

### Sorting - Time Complexities

	Best	Average	Worst	Space
<b>Merge Sort</b>	$O(n \log(n))$	$O(n \log(n))$	$O(n \log(n))$	$O(n)$
<b>Quick Sort</b>	$O(n \log(n))$	$O(n \log(n))$	$O(n^2)$	$O(\log(n))$
<b>Bubble Sort</b>	$O(n)$	$O(n^2)$	$O(n^2)$	$O(1)$
<b>Insertion Sort</b>	$O(n)$	$O(n^2)$	$O(n^2)$	$O(1)$

### Insertion Sort

Split the collection into an ordered & unordered part.

At first the ordered part is just the first element.

Start from the first unordered index and move the item left until you find its place.

### Insertion Sort

```
function insertionSort(arr) {
  let currentVal
  for (let i = 1; i < arr.length; i++) {
    currentVal = arr[i]
    for (let j = i - 1; j >= 0 && arr[j] > currentVal; j--) {
      arr[j+1] = arr[j]
    }
    arr[j+1] = currentVal
  }
  return arr
}
```

### Quick Sort

```
// place pivot in the right index and return pivot
index
function pivot(arr: number[], start = 0, end =
arr.length - 1) {
  const pivot = arr[start];
  let pivotIndex = start;

  for (let i = start + 1; i < end; i++) {
    if (arr[i] < pivot) {
      pivotIndex++;
      [arr[pivotIndex], arr[i]] = [arr[i], arr[pivotIndex]];
    }
  }
  [arr[start], arr[pivotIndex]] = [arr[pivotIndex], arr[start]];
}

function quickSort(arr: number[], start = 0, end =
arr.length - 1) {
  if (left < right) {
    const pivot = pivot(arr, start, end);

    // left
    quickSort(arr, start, pivotIndex - 1);
    // right
    quickSort(arr, pivotIndex + 1, end);
  }

  return arr;
}
```

### Merge Sort

```
function merge(arr1: number[], arr2: number[]):
number[] {
  let result = []
  let i = 0
  let j = 0

  while (i < arr1.length && j < arr2.length) {
    if (arr1[i] < arr2[j]) {
```



By menos

[cheatography.com/menos/](https://cheatography.com/menos/)

Not published yet.

Last updated 30th June, 2022.

Page 1 of 3.

Sponsored by [CrosswordCheats.com](https://CrosswordCheats.com)

Learn to solve cryptic crosswords!

<http://crosswordcheats.com>

### Merge Sort (cont)

```

    result.push(arr1[i])
    i++
  } else {
    result.push(arr2[j])
    j++
  }
}

while (i < arr1.length) {
  result.push(arr1[i])
  i++
}

while (j < arr2.length) {
  result.push(arr2[j])
  j++
}

return result
}

function mergeSort(arr: number[]): number[] {
  if (arr.length <= 1) return arr

  const middle = Math.floor(arr.length / 2)

  const left = mergeSort(arr.slice(0, middle))
  const right = mergeSort(arr.slice(middle))

  return merge(left, right)
}

```

### BFS vs DFS

Breadth-first	Depth-first
Use for finding the shortest path between 2 nodes	Getting all the way to the end of a tree
Level-order	Recursive
Better when target is far from the source	Better when target is closer
Generally slower	Generally faster
Queue (FIFO)	Stack (LIFO)

### BFS: Max Depth of Binary Tree

```

var maxDepth = function(root) {

  if (!root) return 0

  let queue = [root]
  let max = 0

  while (queue.length > 0) {

    max++
    let len = queue.length

    for (let i = 0; i < len; i++) {
      let node = queue.shift()
      if (node.left) queue.push(node.left)
      if (node.right) queue.push(node.right)
    }

  }

  return max
};

```



### Binary Search

```
function binarySearch(sortedArr: number[], value:
number): number {
  let left = 0;
  let right = sortedArr.length - 1;

  while (left <= right) {
    const middle = Math.round((right + left) / 2);

    if (sortedArr[middle] > value) {
      right = middle - 1;
    } else if (sortedArr[middle] < value) {
      left = middle + 1;
    } else {
      return middle;
    }
  }
  return -1;
}
```

### Min Window Substring (Sliding window)

```
/**
Given two strings s and t of lengths m and n
respectively,
return the minimum window substring of s such that
every character in t (including duplicates)
is included in the window.
If there is no such substring, return the empty
string "".
*/
var minWindow = function(s, t) {

  let map = new Map()

  for (let c of t) {
    map.set(c, map.has(c) ? map.get(c) + 1 : 1)
  }

  let start = 0
```

### Min Window Substring (Sliding window) (cont)

```
let end = 0
let charsNeeded = t.length
let minSubstring = s + ' '

while (end < s.length) {

  if (map.has(s[end])) {
    map.set(s[end], map.get(s[end]) - 1)
    if (map.get(s[end]) >= 0) {
      charsNeeded--
    }
  }

  while(charsNeeded === 0 && start <= end) {
    let currentString = s.slice(start, end+1)
    if (currentString.length <= minSubstring.le-
ngth) {
      minSubstring = currentString
    }

    if (map.has(s[start])) {
      map.set( s[start], map.get(s[start]) + 1 )
      if (map.get(s[start]) > 0) {
        charsNeeded++
      }
    }
    start++
  }

  end++
}

return minSubstring.length > s.length ? '' :
minSubstring
};
```



By menos

[cheatography.com/menos/](https://cheatography.com/menos/)

Not published yet.

Last updated 30th June, 2022.

Page 3 of 3.

Sponsored by [CrosswordCheats.com](https://CrosswordCheats.com)

Learn to solve cryptic crosswords!

<http://crosswordcheats.com>