File Edit View Help

File Edit View Help

Searching, Sorting, Complexity Part 02

File Edit View Help

Efficiency

- Efficiency
  - Producing desired results with little to no waste
  - Well organized and prevents wasteful use of a resource
- Resources
  - Time
  - Space
- How do we measure efficiency?
  - Algorithms do not require computers

File Edit View Help

Complexity

- Complexity
  - Classifies Computational Problems based on inherent difficulty
  - Relates problems to each other
  - Time and Space
- Asymptotic Analysis
  - A way to describe a *limiting* behavior / function
  - Limits in math are a value that a function *approaches* as the input *approaches* some value
  - Time and Space Complexity

File Edit View Help

Big O Notation

- Theoretical upper bound of an algorithm
- The "Worst Case" scenario
- Let f and g be functions defined on some subset of real numbers

$$f(n) = O(g(n))\ where\ n \in \mathbb{R}\ \text{as}\ n \to \infty$$

- Let M be a constant that's sufficiently large then we can say

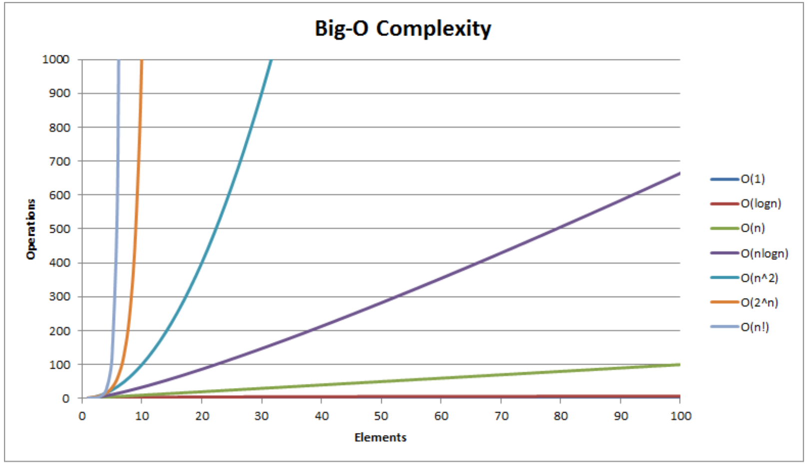
$$|f(n)| \leq M|g(n)|\ \ for\ all\ n \geq n_0$$

File Edit View Help

Big O Notation

- Common Big O Complexities
  - $O(1)$ – Constant
  - $O(\log(n))$ – Logarithmic
  - $O(n)$ – Linear
  - $O(n\log n)$ – Linearithmic
  - $O(n^2)$ – Quadratic
  - $O(2^n)$ – Exponential “Bad”
  - $O(n!)$ – Factorial “Really Bad”

Big O



File Edit View Help

Sorting Algorithms

- Problem:
  - Given any array of integers, develop an algorithm that sorts the values from smallest to largest.
- Selection Sort
  1. Start from index 0
  2. Assume the starting index has the smallest value and record that index
  3. Sequentially check every other value
  4. If a value is found that is smaller at another index, then record that current index
  5. Once all values have been checked if the recorded index does not match the current index then swap those values
  6. Increase the starting index by 1
  7. Repeat 2 through 6 until the staring index >= length

Example

Index	0	1	2	3	4	5	6	7
Value	10	8	7	6	12	5	11	9

File Edit View Help

Sorting Algorithms

- Problem:
  - Given any array of integers, develop an algorithm that sorts the values from smallest to largest.
- Selection Sort
  1. Start from index 0
  2. Assume the starting index has the smallest value and record that index
  3. Sequentially check every other value
  4. If a value is found that is smaller at another index, then record that current index
  5. Once all values have been checked if the recorded index does not match the current index then swap those values
  6. Increase the starting index by 1
  7. Repeat 2 through 6 until the staring index >= length

Example

Index	0	1	2	3	4	5	6	7
Value	10	8	7	6	12	5	11	9

File Edit View Help

Sorting Algorithms

- Problem:
  - Given any array of integers, develop an algorithm that sorts the values from smallest to largest.
- Selection Sort
  1. Start from index 0
  2. Assume the starting index has the smallest value and record that index
  3. Sequentially check every other value
  4. If a value is found that is smaller at another index, then record that current index
  5. Once all values have been checked if the recorded index does not match the current index then swap those values
  6. Increase the starting index by 1
  7. Repeat 2 through 6 until the staring index >= length

Example

Start

Smallest

Index	0	1	2	3	4	5	6	7
Value	10	8	7	6	12	5	11	9

File Edit View Help

Sorting Algorithms

- Problem:
  - Given any array of integers, develop an algorithm that sorts the values from smallest to largest.
- Selection Sort
  1. Start from index 0
  2. Assume the starting index has the smallest value and record that index
  3. Sequentially check every other value
  4. If a value is found that is smaller at another index, then record that current index
  5. Once all values have been checked if the recorded index does not match the current index then swap those values
  6. Increase the starting index by 1
  7. Repeat 2 through 6 until the staring index >= length

Example

Start

Smallest

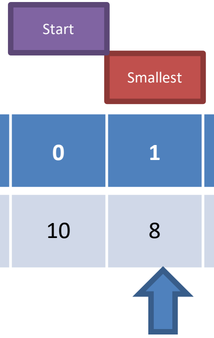
Index	0	1	2	3	4	5	6	7
Value	10	8	7	6	12	5	11	9

File Edit View Help

Sorting Algorithms

- Problem:
  - Given any array of integers, develop an algorithm that sorts the values from smallest to largest.
- Selection Sort
  1. Start from index 0
  2. Assume the starting index has the smallest value and record that index
  3. Sequentially check every other value
  4. If a value is found that is smaller at another index, then record that current index
  5. Once all values have been checked if the recorded index does not match the current index then swap those values
  6. Increase the starting index by 1
  7. Repeat 2 through 6 until the staring index >= length

Example



Index	0	1	2	3	4	5	6	7
Value	10	8	7	6	12	5	11	9

File Edit View Help

Sorting Algorithms

- Problem:
  - Given any array of integers, develop an algorithm that sorts the values from smallest to largest.
- Selection Sort
  1. Start from index 0
  2. Assume the starting index has the smallest value and record that index
  3. Sequentially check every other value
  4. If a value is found that is smaller at another index, then record that current index
  5. Once all values have been checked if the recorded index does not match the current index then swap those values
  6. Increase the starting index by 1
  7. Repeat 2 through 6 until the staring index >= length

Example

Start

Smallest

Index	0	1	2	3	4	5	6	7
Value	10	8	7	6	12	5	11	9

File Edit View Help

Sorting Algorithms

- Problem:
  - Given any array of integers, develop an algorithm that sorts the values from smallest to largest.
- Selection Sort
  1. Start from index 0
  2. Assume the starting index has the smallest value and record that index
  3. Sequentially check every other value
  4. If a value is found that is smaller at another index, then record that current index
  5. Once all values have been checked if the recorded index does not match the current index then swap those values
  6. Increase the starting index by 1
  7. Repeat 2 through 6 until the staring index >= length

Example

Start

Smallest

Index	0	1	2	3	4	5	6	7
Value	10	8	7	6	12	5	11	9

File Edit View Help

Sorting Algorithms

- Problem:
  - Given any array of integers, develop an algorithm that sorts the values from smallest to largest.
- Selection Sort
  1. Start from index 0
  2. Assume the starting index has the smallest value and record that index
  3. Sequentially check every other value
  4. If a value is found that is smaller at another index, then record that current index
  5. Once all values have been checked if the recorded index does not match the current index then swap those values
  6. Increase the starting index by 1
  7. Repeat 2 through 6 until the staring index >= length

Example

Start

Smallest

Index	0	1	2	3	4	5	6	7
Value	10	8	7	6	12	5	11	9

File Edit View Help

Sorting Algorithms

- Problem:
  - Given any array of integers, develop an algorithm that sorts the values from smallest to largest.
- Selection Sort
  1. Start from index 0
  2. Assume the starting index has the smallest value and record that index
  3. Sequentially check every other value
  4. If a value is found that is smaller at another index, then record that current index
  5. Once all values have been checked if the recorded index does not match the current index then swap those values
  6. Increase the starting index by 1
  7. Repeat 2 through 6 until the staring index >= length

Example

Start

Smallest

Index	0	1	2	3	4	5	6	7
Value	10	8	7	6	12	5	11	9

File Edit View Help

Sorting Algorithms

- Problem:
  - Given any array of integers, develop an algorithm that sorts the values from smallest to largest.
- Selection Sort
  1. Start from index 0
  2. Assume the starting index has the smallest value and record that index
  3. Sequentially check every other value
  4. If a value is found that is smaller at another index, then record that current index
  5. Once all values have been checked if the recorded index does not match the current index then swap those values
  6. Increase the starting index by 1
  7. Repeat 2 through 6 until the staring index >= length

Example

Start

Smallest

Index	0	1	2	3	4	5	6	7
Value	10	8	7	6	12	5	11	9

File Edit View Help

Sorting Algorithms

- Problem:
  - Given any array of integers, develop an algorithm that sorts the values from smallest to largest.
- Selection Sort
  1. Start from index 0
  2. Assume the starting index has the smallest value and record that index
  3. Sequentially check every other value
  4. If a value is found that is smaller at another index, then record that current index
  5. Once all values have been checked if the recorded index does not match the current index then swap those values
  6. Increase the starting index by 1
  7. Repeat 2 through 6 until the staring index >= length

Example

Start

Smallest

Index	0	1	2	3	4	5	6	7
Value	10	8	7	6	12	5	11	9

File Edit View Help

Sorting Algorithms

- Problem:
  - Given any array of integers, develop an algorithm that sorts the values from smallest to largest.
- Selection Sort
  1. Start from index 0
  2. Assume the starting index has the smallest value and record that index
  3. Sequentially check every other value
  4. If a value is found that is smaller at another index, then record that current index
  5. Once all values have been checked if the recorded index does not match the current index then swap those values
  6. Increase the starting index by 1
  7. Repeat 2 through 6 until the staring index >= length

Example

Start

Smallest

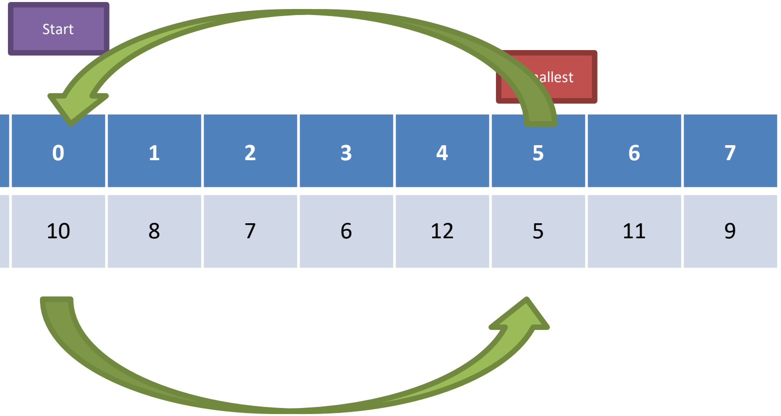
Index	0	1	2	3	4	5	6	7
Value	10	8	7	6	12	5	11	9

File Edit View Help

Sorting Algorithms

- Problem:
  - Given any array of integers, develop an algorithm that sorts the values from smallest to largest.
- Selection Sort
  1. Start from index 0
  2. Assume the starting index has the smallest value and record that index
  3. Sequentially check every other value
  4. If a value is found that is smaller at another index, then record that current index
  5. Once all values have been checked if the recorded index does not match the current index then swap those values
  6. Increase the starting index by 1
  7. Repeat 2 through 6 until the staring index >= length

Example



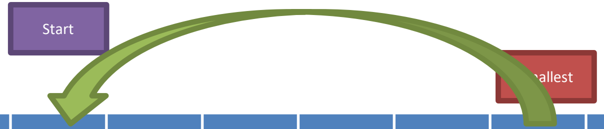
Index	0	1	2	3	4	5	6	7
Value	10	8	7	6	12	5	11	9

File Edit View Help

Sorting Algorithms

- Problem:
  - Given any array of integers, develop an algorithm that sorts the values from smallest to largest.
- Selection Sort
  1. Start from index 0
  2. Assume the starting index has the smallest value and record that index
  3. Sequentially check every other value
  4. If a value is found that is smaller at another index, then record that current index
  5. Once all values have been checked if the recorded index does not match the current index then swap those values
  6. Increase the starting index by 1
  7. Repeat 2 through 6 until the staring index >= length

Example



Index	0	1	2	3	4	5	6	7
Value	5	8	7	6	12	10	11	9

File Edit View Help

Sorting Algorithms

- Problem:
  - Given any array of integers, develop an algorithm that sorts the values from smallest to largest.
- Selection Sort
  1. Start from index 0
  2. Assume the starting index has the smallest value and record that index
  3. Sequentially check every other value
  4. If a value is found that is smaller at another index, then record that current index
  5. Once all values have been checked if the recorded index does not match the current index then swap those values
  6. Increase the starting index by 1
  7. Repeat 2 through 6 until the staring index >= length

Example

Start

Smallest

Index	0	1	2	3	4	5	6	7
Value	5	8	7	6	12	10	11	9

File Edit View Help

Sorting Algorithms

- Problem:
  - Given any array of integers, develop an algorithm that sorts the values from smallest to largest.
- Selection Sort
  1. Start from index 0
  2. Assume the starting index has the smallest value and record that index
  3. Sequentially check every other value
  4. If a value is found that is smaller at another index, then record that current index
  5. Once all values have been checked if the recorded index does not match the current index then swap those values
  6. Increase the starting index by 1
  7. Repeat 2 through 6 until the staring index >= length

Example

Start

Smallest

Index	0	1	2	3	4	5	6	7
Value	5	8	7	6	12	10	11	9

File Edit View Help

Sorting Algorithms

- Problem:
  - Given any array of integers, develop an algorithm that sorts the values from smallest to largest.
- Selection Sort
  1. Start from index 0
  2. Assume the starting index has the smallest value and record that index
  3. Sequentially check every other value
  4. If a value is found that is smaller at another index, then record that current index
  5. Once all values have been checked if the recorded index does not match the current index then swap those values
  6. Increase the starting index by 1
  7. Repeat 2 through 6 until the staring index >= length

Example

Start

Smallest

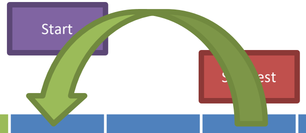
Index	0	1	2	3	4	5	6	7
Value	5	8	7	6	12	10	11	9

File Edit View Help

Sorting Algorithms

- Problem:
  - Given any array of integers, develop an algorithm that sorts the values from smallest to largest.
- Selection Sort
  1. Start from index 0
  2. Assume the starting index has the smallest value and record that index
  3. Sequentially check every other value
  4. If a value is found that is smaller at another index, then record that current index
  5. Once all values have been checked if the recorded index does not match the current index then swap those values
  6. Increase the starting index by 1
  7. Repeat 2 through 6 until the staring index >= length

Example



Index	0	1	2	3	4	5	6	7
Value	5	8	7	6	12	10	11	9

Sorting Algorithms

- Problem:
  - Given any array of integers, develop an algorithm that sorts the values from smallest to largest.
- Selection Sort
  1. Start from index 0
  2. Assume the starting index has the smallest value and record that index
  3. Sequentially check every other value
  4. If a value is found that is smaller at another index, then record that current index
  5. Once all values have been checked if the recorded index does not match the current index then swap those values
  6. Increase the starting index by 1
  7. Repeat 2 through 6 until the staring index >= length

Example

Start

Smallest

Index	0	1	2	3	4	5	6	7
Value	5	6	7	8	12	10	11	9

Sorting Algorithms

- Problem:
  - Given any array of integers, develop an algorithm that sorts the values from smallest to largest.
- Selection Sort
  1. Start from index 0
  2. Assume the starting index has the smallest value and record that index
  3. Sequentially check every other value
  4. If a value is found that is smaller at another index, then record that current index
  5. Once all values have been checked if the recorded index does not match the current index then swap those values
  6. Increase the starting index by 1
  7. Repeat 2 through 6 until the staring index >= length

Example

Start
Smallest

Index	0	1	2	3	4	5	6	7
Value	5	6	7	8	12	10	11	9

File Edit View Help

A Few Swaps Later

File Edit View Help

Sorting Algorithms

- Problem:
  - Given any array of integers, develop an algorithm that sorts the values from smallest to largest.
- Selection Sort
  1. Start from index 0
  2. Assume the starting index has the smallest value and record that index
  3. Sequentially check every other value
  4. If a value is found that is smaller at another index, then record that current index
  5. Once all values have been checked if the recorded index does not match the current index then swap those values
  6. Increase the starting index by 1
  7. Repeat 2 through 6 until the staring index >= length

Example

Index	0	1	2	3	4	5	6	7
Value	5	6	7	8	9	10	11	12

File Edit View Help

Selection Sort Complexity

- Worst Case
  - Sorted in Descending Order
- Operations
  - Search for smallest value = n
  - Search for the next smallest value = n – 1
  - Search for the next smallest value = n – 2
  - …
  - Search for the largest element = 1

<u>Complexity</u>

File Edit View Help

Selection Sort Complexity

- Worst Case
  - Sorted in Descending Order
- Operations
  - Search for smallest value = n
  - Search for the next smallest value = n – 1
  - Search for the next smallest value = n – 2
  - …
  - Search for the largest element = 1

<u>Complexity</u>

$$O(n^2)$$

Sorting Algorithms

- Bubble Sort
    1. Start from index 0
    2. Check each index with its neighbor (index+1)
    3. If that neighbor's value is smaller then swap with the current index's value
    4. Repeat step 1 until no swaps have been made

Example

Index	0	1	2	3	4	5	6	7
Value	10	8	7	6	12	5	11	9

File Edit View Help

Sorting Algorithms

- Bubble Sort
  1. Start from index 0
  2. Check each index with its neighbor (index+1)
  3. If that neighbor's value is smaller then swap with the current index's value
  4. Repeat step 1 until no swaps have been made

Example

Index	0	1	2	3	4	5	6	7
Value	10	8	7	6	12	5	11	9

File Edit View Help

Sorting Algorithms

- Bubble Sort
  1. Start from index 0
  2. Check each index with its neighbor (index+1)
  3. If that neighbor's value is smaller then swap with the current index's value
  4. Repeat step 1 until no swaps have been made

Example

Index	0	1	2	3	4	5	6	7
Value	10	8	7	6	12	5	11	9

File Edit View Help

Sorting Algorithms

- Bubble Sort
  1. Start from index 0
  2. Check each index with its neighbor (index+1)
  3. If that neighbor's value is smaller then swap with the current index's value
  4. Repeat step 1 until no swaps have been made

Example

Index	0	1	2	3	4	5	6	7
Value	8	10	7	6	12	5	11	9

File Edit View Help

Sorting Algorithms

- Bubble Sort
  1. Start from index 0
  2. Check each index with its neighbor (index+1)
  3. If that neighbor's value is smaller then swap with the current index's value
  4. Repeat step 1 until no swaps have been made

Example

Index	0	1	2	3	4	5	6	7
Value	8	10	7	6	12	5	11	9

Sorting Algorithms

- Bubble Sort
  1. Start from index 0
  2. Check each index with its neighbor (index+1)
  3. If that neighbor's value is smaller then swap with the current index's value
  4. Repeat step 1 until no swaps have been made

Example

Index	0	1	2	3	4	5	6	7
Value	8	10	7	6	12	5	11	9

File Edit View Help

Sorting Algorithms

- Bubble Sort
  1. Start from index 0
  2. Check each index with its neighbor (index+1)
  3. If that neighbor's value is smaller then swap with the current index's value
  4. Repeat step 1 until no swaps have been made

Example

Index	0	1	2	3	4	5	6	7
Value	8	7	10	6	12	5	11	9

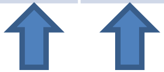
File Edit View Help

Sorting Algorithms

- Bubble Sort
  1. Start from index 0
  2. Check each index with its neighbor (index+1)
  3. If that neighbor's value is smaller then swap with the current index's value
  4. Repeat step 1 until no swaps have been made

Example

Index	0	1	2	3	4	5	6	7
Value	8	7	10	6	12	5	11	9



File Edit View Help

Sorting Algorithms

- Bubble Sort
  1. Start from index 0
  2. Check each index with its neighbor (index+1)
  3. If that neighbor's value is smaller then swap with the current index's value
  4. Repeat step 1 until no swaps have been made

Example

Index	0	1	2	3	4	5	6	7
Value	8	7	6	10	12	5	11	9

File Edit View Help

Sorting Algorithms

- Bubble Sort
  1. Start from index 0
  2. Check each index with its neighbor (index+1)
  3. If that neighbor's value is smaller then swap with the current index's value
  4. Repeat step 1 until no swaps have been made

Example

Index	0	1	2	3	4	5	6	7
Value	8	7	6	10	12	5	11	9

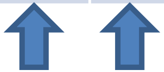
File Edit View Help

Sorting Algorithms

- Bubble Sort
  1. Start from index 0
  2. Check each index with its neighbor (index+1)
  3. If that neighbor's value is smaller then swap with the current index's value
  4. Repeat step 1 until no swaps have been made

Example

Index	0	1	2	3	4	5	6	7
Value	8	7	6	10	12	5	11	9



File Edit View Help

Sorting Algorithms

- Bubble Sort
  1. Start from index 0
  2. Check each index with its neighbor (index+1)
  3. If that neighbor's value is smaller then swap with the current index's value
  4. Repeat step 1 until no swaps have been made

Example

Index	0	1	2	3	4	5	6	7
Value	8	7	6	10	5	12	11	9

File Edit View Help

Sorting Algorithms

- Bubble Sort
  1. Start from index 0
  2. Check each index with its neighbor (index+1)
  3. If that neighbor's value is smaller then swap with the current index's value
  4. Repeat step 1 until no swaps have been made

Example

Index	**0**	**1**	**2**	**3**	**4**	**5**	**6**	**7**
Value	8	7	6	10	5	12	11	9

File Edit View Help

Sorting Algorithms

- Bubble Sort
  1. Start from index 0
  2. Check each index with its neighbor (index+1)
  3. If that neighbor's value is smaller then swap with the current index's value
  4. Repeat step 1 until no swaps have been made

Example

Index	0	1	2	3	4	5	6	7
Value	8	7	6	10	5	11	12	9

File Edit View Help

Sorting Algorithms

- Bubble Sort
  1. Start from index 0
  2. Check each index with its neighbor (index+1)
  3. If that neighbor's value is smaller then swap with the current index's value
  4. Repeat step 1 until no swaps have been made

Example

Index	0	1	2	3	4	5	6	7
Value	8	7	6	10	5	11	12	9

File Edit View Help

Sorting Algorithms

- Bubble Sort
  1. Start from index 0
  2. Check each index with its neighbor (index+1)
  3. If that neighbor's value is smaller then swap with the current index's value
  4. Repeat step 1 until no swaps have been made

Example

Index	0	1	2	3	4	5	6	7
Value	8	7	6	10	5	11	9	12

File Edit View Help

Sorting Algorithms

- Bubble Sort
  1. Start from index 0
  2. Check each index with its neighbor (index+1)
  3. If that neighbor's value is smaller then swap with the current index's value
  4. Repeat step 1 until no swaps have been made

Example

Index	0	1	2	3	4	5	6	7
Value	8	7	6	10	5	11	9	12

A Few Swaps Later

File Edit View Help

Sorting Algorithms

- Bubble Sort
  1. Start from index 0
  2. Check each index with its neighbor (index+1)
  3. If that neighbor's value is smaller then swap with the current index's value
  4. Repeat step 1 until no swaps have been made

Example

Index	0	1	2	3	4	5	6	7
Value	5	6	7	8	9	10	11	12

File Edit View Help

Bubble Sort Complexity

- Worst Case
  - Sorted in Descending Order
- Operations
  - Bubble Up Largest Value = n
  - Bubble Up Next Largest Value = n – 1
  - Bubble Up Next Largest Value = n – 2
  - …
  - Smallest Value = 1

Complexity

File Edit View Help

Bubble Sort Complexity

- Worst Case
  - Sorted in Descending Order
- Operations
  - Bubble Up Largest Value = n
  - Bubble Up Next Largest Value = n – 1
  - Bubble Up Next Largest Value = n – 2
  - …
  - Smallest Value = 1

Complexity

$$O(n^2)$$

File Edit View Help

Can we do better?

File Edit View Help

Sorting Algorithms

- Merge Sort
  1. Recursively split the array in half until single elements remain
  2. Merge two smaller arrays and return the sorted result
     1. Create an array of combined size
     2. Add elements from the two smaller arrays into the combined array in sorted order
  3. Repeat Step 2 until the final array is reached

Example

Index	0	1	2	3	4	5	6	7
Value	10	8	7	6	12	5	11	9

File Edit View Help

Sorting Algorithms

- Merge Sort
  1. Recursively split the array in half until single elements remain
  2. Merge two smaller arrays and return the sorted result
     1. Create an array of combined size
     2. Add elements from the two smaller arrays into the combined array in sorted order
  3. Repeat Step 2 until the final array is reached

Example

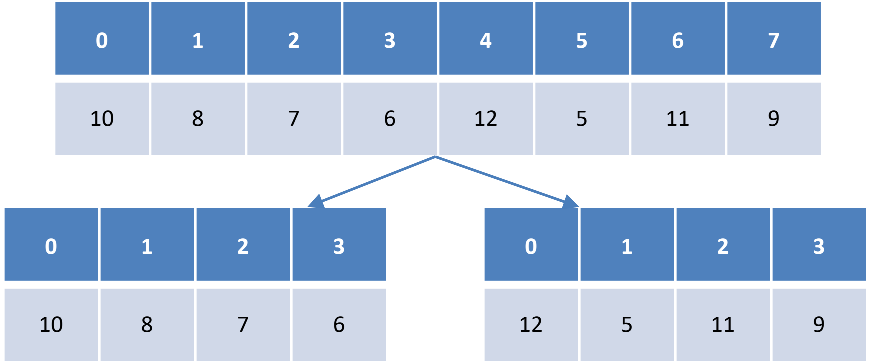
0	1	2	3	4	5	6	7
10	8	7	6	12	5	11	9

File Edit View Help

Sorting Algorithms

- Merge Sort
    1. Recursively split the array in half until single elements remain
    2. Merge two smaller arrays and return the sorted result
        1. Create an array of combined size
        2. Add elements from the two smaller arrays into the combined array in sorted order
    3. Repeat Step 2 until the final array is reached

Example

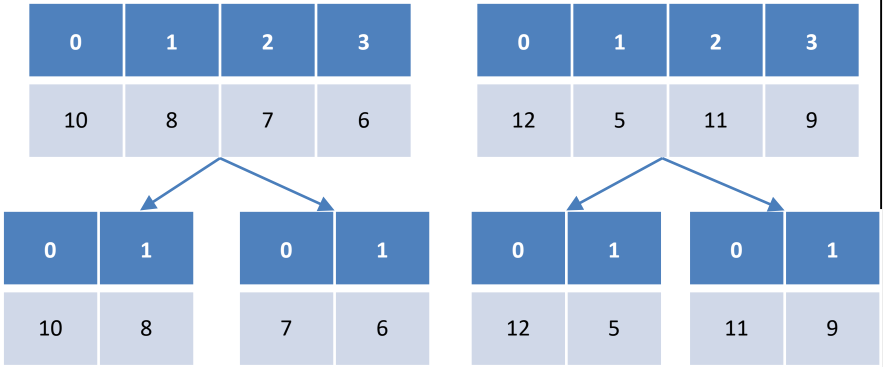


File Edit View Help

Sorting Algorithms

- Merge Sort
  1. Recursively split the array in half until single elements remain
  2. Merge two smaller arrays and return the sorted result
     1. Create an array of combined size
     2. Add elements from the two smaller arrays into the combined array in sorted order
  3. Repeat Step 2 until the final array is reached

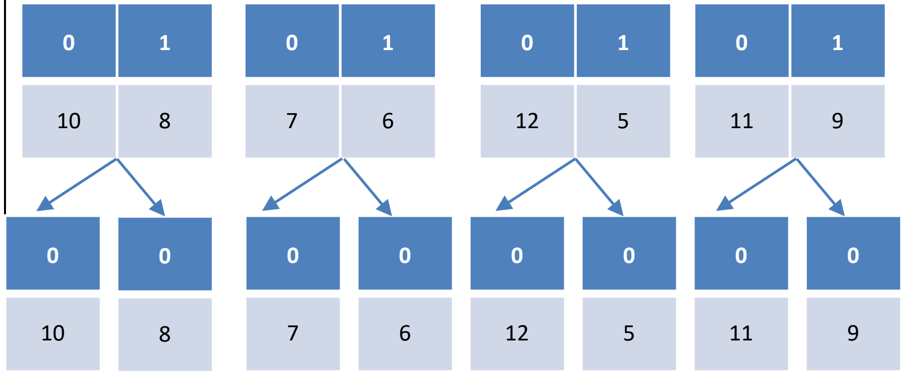
Example



Sorting Algorithms

- Merge Sort
  1. Recursively split the array in half until single elements remain
  2. Merge two smaller arrays and return the sorted result
     1. Create an array of combined size
     2. Add elements from the two smaller arrays into the combined array in sorted order
  3. Repeat Step 2 until the final array is reached

Example



File Edit View Help

Sorting Algorithms

- Merge Sort
  1. Recursively split the array in half until single elements remain
  2. Merge two smaller arrays and return the sorted result
     1. Create an array of combined size
     2. Add elements from the two smaller arrays into the combined array in sorted order
  3. Repeat Step 2 until the final array is reached

Example

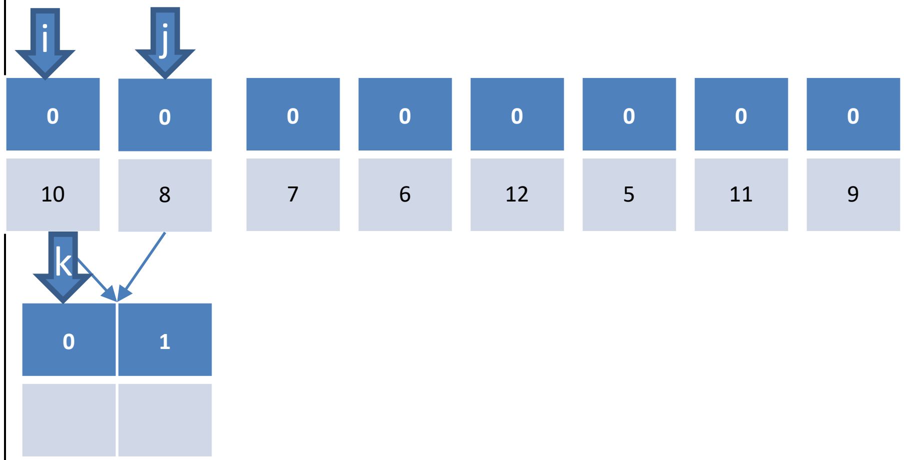
0	0	0	0	0	0	0	0
10	8	7	6	12	5	11	9

File Edit View Help

Sorting Algorithms

- Merge Sort
  1. Recursively split the array in half until single elements remain
  2. Merge two smaller arrays and return the sorted result
     1. Create an array of combined size
     2. Add elements from the two smaller arrays into the combined array in sorted order
  3. Repeat Step 2 until the final array is reached

Example

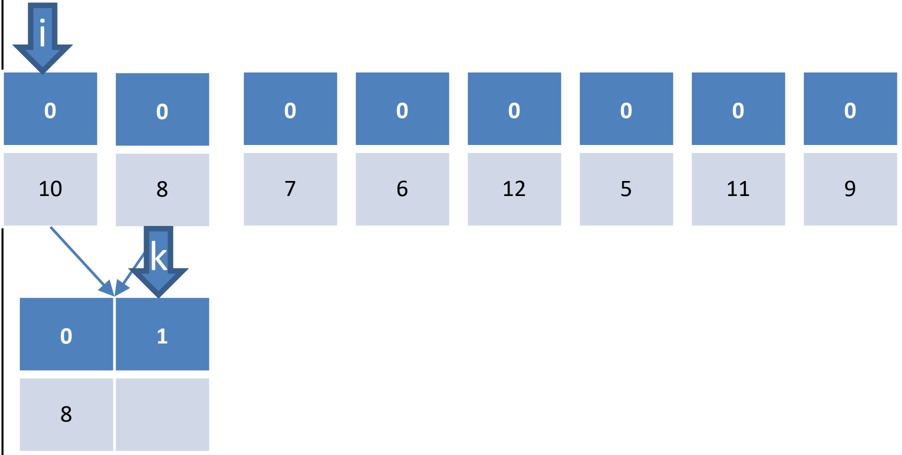


File Edit View Help

Sorting Algorithms

- Merge Sort
  1. Recursively split the array in half until single elements remain
  2. Merge two smaller arrays and return the sorted result
     1. Create an array of combined size
     2. Add elements from the two smaller arrays into the combined array in sorted order
  3. Repeat Step 2 until the final array is reached

Example

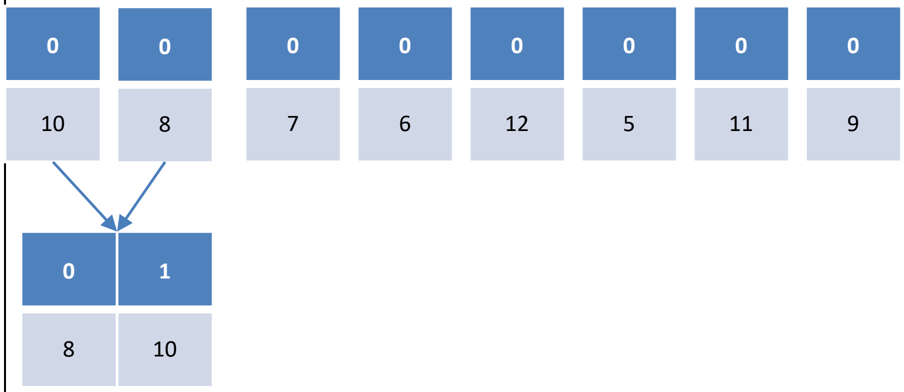


File Edit View Help

Sorting Algorithms

- Merge Sort
  1. Recursively split the array in half until single elements remain
  2. Merge two smaller arrays and return the sorted result
     1. Create an array of combined size
     2. Add elements from the two smaller arrays into the combined array in sorted order
  3. Repeat Step 2 until the final array is reached

Example

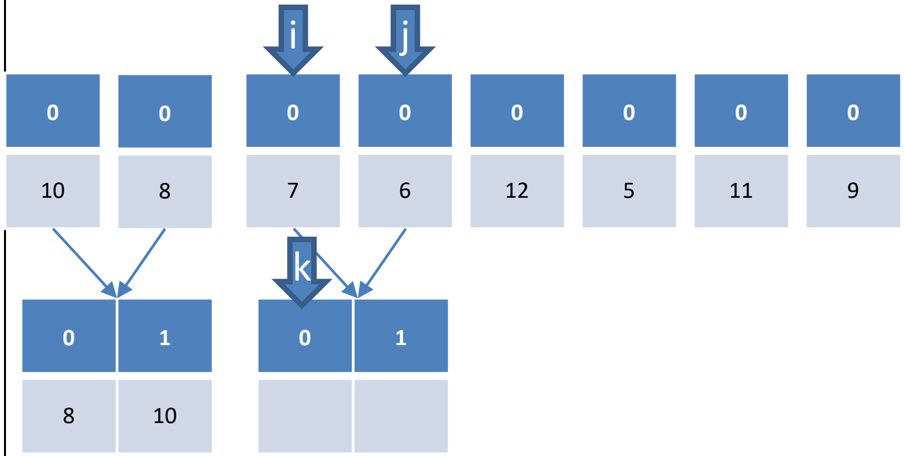


File Edit View Help

Sorting Algorithms

- Merge Sort
  1. Recursively split the array in half until single elements remain
  2. Merge two smaller arrays and return the sorted result
     1. Create an array of combined size
     2. Add elements from the two smaller arrays into the combined array in sorted order
  3. Repeat Step 2 until the final array is reached

Example

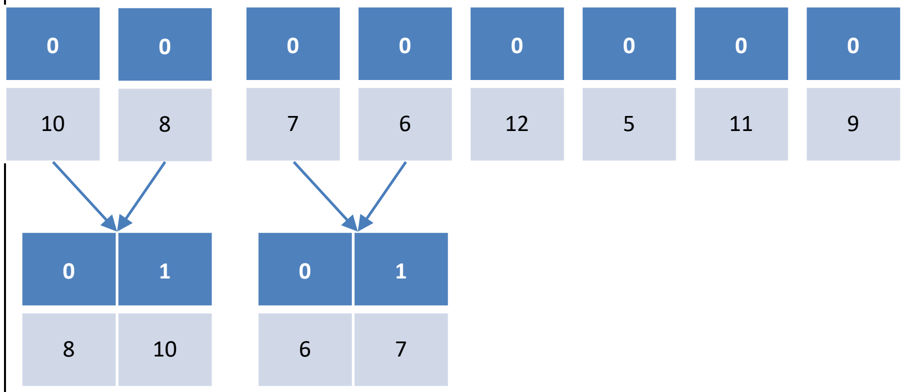


File Edit View Help

Sorting Algorithms

- Merge Sort
  1. Recursively split the array in half until single elements remain
  2. Merge two smaller arrays and return the sorted result
     1. Create an array of combined size
     2. Add elements from the two smaller arrays into the combined array in sorted order
  3. Repeat Step 2 until the final array is reached

Example

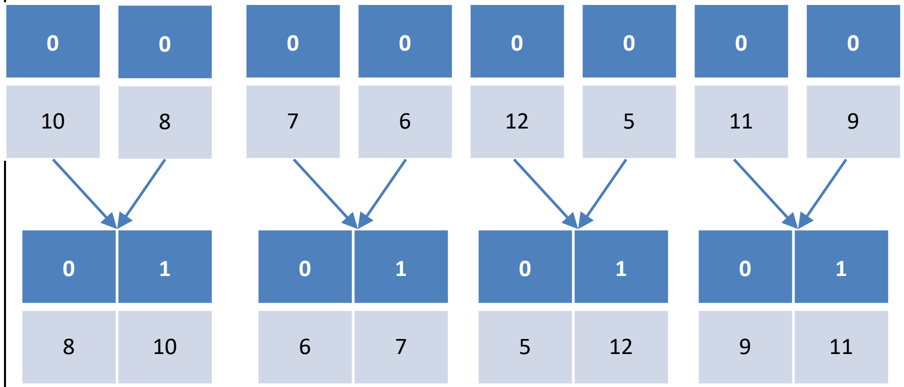


File Edit View Help

Sorting Algorithms

- Merge Sort
  1. Recursively split the array in half until single elements remain
  2. Merge two smaller arrays and return the sorted result
     1. Create an array of combined size
     2. Add elements from the two smaller arrays into the combined array in sorted order
  3. Repeat Step 2 until the final array is reached

Example



File Edit View Help

Sorting Algorithms

- Merge Sort
  1. Recursively split the array in half until single elements remain
  2. Merge two smaller arrays and return the sorted result
     1. Create an array of combined size
     2. Add elements from the two smaller arrays into the combined array in sorted order
  3. Repeat Step 2 until the final array is reached

Example

0	1
8	10

0	1
6	7

0	1
5	12

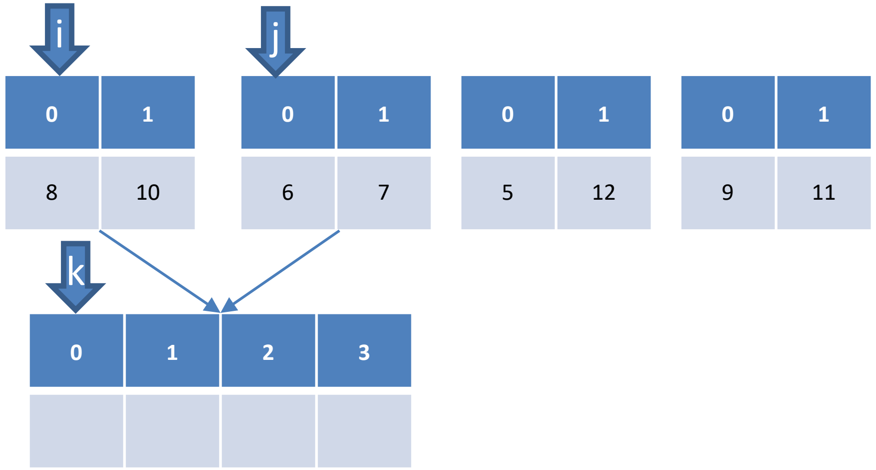
0	1
9	11

File Edit View Help

Sorting Algorithms

- Merge Sort
  1. Recursively split the array in half until single elements remain
  2. Merge two smaller arrays and return the sorted result
     1. Create an array of combined size
     2. Add elements from the two smaller arrays into the combined array in sorted order
  3. Repeat Step 2 until the final array is reached

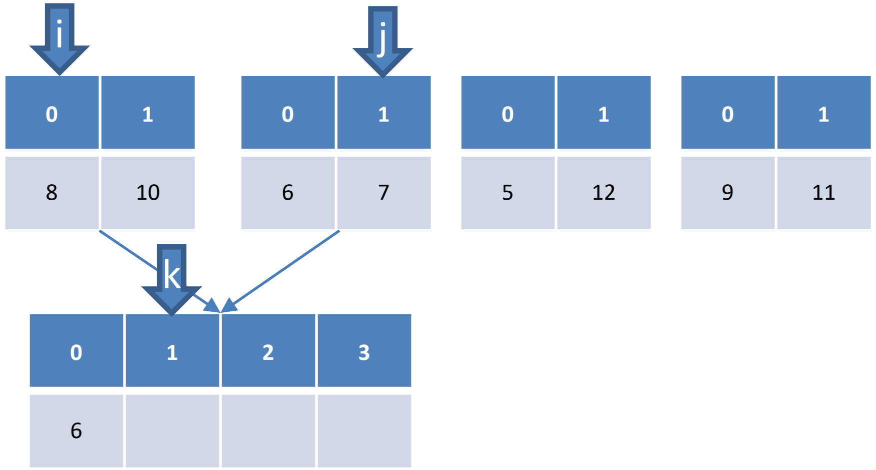
Example



Sorting Algorithms

- Merge Sort
  1. Recursively split the array in half until single elements remain
  2. Merge two smaller arrays and return the sorted result
     1. Create an array of combined size
     2. Add elements from the two smaller arrays into the combined array in sorted order
  3. Repeat Step 2 until the final array is reached

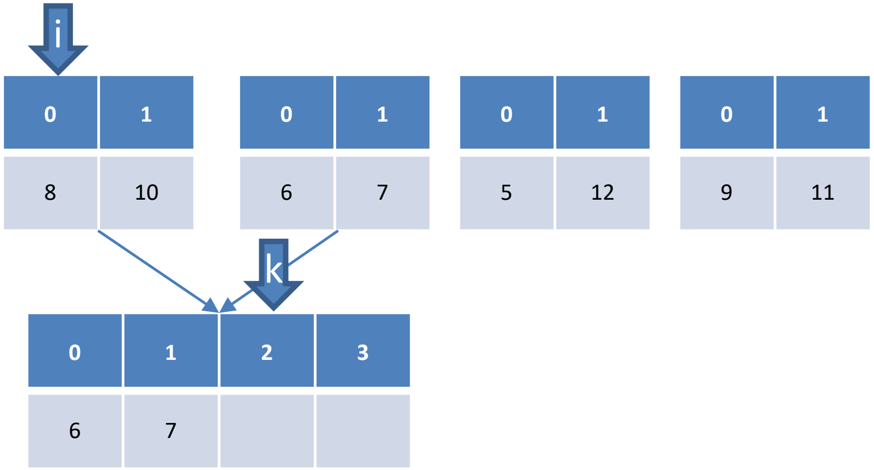
Example



Sorting Algorithms

- Merge Sort
  1. Recursively split the array in half until single elements remain
  2. Merge two smaller arrays and return the sorted result
     1. Create an array of combined size
     2. Add elements from the two smaller arrays into the combined array in sorted order
  3. Repeat Step 2 until the final array is reached

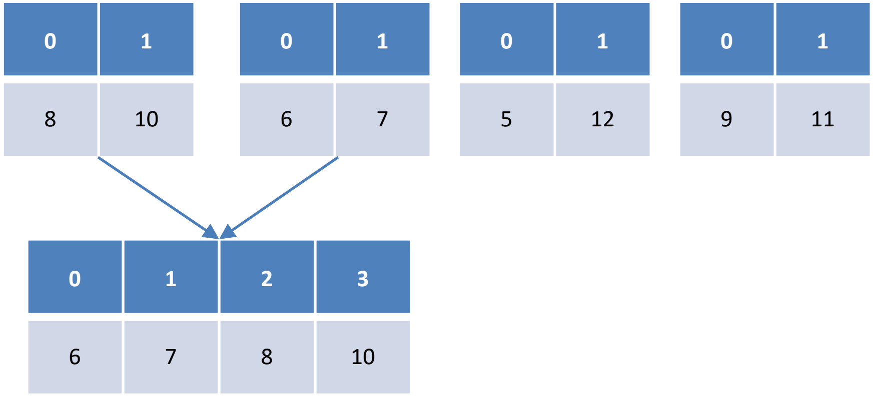
Example



Sorting Algorithms

- Merge Sort
  1. Recursively split the array in half until single elements remain
  2. Merge two smaller arrays and return the sorted result
     1. Create an array of combined size
     2. Add elements from the two smaller arrays into the combined array in sorted order
  3. Repeat Step 2 until the final array is reached

Example

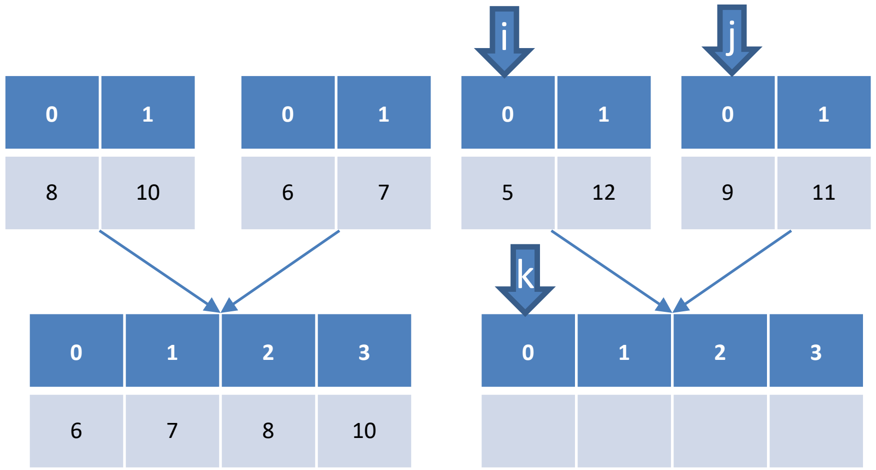


File Edit View Help

Sorting Algorithms

- Merge Sort
  1. Recursively split the array in half until single elements remain
  2. Merge two smaller arrays and return the sorted result
     1. Create an array of combined size
     2. Add elements from the two smaller arrays into the combined array in sorted order
  3. Repeat Step 2 until the final array is reached

Example

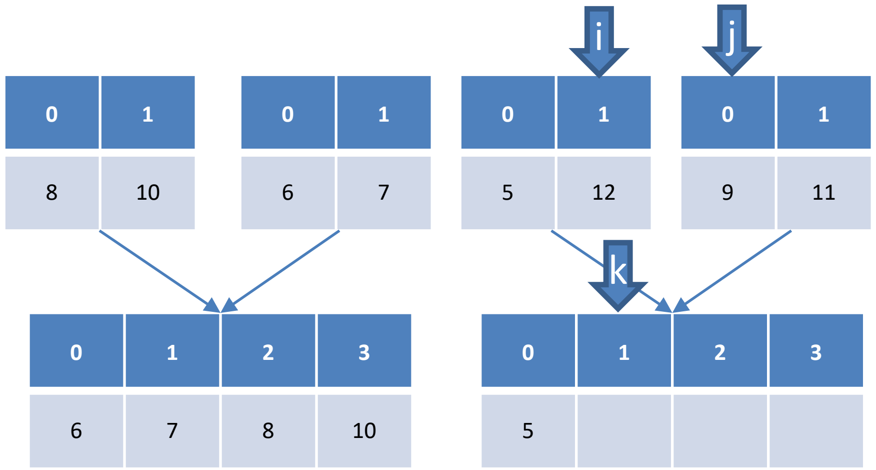


File Edit View Help

Sorting Algorithms

- Merge Sort
    1. Recursively split the array in half until single elements remain
    2. Merge two smaller arrays and return the sorted result
        1. Create an array of combined size
        2. Add elements from the two smaller arrays into the combined array in sorted order
    3. Repeat Step 2 until the final array is reached

Example

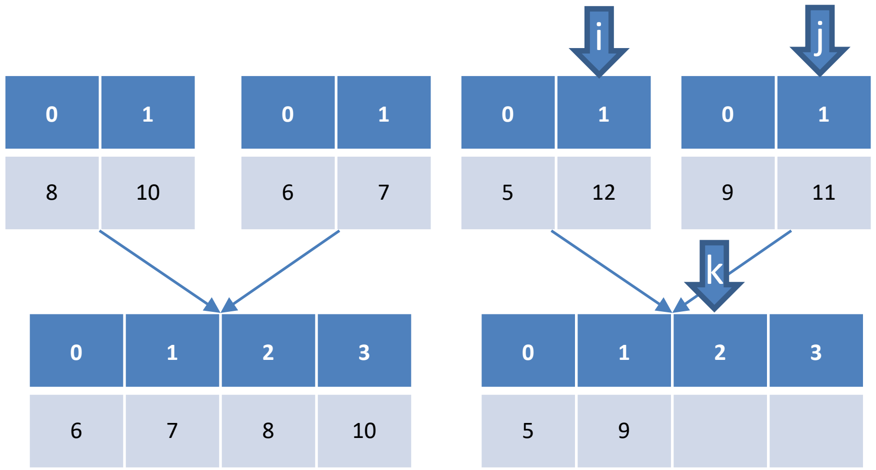


File Edit View Help

Sorting Algorithms

- Merge Sort
    1. Recursively split the array in half until single elements remain
    2. Merge two smaller arrays and return the sorted result
        1. Create an array of combined size
        2. Add elements from the two smaller arrays into the combined array in sorted order
    3. Repeat Step 2 until the final array is reached

Example

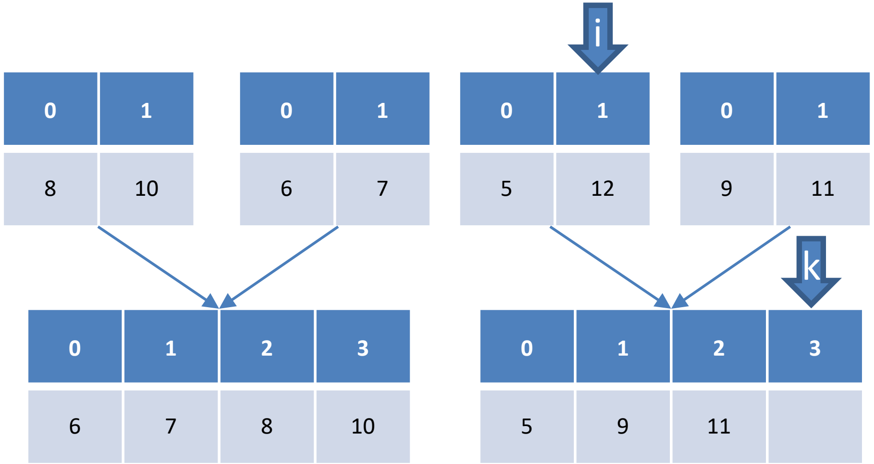


File Edit View Help

Sorting Algorithms

- Merge Sort
    1. Recursively split the array in half until single elements remain
    2. Merge two smaller arrays and return the sorted result
        1. Create an array of combined size
        2. Add elements from the two smaller arrays into the combined array in sorted order
    3. Repeat Step 2 until the final array is reached

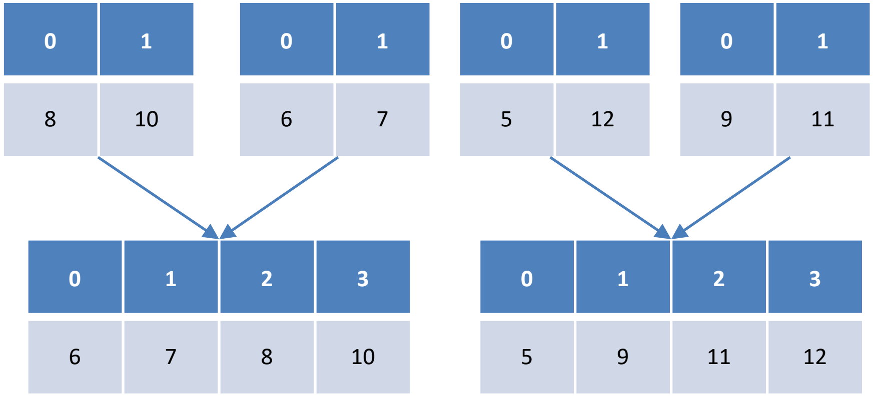
Example



Sorting Algorithms

- Merge Sort
  1. Recursively split the array in half until single elements remain
  2. Merge two smaller arrays and return the sorted result
     1. Create an array of combined size
     2. Add elements from the two smaller arrays into the combined array in sorted order
  3. Repeat Step 2 until the final array is reached

Example

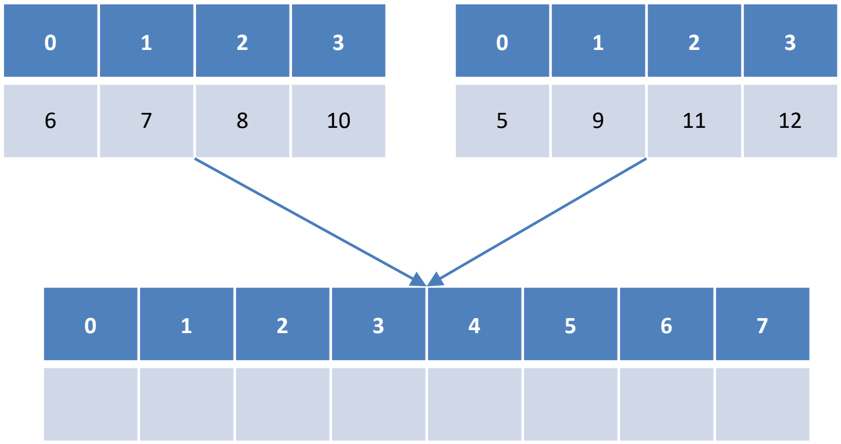


File Edit View Help

Sorting Algorithms

- Merge Sort
  1. Recursively split the array in half until single elements remain
  2. Merge two smaller arrays and return the sorted result
     1. Create an array of combined size
     2. Add elements from the two smaller arrays into the combined array in sorted order
  3. Repeat Step 2 until the final array is reached

Example

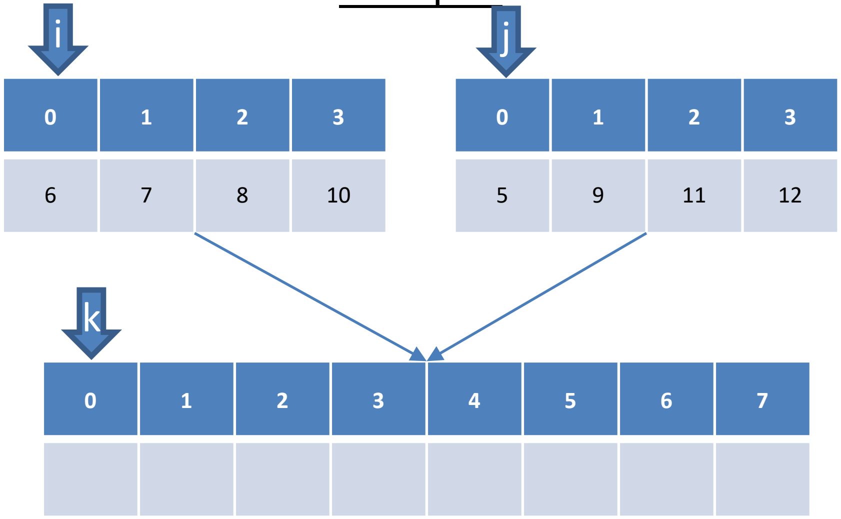


File Edit View Help

Sorting Algorithms

- Merge Sort
    1. Recursively split the array in half until single elements remain
    2. Merge two smaller arrays and return the sorted result
        1. Create an array of combined size
        2. Add elements from the two smaller arrays into the combined array in sorted order
    3. Repeat Step 2 until the final array is reached

Example

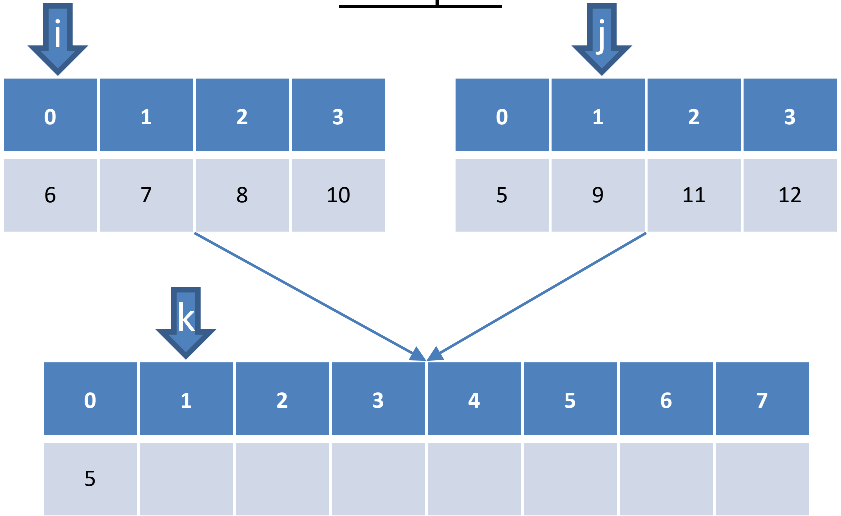


File Edit View Help

Sorting Algorithms

- Merge Sort
  1. Recursively split the array in half until single elements remain
  2. Merge two smaller arrays and return the sorted result
     1. Create an array of combined size
     2. Add elements from the two smaller arrays into the combined array in sorted order
  3. Repeat Step 2 until the final array is reached

Example

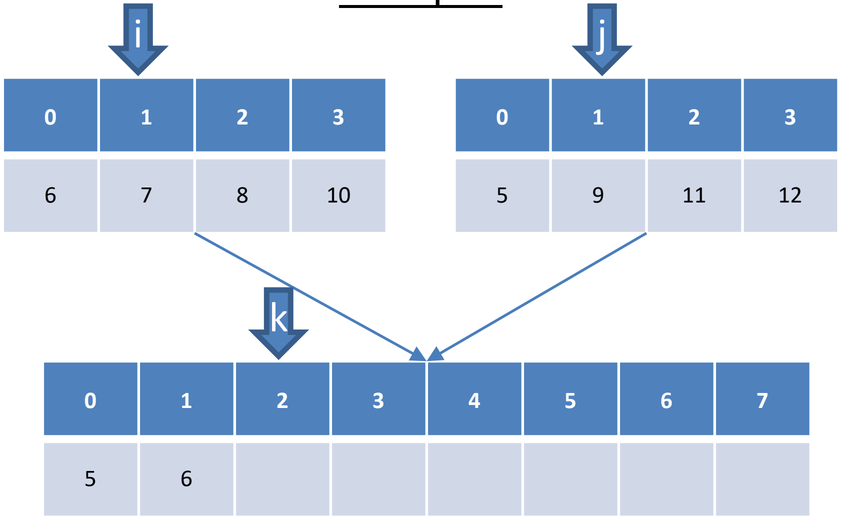


File Edit View Help

Sorting Algorithms

- Merge Sort
  1. Recursively split the array in half until single elements remain
  2. Merge two smaller arrays and return the sorted result
     1. Create an array of combined size
     2. Add elements from the two smaller arrays into the combined array in sorted order
  3. Repeat Step 2 until the final array is reached

Example

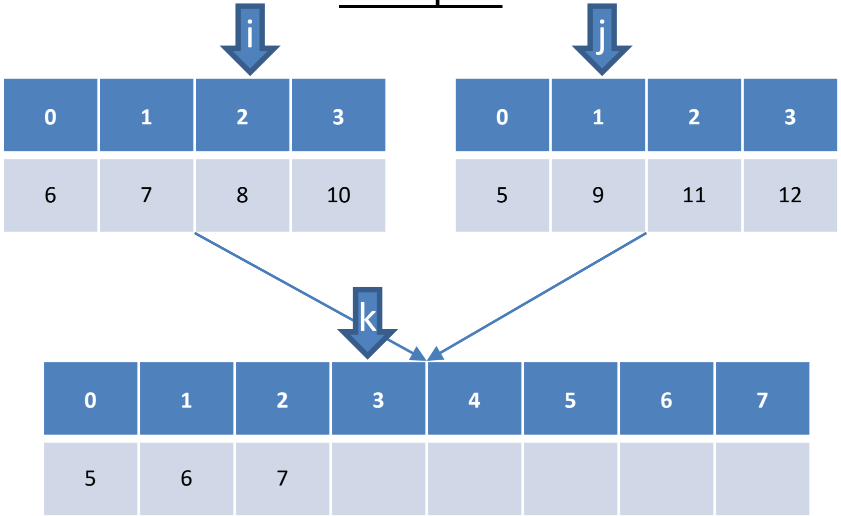


File Edit View Help

Sorting Algorithms

- Merge Sort
    1. Recursively split the array in half until single elements remain
    2. Merge two smaller arrays and return the sorted result
        1. Create an array of combined size
        2. Add elements from the two smaller arrays into the combined array in sorted order
    3. Repeat Step 2 until the final array is reached

Example

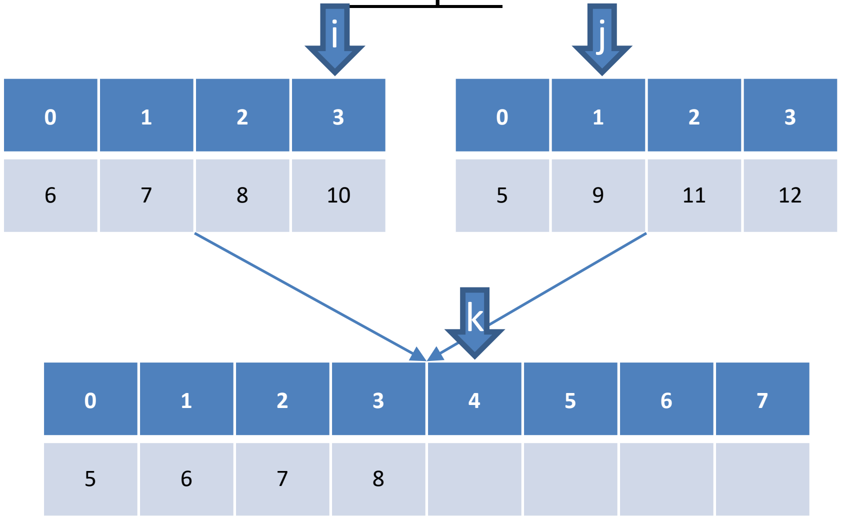


File Edit View Help

Sorting Algorithms

- Merge Sort
  1. Recursively split the array in half until single elements remain
  2. Merge two smaller arrays and return the sorted result
     1. Create an array of combined size
     2. Add elements from the two smaller arrays into the combined array in sorted order
  3. Repeat Step 2 until the final array is reached

Example

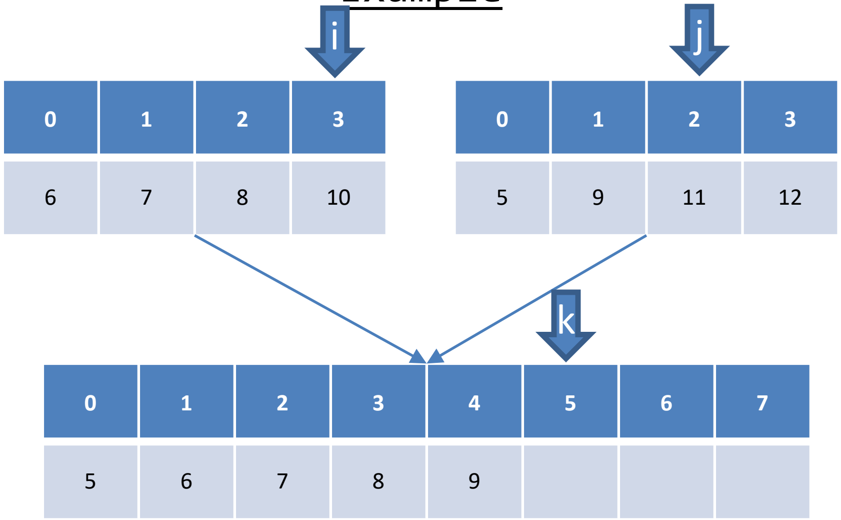


File Edit View Help

Sorting Algorithms

- Merge Sort
  1. Recursively split the array in half until single elements remain
  2. Merge two smaller arrays and return the sorted result
     1. Create an array of combined size
     2. Add elements from the two smaller arrays into the combined array in sorted order
  3. Repeat Step 2 until the final array is reached

Example

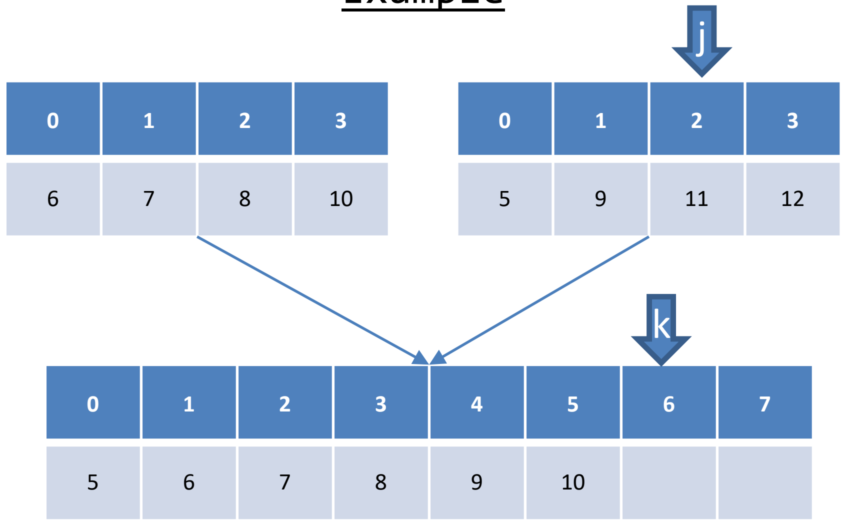


File Edit View Help

Sorting Algorithms

- Merge Sort
  1. Recursively split the array in half until single elements remain
  2. Merge two smaller arrays and return the sorted result
     1. Create an array of combined size
     2. Add elements from the two smaller arrays into the combined array in sorted order
  3. Repeat Step 2 until the final array is reached

Example

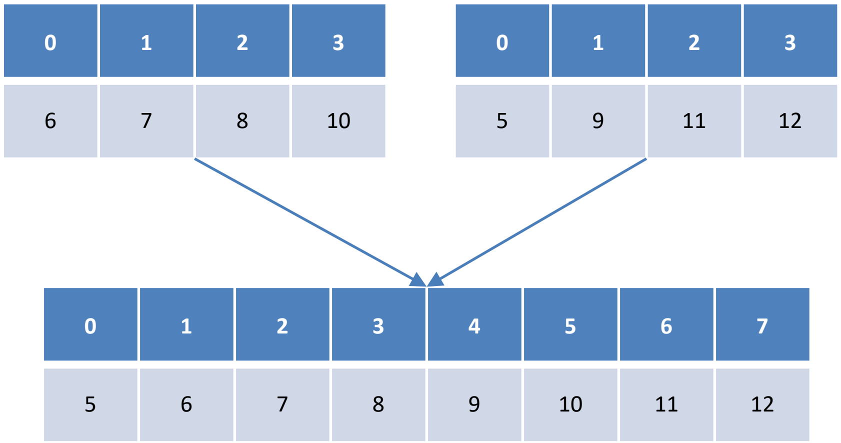


File Edit View Help

Sorting Algorithms

- Merge Sort
  1. Recursively split the array in half until single elements remain
  2. Merge two smaller arrays and return the sorted result
     1. Create an array of combined size
     2. Add elements from the two smaller arrays into the combined array in sorted order
  3. Repeat Step 2 until the final array is reached

Example



File Edit View Help

Sorting Algorithms

- Merge Sort
    1. Recursively split the array in half until single elements remain
    2. Merge two smaller arrays and return the sorted result
        1. Create an array of combined size
        2. Add elements from the two smaller arrays into the combined array in sorted order
    3. Repeat Step 2 until the final array is reached

Example

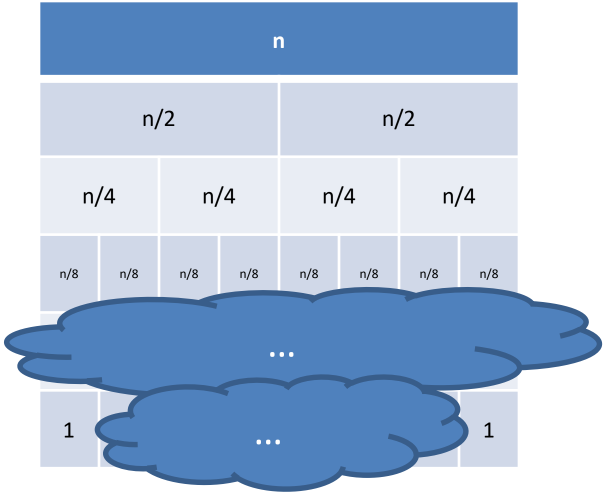
0	1	2	3	4	5	6	7
5	6	7	8	9	10	11	12

File Edit View Help

Merge Sort Complexity

- Worst Case
  - Sorted in Descending Order
- Operations
  - Split
  - Merge

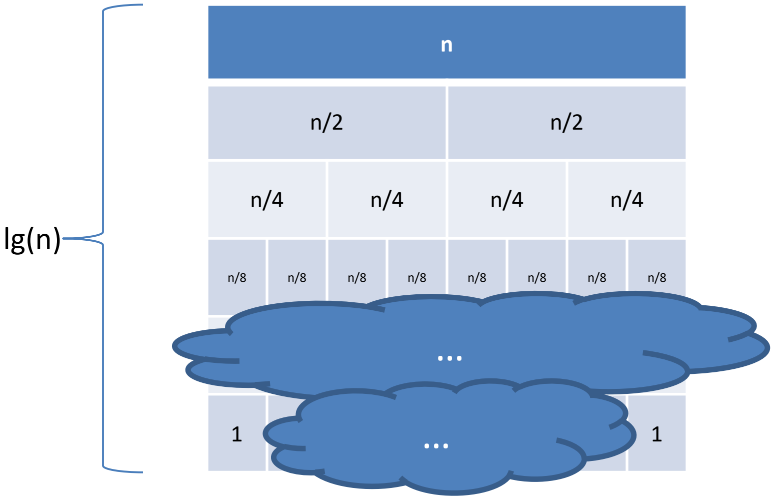
Complexity Visual



Merge Sort Complexity

- Worst Case
  - Sorted in Descending Order
- Operations
  - Split
  - Merge

Complexity Visual

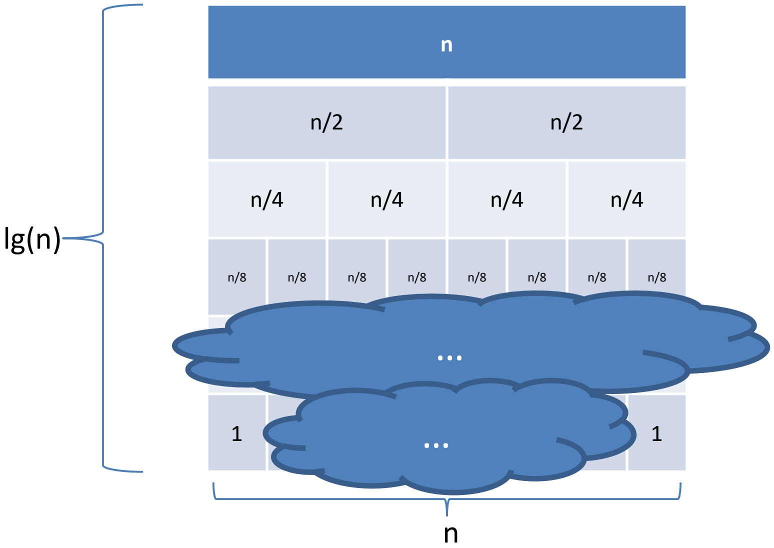


File Edit View Help

Merge Sort Complexity

- Worst Case
  - Sorted in Descending Order
- Operations
  - Split
  - Merge

Complexity Visual



File Edit View Help

Merge Sort Complexity

- Worst Case
  - Sorted in Descending Order
- Operations
  - Split
  - Merge

Complexity

$$O(n\lg(n))$$

File Edit View Help

Sorting Algorithms

- Quick Sort
  1. Pick an arbitrary value called a "pivot" from the array
  2. Using the pivot value "partition" the array
     1. Reorder the array where smaller values are to the left of the pivot, and large / equal values are to the right
     2. Once it has been partitioned the pivot value is where it should be
  3. Recursively continue now with the array to the left, and the array to the right of the pivot

Example

Index	0	1	2	3	4	5	6	7
Value	10	8	7	6	12	5	11	9

File Edit View Help

Sorting Algorithms

- Quick Sort
  1. Pick an arbitrary value called a “pivot” from the array
  2. Using the pivot value “partition” the array
     1. Reorder the array where smaller values are to the left of the pivot, and large / equal values are to the right
     2. Once it has been partitioned the pivot value is where it should be
  3. Recursively continue now with the array to the left, and the array to the right of the pivot

Example

Index	0	1	2	3	4	5	6	7
Value	10	8	7	6	12	5	11	9 pivot

File Edit View Help

Sorting Algorithms

- Quick Sort
  1. Pick an arbitrary value called a "pivot" from the array
  2. Using the pivot value "partition" the array
     1. Reorder the array where smaller values are to the left of the pivot, and large / equal values are to the right
     2. Once it has been partitioned the pivot value is where it should be
  3. Recursively continue now with the array to the left, and the array to the right of the pivot

Example

Index	0	1	2	3	4	5	6	7
Value	10	8	7	6	12	5	11	9 pivot

If value at j is smaller than the pivot then swap values at i and j and increase i by 1

File Edit View Help

Sorting Algorithms

- Quick Sort
  1. Pick an arbitrary value called a “pivot” from the array
  2. Using the pivot value “partition” the array
     1. Reorder the array where smaller values are to the left of the pivot, and large / equal values are to the right
     2. Once it has been partitioned the pivot value is where it should be
  3. Recursively continue now with the array to the left, and the array to the right of the pivot

Example

Index	0	1	2	3	4	5	6	7
Value	10	8	7	6	12	5	11	9 pivot

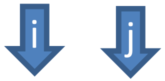
If value at j is smaller than the pivot then swap values at i and j and increase i by 1

File Edit View Help

Sorting Algorithms

- Quick Sort
  1. Pick an arbitrary value called a "pivot" from the array
  2. Using the pivot value "partition" the array
     1. Reorder the array where smaller values are to the left of the pivot, and large / equal values are to the right
     2. Once it has been partitioned the pivot value is where it should be
  3. Recursively continue now with the array to the left, and the array to the right of the pivot

Example



Index	0	1	2	3	4	5	6	7
Value	8	10	7	6	12	5	11	9 pivot

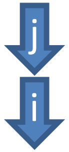
If value at j is smaller than the pivot then swap values at i and j and increase i by 1

File Edit View Help

Sorting Algorithms

- Quick Sort
  1. Pick an arbitrary value called a "pivot" from the array
  2. Using the pivot value "partition" the array
     1. Reorder the array where smaller values are to the left of the pivot, and large / equal values are to the right
     2. Once it has been partitioned the pivot value is where it should be
  3. Recursively continue now with the array to the left, and the array to the right of the pivot

Example



Index	0	1	2	3	4	5	6	7
Value	8	10	7	6	12	5	11	9 pivot

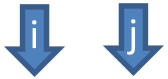
If value at j is smaller than the pivot then swap values at i and j and increase i by 1

File Edit View Help

Sorting Algorithms

- Quick Sort
  1. Pick an arbitrary value called a “pivot” from the array
  2. Using the pivot value “partition” the array
     1. Reorder the array where smaller values are to the left of the pivot, and large / equal values are to the right
     2. Once it has been partitioned the pivot value is where it should be
  3. Recursively continue now with the array to the left, and the array to the right of the pivot

Example



Index	0	1	2	3	4	5	6	7
Value	8	10	7	6	12	5	11	9 pivot

If value at j is smaller than the pivot then swap values at i and j and increase i by 1

File Edit View Help

Sorting Algorithms

- Quick Sort
  1. Pick an arbitrary value called a “pivot” from the array
  2. Using the pivot value “partition” the array
     1. Reorder the array where smaller values are to the left of the pivot, and large / equal values are to the right
     2. Once it has been partitioned the pivot value is where it should be
  3. Recursively continue now with the array to the left, and the array to the right of the pivot

Example

Index	0	1	2	3	4	5	6	7
Value	8	10	7	6	12	5	11	9 pivot

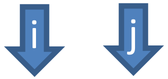
If value at j is smaller than the pivot then swap values at i and j and increase i by 1

File Edit View Help

Sorting Algorithms

- Quick Sort
  1. Pick an arbitrary value called a "pivot" from the array
  2. Using the pivot value "partition" the array
     1. Reorder the array where smaller values are to the left of the pivot, and large / equal values are to the right
     2. Once it has been partitioned the pivot value is where it should be
  3. Recursively continue now with the array to the left, and the array to the right of the pivot

Example



Index	0	1	2	3	4	5	6	7
Value	8	7	10	6	12	5	11	9 pivot

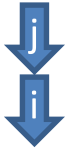
If value at j is smaller than the pivot then swap values at i and j and increase i by 1

File Edit View Help

Sorting Algorithms

- Quick Sort
  1. Pick an arbitrary value called a "pivot" from the array
  2. Using the pivot value "partition" the array
     1. Reorder the array where smaller values are to the left of the pivot, and large / equal values are to the right
     2. Once it has been partitioned the pivot value is where it should be
  3. Recursively continue now with the array to the left, and the array to the right of the pivot

Example



Index	0	1	2	3	4	5	6	7
Value	8	7	10	6	12	5	11	9 pivot

If value at j is smaller than the pivot then swap values at i and j and increase i by 1

File Edit View Help

Sorting Algorithms

- Quick Sort
  1. Pick an arbitrary value called a “pivot” from the array
  2. Using the pivot value “partition” the array
     1. Reorder the array where smaller values are to the left of the pivot, and large / equal values are to the right
     2. Once it has been partitioned the pivot value is where it should be
  3. Recursively continue now with the array to the left, and the array to the right of the pivot

Example

Index	0	1	2	3	4	5	6	7
Value	8	7	10	6	12	5	11	9 pivot

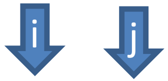
If value at j is smaller than the pivot then swap values at i and j and increase i by 1

File Edit View Help

Sorting Algorithms

- Quick Sort
  1. Pick an arbitrary value called a “pivot” from the array
  2. Using the pivot value “partition” the array
     1. Reorder the array where smaller values are to the left of the pivot, and large / equal values are to the right
     2. Once it has been partitioned the pivot value is where it should be
  3. Recursively continue now with the array to the left, and the array to the right of the pivot

Example



Index	0	1	2	3	4	5	6	7
Value	8	7	6	10	12	5	11	9 pivot

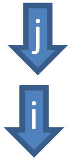
If value at j is smaller than the pivot then swap values at i and j and increase i by 1

File Edit View Help

Sorting Algorithms

- Quick Sort
  1. Pick an arbitrary value called a “pivot” from the array
  2. Using the pivot value “partition” the array
     1. Reorder the array where smaller values are to the left of the pivot, and large / equal values are to the right
     2. Once it has been partitioned the pivot value is where it should be
  3. Recursively continue now with the array to the left, and the array to the right of the pivot

Example



Index	0	1	2	3	4	5	6	7
Value	8	7	6	10	12	5	11	9 pivot

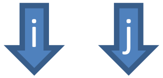
If value at j is smaller than the pivot then swap values at i and j and increase i by 1

File Edit View Help

Sorting Algorithms

- Quick Sort
  1. Pick an arbitrary value called a “pivot” from the array
  2. Using the pivot value “partition” the array
     1. Reorder the array where smaller values are to the left of the pivot, and large / equal values are to the right
     2. Once it has been partitioned the pivot value is where it should be
  3. Recursively continue now with the array to the left, and the array to the right of the pivot

Example



Index	0	1	2	3	4	5	6	7
Value	8	7	6	10	12	5	11	9 pivot

If value at j is smaller than the pivot then swap values at i and j and increase i by 1

File Edit View Help

Sorting Algorithms

- Quick Sort
  1. Pick an arbitrary value called a "pivot" from the array
  2. Using the pivot value "partition" the array
     1. Reorder the array where smaller values are to the left of the pivot, and large / equal values are to the right
     2. Once it has been partitioned the pivot value is where it should be
  3. Recursively continue now with the array to the left, and the array to the right of the pivot

Example

Index	0	1	2	3	4	5	6	7
Value	8	7	6	10	12	5	11	9 pivot

If value at j is smaller than the pivot then swap values at i and j and increase i by 1

File Edit View Help

Sorting Algorithms

- Quick Sort
    1. Pick an arbitrary value called a "pivot" from the array
    2. Using the pivot value "partition" the array
        1. Reorder the array where smaller values are to the left of the pivot, and large / equal values are to the right
        2. Once it has been partitioned the pivot value is where it should be
    3. Recursively continue now with the array to the left, and the array to the right of the pivot

Example

Index	0	1	2	3	4	5	6	7
Value	8	7	6	10	12	5	11	9 pivot

If value at j is smaller than the pivot then swap values at i and j and increase i by 1

File Edit View Help

Sorting Algorithms

- Quick Sort
  1. Pick an arbitrary value called a “pivot” from the array
  2. Using the pivot value “partition” the array
     1. Reorder the array where smaller values are to the left of the pivot, and large / equal values are to the right
     2. Once it has been partitioned the pivot value is where it should be
  3. Recursively continue now with the array to the left, and the array to the right of the pivot

Example

Index	0	1	2	3	4	5	6	7
Value	8	7	6	5	12	10	11	9 pivot

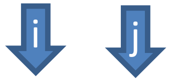
If value at j is smaller than the pivot then swap values at i and j and increase i by 1

File Edit View Help

Sorting Algorithms

- Quick Sort
  1. Pick an arbitrary value called a "pivot" from the array
  2. Using the pivot value "partition" the array
     1. Reorder the array where smaller values are to the left of the pivot, and large / equal values are to the right
     2. Once it has been partitioned the pivot value is where it should be
  3. Recursively continue now with the array to the left, and the array to the right of the pivot

Example



Index	0	1	2	3	4	5	6	7
Value	8	7	6	5	12	10	11	9 pivot

If value at j is smaller than the pivot then swap values at i and j and increase i by 1

File Edit View Help

Sorting Algorithms

- Quick Sort
  1. Pick an arbitrary value called a “pivot” from the array
  2. Using the pivot value “partition” the array
     1. Reorder the array where smaller values are to the left of the pivot, and large / equal values are to the right
     2. Once it has been partitioned the pivot value is where it should be
  3. Recursively continue now with the array to the left, and the array to the right of the pivot

Example

Index	0	1	2	3	4	5	6	7
Value	8	7	6	5	12	10	11	9 pivot

If value at j is smaller than the pivot then swap values at i and j and increase i by 1

File Edit View Help

Sorting Algorithms

- Quick Sort
  1. Pick an arbitrary value called a “pivot” from the array
  2. Using the pivot value “partition” the array
     1. Reorder the array where smaller values are to the left of the pivot, and large / equal values are to the right
     2. Once it has been partitioned the pivot value is where it should be
  3. Recursively continue now with the array to the left, and the array to the right of the pivot

Example

					i			j
Index	**0**	**1**	**2**	**3**	**4**	**5**	**6**	**7**
Value	8	7	6	5	12	10	11	9 pivot

If value at j is smaller than the pivot then swap values at i and j and increase i by 1

File Edit View Help

Sorting Algorithms

- Quick Sort
  1. Pick an arbitrary value called a “pivot” from the array
  2. Using the pivot value “partition” the array
     1. Reorder the array where smaller values are to the left of the pivot, and large / equal values are to the right
     2. Once it has been partitioned the pivot value is where it should be
  3. Recursively continue now with the array to the left, and the array to the right of the pivot

Example

Index	0	1	2	3	4	5	6	7
Value	8	7	6	5	12	10	11	9 pivot

Swap values at i and the pivot

File Edit View Help

Sorting Algorithms

- Quick Sort
  1. Pick an arbitrary value called a “pivot” from the array
  2. Using the pivot value “partition” the array
     1. Reorder the array where smaller values are to the left of the pivot, and large / equal values are to the right
     2. Once it has been partitioned the pivot value is where it should be
  3. Recursively continue now with the array to the left, and the array to the right of the pivot

Example

Index	0	1	2	3	4	5	6	7
Value	8	7	6	5	12	10	11	9 pivot

Swap values at i and the pivot

File Edit View Help

Sorting Algorithms

- Quick Sort
  1. Pick an arbitrary value called a “pivot” from the array
  2. Using the pivot value “partition” the array
     1. Reorder the array where smaller values are to the left of the pivot, and large / equal values are to the right
     2. Once it has been partitioned the pivot value is where it should be
  3. Recursively continue now with the array to the left, and the array to the right of the pivot

Example

Index	0	1	2	3	4	5	6	7
Value	8	7	6	5	9	10	11	12

Swap values at i and the pivot

File Edit View Help

Sorting Algorithms

- Quick Sort
  1. Pick an arbitrary value called a “pivot” from the array
  2. Using the pivot value “partition” the array
     1. Reorder the array where smaller values are to the left of the pivot, and large / equal values are to the right
     2. Once it has been partitioned the pivot value is where it should be
  3. Recursively continue now with the array to the left, and the array to the right of the pivot

Example

i

Index	0	1	2	3	4	5	6	7
Value	8	7	6	5	9	10	11	12

Recursively do the same for the values to the left of the partition and to the right of the partition

File Edit View Help

Sorting Algorithms

- Quick Sort
  1. Pick an arbitrary value called a “pivot” from the array
  2. Using the pivot value “partition” the array
     1. Reorder the array where smaller values are to the left of the pivot, and large / equal values are to the right
     2. Once it has been partitioned the pivot value is where it should be
  3. Recursively continue now with the array to the left, and the array to the right of the pivot

Example

Index	0	1	2	3	4	5	6	7
Value	8	7	6	5 pivot	9	10	11	12 pivot

Recursively do the same for the values to the left of the partition and to the right of the partition

File Edit View Help

Sorting Algorithms

- Quick Sort
  1. Pick an arbitrary value called a “pivot” from the array
  2. Using the pivot value “partition” the array
     1. Reorder the array where smaller values are to the left of the pivot, and large / equal values are to the right
     2. Once it has been partitioned the pivot value is where it should be
  3. Recursively continue now with the array to the left, and the array to the right of the pivot

Example

Index	0	1	2	3	4	5	6	7
Value	8	7	6	5 pivot	9	10	11	12 pivot

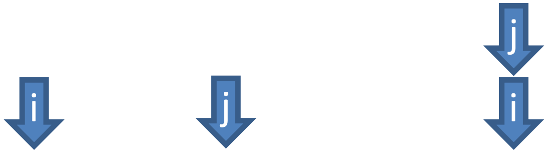
Recursively do the same for the values to the left of the partition and to the right of the partition

File Edit View Help

Sorting Algorithms

- Quick Sort
  1. Pick an arbitrary value called a “pivot” from the array
  2. Using the pivot value “partition” the array
     1. Reorder the array where smaller values are to the left of the pivot, and large / equal values are to the right
     2. Once it has been partitioned the pivot value is where it should be
  3. Recursively continue now with the array to the left, and the array to the right of the pivot

Example



Index	0	1	2	3	4	5	6	7
Value	8	7	6	5 pivot	9	10	11	12 pivot

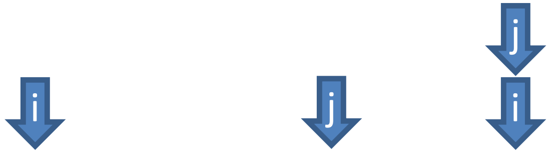
Recursively do the same for the values to the left of the partition and to the right of the partition

File Edit View Help

Sorting Algorithms

- Quick Sort
  1. Pick an arbitrary value called a “pivot” from the array
  2. Using the pivot value “partition” the array
     1. Reorder the array where smaller values are to the left of the pivot, and large / equal values are to the right
     2. Once it has been partitioned the pivot value is where it should be
  3. Recursively continue now with the array to the left, and the array to the right of the pivot

Example



Index	0	1	2	3	4	5	6	7
Value	8	7	6	5 pivot	9	10	11	12 pivot

Recursively do the same for the values to the left of the partition and to the right of the partition

File Edit View Help

Sorting Algorithms

- Quick Sort
    1. Pick an arbitrary value called a “pivot” from the array
    2. Using the pivot value “partition” the array
        1. Reorder the array where smaller values are to the left of the pivot, and large / equal values are to the right
        2. Once it has been partitioned the pivot value is where it should be
    3. Recursively continue now with the array to the left, and the array to the right of the pivot

Example

Index	0	1	2	3	4	5	6	7
Value	5	7	6	8	9	10	11	12 pivot

Recursively do the same for the values to the left of the partition and to the right of the partition

File Edit View Help

Sorting Algorithms

- Quick Sort
  1. Pick an arbitrary value called a “pivot” from the array
  2. Using the pivot value “partition” the array
     1. Reorder the array where smaller values are to the left of the pivot, and large / equal values are to the right
     2. Once it has been partitioned the pivot value is where it should be
  3. Recursively continue now with the array to the left, and the array to the right of the pivot

Example

Index	0	1	2	3	4	5	6	7
Value	5	7	6	8 pivot	9	10	11	12 pivot

Recursively do the same for the values to the left of the partition and to the right of the partition

File Edit View Help

Sorting Algorithms

- Quick Sort
  1. Pick an arbitrary value called a “pivot” from the array
  2. Using the pivot value “partition” the array
     1. Reorder the array where smaller values are to the left of the pivot, and large / equal values are to the right
     2. Once it has been partitioned the pivot value is where it should be
  3. Recursively continue now with the array to the left, and the array to the right of the pivot

Example

Index	0	1	2	3	4	5	6	7
Value	5	7	6	8 pivot	9	10	11	12 pivot

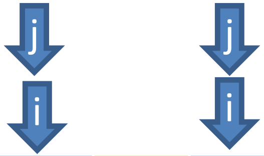
Recursively do the same for the values to the left of the partition and to the right of the partition

File Edit View Help

Sorting Algorithms

- Quick Sort
  1. Pick an arbitrary value called a "pivot" from the array
  2. Using the pivot value "partition" the array
     1. Reorder the array where smaller values are to the left of the pivot, and large / equal values are to the right
     2. Once it has been partitioned the pivot value is where it should be
  3. Recursively continue now with the array to the left, and the array to the right of the pivot

Example



Index	0	1	2	3	4	5	6	7
Value	5	7	6	8 pivot	9	10	11	12 pivot

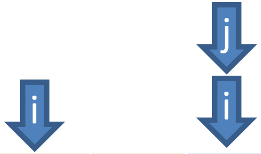
Recursively do the same for the values to the left of the partition and to the right of the partition

File Edit View Help

Sorting Algorithms

- Quick Sort
  1. Pick an arbitrary value called a “pivot” from the array
  2. Using the pivot value “partition” the array
     1. Reorder the array where smaller values are to the left of the pivot, and large / equal values are to the right
     2. Once it has been partitioned the pivot value is where it should be
  3. Recursively continue now with the array to the left, and the array to the right of the pivot

Example



Index	0	1	2	3	4	5	6	7
Value	5	7	6	8	9	10	11	12 pivot

Recursively do the same for the values to the left of the partition and to the right of the partition

File Edit View Help

Sorting Algorithms

- Quick Sort
  1. Pick an arbitrary value called a "pivot" from the array
  2. Using the pivot value "partition" the array
     1. Reorder the array where smaller values are to the left of the pivot, and large / equal values are to the right
     2. Once it has been partitioned the pivot value is where it should be
  3. Recursively continue now with the array to the left, and the array to the right of the pivot

Example

Index	0	1	2	3	4	5	6	7
Value	5	7	6 pivot	8	9	10	11	12 pivot

Recursively do the same for the values to the left of the partition and to the right of the partition

Sorting Algorithms

- Quick Sort
  1. Pick an arbitrary value called a "pivot" from the array
  2. Using the pivot value "partition" the array
     1. Reorder the array where smaller values are to the left of the pivot, and large / equal values are to the right
     2. Once it has been partitioned the pivot value is where it should be
  3. Recursively continue now with the array to the left, and the array to the right of the pivot

Example

Index	0	1	2	3	4	5	6	7
Value	5	7	6 pivot	8	9	10	11	12 pivot

Recursively do the same for the values to the left of the partition and to the right of the partition

File Edit View Help

Sorting Algorithms

- Quick Sort
  1. Pick an arbitrary value called a "pivot" from the array
  2. Using the pivot value "partition" the array
     1. Reorder the array where smaller values are to the left of the pivot, and large / equal values are to the right
     2. Once it has been partitioned the pivot value is where it should be
  3. Recursively continue now with the array to the left, and the array to the right of the pivot

Example

Index	0	1	2	3	4	5	6	7
Value	5	6	7	8	9	10	11	12 pivot

Recursively do the same for the values to the left of the partition and to the right of the partition

File Edit View Help

Sorting Algorithms

- Quick Sort
  1. Pick an arbitrary value called a “pivot” from the array
  2. Using the pivot value “partition” the array
     1. Reorder the array where smaller values are to the left of the pivot, and large / equal values are to the right
     2. Once it has been partitioned the pivot value is where it should be
  3. Recursively continue now with the array to the left, and the array to the right of the pivot

Example

Index	0	1	2	3	4	5	6	7
Value	5	6	7 pivot	8	9	10	11	12 pivot

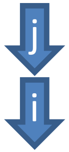
Recursively do the same for the values to the left of the partition and to the right of the partition

File Edit View Help

Sorting Algorithms

- Quick Sort
  1. Pick an arbitrary value called a “pivot” from the array
  2. Using the pivot value “partition” the array
     1. Reorder the array where smaller values are to the left of the pivot, and large / equal values are to the right
     2. Once it has been partitioned the pivot value is where it should be
  3. Recursively continue now with the array to the left, and the array to the right of the pivot

Example



Index	0	1	2	3	4	5	6	7
Value	5	6	7	8	9	10	11	12 pivot

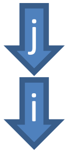
Recursively do the same for the values to the left of the partition and to the right of the partition

File Edit View Help

Sorting Algorithms

- Quick Sort
  1. Pick an arbitrary value called a "pivot" from the array
  2. Using the pivot value "partition" the array
     1. Reorder the array where smaller values are to the left of the pivot, and large / equal values are to the right
     2. Once it has been partitioned the pivot value is where it should be
  3. Recursively continue now with the array to the left, and the array to the right of the pivot

Example



Index	0	1	2	3	4	5	6	7
Value	5	6	7	8	9	10	11	12 pivot

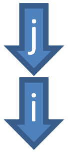
Recursively do the same for the values to the left of the partition and to the right of the partition

File Edit View Help

Sorting Algorithms

- Quick Sort
  1. Pick an arbitrary value called a “pivot” from the array
  2. Using the pivot value “partition” the array
     1. Reorder the array where smaller values are to the left of the pivot, and large / equal values are to the right
     2. Once it has been partitioned the pivot value is where it should be
  3. Recursively continue now with the array to the left, and the array to the right of the pivot

Example



Index	0	1	2	3	4	5	6	7
Value	5	6	7	8	9	10	11	12 pivot

Recursively do the same for the values to the left of the partition and to the right of the partition

File Edit View Help

Sorting Algorithms

- Quick Sort
    1. Pick an arbitrary value called a “pivot” from the array
    2. Using the pivot value “partition” the array
        1. Reorder the array where smaller values are to the left of the pivot, and large / equal values are to the right
        2. Once it has been partitioned the pivot value is where it should be
    3. Recursively continue now with the array to the left, and the array to the right of the pivot

Example

Index	0	1	2	3	4	5	6	7
Value	5	6	7	8	9	10	11	12

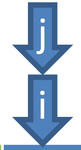
Recursively do the same for the values to the left of the partition and to the right of the partition

File Edit View Help

Sorting Algorithms

- Quick Sort
  1. Pick an arbitrary value called a “pivot” from the array
  2. Using the pivot value “partition” the array
     1. Reorder the array where smaller values are to the left of the pivot, and large / equal values are to the right
     2. Once it has been partitioned the pivot value is where it should be
  3. Recursively continue now with the array to the left, and the array to the right of the pivot

Example



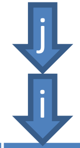
Index	0	1	2	3	4	5	6	7
Value	5	6	7	8	9	10	11 pivot	12

Recursively do the same for the values to the left of the partition and to the right of the partition

Sorting Algorithms

- Quick Sort
  1. Pick an arbitrary value called a "pivot" from the array
  2. Using the pivot value "partition" the array
     1. Reorder the array where smaller values are to the left of the pivot, and large / equal values are to the right
     2. Once it has been partitioned the pivot value is where it should be
  3. Recursively continue now with the array to the left, and the array to the right of the pivot

Example



Index	0	1	2	3	4	5	6	7
Value	5	6	7	8	9	10	11 pivot	12

Recursively do the same for the values to the left of the partition and to the right of the partition

File Edit View Help

Sorting Algorithms

- Quick Sort
  1. Pick an arbitrary value called a “pivot” from the array
  2. Using the pivot value “partition” the array
     1. Reorder the array where smaller values are to the left of the pivot, and large / equal values are to the right
     2. Once it has been partitioned the pivot value is where it should be
  3. Recursively continue now with the array to the left, and the array to the right of the pivot

Example

i

Index	0	1	2	3	4	5	6	7
Value	5	6	7	8	9	10	11	12

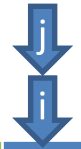
Recursively do the same for the values to the left of the partition and to the right of the partition

File Edit View Help

Sorting Algorithms

- Quick Sort
  1. Pick an arbitrary value called a “pivot” from the array
  2. Using the pivot value “partition” the array
     1. Reorder the array where smaller values are to the left of the pivot, and large / equal values are to the right
     2. Once it has been partitioned the pivot value is where it should be
  3. Recursively continue now with the array to the left, and the array to the right of the pivot

Example



Index	0	1	2	3	4	5	6	7
Value	5	6	7	8	9	10 pivot	11	12

Recursively do the same for the values to the left of the partition and to the right of the partition

File Edit View Help

Sorting Algorithms

- Quick Sort
  1. Pick an arbitrary value called a "pivot" from the array
  2. Using the pivot value "partition" the array
     1. Reorder the array where smaller values are to the left of the pivot, and large / equal values are to the right
     2. Once it has been partitioned the pivot value is where it should be
  3. Recursively continue now with the array to the left, and the array to the right of the pivot

Example

Index	0	1	2	3	4	5	6	7
Value	5	6	7	8	9	10	11	12

Recursively do the same for the values to the left of the partition and to the right of the partition

Quick Sort Complexity

- Worst Case
  - Sorted in Ascending Order
  - Assuming pivot is always picked from the last index
- Operations
  - The first index moves n spaces
  - The first index moves n-1 spaces
  - The first index moves n-2 spaces
  - …
  - The first index moves 1 space

Complexity Example

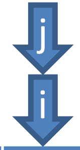
Index	0	1	2	3	4	5	6	7
Value	5	6	7	8	9	10	11	12

File Edit View Help

Quick Sort Complexity

- Worst Case
  - Sorted in Ascending Order
  - Assuming pivot is always picked from the last index
- Operations
  - The first index moves n spaces
  - The first index moves n-1 spaces
  - The first index moves n-2 spaces
  - …
  - The first index moves 1 space

Complexity Example



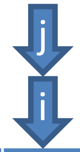
Index	0	1	2	3	4	5	6	7
Value	5	6	7	8	9	10	11	12 pivot

File Edit View Help

Quick Sort Complexity

- Worst Case
  - Sorted in Ascending Order
  - Assuming pivot is always picked from the last index
- Operations
  - The first index moves n spaces
  - The first index moves n-1 spaces
  - The first index moves n-2 spaces
  - …
  - The first index moves 1 space

Complexity Example



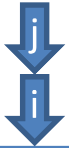
Index	0	1	2	3	4	5	6	7
Value	5	6	7	8	9	10	11	12 pivot

File Edit View Help

Quick Sort Complexity

- Worst Case
  - Sorted in Ascending Order
  - Assuming pivot is always picked from the last index
- Operations
  - The first index moves n spaces
  - The first index moves n-1 spaces
  - The first index moves n-2 spaces
  - …
  - The first index moves 1 space

Complexity Example



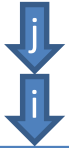
Index	0	1	2	3	4	5	6	7
Value	5	6	7	8	9	10	11	12 pivot

File Edit View Help

Quick Sort Complexity

- Worst Case
  - Sorted in Ascending Order
  - Assuming pivot is always picked from the last index
- Operations
  - The first index moves n spaces
  - The first index moves n-1 spaces
  - The first index moves n-2 spaces
  - …
  - The first index moves 1 space

Complexity Example



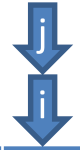
Index	0	1	2	3	4	5	6	7
Value	5	6	7	8	9	10	11	12 pivot

File Edit View Help

Quick Sort Complexity

- Worst Case
  - Sorted in Ascending Order
  - Assuming pivot is always picked from the last index
- Operations
  - The first index moves n spaces
  - The first index moves n-1 spaces
  - The first index moves n-2 spaces
  - …
  - The first index moves 1 space

Complexity Example



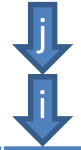
Index	0	1	2	3	4	5	6	7
Value	5	6	7	8	9	10	11	12 pivot

File Edit View Help

Quick Sort Complexity

- Worst Case
  - Sorted in Ascending Order
  - Assuming pivot is always picked from the last index
- Operations
  - The first index moves n spaces
  - The first index moves n-1 spaces
  - The first index moves n-2 spaces
  - …
  - The first index moves 1 space

Complexity Example



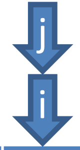
Index	0	1	2	3	4	5	6	7
Value	5	6	7	8	9	10	11	12 pivot

File Edit View Help

Quick Sort Complexity

- Worst Case
  - Sorted in Ascending Order
  - Assuming pivot is always picked from the last index
- Operations
  - The first index moves n spaces
  - The first index moves n-1 spaces
  - The first index moves n-2 spaces
  - …
  - The first index moves 1 space

Complexity Example



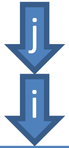
Index	0	1	2	3	4	5	6	7
Value	5	6	7	8	9	10	11	12 pivot

File Edit View Help

Quick Sort Complexity

- Worst Case
  - Sorted in Ascending Order
  - Assuming pivot is always picked from the last index
- Operations
  - The first index moves n spaces
  - The first index moves n-1 spaces
  - The first index moves n-2 spaces
  - …
  - The first index moves 1 space

Complexity Example



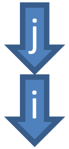
Index	0	1	2	3	4	5	6	7
Value	5	6	7	8	9	10	11	12 pivot

File Edit View Help

Quick Sort Complexity

- Worst Case
  - Sorted in Ascending Order
  - Assuming pivot is always picked from the last index
- Operations
  - The first index moves n spaces
  - The first index moves n-1 spaces
  - The first index moves n-2 spaces
  - …
  - The first index moves 1 space

Complexity Example



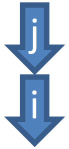
Index	0	1	2	3	4	5	6	7
Value	5	6	7	8	9	10	11 pivot	12

File Edit View Help

Quick Sort Complexity

- Worst Case
  - Sorted in Ascending Order
  - Assuming pivot is always picked from the last index
- Operations
  - The first index moves n spaces
  - The first index moves n-1 spaces
  - The first index moves n-2 spaces
  - …
  - The first index moves 1 space

Complexity Example



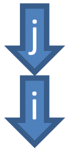
Index	0	1	2	3	4	5	6	7
Value	5	6	7	8	9	10 pivot	11	12

File Edit View Help

Quick Sort Complexity

- Worst Case
  - Sorted in Ascending Order
  - Assuming pivot is always picked from the last index
- Operations
  - The first index moves n spaces
  - The first index moves n-1 spaces
  - The first index moves n-2 spaces
  - …
  - The first index moves 1 space

Complexity Example



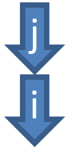
Index	0	1	2	3	4	5	6	7
Value	5	6	7	8	9 pivot	10	11	12

File Edit View Help

Quick Sort Complexity

- Worst Case
  - Sorted in Ascending Order
  - Assuming pivot is always picked from the last index
- Operations
  - The first index moves n spaces
  - The first index moves n-1 spaces
  - The first index moves n-2 spaces
  - …
  - The first index moves 1 space

Complexity Example



Index	0	1	2	3	4	5	6	7
Value	5	6	7	8	9	10	11	12

File Edit View Help

Quick Sort Complexity

- Worst Case
  - Sorted in Ascending Order
  - Assuming pivot is always picked from the last index
- Operations
  - The first index moves n spaces
  - The first index moves n-1 spaces
  - The first index moves n-2 spaces
  - …
  - The first index moves 1 space

Complexity

$$O(n^2)$$

File Edit View Help

Quick Sort Merge Sort Compared

Merge Sort

- Worst Time Complexity = $O(n\lg(n))$
- Average Time Complexity = $\Theta(n\lg(n))$
- Worst Space Complexity = $O(n)$ additional

Quick Sort

- Worst Time Complexity = $O(n^2)$
- Average Time Complexity = $\Theta(n\lg(n))$
- Worst Space Complexity = $O(\lg(n))$ additional