Quicksort

Part 2: Proof of Correctness
What do we need to do?

1. Prove that PARTITION works
2. Prove that Quicksort works
Not a copy! (In-place)

(partition)

(doesn't match function from slides)
Not a copy!

(partition)

(base)

(base)

(base)

(base)

(base)

(base)

(base)

(base)

(base)

(base)

(base)

(base)

(base)

(base)

(base)

(base)

(base)

(base)

(base)

(base)

(base)

(base)

(base)

(base)

(base)

(base)

(base)

(base)

(base)

(base)

(base)

(base)

(base)

(base)

(base)

(base)

(base)

(base)

(base)

(base)

(base)

(base)

(base)

(base)

(base)

(base)

(base)

(base)

(base)

(base)

(base)

(base)

(base)

(base)

(base)

(base)

(base)

(base)

(base)

(base)

(base)

(base)

(base)

(base)

(base)

(base)

(base)

(base)

(base)

(base)

(base)

(base)

(base)

(base)

(base)

(base)

(base)

(base)

(base)

(base)

(base)

(base)

(base)

(base)

(base)

(base)

(base)

(base)

(base)

(base)

(base)

(base)

(base)

(base)

(base)

(base)

(base)

(base)

(base)

(base)

(base)

(base)

(base)

(base)

(base)

(base)

(base)

(base)

(base)

(base)

(base)

(base)

(base)

(base)

(base)

(base)

(base)

(base)

(base)

(base)

(base)

(base)

(base)

(base)

(base)

(base)

(base)

(base)

(base)

(base)

(base)

(base)

(base)

(base)

(base)

(base)

(base)

(base)

(base)

(base)

(base)

(base)

(base)

(base)

(base)

(base)

(base)

(base)

(base)

(base)

(base)

(base)

(base)

(base)

(base)

(base)

(base)

(base)

(base)

(base)

(base)

(base)

(base)

(base)

(base)

(base)

(base)

(base)

(base)

(base)

(base)

(base)

(base)

(base)

(base)

(base)

(base)

(base)

(base)

(base)

(base)

(base)

(base)

(base)

(base)

(base)

(base)

(base)

(base)

(base)

(base)

(base)

(base)

(base)

(base)

(base)

(base)

(base)

(base)

(base)

(base)

(base)

(base)

(base)

(base)

(base)

(base)

(base)

(base)

(base)

(base)

(base)

(base)

(base)

(base)

(base)

(base)

(base)

(base)

(base)

(base)

(base)

(base)

(base)

(base)

(base)

(base)

(base)

(base)

(base)

(base)

(base)

(base)

(base)

(base)

(base)

(base)

(base)

(base)

(base)

(base)

(base)

(base)

(base)

(base)

(base)

(base)

(base)

(base)

(base)

(base)

(base)

(base)

(base)

(base)

(base)

(base)

(base)

(base)

(base)

(base)

(base)

(base)

(base)

(base)

(base)

(base)

(base)

(base)

(base)

(base)

(base)

(base)

(base)

(base)

(base)

(base)

(base)

(base)

(base)

(base)

(base)

(base)

(base)

(base)

(base)

(base)

(base)

(base)

(base)

(base)

(base)

(base)

(base)

(base)

(base)

(base)

(base)

(base)

(base)

(base)

(base)

(base)

(base)

(base)

(base)

(base)

(base)

(base)

(base)

(base)

(base)

(base)

(base)

(base)

(base)

(base)

(base)

(base)

(base)

(base)

(base)

(base)

(base)

(base)

(base)

(base)

(base)

(base)

(base)

(base)

(base)

(base)

(base)

(base)

(base)

(base)

(base)

(base)

(base)

(base)

(base)

(base)

(base)

(base)

(base)

(base)

(base)

(base)

(base)

(base)

(base)

(base)

(base)

(base)

(base)

(base)

(base)

(base)

(base)

(base)

(base)

(base)

(base)

(base)

(base)

(base)

(base)

(base)

(base)

(base)

(base)

(base)

(base)

(base)

(base)

(base)

(base)

(base)

(base)

(base)

(base)

(base)

(base)

(base)

(base)

(base)

(base)

(base)

(base)

(base)

(base)

(base)

(base)

(base)

(base)

(base)

(base)
Not a copy!

11 23 31 51 89 67 91 47

(partition)

(base)

Not a copy!

11 23 31 51 89 67 91 47

(doesn't match function from slides)

(base)
Not a copy!

(base) 11 23 31 51 89 67 91 47

(partition) 11 23 31 51 89 67 91 47

(base) 11 23 31 51 89 67 91 47

Not a copy!

(base) 11 23 31 51 89 67 91 47

(base) 11 23 31 51 89 67 91 47

(base) 11 23 31 51 89 67 91 47

Not a copy!

(base) 11 23 31 51 89 67 91 47

(base) 11 23 31 51 89 67 91 47

(base) 11 23 31 51 89 67 91 47
Not a copy!

(base)

(partition)

Not a copy!

(base)
Not a copy!

(partition)

(base)

Not a copy!

(base)

(base)

(base)
Not a copy!

(partition)

(base)

(base)

(base)

(base)

(base)

(base)
Partition proof of correctness

```
PARTITION (A, l, r)
pivot = A[l]
i = l + 1
for j = (l + 1) .. r
  if A[j] < pivot
    swap A[j] and A[i]
i += 1
swap A[l] and A[i-1]
return i-1
```

How do we prove that PARTITION is correct?

**Loop Invariant:** At the beginning of each iteration of the for-loop the following conditions are met:

1. All items in $A[l + 1 .. i - 1]$ are $< pivot$
2. All items in $A[i .. j - 1]$ are $\geq pivot$

<table>
<thead>
<tr>
<th>Value</th>
<th>67</th>
<th>44</th>
<th>…</th>
<th>21</th>
<th>-87</th>
<th>…</th>
<th>5</th>
<th>101</th>
<th>…</th>
<th>4</th>
</tr>
</thead>
<tbody>
<tr>
<td>Index</td>
<td>0</td>
<td>1</td>
<td>…</td>
<td>left</td>
<td>left + 1</td>
<td>…</td>
<td>right - 1</td>
<td>right</td>
<td>…</td>
<td>n - 1</td>
</tr>
</tbody>
</table>
Partition Proof

**Loop Invariant:** At the beginning of each iteration of the *for-loop* the following conditions are met:
1. All items in $A[l + 1..i - 1]$ are $<$ pivot
2. All items in $A[i..j - 1]$ are $\geq$ pivot

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

- $pivot = A[l]$
- $i = l + 1$
- for $j = (l + 1) .. r$
  - if $A[j] < pivot$
    - swap $A[j]$ and $A[i]$
    - $i += 1$
- swap $A[l]$ and $A[i - 1]$
- return $i - 1$
**Initialization:**
1. No numbers in $A[l + 1 .. i - 1]$
2. No numbers in $A[i .. j - 1]$

**Loop Invariant:** At the beginning of each iteration of the for-loop the following conditions are met:
1. All items in $A[l + 1 .. i - 1]$ are $< \text{pivot}$
2. All items in $A[i .. j - 1]$ are $\geq \text{pivot}$

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

```
  pivot = A[l]
i = l + 1
  for j = (l + 1) .. r
      if A[j] < pivot
          swap A[j] and A[i]
i += 1
  swap A[l] and A[i - 1]
return i - 1
```
Partition Proof

Maintenance (case 1):
- Suppose conditions 1 and 2 are met.
- Now, suppose $A[j] < \text{pivot}$

Loop Invariant: At the beginning of each iteration of the for-loop the following conditions are met:
1. All items in $A[l + 1 .. i - 1]$ are $< \text{pivot}$
2. All items in $A[i .. j - 1]$ are $\geq \text{pivot}$

PARTITION ($A, l, r$)
pivot = $A[l]$
i = $l + 1$
for $j = (l + 1) .. r$
  if $A[j] < \text{pivot}$
    swap $A[j]$ and $A[i]$
    $i += 1$
  true
swap $A[l]$ and $A[i - 1]$
return $i - 1$
**Partition Proof**

**Loop Invariant:** At the beginning of each iteration of the for-loop the following conditions are met:
1. All items in $A[l + 1 .. i - 1]$ are $< pivot$
2. All items in $A[i .. j - 1]$ are $\geq pivot$

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

- $pivot = A[l]$
- $i = l + 1$

for $j = (l + 1) .. r$

true

if $A[j] < pivot$

swaps $A[j]$ and $A[i]$

$i += 1$

swaps $A[l]$ and $A[i - 1]$

return $i - 1$

**Maintenance (case 1):**

- Suppose conditions 1 and 2 are met.
- Now, suppose $A[j] < pivot$
- Then $A[j]$ and $A[i]$ are swapped
- By (2), $A[i]$ was $> pivot$ so now $A[i] < pivot$ and $A[j] > pivot$
Maintenance (case 1):
- Suppose conditions 1 and 2 are met.
- Now, suppose $A[j] < \text{pivot}$
- Then $A[j]$ and $A[i]$ are swapped
- By (2), $A[i]$ was $> \text{pivot}$ so now $A[i] < \text{pivot}$ and $A[j] > \text{pivot}$

**Loop Invariant:** At the beginning of each iteration of the for-loop the following conditions are met:
1. All items in $A[l + 1 .. i - 1]$ are $< \text{pivot}$
2. All items in $A[i .. j - 1]$ are $\geq \text{pivot}$

**PARTITION** $(A, l, r)$
- $\text{pivot} = A[l]$
- $i = l + 1$
- **for** $j = (l + 1) .. r$
  - **true** if $A[j] < \text{pivot}$
  - swap $A[j]$ and $A[i]$
  - $i += 1$
  - swap $A[l]$ and $A[i - 1]$
return $i - 1$
Partition Proof

Partition Proof

Maintenance (case 1):

• Suppose conditions 1 and 2 are met.
• Now, suppose A[j] < pivot
• Then A[j] and A[i] are swapped
• By (2), A[i] was > pivot so now
• Incrementing i and j satisfies 1 and 2

Loop Invariant: At the beginning of each iteration of the for-loop the following conditions are met:
1. All items in A[l + 1 .. i - 1] are < pivot
2. All items in A[i .. j - 1] are ≥ pivot

PARTITION (A, l, r)
pivot = A[l]
i = l + 1
for j = (l + 1) .. r
  if A[j] < pivot
    swap A[j] and A[i]
i += 1
  swap A[l] and A[i - 1]
return i - 1
Partition Proof

Maintenance (case 2):
• Suppose conditions 1 and 2 are met.
• Now, suppose A[j] >= pivot

Loop Invariant: At the beginning of each iteration of the for-loop the following conditions are met:
1. All items in A[l + 1 .. i - 1] are < pivot
2. All items in A[i .. j - 1] are ≥ pivot

PARTITION (A, l, r)
  pivot = A[l]
  i = l + 1
  for j = (l + 1) .. r
  if A[j] < pivot
     swap A[j] and A[i]
     i += 1
  swap A[l] and A[i - 1]
  return i - 1
Partition Proof

Maintenance (case 2):
• Suppose conditions 1 and 2 are met.
• Now, suppose $A[j] \geq pivot$
• We do not change $i$ so (1) holds

Loop Invariant: At the beginning of each iteration of the for-loop the following conditions are met:
1. All items in $A[l + 1 .. i - 1]$ are $< pivot$
2. All items in $A[i .. j - 1]$ are $\geq pivot$

PARTITION ($A, l, r$)
$$pivot = A[l]$$
$$i = l + 1$$
$$for \ j = (l + 1) .. r$$
$$false \ if \ A[j] < pivot$$
$$\ swap \ A[j] \ and \ A[i]$$
$$i += 1$$
$$\ swap \ A[l] \ and \ A[i - 1]$$
return $i - 1$
Partition Proof

Maintenance (case 2):
- Suppose conditions 1 and 2 are met.
- Now, suppose $A[j] \geq pivot$
- We do not change $i$ so (1) holds
- We increment $j$ so (2) holds

Loop Invariant: At the beginning of each iteration of the for-loop the following conditions are met:
1. All items in $A[l + 1 .. i - 1]$ are $< pivot$
2. All items in $A[i .. j - 1]$ are $\geq pivot$

PARTITION $(A, l, r)$
- $pivot = A[l]$
- $i = l + 1$
- $for$ $j = (l + 1) .. r$
  - $false$ $if$ $A[j] < pivot$
  - swap $A[j]$ and $A[i]$
    - $i += 1$
  - swap $A[l]$ and $A[i - 1]$
  - return $i - 1$
Partition Proof

**Partition (A, l, r)**

\[ \text{pivot} = A[l] \]

\[ i = l + 1 \]

\[ \text{for } j = (l + 1) \ldots r \]

\[ \text{if } A[j] < \text{pivot} \]

\[ \text{swap } A[j] \text{ and } A[i] \]

\[ i += 1 \]

\[ \text{swap } A[l] \text{ and } A[i - 1] \]

\[ \text{return } i - 1 \]

**Loop Invariant:** At the beginning of each iteration of the for-loop the following conditions are met:

1. All items in \( A[l + 1 \ldots i - 1] \) are < pivot
2. All items in \( A[i \ldots j - 1] \) are ≥ pivot

**Termination:**

- Now \( j = r + 1 \)
- All items have been considered
- All items in \( A[l + 1 \ldots i - 1] \) are < pivot
- All items in \( A[i \ldots j - 1] \) are ≥ pivot
### Partition Proof

#### PARTITION \((A, l, r)\)

1. \(\text{pivot} = A[1]\)
2. \(i = l + 1\)
3. \(\text{for } j = (l + 1) \ldots r\)
   1. \(\text{if } A[j] < \text{pivot}\)
      1. \(\text{swap } A[j] \text{ and } A[i]\)
      2. \(i += 1\)
   4. \(\text{swap } A[l] \text{ and } A[i - 1]\)
5. \(\text{return } i - 1\)

---

#### Loop Invariant:

At the beginning of each iteration of the for-loop the following conditions are met:

1. All items in \(A[l + 1 \ldots i - 1]\) are < pivot
2. All items in \(A[i \ldots j - 1]\) are ≥ pivot

---

After the loop we do the final swap
What do we need to do?

1. Prove that PARTITION works
   • Proof by loop invariant

2. Prove that Quicksort works
   • Proof by induction
Proof by Induction in General

Some property $P$ that we want to prove
• A **base case**: some statement regarding $P(1)$
• An **inductive hypothesis**: assume we know that $P(n)$ is true
• An **inductive step**: if $P(n)$ is correct then so is $P(n+1)$ because...

For quicksort we are going to use a slightly different form
 • If $P(k)$ where $k < n$ is correct then $P(n)$ is also correct
• An **inductive hypothesis**: assume we know that $P(n-1)$ is true
• An **inductive step**: if $P(n-1)$ is correct then so is $P(n)$
Proof by Induction Cheat-sheet

Proof by induction that $P(n)$ holds for all $n$

1. $P(1)$ holds because <something about the code/problem>
2. Let’s assume that $P(k)$ (where $k < n$) holds.
3. $P(n)$ holds because of $P(k)$ and <something about the code>
4. Thus, by induction, $P(n)$ holds for all $n$
Quicksort Proof

P(n) = Quicksort is always correct for arrays of length n.

• P(1) is an array of one element, and any such array is always sorted.
• Assume (hypothesis) that P(k) is correct for k < n
• P(n) holds because:

Proof by induction that P(n) holds for all n

• P(1) holds because ...
• Let’s assume that P(k) (where k < n) holds.
• P(n) holds because of P(k) and ...
• Thus, by induction, P(n) holds for all n
Quicksort Proof

P(n) = Quicksort is always correct for arrays of length n.
- P(1) is an array of one element, and any such array is always sorted.
- Assume (hypothesis) that P(k) is correct for k < n
- P(n) holds because:
  - Let $k_{\text{left}}$, $k_{\text{right}}$ = the lengths of the left and right subarrays
  - $k_{\text{left}}$, $k_{\text{right}}$ < n (strictly less than n)
  - By our hypothesis, the left and right subarrays are correctly sorted
  - The partition loop-invariant guarantees that the pivot is in the correct spot
- Thus, by induction, P(n) holds for all n

Proof by induction that P(n) holds for all n
- P(1) holds because ...
- Let's assume that P(k) (where k < n) holds.
- P(n) holds because of P(k) and ...
- Thus, by induction, P(n) holds for all n
Quicksort Proof

\[ P(n) = \text{Quicksort is always correct for arrays of length } n. \]

- \( P(1) \) is an array of one element, and any such array is always sorted. **Base case**

- Assume (hypothesis) that \( P(k) \) is correct for \( k < n \) **Inductive Hypothesis**

- \( P(n) \) holds because:
  - Let \( k_{\text{left}} \), \( k_{\text{right}} \) = the lengths of the left and right subarrays
  - \( k_{\text{left}}, k_{\text{right}} < n \) (strictly less than \( n \))
  - By our hypothesis, the left and right subarrays are correctly sorted
  - The partition loop-invariant guarantees that the pivot is in the correct spot **Inductive Step**

- Thus, by induction, \( P(n) \) holds for all \( n \)

Proof by induction that \( P(n) \) holds for all \( n \)
- \( P(1) \) holds because ...
- Let’s assume that \( P(k) \) (where \( k < n \)) holds.
- \( P(n) \) holds because of \( P(k) \) and ...

\[ < P \quad P \quad > P \]
What do we need to do?

1. Prove that PARTITION works
   • Proof by loop invariant

2. Prove that Quicksort works
   • Proof by induction