alan G. Labouseur, Ph.D.
Alan.Labouseur@Marist.edu
Divide and Conquer

Take a big problem and divide it into two smaller problems. Take a those problems and divide them into two smaller problems. Take a those problems and divide them into two smaller problems. Take a those problems and divide them into two smaller problems. Take a those problems and divide them into two smaller problems. Take a those problems and divide them into two smaller problems. Take a those problems and divide them into two smaller problems. Take a those problems and divide them into two smaller problems. Take a those problems and divide them into two smaller problems. Take a those problems and divide them into two smaller problems. Take a those problems and divide them into two smaller problems. Take a those problems and divide them into two smaller problems. Take a those problems and divide them into two smaller problems. Take a those problems and divide them into two smaller problems. Take a those problems and divide them into two smaller problems.

... until the problems get small enough that they are solved. Then ...

... until the problems get small enough that they are solved. Then ...
Divide and Conquer

Big Problem
Divide and Conquer

Big Problem

Smaller Problems

Smaller Problems
Divide and Conquer

Big Problem

Smaller Problems

Still Smaller Problems

Still Smaller Problems

Still Smaller Problems

Still Smaller Problems
Divide and Conquer

Big Problem

Still Smaller Problems

Still Smaller Problems

Still Smaller Problems

Solve the smallest problem

Solve the smallest problem

Solve the smallest problem

Solve the smallest problem

Solve the smallest problem

Solve the smallest problem

Solve the smallest problem
Divide and Conquer

Big Problem

Smaller Problems
Solved Still Smaller Problems
Solve the smallest problem
Solve the smallest problem

Smaller Problems
Solved Still Smaller Problems
Solve the smallest problem
Solve the smallest problem

Smaller Problems
Solved Still Smaller Problems
Solve the smallest problem
Solve the smallest problem

Smaller Problems
Solved Still Smaller Problems
Solve the smallest problem
Solve the smallest problem
Divide and Conquer

Big Problem

Solved Smaller Problems

Solved Still Smaller Problems

Solve the smallest problem

Solve the smallest problem

Solved Smaller Problems

Solved Still Smaller Problems

Solve the smallest problem

Solve the smallest problem

Solved Smaller Problems

Solved Still Smaller Problems

Solve the smallest problem

Solve the smallest problem

Solved Smaller Problems

Solved Still Smaller Problems

Solve the smallest problem

Solve the smallest problem

Solve the smallest problem

Solve the smallest problem
Divide and Conquer

Solved Big Problem

Solved Smaller Problems

Solved Still Smaller Problems

Solved Still Smaller Problems

Solved Still Smaller Problems

Solved Still Smaller Problems

Solve the smallest problem

Solve the smallest problem

Solve the smallest problem

Solve the smallest problem

Solve the smallest problem

Solve the smallest problem

Solve the smallest problem

Solve the smallest problem
Divide and Conquer :: Merge Sort
Divide and Conquer :: Merge Sort

Given an array that you want to sort . . .

Recursively **divide** the array into sub-arrays half the size until you have arrays of size 1. Note: an array of size 1 is sorted.

Then **conquer** by merging the (technically sorted) single-element arrays into progressively larger sorted sub-arrays as the recursion “unwinds”.
Divide and Conquer :: Merge Sort

Given an array that you want to sort . . .

<table>
<thead>
<tr>
<th>[0]</th>
<th>[1]</th>
<th>[2]</th>
<th>[3]</th>
<th>[4]</th>
<th>[5]</th>
<th>[6]</th>
<th>[7]</th>
</tr>
</thead>
<tbody>
<tr>
<td>2</td>
<td>8</td>
<td>7</td>
<td>1</td>
<td>3</td>
<td>6</td>
<td>5</td>
<td>4</td>
</tr>
</tbody>
</table>
Divide and Conquer :: Merge Sort

Recursively **divide** the array into sub-arrays half the size

<table>
<thead>
<tr>
<th></th>
<th>[0]</th>
<th>[1]</th>
<th>[2]</th>
<th>[3]</th>
<th>[4]</th>
<th>[5]</th>
<th>[6]</th>
<th>[7]</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>2</td>
<td>8</td>
<td>7</td>
<td>1</td>
<td>3</td>
<td>6</td>
<td>5</td>
<td>4</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th></th>
<th>2</th>
<th>8</th>
<th>7</th>
<th>1</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>3</td>
<td>6</td>
<td>5</td>
<td>4</td>
</tr>
</tbody>
</table>
Divide and Conquer :: Merge Sort

Recursively **divide** the array into sub-arrays half the size . .
Divide and Conquer :: Merge Sort

Recursively **divide** the array into sub-arrays half the size . . .

\[
\begin{array}{cccccc}
0 & 1 & 2 & 3 & 4 & 5 \\
2 & 8 & 7 & 1 & 3 & 6 & 5 & 4
\end{array}
\]

\[
\begin{array}{cccc}
0 & 1 & 2 & 3 \\
2 & 8 & 7 & 1
\end{array}
\quad
\begin{array}{cccc}
4 & 5 & 6 & 7 \\
3 & 6 & 5 & 4
\end{array}
\]

\[
\begin{array}{cccc}
0 & 1 & 2 & 3 \\
2 & 8 & 7 & 1
\end{array}
\quad
\begin{array}{cccc}
4 & 5 & 6 & 7 \\
3 & 6 & 5 & 4
\end{array}
\]

\[
\begin{array}{cccc}
0 & 1 & 2 & 3 \\
2 & 8 & 7 & 1
\end{array}
\quad
\begin{array}{cccc}
4 & 5 & 6 & 7 \\
3 & 6 & 5 & 4
\end{array}
\]

. . . until you have arrays of size 1. (Arrays of size 1 are sorted.)
Divide and Conquer :: Merge Sort

Conquer by merging the sub-arrays into progressively larger sorted arrays.
Divide and Conquer :: Merge Sort

Conquer by merging the sub-arrays into progressively larger sorted arrays . .
Divide and Conquer :: Merge Sort

Conquer by merging the sub-arrays into progressively larger sorted arrays . . . until the entire thing is sorted.
Divide and Conquer :: Merge Sort

<p>| | | | | | | | |</p>
<table>
<thead>
<tr>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>2</td>
<td>3</td>
<td>4</td>
<td>5</td>
<td>6</td>
<td>7</td>
<td>8</td>
</tr>
</tbody>
</table>

The sorting work is done in the merge steps.
“There are two ways of constructing a software design. One way is to make it so simple that there are obviously no deficiencies. And the other way is to make it so complicated that there are no obvious deficiencies.”

- C.A.R Hoare
Divide and Conquer :: Quick Sort

Given an array that you want to sort . . .

Recursively divide the array into halves — conquering by partitioning those halves around a “pivot” value — until the smallest sub-arrays are sorted.
Divide and Conquer :: Quick Sort

Given an array that you want to sort . . .

<table>
<thead>
<tr>
<th>[0]</th>
<th>[1]</th>
<th>[2]</th>
<th>[3]</th>
<th>[4]</th>
<th>[5]</th>
<th>[6]</th>
<th>[7]</th>
</tr>
</thead>
<tbody>
<tr>
<td>2</td>
<td>8</td>
<td>7</td>
<td>1</td>
<td>3</td>
<td>6</td>
<td>5</td>
<td>4</td>
</tr>
</tbody>
</table>
Divide and Conquer :: Quick Sort

Given an array that you want to sort . . .

Randomly select an index to provide the pivot value . . .
Divide and Conquer :: Quick Sort

Given an array that you want to sort . . .

Randomly select an index to provide the pivot value . . . and divide the array into halves — conquering by partitioning those halves around a “pivot” value.
Divide and Conquer :: Quick Sort

Given an array that you want to sort . . .

Randomly select an index to provide the pivot value . . . and divide
the array into halves — conquering by partitioning those halves
around a “pivot” value.
Divide and Conquer :: Quick Sort

Given an array that you want to sort . . .

Randomly select an index to provide the pivot value . . . and divide the array into halves — conquering by partitioning those halves around a “pivot” value.
Divide and Conquer :: Quick Sort

We are done when all the sub-arrays are of size 1.

The **sorting work** is done in the **partition** steps.
Divide and Conquer :: Quick Sort

Let’s look at Quicksort again, this time focused on what the array looks like at each step.
Divide and Conquer :: Quick Sort

Let’s look at Quicksort again, this time focused on what the array looks like at each step.
Divide and Conquer :: Quick Sort

Let’s look at Quicksort again, this time focused on what the array looks like at each step.
Divide and Conquer :: Quick Sort

Let's look at Quicksort again, this time focused on what the array looks like at each step.
Divide and Conquer :: Quick Sort

Let’s look at Quicksort again, this time focused on what the array looks like at each step.

Sub-arrays of size 1. Done.
Divide and Conquer :: Quick Sort

Let’s look at Quicksort again, this time focused on what the array looks like at each step.
Divide and Conquer :: Quick Sort

Let’s look at Quicksort again, this time focused on what the array looks like at each step.
Divide and Conquer :: Quick Sort

Let’s look at Quicksort again, this time focused on what the array looks like at each step.

![Diagram of Quicksort steps]

Sorted!
Divide and Conquer :: Merge Sort and Quick Sort

Both Merge Sort and QuickSort tend to be $O(n \times \log_2 n)$.

Why?