Javascript Algorithmic Cookbook Cheatsheet Cheat Sheet

by BriYvonne01 (briyvonne01) via cheatography.com/61413/cs/31853/

Involving collections

Two Pointer Technique	Two pointers, each starting from the beginning and the end until they both meet OR One pointer moving at a slow pace, while the other pointer moves at twice the speed

Collections are arrays, string, or linked lists. Other application: searching for pairs in an array, two sorted arrays need to be merged (2 pointers for both arrays), the pattern of using a fast pointer and a slow pointer (detecting cycles in a linked list)

The Two Heaps Technique

Scheduling	find the earliest or latest meeting at any given time

Finding the median in a large collection.

Largest and smallest values in a set.

To implement a priority queue

Binary Search Technique

Binary search is a technique for finding out whether an item is present in a sorted array or not

It makes use of the fact that if a value is greater than array[i] and the array is sorted, then the value has to lie at index (i+1) or higher. Similarly, if the value is less than array[i], then it has to lie at an index less than i.

Binary search works by comparing the value to search for with the middle item of the array. If the value is higher than the middle item, then the search proceeds in the right half of the array. If the value is less than the middle, then the left sub-array is searched. This process is repeated, and each time, the sub-array within which the value is being searched is reduced by half.

This gives an overall time complexity of O(log n).

Merge Sort Algorithm

Divides the input collection into two portions, recursively calls itself for the two divided portions, and then merges the two sorted portions. Merge Sort Algorithm works in three parts:

Divide: At first, it divides the given collection of elements into two equal halves and then recursively passes these halves to itself. Sorting: Unless, there is only one element left in each half, the division continues, after that, those two end elements (leaf nodes) are sorted. Merging: Finally, the sorted arrays from the top step are merged in order to form the sorted collection of elements

Time Complexity: O(NlogN)

Space Complexity: O(N)

Bubble Sort Algorithm

The adjacent elements in the collection are compared and the positions are swapped if the first element is greater than the second one. In this way, the largest valued element "bubbles" to the top.

Usually, after each loop, the elements furthest to the right are in sorted order. The process is continued until all the elements are in sorted order.

Time Complexity: O(N2)

Space Complexity: O(1)

Keeping track of the "balancing"

When you want to keep track of what's "coming in" or incoming and what's what's "going out" or outgoing

String methods

length	gets the string length
indexOf()	gets the index of first occurence of the character or a substring in the string.
toLowe-rCase()	converts the string to lowercase (or similarly, to upperc-ase).
substring	gets a substring from defined index point in the string.

Javascript Algorithmic Cookbook Cheatsheet Cheat Sheet

by BriYvonne01 (briyvonne01) via cheatography.com/61413/cs/31853/

Array Methods

pop	removes (pops) the last element of an array
push	method adds new items to the end of an array
shift	method removes the first item of an array
unshift	adds new elements to the beginning of an array
sort	sorts the elements of an array in alphabetical in ascending order
filter	method creates a new array filled with elements that pass a test provided by a function. array.filter(function(curre-ntValue, index, arr), thisValue)
map	creates a new array from calling a function for every array element.
foreach	method calls a function for each element in an array. array.forEach(function(currentValue, index, arr), thisValue)
find	method returns the value of the first element that passes a test. array.find(function(currentValue, index, arr),this-Value)
includes	returns true if an array contains a specified value. "fru-its.includes("Mango");"
reverse	reverses the order of the elements in an array
every	executes a function for each array element. returns true/false if the function returns true/false for all elements.
join	method returns an array as a string

When to use BFS instead of DFS?

When you need to find the shortest path between any two nodes in an unweighted graph.

If you're solving a problem, and know a solution is not far from the root of the tree, BFS will likely get you there faster.

If the tree is very deep, DFS with heuristics might be faster.

Depth First Search (DFS)

Preorder Traversal	We start from the root node and search the tree vertically (top to bottom), traversing from left to right. The order of visits is ROOT-LEFT-RIGHT
In-order Traversal	Start from the leftmost node and move to the right nodes traversing from top to bottom. The order of visits is LEFT-ROOT-RIGHT
Post-order Traversal	Start from the leftmost node and move to the right nodes, traversing from bottom to top. The order of visits is LEFT-RIGHT-ROOT

In Depth First Search (DFS), a tree is traversed vertically from top to bottom or bottom to top. As you might infer from its namesake, we will traverse as deeply as possible, before moving on to a neighbor node. There are three main ways to apply Depth First Search to a tree.

Time complexity of an array (worst case)

Access	O(1)
Search	O(n)
Insertion	O(n)
append	O(1)
Deletion	O(n)

Insertion Sort Algorithm

The collection is virtually split into an ordered and an unordered part. Elements from the unordered portion are picked and placed at the correct index in the ordered part.

Time Complexity: O(N2)

Space Complexity: O(1)