CSE 548 / AMS 542: Analysis of Algorithms

Prerequisites Review 3
( Deterministic Quicksort and Average-case Analysis )

Rezaul Chowdhury
Department of Computer Science
SUNY Stony Brook
Fall 2023
The Divide-and-Conquer Process in Merge Sort

Suppose we want to sort a typical subarray $A[p..r]$. 

**DIVIDE:** Split $A[p..r]$ at midpoint $q$ into two subarrays $A[p..q]$ and $A[q+1..r]$ of equal or almost equal length.

**CONQUER:** Recursively sort $A[p..q]$ and $A[q+1..r]$.

**COMBINE:** Merge the two sorted subarrays $A[p..q]$ and $A[q+1..r]$ to obtain a longer sorted subarray $A[p..r]$.

The **DIVIDE** step is cheap — takes only $\Theta(1)$ time.

But the **COMBINE** step is costly — takes $\Theta(n)$ time, where $n$ is the length of $A[p..r]$. 
The Divide-and-Conquer Process in Quicksort

Suppose we want to sort a typical subarray $A[p..r]$.

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

**Conquer:** Recursively sort $A[p..q - 1]$ and $A[q + 1..r]$.

**Combine:** Since $A[q]$ is “equal or larger” and “equal or smaller” than everything to its left and right, respectively, and both left and right parts are sorted, subarray $A[p..r]$ is also sorted.

The **Combine** step is cheap — takes only $\Theta(1)$ time.

But the **Divide** step is costly — takes $\Theta(n)$ time, where $n$ is the length of $A[p..r]$. 
Quicksort

Input: A subarray $A[p : r]$ of $r - p + 1$ numbers, where $p \leq r$.


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

1. *if* $p < r$ *then*
2. // partition $A[p..r]$ into $A[p..q - 1]$ and $A[q + 1..r]$ such that everything in $A[p..q - 1]$ is $\leq A[q]$ and everything in $A[q + 1..r]$ is $\geq A[q]$
3. $q = \text{PARTITION} (A, p, r)$
4. // recursively sort the left part
5. **QUICKSORT** ($A, p, q - 1$)
6. // recursively sort the right part
7. **QUICKSORT** ($A, q + 1, r$)
### Partition

**Input:**

\[ x = A[15] = 10 \]

- \( j = 1 \)
- \( i = 0 \)

<table>
<thead>
<tr>
<th>1</th>
<th>2</th>
<th>3</th>
<th>4</th>
<th>5</th>
<th>6</th>
<th>7</th>
<th>8</th>
<th>9</th>
<th>10</th>
<th>11</th>
<th>12</th>
<th>13</th>
<th>14</th>
<th>15</th>
</tr>
</thead>
<tbody>
<tr>
<td>9</td>
<td>1</td>
<td>13</td>
<td>20</td>
<td>4</td>
<td>15</td>
<td>19</td>
<td>6</td>
<td>12</td>
<td>6</td>
<td>3</td>
<td>17</td>
<td>15</td>
<td>5</td>
<td>10</td>
</tr>
</tbody>
</table>

- Item known to be \( \leq x \)
- Item known to be \( > x \)
- Item known to be at correct sorted location
- Item yet to be compared with \( x \)
Input: $A = 9, 1, 13, 20, 4, 15, 19, 6, 12, 6, 3, 17, 15, 5, 10$

Partition

$j = 1$

$A[j] \leq x$


$i = 0$

item known to be $\leq x$

item known to be $> x$

item known to be at correct sorted location

item yet to be compared with $x$
Input: \[ A = \begin{array}{ccccccccccccccc} 9 & 1 & 13 & 20 & 4 & 15 & 19 & 6 & 12 & 6 & 3 & 17 & 15 & 5 & 10 \end{array} \]

\[ j = 2 \]

\[ i = 1 \]

\[ x = A[15] = 10 \]

- item known to be $\leq x$
- item known to be $> x$
- item known to be at correct sorted location
- item yet to be compared with $x$
Input: $A = 9 1 13 20 4 15 19 6 12 6 3 17 15 5 10$

Partition

$j = 2$

$A[j] \leq x$


item known to be $\leq x$

item known to be $> x$

item known to be at correct sorted location

item yet to be compared with $x$
Input: \( A = [9, 1, 13, 20, 4, 15, 19, 6, 12, 6, 3, 17, 15, 5, 10] \)

\( j = 3 \)

\( i = 2 \)

\( x = A[15] = 10 \)

- item known to be \( \leq x \)
- item known to be \( > x \)
- item known to be at correct sorted location
- item yet to be compared with \( x \)
**Partition**

Input: \( A = [9, 1, 13, 20, 4, 15, 19, 6, 12, 6, 3, 17, 15, 5, 10] \)

- \( j = 3 \)
- \( A[j] > x \)
- \( x = A[15] = 10 \)

<table>
<thead>
<tr>
<th>1</th>
<th>2</th>
<th>3</th>
<th>4</th>
<th>5</th>
<th>6</th>
<th>7</th>
<th>8</th>
<th>9</th>
<th>10</th>
<th>11</th>
<th>12</th>
<th>13</th>
<th>14</th>
<th>15</th>
</tr>
</thead>
<tbody>
<tr>
<td>9</td>
<td>1</td>
<td>13</td>
<td>20</td>
<td>4</td>
<td>15</td>
<td>19</td>
<td>6</td>
<td>12</td>
<td>6</td>
<td>3</td>
<td>17</td>
<td>15</td>
<td>5</td>
<td>10</td>
</tr>
</tbody>
</table>

- Item known to be \( \leq x \)
- Item known to be \( > x \)
- Item known to be at correct sorted location
- Item yet to be compared with \( x \)
Partition

Input:\n\[ x = A[15] = 10 \]

\[ i = 2 \]
\[ j = 4 \]

- item known to be \( \leq x \)
- item known to be \( > x \)
- item known to be at correct sorted location
- item yet to be compared with \( x \)
Partition

Input:
\[ x = A_{15} = 10 \]
\[ j = 4 \]
\[ A[j] > x \]

\[ i = 2 \]

- Item known to be \( \leq x \)
- Item known to be \( > x \)
- Item known to be at correct sorted location
- Item yet to be compared with \( x \)
Partition

Input: $A = [9, 1, 13, 20, 4, 15, 19, 6, 12, 6, 3, 17, 15, 5, 10]$


$j = 5$

$i = 2$

item known to be $\leq x$

item known to be $> x$

item known to be at correct sorted location

item yet to be compared with $x$
Input: \[ A = 9 \quad 1 \quad 13 \quad 20 \quad 4 \quad 15 \quad 19 \quad 6 \quad 12 \quad 6 \quad 3 \quad 17 \quad 15 \quad 5 \quad 10 \]

- \( j = 5 \)
- \( A[j] \leq x \)
- \( i = 2 \)

- Item known to be \( \leq x \)
- Item known to be \( > x \)
- Item known to be at correct sorted location
- Item yet to be compared with \( x \)
Input: \( A \)

\[
\begin{align*}
\text{swap} \quad A[j] &\leq x \\
i &= 3 \\
j &= 5
\end{align*}
\]

The diagram illustrates the partition algorithm with the given input array. Each item is categorized into:
- **Item known to be \( \leq x \)**
- **Item known to be \( > x \)**
- **Item known to be at correct sorted location**
- **Item yet to be compared with \( x \)**
**Partition**

Input: \[ A = 9 \ 1 \ 4 \ 20 \ 13 \ 15 \ 19 \ 6 \ 12 \ 6 \ 3 \ 17 \ 15 \ 5 \ 10 \]

- \( i = 3 \)
- \( j = 5 \)
- \( x = A[15] = 10 \)
- Item known to be \( \leq x \)
- Item known to be \( > x \)
- Item known to be at correct sorted location
- Item yet to be compared with \( x \)
Input: \( A \)

\[
\begin{array}{cccccccccccccccc}
1 & 2 & 3 & 4 & 5 & 6 & 7 & 8 & 9 & 10 & 11 & 12 & 13 & 14 & 15 \\
\hline
9 & 1 & 4 & 20 & 13 & 15 & 19 & 6 & 12 & 6 & 3 & 17 & 15 & 5 & 10
\end{array}
\]

\( j = 6 \)

\( i = 3 \)

\( x = A[15] = 10 \)

- item known to be \( \leq x \)
- item known to be \( > x \)
- item known to be at correct sorted location
- item yet to be compared with \( x \)
Partition

Input: $A$

\[ j = 6 \]

$A[j] > x$


1 2 3 4 5 6 7 8 9 10 11 12 13 14 15

$\text{item known to be } \leq x$

$\text{item known to be } > x$

$\text{item known to be at correct sorted location}$

$\text{item yet to be compared with } x$
Input: $A = [9, 1, 4, 20, 13, 15, 19, 6, 12, 6, 3, 17, 15, 5, 10]$


$j = 7$

$i = 3$

- item known to be $\leq x$
- item known to be $> x$
- item known to be at correct sorted location
- item yet to be compared with $x$
Input: $\mathbf{A}$

\[
\begin{array}{ccccccccccccccc}
9 & 1 & 4 & 20 & 13 & 15 & 19 & 6 & 12 & 6 & 3 & 17 & 15 & 5 & 10 \\
\end{array}
\]

Partition

$j = 7$

$A[j] > x$

$i = 3$


- Item known to be $\leq x$
- Item known to be $> x$
- Item known to be at correct sorted location
- Item yet to be compared with $x$
**Partition**

Input: \[ A = \begin{array}{ccccccccccccc}
9 & 1 & 4 & 20 & 13 & 15 & 19 & 6 & 12 & 6 & 3 & 17 & 15 & 5 & 10
\end{array} \]

- \( x = A[15] = 10 \)
- \( j = 8 \)
- \( i = 3 \)

**Legend:**
- Item known to be \( \leq x \)
- Item known to be \( > x \)
- Item known to be at correct sorted location
- Item yet to be compared with \( x \)
Partition

Input: \( A \)

\[
\]

\( j = 8 \)

\( A[j] \leq x \)

...item known to be \( \leq x \)

...item known to be \( > x \)

...item known to be at correct sorted location

...item yet to be compared with \( x \)
Input: $A = [9, 1, 4, 20, 13, 15, 19, 6, 12, 6, 3, 17, 15, 5, 10]$


$i = 4$

$j = 8$

$A[j] \leq x$

Swap item known to be at correct sorted location with item yet to be compared with $x$.
Partition

Input: \[ A = [9, 1, 4, 6, 13, 15, 19, 20, 12, 6, 3, 17, 15, 5, 10] \]

\[ x = A[15] = 10 \]

\[ j = 8 \]

\[ i = 4 \]

- item known to be \( \leq x \)
- item known to be \( > x \)
- item known to be at correct sorted location
- item yet to be compared with \( x \)
**Partition**

Input: \( A = \begin{bmatrix} 9 & 1 & 4 & 6 & 13 & 15 & 19 & 20 & 12 & 6 & 3 & 17 & 15 & 5 & 10 \end{bmatrix} \)

- \( i = 4 \)
- \( j = 9 \)
- \( x = A[15] = 10 \)

- item known to be \( \leq x \)
- item known to be \( > x \)
- item known to be at correct sorted location
- item yet to be compared with \( x \)
Input: \( A \)

\[
\begin{array}{cccccccccccccc}
9 & 1 & 4 & 6 & 13 & 15 & 19 & 20 & 12 & 6 & 3 & 17 & 15 & 5 & 10 \\
\end{array}
\]

\( j = 9 \)

\( A[j] > x \)

\( i = 4 \)

\( x = A[15] = 10 \)

- Item known to be \( \leq x \)
- Item known to be \( > x \)
- Item known to be at correct sorted location
- Item yet to be compared with \( x \)
Input: \( A = [9, 1, 4, 6, 13, 15, 19, 20, 12, 6, 3, 17, 15, 5, 10] \)

\( x = A[15] = 10 \)

\( j = 10 \)

\( i = 4 \)

- Item known to be \( \leq x \)
- Item known to be \( > x \)
- Item known to be at correct sorted location
- Item yet to be compared with \( x \)
**Partition**

Input: \( A \)

<table>
<thead>
<tr>
<th>1</th>
<th>2</th>
<th>3</th>
<th>4</th>
<th>5</th>
<th>6</th>
<th>7</th>
<th>8</th>
<th>9</th>
<th>10</th>
<th>11</th>
<th>12</th>
<th>13</th>
<th>14</th>
<th>15</th>
</tr>
</thead>
<tbody>
<tr>
<td>9</td>
<td>1</td>
<td>4</td>
<td>6</td>
<td>13</td>
<td>15</td>
<td>19</td>
<td>20</td>
<td>12</td>
<td>6</td>
<td>3</td>
<td>17</td>
<td>15</td>
<td>5</td>
<td>10</td>
</tr>
</tbody>
</table>

- Item known to be \( \leq x \)
- Item known to be \( > x \)
- Item known to be at correct sorted location
- Item yet to be compared with \( x \)

\( i = 4 \)

\( j = 10 \)

\( A[j] \leq x \)

\( x = A[15] = 10 \)
Partition

Input: \(A\)

\[
\begin{align*}
A &= \begin{bmatrix}
9 & 1 & 4 & 6 & 13 & 15 & 19 & 20 & 12 & 6 & 3 & 17 & 15 & 5 & 10
\end{bmatrix}
\end{align*}
\]

\(i = 5\)

\(j = 10\)

\(A[j] \leq x\)

\(x = A[15] = 10\)

- Item known to be \(\leq x\)
- Item known to be \(> x\)
- Item known to be at correct sorted location
- Item yet to be compared with \(x\)
Input: \( A = [9, 1, 4, 6, 6, 15, 19, 20, 12, 13, 3, 17, 15, 5, 10] \)

\[ i = 5 \]

\[ j = 10 \]

\[ x = A[15] = 10 \]

- item known to be \( \leq x \)
- item known to be \( > x \)
- item known to be at correct sorted location
- item yet to be compared with \( x \)
Input:

\[ x = A[15] = 10 \]

\[ j = 11 \]

\[ i = 5 \]

**Item known to be \( \leq x \)**

**Item known to be \( > x \)**

**Item known to be at correct sorted location**

**Item yet to be compared with \( x \)**
Input: \( A \)

\[
A = [9, 1, 4, 6, 6, 15, 19, 20, 12, 13, 3, 17, 15, 5, 10]
\]

\( j = 11 \)
\( A[j] \leq x \)

\( x = A[15] = 10 \)

\( i = 5 \)

- item known to be \( \leq x \)
- item known to be \( > x \)
- item known to be at correct sorted location
- item yet to be compared with \( x \)
Partition

Input: $A$

\[x = A[15] = 10\]

\[j = 11\]

\[A[j] \leq x\]

\[i = 6\]

item known to be $\leq x$

item known to be $> x$

item known to be at correct sorted location

item yet to be compared with $x$
**Partition**

Input: $A = [9, 1, 4, 6, 6, 3, 19, 20, 12, 13, 15, 17, 15, 5, 10]$

- $i = 6$
- $j = 11$

- Item known to be $\leq x$
- Item known to be $> x$
- Item known to be at correct sorted location
- Item yet to be compared with $x$
**Partition**

Input: \( A \)

\[
\begin{align*}
\text{Input: } & \\
& 9 & 1 & 4 & 6 & 6 & 3 & 19 & 20 & 12 & 13 & 15 & 17 & 15 & 5 & 10 \\
\end{align*}
\]

\[
\begin{align*}
\text{Partition: } & \\
& j = 12 \\
& i = 6 \\
\end{align*}
\]

- item known to be \( \leq x \)
- item known to be \( > x \)
- item known to be at correct sorted location
- item yet to be compared with \( x \)
Input: $A = [9, 1, 4, 6, 6, 3, 19, 20, 12, 13, 15, 17, 15, 5, 10]$

$, j = 12$

$A[j] > x$


$i = 6$

item known to be $\leq x$

item known to be $> x$

item known to be at correct sorted location

item yet to be compared with $x$
Partition

Input: \( A = [9, 1, 4, 6, 6, 3, 19, 20, 12, 13, 15, 17, 15, 5, 10] \)

- \( i = 6 \)
- \( j = 13 \)
- \( x = A[15] = 10 \)

- Item known to be \( \leq x \)
- Item known to be \( > x \)
- Item known to be at correct sorted location
- Item yet to be compared with \( x \)
Input: \[ A = [9, 1, 4, 6, 6, 3, 19, 20, 12, 13, 15, 17, 15, 5, 10] \]

**Partition**

- \( i = 6 \)
- \( j = 13 \)
- \( A[j] > x \)
- \( x = A[15] = 10 \)

**Legend:**
- Item known to be \( \leq x \)
- Item known to be \( > x \)
- Item known to be at correct sorted location
- Item yet to be compared with \( x \)
Input: \[ A = 9 \ 1 \ 4 \ 6 \ 6 \ 3 \ 19 \ 20 \ 12 \ 13 \ 15 \ 17 \ 15 \ 5 \ 10 \]

\[ x = A[15] = 10 \]

\[ j = 14 \]

\[ i = 6 \]

- Item known to be \( \leq x \)
- Item known to be \( > x \)
- Item known to be at correct sorted location
- Item yet to be compared with \( x \)
Input: \( A \)

\[
\begin{array}{cccccccccccccc}
9 & 1 & 4 & 6 & 6 & 3 & 19 & 20 & 12 & 13 & 15 & 17 & 15 & 5 & 10 \\
\end{array}
\]

\( j = 14 \)

\( A[j] \leq x \)

\( x = A[15] = 10 \)

\( i = 6 \)

- item known to be \( \leq x \)
- item known to be \( > x \)
- item known to be at correct sorted location
- item yet to be compared with \( x \)
### Partition

Input: \[ A = [9, 1, 4, 6, 6, 3, 19, 20, 12, 13, 15, 17, 15, 5, 10] \]

- \( i = 7 \)
- \( j = 14 \)
- \( x = A[15] = 10 \)
- \( A[j] \leq x \)

**Legend:**
- Item known to be \( \leq x \)
- Item known to be \( > x \)
- Item known to be at correct sorted location
- Item yet to be compared with \( x \)
Input: \( A \) = 9 1 4 6 6 3 5 20 12 13 15 17 15 19 10

\( j = 14 \)

\( x = A[15] = 10 \)

\( i = 7 \)

**Partition**

- Item known to be \( \leq x \)
- Item known to be \( > x \)
- Item known to be at correct sorted location
- Item yet to be compared with \( x \)
Input: $A = [9, 1, 4, 6, 6, 3, 5, 20, 12, 13, 15, 17, 15, 19, 10]$


$i = 7$

- Item known to be $\leq x$
- Item known to be $> x$
- Item known to be at correct sorted location
- Item yet to be compared with $x$

\[ i = 7 \]

- Item known to be \( \leq x \)
- Item known to be \( > x \)
- Item known to be at correct sorted location
- Item yet to be compared with \( x \)
Input: \[ A = [9, 1, 4, 6, 6, 3, 5, 10, 12, 13, 15, 17, 15, 19, 20] \]

Partition

\[ i = 7 \]

- Item known to be \( \leq x \)
- Item known to be \( > x \)
- Item known to be at correct sorted location
- Item yet to be compared with \( x \)
Partition

Input:  

\[ A = [9, 1, 4, 6, 6, 3, 5, 10, 12, 13, 15, 17, 15, 19, 20] \]

\[ i = 7 \]
**Partition**

**Input:** A subarray $A[p : r]$ of $r - p + 1$ numbers, where $p \leq r$.

**Output:** Elements of $A[p : r]$ are rearranged such that for some $q \in [p, r]$ everything in $A[p : q - 1]$ is $\leq A[q]$ and everything in $A[q + 1 : r]$ is $\geq A[q]$. Index $q$ is returned.

```
PARTITION ( A, p, r )
1.   x = A[r]
2.   i = p - 1
3.   for j = p to r - 1
4.       if A[j] $\leq$ x
5.           i = i + 1
7.       exchange A[i + 1] with A[r]
8.   return i + 1
```
Running Time of Partition

**Input:** A subarray $A[p : r]$ of $r - p + 1$ numbers, where $p \leq r$.

**Output:** Elements of $A[p : r]$ are rearranged such that for some $q \in [p, r]$ everything in $A[p : q - 1]$ is $\leq A[q]$ and everything in $A[q + 1 : r]$ is $\geq A[q]$. Index $q$ is returned.

Let $n = r - p + 1$.

The loop of lines 3–6 takes $\Theta(r - 1 - p + 1) = \Theta(n)$ time.

Lines 1, 2, 7 and 8 take $\Theta(1)$ time each.

Hence, the overall running time is $\Theta(n)$. 

---

**Partition (A, p, r)**

1. $x = A[r]$
2. $i = p - 1$
3. *for* $j = p$ *to* $r - 1$
4. *if* $A[j] \leq x$
5. $i = i + 1$
8. *return* $i + 1$
Running Time of Partition

Input: A subarray $A[p : r]$ of $r - p + 1$ numbers, where $p \leq r$.


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

1. $x = A[r]$
2. $i = p - 1$
3. \textbf{for} $j = p \textbf{ to } r - 1$
4. \hspace{1em} \textbf{if} $A[j] \leq x$
5. \hspace{1.5em} $i = i + 1$
6. \hspace{1em} exchange $A[i]$ with $A[j]$
7. exchange $A[i + 1]$ with $A[r]$
8. \textbf{return} $i + 1$

Let $n = r - p + 1$.

The loop of lines 3–6 takes $\Theta(r - 1 - p + 1) = \Theta(n)$ time.

Lines 1, 2, 7 and 8 take $\Theta(1)$ time each.

Hence, the overall running time is $\Theta(n)$. 

49
Quicksort

Input:

<table>
<thead>
<tr>
<th></th>
<th>1</th>
<th>2</th>
<th>3</th>
<th>4</th>
<th>5</th>
<th>6</th>
<th>7</th>
<th>8</th>
<th>9</th>
<th>10</th>
<th>11</th>
<th>12</th>
<th>13</th>
<th>14</th>
<th>15</th>
</tr>
</thead>
<tbody>
<tr>
<td>A</td>
<td>9</td>
<td>1</td>
<td>13</td>
<td>20</td>
<td>4</td>
<td>15</td>
<td>19</td>
<td>6</td>
<td>12</td>
<td>6</td>
<td>3</td>
<td>17</td>
<td>15</td>
<td>5</td>
<td>10</td>
</tr>
</tbody>
</table>

item determined to be at correct sorted location
Quicksort

Partition \( A[1..15] \) around 10

Input:

\[ A = [9, 1, 13, 20, 4, 15, 19, 6, 12, 6, 3, 17, 15, 5, 10] \]

- Item determined to be at correct sorted location
### Quicksort

Partition $A[1..15]$ around 10

**Input:**

<table>
<thead>
<tr>
<th></th>
<th>1</th>
<th>2</th>
<th>3</th>
<th>4</th>
<th>5</th>
<th>6</th>
<th>7</th>
<th>8</th>
<th>9</th>
<th>10</th>
<th>11</th>
<th>12</th>
<th>13</th>
<th>14</th>
<th>15</th>
</tr>
</thead>
<tbody>
<tr>
<td>A</td>
<td>9</td>
<td>1</td>
<td>4</td>
<td>6</td>
<td>6</td>
<td>3</td>
<td>5</td>
<td><strong>10</strong></td>
<td>12</td>
<td>13</td>
<td>15</td>
<td>17</td>
<td>15</td>
<td>19</td>
<td>20</td>
</tr>
</tbody>
</table>

- Item determined to be at correct sorted location
Quicksort

Partition $A[1..7]$ around 5

Input: $A = \begin{array}{ccccccccccccc}
9 & 1 & 4 & 6 & 6 & 3 & \textbf{5} & 10 & 12 & 13 & 15 & 17 & 15 & 19 & 20
\end{array}$

- item determined to be at correct sorted location
**Quicksort**

Partition $A[1..7]$ around 5

Input: $A$

1 4 3 5 6 9 6 10 12 13 15 17 15 19 20

- **Green item:** determined to be at correct sorted location
Quicksort

Partition $A[1..3]$ around 3

Input: $A = [1, 4, 3, 5, 6, 9, 6, 10, 12, 13, 15, 17, 15, 19, 20]$

- item determined to be at correct sorted location
QuickSort

Input: A

Partition $A[1..3]$ around 3

item determined to be at correct sorted location
Quicksort

Input: A

A[1..1] is trivially sorted

item determined to be at correct sorted location
Quicksort

Input: $A$

$A[1..1]$ is trivially sorted

item determined to be at correct sorted location
**Quicksort**

Input: $A = [1, 3, 4, 5, 6, 9, 6, 10, 12, 13, 15, 17, 15, 19, 20]$

$A[3..3]$ is trivially sorted

item determined to be at correct sorted location
Quicksort

Input: $A = \begin{array}{c}
1 & 3 & 4 & 5 & 6 & 9 & 6 & 10 & 12 & 13 & 15 & 17 & 15 & 19 & 20 \\
\end{array}$

$A[3..3]$ is trivially sorted

Item determined to be at correct sorted location
**Quicksort**

Partition $A[5..7]$ around 6

Input: $A = [1, 3, 4, 5, 6, 9, 6, 10, 12, 13, 15, 17, 15, 19, 20]$

Item determined to be at correct sorted location
QuickSort

Input: A

Partition A[5..7] around 6

item determined to be at correct sorted location
QuickSort

Input: A

A[5..5] is trivially sorted

item determined to be at correct sorted location
**Quicksort**

Input: \( A \)

\[
\begin{array}{cccccccccccccc}
1 & 3 & 4 & 5 & 6 & 6 & 9 & 10 & 12 & 13 & 15 & 17 & 15 & 19 & 20 \\
\end{array}
\]

\( A[5..5] \) is trivially sorted

item determined to be at correct sorted location
A[7..7] is trivially sorted

Input: A

| 1 | 3 | 4 | 5 | 6 | 6 | 9 | 10 | 12 | 13 | 15 | 17 | 15 | 19 | 20 |

item determined to be at correct sorted location
Quicksort

Input: $A$

$A[7..7]$ is trivially sorted

item determined to be at correct sorted location
Quicksort


Input: $A = [1 \ 3 \ 4 \ 5 \ 6 \ 6 \ 9 \ 10 \ 12 \ 13 \ 15 \ 17 \ 15 \ 19 \ 20]$

item determined to be at correct sorted location
Quicksort


Input: $A$

1 3 4 5 6 6 9 10 12 13 15 17 15 19 20

item determined to be at correct sorted location
**Quicksort**

Input: $A = 1, 3, 4, 5, 6, 6, 9, 10, 12, 13, 15, 17, 15, 19, 20$

Partition $A[9..14]$ around 19

- item determined to be at correct sorted location
Quicksort

Partition $A[9..14]$ around 19

Input: $A = [1, 3, 4, 5, 6, 6, 9, 10, 12, 13, 15, 17, 15, 19, 20]$

- Item determined to be at correct sorted location
Quicksort

Partition $A[9..13]$ around 15

Input: $A = [1, 3, 4, 5, 6, 6, 9, 10, 12, 13, 15, 17, 15, 19, 20]$

item determined to be at correct sorted location
Quicksort

Partition $A[9..13]$ around 15

Input: $A = [1, 3, 4, 5, 6, 6, 9, 10, 12, 13, 15, 15, 17, 19, 20]$

item determined to be at correct sorted location
QuickSort

Partition \( A[9..11] \) around 15

Input: \[ A = \begin{array}{cccccccccccccccc}
1 & 3 & 4 & 5 & 6 & 6 & 9 & 10 & 12 & 13 & 15 & 15 & 17 & 19 & 20
\end{array} \]

item determined to be at correct sorted location
Quicksort

Input: A

Partition \( A[9..11] \) around 15

Item determined to be at correct sorted location
Quicksort

Partition $A[9..10]$ around 13

Input: $A$

1 2 3 4 5 6 6 9 10 12 13 15 15 17 19 20

item determined to be at correct sorted location
**Quicksort**

Input: A

| 1 | 3 | 4 | 5 | 6 | 6 | 9 | 10 | 12 | 13 | 15 | 15 | 17 | 19 | 20 |

Partition A[9..10] around 13

Item determined to be at correct sorted location
Quicksort

Input: $A$

| 1 | 3 | 4 | 5 | 6 | 6 | 9 | 10 | 12 | 13 | 15 | 15 | 17 | 19 | 20 |

$A[9..9]$ is trivially sorted

item determined to be at correct sorted location
Quicksort

Input: A

A[9..9] is trivially sorted

item determined to be at correct sorted location
Quicksort

Input: A

| 1 | 3 | 4 | 5 | 6 | 6 | 9 | 10 | 12 | 13 | 15 | 15 | 17 | 19 | 20 |

A[13..13] is trivially sorted

item determined to be at correct sorted location
Quicksort

Input: \(A\)

| 1 | 3 | 4 | 5 | 6 | 6 | 9 | 10 | 12 | 13 | 15 | 15 | 17 | 19 | 20 |

\(A[13..13]\) is trivially sorted

item determined to be at correct sorted location
Quicksort

Input: A

\[ A[1..15] \text{ is now fully sorted} \]

item determined to be at correct sorted location
Quicksort

**Input:** A subarray $A[ p : r ]$ of $r - p + 1$ numbers, where $p \leq r$.


Quicksort ($A, p, r$)

1. if $p < r$ then
2. 
   // partition $A[p..r]$ into $A[p..q - 1]$ and $A[q + 1..r]$ such that everything in $A[p..q - 1]$ is $\leq A[q]$ and everything in $A[q + 1..r]$ is $\geq A[q]$ 
3. $q =$ Partition ($A, p, r$)
4. 
   // recursively sort the left part 
5. Quicksort ($A, p, q - 1$)
6. 
   // recursively sort the right part 
7. Quicksort ($A, q + 1, r$)
Worst-case Running Time of Quicksort

Assuming \( n = r - p + 1 \), the worst-case running time of quicksort:

\[
T(n) = \begin{cases} 
\Theta(1) & \text{if } n = 1, \\
\max_{p\leq q\leq r} \{T(q - p) + T(r - q)\} + \Theta(n) & \text{if } n > 1.
\end{cases}
\]

Replacing \( q \) with \( k + p - 1 \), we get:

\[
T(n) = \begin{cases} 
\Theta(1) & \text{if } n = 1, \\
\max_{1\leq k\leq n} \{T(k - 1) + T(n - k)\} + \Theta(n) & \text{if } n > 1.
\end{cases}
\]
Worst-case Running Time of Quicksort (Upper Bound)

For \( n > 1 \) and a constant \( c > 0 \),

\[
T(n) = \max_{1 \leq k \leq n} \{T(k - 1) + T(n - k)\} + cn
\]

Our guess for upper bound: \( T(n) \leq c_1 n^2 \) for constant \( c_1 > 0 \).

Using this bound on the right side of the recurrence equation, we get.

\[
T(n) \leq \max_{1 \leq k \leq n} \left\{ c_1 (k - 1)^2 + c_1 (n - k)^2 \right\} + cn
\]

\[
\Rightarrow T(n) \leq c_1 \max_{1 \leq k \leq n} \left\{ (k - 1)^2 + (n - k)^2 \right\} + cn
\]

But \( (k - 1)^2 + (n - k)^2 \) reaches its maximum value for \( k = 1 \) and \( k = n \).

Hence,

\[
T(n) \leq c_1 ((1 - 1)^2 + (n - 1)^2) + cn
\]

\[
\Rightarrow T(n) \leq c_1 (n - 1)^2 + cn
\]

\[
\Rightarrow T(n) \leq c_1 n^2 - (c_1 (2n - 1) - cn)
\]
Worst-case Running Time of Quicksort (Upper Bound)

But for $c_1 \geq c$, we have,

$$c_1(2n - 1) \geq c(2n - 1)$$

$$\Rightarrow c_1(2n - 1) \geq 2cn - c$$

$$\Rightarrow c_1(2n - 1) - cn \geq cn - c$$

But $n \geq 1 \Rightarrow cn \geq c \Rightarrow cn - c \geq 0$, and thus

$$c_1(2n - 1) - cn \geq 0$$

$$\Rightarrow -(c_1(2n - 1) - cn) \leq 0$$

$$\Rightarrow c_1n^2 - (c_1(2n - 1) - cn) \leq c_1n^2$$

But $T(n) \leq c_1n^2 - (c_1(2n - 1) - cn)$.

Hence, $T(n) \leq c_1n^2$ for $c_1 \geq c$. 
For \( n > 1 \) and a constant \( c > 0 \),

\[
T(n) = \max_{1 \leq k \leq n} \{T(k - 1) + T(n - k)\} + cn
\]

Our guess for lower bound: \( T(n) \geq c_2 n^2 \) for constant \( c_2 > 0 \).

Using this bound on the right side of the recurrence equation, we get.

\[
T(n) \geq \max_{1 \leq k \leq n} \left\{c_2(k - 1)^2 + c_1(n - k)^2\right\} + cn
\]

\[
\Rightarrow T(n) \geq c_2 \max_{1 \leq k \leq n} \{(k - 1)^2 + (n - k)^2\} + cn
\]

But \((k - 1)^2 + (n - k)^2\) reaches its maximum value for \( k = 1 \) and \( k = n \).

Hence,

\[
T(n) \geq c_2 ((1 - 1)^2 + (n - 1)^2) + cn
\]

\[
\Rightarrow T(n) \geq c_2 (n - 1)^2 + cn
\]

\[
\Rightarrow T(n) \geq c_2 n^2 + (cn - c_2 (2n - 1))
\]
But for $c_2 \leq \frac{c}{2}$, we have,

$$c_2(2n - 1) \leq \frac{c}{2} (2n - 1)$$

$$\Rightarrow c_2(2n - 1) \leq cn - \frac{c}{2}$$

$$\Rightarrow cn - c_2(2n - 1) \geq \frac{c}{2}$$

But $c > 0$, and thus

$$cn - c_2(2n - 1) > 0$$

$$\Rightarrow c_2 n^2 + (cn - c_2(2n - 1)) > c_2 n^2$$

But $T(n) \geq c_2 n^2 + (cn - c_2(2n - 1))$.

Hence, $T(n) \geq c_2 n^2$ for $c_2 \leq \frac{c}{2}$. 

Worst-case Running Time of Quicksort (Lower Bound)
Worst-case Running Time of Quicksort (Tight Bound)

We have proved that

\[ T(n) \leq c_1 n^2 \text{ for } c_1 \geq c, \]

and \[ T(n) \geq c_2 n^2 \text{ for } c_2 \leq \frac{c}{2}. \]

Thus \[ c_2 n^2 \leq T(n) \leq c_1 n^2 \] for constants \( c_1 \geq c \) and \( c_2 \leq \frac{c}{2} \).

Hence, \( T(n) = \Theta(n^2) \).
Optional
Average Case Running Time of Quicksort
Average Case Running Time of Quicksort

\textbf{QUICKSORT} \ (A, p, r)

1. \textbf{if} \ p < r \ \textbf{then}
2. \ \ // partition \ A[p..r] \ into \ A[p..q-1]
   \ \ and \ A[q+1..r] \ such \ that \ everything
   \ \ in \ A[p..q-1] \ is \ \leq \ A[q] \ and \ everything
   \ \ in \ A[q+1..r] \ is \ \geq \ A[q]
3. \ q = \textbf{PARTITION} \ (A, p, r)
4. \ \ // \ recursively \ sort \ the \ left \ part
5. \ \textbf{QUICKSORT} \ (A, p, q-1)
6. \ \ // \ recursively \ sort \ the \ right \ part
7. \ \textbf{QUICKSORT} \ (A, q+1, r)

\[
T(n) = \begin{cases} 
\theta(1) & \text{if } n = 1, \\
\frac{1}{n} \sum_{1 \leq k \leq n} \{T(k-1) + T(n-k)} + \Theta(n) & \text{if } n > 1.
\end{cases}
\]
Average Case Running Time of Quicksort

For \( n > 1 \) and a constant \( c > 0 \),

\[
T(n) = \frac{1}{n} \sum_{1 \leq k \leq n} \{T(k - 1) + T(n - k)\} + cn
\]

\[
\Rightarrow nT(n) = \sum_{1 \leq k \leq n} \{T(k - 1) + T(n - k)\} + cn^2
\]

\[
\Rightarrow nT(n) = 2 \sum_{0 \leq k \leq n-1} T(k) + cn^2 \quad \cdots (1)
\]

Replacing \( n \) with \( n - 1 \),

\[
\Rightarrow (n - 1)T(n - 1) = 2 \sum_{0 \leq k \leq n-2} T(k) + c(n - 1)^2 \quad \cdots (2)
\]

Subtracting equation (2) from equation (1), we get

\[
nT(n) - (n - 1)T(n - 1) = 2T(n - 1) + c(2n - 1)
\]

\[
\Rightarrow nT(n) - (n + 1)T(n - 1) = c(2n - 1)
\]

Dividing both sides by \( n(n + 1) \), we get

\[
\frac{T(n)}{n+1} - \frac{T(n-1)}{n} = \frac{c(2n-1)}{n(n+1)}
\]
Average Case Running Time of Quicksort

Assuming \( \frac{T(n)}{n+1} = A(n) \), we get from the equation from the previous slide,

\[
A(n) - A(n - 1) = \frac{c(2n-1)}{n(n+1)}
\]

\[
\Rightarrow A(n) = A(n - 1) + \frac{c(2n-1)}{n(n+1)}
\]

\[
\Rightarrow A(n) = A(n - 1) + \frac{2c}{n+1} - \frac{c}{n(n+1)}
\]

\[
\Rightarrow A(n) < A(n - 1) + \frac{2c}{n+1}
\]

\[
\Rightarrow A(n) < A(n - 2) + \frac{2c}{n} + \frac{2c}{n+1}
\]

\[
\Rightarrow A(n) < A(n - 3) + \frac{2c}{n-1} + \frac{2c}{n} + \frac{2c}{n+1}
\]

\[
\Rightarrow A(n) < A(n - k) + \frac{2c}{n-k+2} + \frac{2c}{n-k+3} + \cdots + \frac{2c}{n} + \frac{2c}{n+1}
\]

\[
\Rightarrow A(n) < A(1) + \frac{2c}{3} + \frac{2c}{4} + \cdots + \frac{2c}{n} + \frac{2c}{n+1}
\]
Average Case Running Time of Quicksort

Since \(A(1) = \frac{T(1)}{2} = \Theta(1)\), we get,

\[
\Rightarrow A(n) < \Theta(1) + 2c \left( \frac{1}{3} + \frac{1}{4} + \cdots + \frac{1}{n} + \frac{1}{n+1} \right)
\]

\[
\Rightarrow A(n) < \Theta(1) + 2c \left( 1 + \frac{1}{2} + \frac{1}{3} + \cdots + \frac{1}{n} + \frac{1}{n+1} \right) - 2c \left( 1 + \frac{1}{2} \right)
\]

But \(H_{n+1} = 1 + \frac{1}{2} + \frac{1}{3} + \cdots + \frac{1}{n} + \frac{1}{n+1}\) is the \(n+1\)'st Harmonic Number, and \(\lim_{n \to \infty} H_{n+1} = \ln(n + 1) + \gamma\), where \(\gamma \approx 0.5772\) is known as the Euler-Mascheroni constant.

Hence, for \(n \to \infty\): \(A(n) < 2c(\ln(n + 1) + \gamma) - 3c + \Theta(1)\)

\[
\Rightarrow A(n) < 2c \ln(n + 1) + \Theta(1)
\]

\[
\Rightarrow \frac{T(n)}{n+1} < 2c \ln(n + 1) + \Theta(1)
\]

\[
\Rightarrow T(n) < 2c (n + 1) \ln(n + 1) + \Theta(n)
\]

\[
\Rightarrow T(n) = O(n \log n)
\]
[ Optional ]

Proof of Correctness of Partition
Correctness of Partition

**Input:** A subarray $A[ p : r ]$ of $r - p + 1$ numbers, where $p \leq r$.

**Output:** Elements of $A[ p : r ]$ are rearranged such that for some $q \in [p, r]$ everything in $A[ p : q - 1 ]$ is $\leq A[q]$ and everything in $A[ q + 1 : r ]$ is $\geq A[q]$. Index $q$ is returned.

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

1. $x = A[r]$
2. $i = p - 1$
3. for $j = p$ to $r - 1$
4. if $A[j] \leq x$
5. $i = i + 1$
7. exchange $A[i + 1]$ with $A[r]$
8. return $i + 1$

**Loop Invariant**

At the start of each iteration of the for loop of lines 3–6, for any array index $k$,

1. if $p \leq k \leq i$,
   
   then $A[k] \leq x$.

2. if $i + 1 \leq k \leq j - 1$,
   
   then $A[k] > x$.

3. if $k = r$,
   
   then $A[k] = x$.