Design and Analysis of Algorithms

CSE 5311

Lecture 4  Quicksort

Song Jiang, Ph.D.
Department of Computer Science and Engineering
Quicksort

• Divide-and-conquer algorithm.
• Sorts “in place” (like insertion sort, but not like merge sort).
• Very practical (with tuning).
Divide and conquer

Quicksort an $n$-element array:

1. **Divide:** Partition the array into two subarrays around a *pivot* $x$ such that elements in lower subarray $\leq x \leq$ elements in upper subarray.

   $\leq x \quad x \quad \geq x$

2. **Conquer:** Recursively sort the two subarrays.

3. **Combine:** Trivial.

**Key:** *Linear-time partitioning subroutine.*
Partitioning subroutine

\[
\text{PARTITION}(A, p, q) \triangleright A[p \ldots q]
\]

\[x \leftarrow A[p]\] \triangleright \text{pivot} = A[p]

\[i \leftarrow p\]

\textbf{for} \ j \leftarrow p + 1 \ \textbf{to} \ q

\textbf{do} \ \textbf{if} \ A[j] \leq x

\textbf{then} \ i \leftarrow i + 1

\text{exchange} \ A[i] \leftrightarrow A[j]

\text{exchange} \ A[p] \leftrightarrow A[i]

\textbf{return} \ i

\textbf{Invariant:} \quad x \leq p \quad i \quad j \quad \geq q

\text{Running time} = \mathcal{O}(n) \text{ for } n \text{ elements.}
Example of partitioning

\[
\begin{array}{cccccccc}
6 & 10 & 13 & 5 & 8 & 3 & 2 & 11 \\
\end{array}
\]
Example of partitioning

6 10 13 5 8 3 2 11

\(i\) \[\rightarrow\] \(j\)
Example of partitioning

6  10  13  5  8  3  2  11

i  $\rightarrow$  j
Example of partitioning

\[ \begin{array}{ccccccccc}
6 & 10 & 13 & 5 & 8 & 3 & 2 & 11 \\
6 & 5 & 13 & 10 & 8 & 3 & 2 & 11
\end{array} \]

\[ i \quad j \]
Example of partitioning

<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>6</td>
<td>10</td>
<td>13</td>
<td>5</td>
<td>8</td>
<td>3</td>
<td>2</td>
<td>11</td>
</tr>
</tbody>
</table>

<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>6</td>
<td>5</td>
<td>13</td>
<td>10</td>
<td>8</td>
<td>3</td>
<td>2</td>
<td>11</td>
</tr>
</tbody>
</table>

\[ i \quad \longrightarrow \quad j \]
Example of partitioning
Example of partitioning
Example of partitioning
Example of partitioning

\[
\begin{array}{cccccccc}
6 & 10 & 13 & 5 & 8 & 3 & 2 & 11 \\
6 & 5 & 13 & 10 & 8 & 3 & 2 & 11 \\
6 & 5 & 3 & 10 & 8 & 13 & 2 & 11 \\
6 & 5 & 3 & 2 & 8 & 13 & 10 & 11 \\
\end{array}
\]
Example of partitioning

\[ \begin{array}{cccccccc}
6 & 10 & 13 & 5 & 8 & 3 & 2 & 11 \\
6 & 5 & 13 & 10 & 8 & 3 & 2 & 11 \\
6 & 5 & 3 & 10 & 8 & 13 & 2 & 11 \\
6 & 5 & 3 & 2 & 8 & 13 & 10 & 11 \\
\end{array} \]
Example of partitioning
Example of partitioning
Pseudocode for quicksort

\[
\text{QUICKSORT}(A, p, r)
\]

if \( p < r \)

then \( q \leftarrow \text{PARTITION}(A, p, r) \)

\[
\text{QUICKSORT}(A, p, q-1)
\]

\[
\text{QUICKSORT}(A, q+1, r)
\]

Initial call: \( \text{QUICKSORT}(A, 1, n) \)
Analysis of quicksort

• Assume all input elements are distinct.

• In practice, there are better partitioning algorithms for when duplicate input elements may exist.

• Let $T(n)$ = worst-case running time on an array of $n$ elements.
Worst-case of quicksort

- Input sorted or reverse sorted.
- Partition around min or max element.
- One side of partition always has no elements.

\[ T(n) = T(0) + T(n - 1) + \Theta(n) \]
\[ = \Theta(1) + T(n - 1) + \Theta(n) \]
\[ = T(n - 1) + \Theta(n) \]
\[ = \Theta(n^2) \quad (\text{arithmetic series}) \]
Worst-case recursion tree

\[ T(n) = T(0) + T(n-1) + cn \]
Worst-case recursion tree

\[ T(n) = T(0) + T(n-1) + cn \]
Worst-case recursion tree

\[ T(n) = T(0) + T(n-1) + cn \]
Worst-case recursion tree

\[ T(n) = T(0) + T(n-1) + cn \]
Worst-case recursion tree

\[ T(n) = T(0) + T(n-1) + cn \]
Worst-case recursion tree

\[ T(n) = T(0) + T(n-1) + cn \]

\[ \Theta \left( \sum_{k=1}^{n} k \right) = \Theta(n^2) \]
Worst-case recursion tree

\[ T(n) = T(0) + T(n-1) + cn \]

\[ h = n \]

\[ T(n) = \Theta(n) + \Theta(n^2) = \Theta(n^2) \]
Best-case analysis
(For intuition only!)

If we’re lucky, PARTITION splits the array evenly:

\[ T(n) = 2T(n/2) + \Theta(n) \]
\[ = \Theta(n \lg n) \quad \text{(same as merge sort)} \]

What if the split is always \( \frac{1}{10} : \frac{9}{10} \)?

\[ T(n) = T\left(\frac{1}{10} n\right) + T\left(\frac{9}{10} n\right) + \Theta(n) \]

What is the solution to this recurrence?
Analysis of “almost-best” case

\[ T(n) \]
Analysis of “almost-best” case

\[ T\left(\frac{1}{10}n\right) \quad cn \quad T\left(\frac{9}{10}n\right) \]
Analysis of “almost-best” case

\[ T\left(\frac{1}{100} n\right) T\left(\frac{9}{100} n\right) \]

\[ T\left(\frac{9}{100} n\right) T\left(\frac{81}{100} n\right) \]
Analysis of “almost-best” case

\[ \Theta(1) \]

\[ O(n) \text{ leaves} \]

\[ \log_{10/9} n \]

\[ cn \]

\[ \frac{1}{10} cn \]

\[ \frac{9}{10} cn \]

\[ \frac{9}{100} cn \]

\[ \frac{91}{100} cn \]

\[ \frac{81}{100} cn \]
Analysis of “almost-best” case

\[ T(n) = \begin{cases} 
\Theta(1) & \text{if } \log_{10} n \\
\Theta(n) \text{ leaves} & \text{if } \log_{10/9} n \\
\end{cases} \]

\[ cn \log_{10} n \leq T(n) \leq cn \log_{10/9} n + O(n) \]

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

\[ \text{Lucky!} \]
More intuition

Suppose we alternate lucky, unlucky, lucky, unlucky, lucky, …. 

\[ L(n) = 2U(n/2) + \Theta(n) \quad \text{lucky} \]

\[ U(n) = L(n - 1) + \Theta(n) \quad \text{unlucky} \]

Solving:

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

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

\[ = \Theta(n \lg n) \quad \text{Lucky!} \]

How can we make sure we are usually lucky?
Randomized quicksort

**Idea**: Partition around a *random* element.

- Running time is independent of the input order.
- No assumptions need to be made about the input distribution.
- No specific input elicits the worst-case behavior.
- The worst case is determined only by the output of a random-number generator.
Randomized quicksort analysis

Let $T(n) = \text{the random variable for the running time of randomized quicksort on an input of size } n$, assuming random numbers are independent.

For $k = 0, 1, \ldots, n-1$, define the **indicator random variable**

$$X_k = \begin{cases} 
1 & \text{if PARTITION generates a } k : n-k-1 \text{ split,} \\
0 & \text{otherwise.}
\end{cases}$$

$E[X_k] = \Pr\{X_k = 1\} = 1/n$, since all splits are equally likely, assuming elements are distinct.
Analysis (continued)

\[ T(n) = \begin{cases} 
T(0) + T(n-1) + \Theta(n) & \text{if } 0 : n-1 \text{ split}, \\
T(1) + T(n-2) + \Theta(n) & \text{if } 1 : n-2 \text{ split}, \\
\vdots \\
T(n-1) + T(0) + \Theta(n) & \text{if } n-1 : 0 \text{ split}, \\
\end{cases} \]

\[ = \sum_{k=0}^{n-1} X_k (T(k) + T(n - k - 1) + \Theta(n)) \]
Calculating expectation

\[ E[T(n)] = E \left[ \sum_{k=0}^{n-1} X_k (T(k) + T(n-k-1) + \Theta(n)) \right] \]

Take expectations of both sides.
Calculating expectation

\[ E[T(n)] = E \left[ \sum_{k=0}^{n-1} X_k (T(k) + T(n - k - 1) + \Theta(n)) \right] \]

\[ = \sum_{k=0}^{n-1} E[X_k (T(k) + T(n - k - 1) + \Theta(n))] \]

Linearity of expectation.
Calculating expectation

\[
E[T(n)] = E\left[ \sum_{k=0}^{n-1} X_k \left( T(k) + T(n-k-1) + \Theta(n) \right) \right]
\]

\[
= \sum_{k=0}^{n-1} E[X_k \left( T(k) + T(n-k-1) + \Theta(n) \right)]
\]

\[
= \sum_{k=0}^{n-1} E[X_k] \cdot E[T(k) + T(n-k-1) + \Theta(n)]
\]

Independence of \(X_k\) from other random choices.
Calculating expectation

\[
E[T(n)] = E\left[ \sum_{k=0}^{n-1} X_k (T(k) + T(n-k-1) + \Theta(n)) \right]
\]

\[
= \sum_{k=0}^{n-1} E[X_k (T(k) + T(n-k-1) + \Theta(n))]
\]

\[
= \sum_{k=0}^{n-1} E[X_k] \cdot E[T(k) + T(n-k-1) + \Theta(n)]
\]

\[
= \frac{1}{n} \sum_{k=0}^{n-1} E[T(k)] + \frac{1}{n} \sum_{k=0}^{n-1} E[T(n-k-1)] + \frac{1}{n} \sum_{k=0}^{n-1} \Theta(n)
\]

Linearity of expectation; \( E[X_k] = 1/n \).
Calculating expectation

\[
E[T(n)] = E\left[ \sum_{k=0}^{n-1} X_k (T(k) + T(n - k - 1) + \Theta(n)) \right]
\]

\[
= \sum_{k=0}^{n-1} E[X_k (T(k) + T(n - k - 1) + \Theta(n))]
\]

\[
= \sum_{k=0}^{n-1} E[X_k] \cdot E[T(k) + T(n - k - 1) + \Theta(n)]
\]

\[
= \frac{1}{n} \sum_{k=0}^{n-1} E[T(k)] + \frac{1}{n} \sum_{k=0}^{n-1} E[T(n - k - 1)] + \frac{1}{n} \sum_{k=0}^{n-1} \Theta(n)
\]

\[
= \frac{2}{n} \sum_{k=1}^{n-1} E[T(k)] + \Theta(n)
\]

Summations have identical terms.
Hairy recurrence

\[ E[T(n)] = 2^{n-1} \sum_{k=2}^{n} E[T(k)] + \Theta(n) \]

(The \( k = 0, 1 \) terms can be absorbed in the \( \Theta(n) \).)

**Prove:** \( E[T(n)] \leq a n \lg n \) for constant \( a > 0 \).

- Choose \( a \) large enough so that \( a n \lg n \) dominates \( E[T(n)] \) for sufficiently small \( n \geq 2 \).

**Use fact:** \( \sum_{k=2}^{n-1} k \lg k \leq \frac{1}{2} n^2 \lg n - \frac{1}{8} n^2 \) (exercise).
Substitution method

\[ E[T(n)] \leq 2 \sum_{k=2}^{n-1} a k \log k + \Theta(n) \]

Substitute inductive hypothesis.
Substitution method

\[ E[T(n)] \leq 2 \sum_{k=2}^{n-1} ak \lg k + \Theta(n) \]

\[ \leq \frac{2a}{n} \left( \frac{1}{2} n^2 \lg n - \frac{1}{8} n^2 \right) + \Theta(n) \]

Use fact.
Substitution method

\[
E[T(n)] \leq 2 \sum_{k=2}^{n-1} \frac{ak \lg k + \Theta(n)}{n} \\
\leq \frac{2a}{n} \left( \frac{1}{2} n^2 \lg n - \frac{1}{8} n^2 \right) + \Theta(n) \\
= an \lg n - \left( \frac{an}{4} - \Theta(n) \right)
\]

Express as \textit{desired} – \textit{residual}.
Substitution method

\[ E[T(n)] \leq \frac{2}{n} \sum_{k=2}^{n-1} ak \lg k + \Theta(n) \]

\[ = \frac{2a}{n} \left( \frac{1}{2} n^2 \lg n - \frac{1}{8} n^2 \right) + \Theta(n) \]

\[ = an \lg n - \left( \frac{an}{4} - \Theta(n) \right) \]

\[ \leq an \lg n, \]

if \( a \) is chosen large enough so that \( an/4 \) dominates the \( \Theta(n) \).
Quicksort in practice

• Quicksort is a great general-purpose sorting algorithm.

• Quicksort is typically over twice as fast as merge sort.

• Quicksort can benefit substantially from code tuning.

• Quicksort behaves well even with caching and virtual memory.