Quicksort
Sorting by Splitting

- 1960 C.A. Hoare
- An example of sorting by splitting
- \( W(n) = O(n^2) \) but \( A(n) = O(n \log n) \)
- In place
- Also a divide-and-conquer algorithm
What other types of Sorting algorithms are there?

• By Ranking – Rank sort
• By Swapping – Bubble sort, Odd-even sort
• By Selection – Selection, Tournament, Heap
• By Insertion - Insertion
• By Merging - Merge
• By Splitting - Quick
Quicksort Steps

**Divide**: Partition $A[p..r]$ into two (possibly empty) subarrays $A[p...q-1]$ and $A[q+1...r]$ such that each element of $A[p..q-1] \leq A[q] \leq A[q+1...r]$

**Conquer**: Sort the two subarrays recursively

**Combine**: No work needed to combine since subarrays are sorted in place.
Quicksort($A, p, r$)

1. if $p < r$
2. then $q \leftarrow \text{Partition}(A, p, r)$
3. Quicksort($A, p, q - 1$)
4. Quicksort($A, q + 1, r$)
Partitioning

**PARTITION**($A, p, r$)

$x = A[r]$

$i = p - 1$

for $j = p$ to $r - 1$

\[ \text{if } A[j] \leq x \]

\[ i = i + 1 \]

exchange $A[i]$ with $A[j]$

exchange $A[i + 1]$ with $A[r]$

**return** $i + 1$
<p>| | | | | | | |</p>
<table>
<thead>
<tr>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>i</td>
<td>p</td>
<td>j</td>
<td></td>
<td>r</td>
<td></td>
<td></td>
</tr>
<tr>
<td>(a)</td>
<td>2</td>
<td>8</td>
<td>7</td>
<td>1</td>
<td>3</td>
<td>5</td>
</tr>
<tr>
<td>(b)</td>
<td>2</td>
<td>8</td>
<td>7</td>
<td>1</td>
<td>3</td>
<td>5</td>
</tr>
<tr>
<td>(c)</td>
<td>2</td>
<td>8</td>
<td>7</td>
<td>1</td>
<td>3</td>
<td>5</td>
</tr>
<tr>
<td>(d)</td>
<td>2</td>
<td>8</td>
<td>7</td>
<td>1</td>
<td>3</td>
<td>5</td>
</tr>
<tr>
<td>(e)</td>
<td>2</td>
<td>1</td>
<td>7</td>
<td>8</td>
<td>3</td>
<td>5</td>
</tr>
<tr>
<td>(f)</td>
<td>2</td>
<td>1</td>
<td>3</td>
<td>8</td>
<td>7</td>
<td>5</td>
</tr>
<tr>
<td>(g)</td>
<td>2</td>
<td>1</td>
<td>3</td>
<td>8</td>
<td>7</td>
<td>5</td>
</tr>
<tr>
<td>(h)</td>
<td>2</td>
<td>1</td>
<td>3</td>
<td>8</td>
<td>7</td>
<td>5</td>
</tr>
<tr>
<td>(i)</td>
<td>2</td>
<td>1</td>
<td>3</td>
<td>4</td>
<td>7</td>
<td>5</td>
</tr>
</tbody>
</table>

**Figure 7.1** The operation of **PARTITION** on a sample array. Lightly shaded array elements are all in the first partition with values no greater than \( x \). Heavily shaded elements are in the second partition with values greater than \( x \). The unshaded elements have not yet been put in one of the first two partitions, and the final white element is the pivot. (a) The initial array and variable settings. None of the elements have been placed in either of the first two partitions. (b) The value 2 is “swapped with itself” and put in the partition of smaller values. (c)–(d) The values 8 and 7 are added to the partition of larger values. (e) The values 1 and 8 are swapped, and the smaller partition grows. (f) The values 3 and 8 are swapped, and the smaller partition grows. (g)–(h) The larger partition grows to include 5 and 6 and the loop terminates. (i) In lines 7–8, the pivot element is swapped so that it lies between the two partitions.
Partitioning

- As PARTITION runs, the array A is partitioned into 4 regions

![Diagram of partitioned array](image)

**Figure 7.2** The four regions maintained by the procedure PARTITION on a subarray $A[p..r]$. The values in $A[p..i]$ are all less than or equal to $x$, the values in $A[i+1..j-1]$ are all greater than $x$, and $A[r] = x$. The values in $A[j..r-1]$ can take on any values.
Running time

• How long does PARTITION take??

• $\Theta(n)$ where $n=r-p+1$
Analyzing Quicksort

• What is the worst case?
• When a bad partition creates one subproblem with \( n-1 \) elements and another with 0
• If this happens every time
  \[ T(n) = T(n-1) + T(0) + \Theta(n) \]
  \[ = T(n-1) + \Theta(n) \]
• Solve this recurrence to get \( T(n) = \Theta(n^2) \)
Analyzing Quicksort

• What is the best case?
• Perfect split!
• $T(n) = 2 \ T(n/2) + \Theta(n)$

• Solve this recurrence (case 2 Master Theorem)
$T(n)=O\ (n \ \log \ n)$
Analyzing Quicksort

• But what about the Average case?
• We will look at a formal analysis next, but let's begin by intuitively working this out.
• Let's try understanding how the balance of the partitioning changes the recurrence.
Analyzing Quicksort

- Suppose we always get a 9-to-1 split
  \[ T(n) = T(9n/10) + T(n/10) + cn \]

- What does this recursion tree look like?
Figure 7.4  A recursion tree for QUICKSORT in which PARTITION always produces a 9-to-1 split, yielding a running time of $O(n \log n)$. Nodes show subproblem sizes, with per-level costs on the right. The per-level costs include the constant $c$ implicit in the $\Theta(n)$ term.
Analyzing Quicksort

• But... But... a 9-to-1 split is supposed to be bad
• With any split of constant proportionality, the tree is of depth $\log n$
• Even a 99-to-1 split!
Intuition for the average case

- Random input array
- Unlikely that the partition happens the same way at every level
- Some partitions will be balanced and some will be very unbalanced
- Average case – mix of “good” and “bad” splits randomly scattered through the recursion tree
Intuition

• Lets assume good and bad splits alternate
Intuition

• Is this any worse than the balanced case?
Intuition

• Suppose we alternate, good, bad, good, ...

\[ G(n) = 2 \ B(n/2) + \Theta(n) \quad \text{... Good case} \]

\[ B(n) = G(n-1) + \Theta(n) \quad \text{... Bad case} \]

Substituting,

\[ G(n) = 2 \ (G(n/2-1) + \Theta(n/2)) + \Theta(n) \quad \text{.. Recurrence} \]

\[ = 2 \ G(n/2 -1) + \Theta(n) \]

\[ = \Theta(n \ lg \ n) \quad \text{Master’s Theorem} \]
So how can we avoid the worst case?

• What input causes it??
  – Sorted
  – Reverse sorted

• How to avoid getting these inputs?
Randomized Quicksort

- Randomly permute the input
  OR
- Randomly select the pivot

Randomization means:
- Running time independent of input
- No specific input creates worst case
- Worst case is determined only by a random number generator
\textbf{RANDOMIZED-PARTITION}(A, p, r)

1 \hspace{1em} i \leftarrow \text{RANDOM}(p, r)

2 \hspace{1em} \text{exchange} \ A[r] \leftrightarrow A[i]

3 \hspace{1em} \text{return} \ \text{PARTITION}(A, p, r)
RANDOMIZED-QUICKSORT \(A, p, r\)

1. if \(p < r\)
2. then \(q \leftarrow \text{RANDOMIZED-PARTITION}(A, p, r)\)
3. \text{RANDOMIZED-QUICKSORT}(A, p, q - 1)
4. \text{RANDOMIZED-QUICKSORT}(A, q + 1, r)
Analysis – in class