In-place Algorithm
Motivation

• Some devices don’t have enough space
  - Embedded system like PDA, cell phone......

• I/O spends much more time than calculation, and less space usually means fewer I/O
  - Database

• Reducing space usage is important
Simple Reverse

- Problem definition:
  - Given an array $A[0..n]$
  - Output the “reverse” of $A$
    - That is, output an array $B[0..n]$ such that $B[k] = A[n-k]$ for every $k$
  - In this problem, we are not required to keep $A$ after the processing
Simple Reverse

- Solution 1: Use a new array with size equal to the input array size
Simple Reverse

1 2 3 4 5 6 7

7
Simple Reverse

1 2 3 4 5 6 7

7 6
Simple Reverse

1 2 3 4 5 6 7

7 6 5
Simple Reverse

1 2 3 4 5 6 7

7 6 5 4
Simple Reverse

\[
\begin{array}{cccccccc}
1 & 2 & 3 & 4 & 5 & 6 & 7 \\
7 & 6 & 5 & 4 & 3 & 2 \\
\end{array}
\]
Simple Reverse
Simple Reverse

• Needs $O(n)$ extra space

• Can we use less space?
Smarter Reverse

• Solution 2: Exchange the first and the last elements (inside A), and then serve the remaining list
Smarter Reverse

1 2 3 4 5 6 7
Smarter Reverse

```
7  2  3  4  5  6  1
```
Smarter Reverse
Smarter Reverse

7 6 5 4 3 2 1
Smarter Reverse

```
7 6 5 4 3 2 1
```
Smarter Reverse

• Needs $O(1)$ extra space
  - One for exchanging elements
What is In-place Algorithm?

- Algorithm that uses a small constant amount of extra space in addition to the original input
- Usually overwrite the input space
  - Spend more time in some cases
- On the contrary: not-in-place or out-of-place
More Examples

• Do we know any algorithms which are in-place?
  - Insertion sort
  - Selection sort
Insertion Sort

4  2  3  6  7  1  8
Insertion Sort

4 2 3 6 7 1 8
Insertion Sort
Insertion Sort

2  3  4  6  7  1  8
Insertion Sort

\[\begin{array}{|c|c|c|c|c|c|c|}
\hline
2 & 3 & 4 & 6 & 7 & 1 & 8 \\
\hline
\end{array}\]
Insertion Sort

\[
\begin{array}{cccccccc}
2 & 3 & 4 & 6 & 7 & 1 & 8 \\
\end{array}
\]
Insertion Sort

1 2 3 4 6 7 8
Insertion Sort

1 2 3 4 6 7 8
Insertion Sort

- Only needs $O(1)$ extra space
  - One for exchanging

1 2 3 4 6 7 8
Selection Sort

• How about Selection Sort?

• Needs only $O(1)$ extra space
  - For exchanging
Not-in-place Algorithm

• Do we know any algorithms which are not-in-place?
  - Merge sort
    • $O(n)$ extra space for merging
Simple Merge Sort

Input array

Unsorted subarray1

Unsorted subarray2

Sorted subarray1

Sorted subarray2

Sorted array

Merge sort

Merge sort

divide

combine
What’s wrong in simple?

- In the merge step

<table>
<thead>
<tr>
<th>2</th>
<th>6</th>
<th>8</th>
<th>9</th>
<th>1</th>
<th>3</th>
<th>4</th>
<th>7</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>1</td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

1
What’s wrong in simple?

- In the merge step

```
2 6 8 9
1 3 4 7
1 2
```
What’s wrong in simple?

• In the merge step

<table>
<thead>
<tr>
<th>2</th>
<th>6</th>
<th>8</th>
<th>9</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>3</td>
<td>4</td>
<td>7</td>
</tr>
</tbody>
</table>

| 1 | 2 | 3 |   |   |   |
What’s wrong in simple?

• In the merge step

<table>
<thead>
<tr>
<th>2</th>
<th>6</th>
<th>8</th>
<th>9</th>
</tr>
</thead>
</table>

| 1 | 3 | 4 | 7 |

| 1 | 2 | 3 | 4 |   |   |   |
What’s wrong in simple?

- In the merge step

<table>
<thead>
<tr>
<th>2</th>
<th>6</th>
<th>8</th>
<th>9</th>
<th>1</th>
<th>3</th>
<th>4</th>
<th>7</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>2</td>
<td>3</td>
<td>4</td>
<td>6</td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
What’s wrong in simple?

• In the merge step

<table>
<thead>
<tr>
<th>2</th>
<th>6</th>
<th>8</th>
<th>9</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>3</td>
<td>4</td>
<td>7</td>
</tr>
</tbody>
</table>

| 1 | 2 | 3 | 4 | 6 | 7 |
What’s wrong in simple?

• In the merge step

```
  2 6 8 9
```

```
  1 3 4 7
```

```
  1 2 3 4 6 7 8
```
What’s wrong in simple?

• In the merge step

<table>
<thead>
<tr>
<th>2</th>
<th>6</th>
<th>8</th>
<th>9</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>3</td>
<td>4</td>
<td>7</td>
</tr>
</tbody>
</table>

| 1 | 2 | 3 | 4 | 6 | 7 | 8 | 9 |
What’s wrong in simple?

- In the merge step, needs $O(n)$ extra space

```
2 6 8 9
1 3 4 7
1 2 3 4 6 7 8 9
```
MergeSort2

• We design a new function called “inplaceMerge”
• We design a new function called “inplaceMerge”
We design a new function called "inplaceMerge"
• We design a new function called “inplaceMerge”
• We design a new function called “inplaceMerge”
MergeSort2

• We design a new function called "inplaceMerge"
We design a new function called "inplaceMerge"
MergeSort2

• We design a new function called “inplaceMerge”
- We design a new function called "inplaceMerge"
MergeSort2

- We design a new function called “inplaceMerge”
MergeSort2

- We design a new function called "inplaceMerge"
MergeSort2

- We design a new function called "inplaceMerge"
MergeSort2

- We design a new function called “inplaceMerge”
MergeSort2

• We design a new function called “inplaceMerge”
We design a new function called “inplaceMerge”
MergeSort2

• We design a new function called “inplaceMerge”
• We design a new function called "inplaceMerge"
MergeSort2

- We design a new function called "inplaceMerge"
MergeSort2

- We design a new function called "inplaceMerge"
MergeSort2

- We design a new function called "inplaceMerge"
MergeSort2

• We design a new function called “inplaceMerge”
• Time complexity of inplaceMerge: $O(n^2)$
MergerSort2

• Replace the merge function in simple merge with `inplaceMerge`

• Time complexity:
  - $T(n) = 2T(n/2) + n^2$

    By Master theory, $T(n) = \Theta(n^2)$
**MergerSort2**

- Replace the merge function in simple merge with `inplaceMerge`

- Is the algorithm an in-place algorithm?
  - NO, because we recurrently call function
    - It require $O(\log n)$ function call
In-place MergeSort

- So, we re-design the merge sort algorithm

\[ n = 2^m \]
In-place MergeSort

- So, we re-design the merge sort algorithm

$n = 2^m$
In-place MergeSort

• So, we re-design the merge sort algorithm

\[ n = 2^m \]
In-place Merge Sort

- Time complexity:
  \[ \frac{n}{2} \cdot O(2^2) + \frac{n}{4} \cdot O(4^2) + \ldots + 1 \cdot O(n^2) \]
  
  \[ = O(2n) + O(4n) + O(8n) + \ldots + O(n^2) \]
  
  \[ = O(n^2) \]

  ➔ still the same as MergeSort2, but avoid using \( O(\log n) \) function calls
In-place Merge Sort

• Can we do better?

• In fact, there is an In-place Merge Sort algorithm that works faster, using only optimal $O(n \log n)$ time
  - The merging step is a bit complicated, so we do not introduce here ...