**JavaScript Built-in Sort Method**

We can tell the built-in sort method to accept an optional comparator function

We can use this comparator method to tell JS how you want it to sort

The comparator looks at pairs of elements (a and b), determines their sort order based on the return value:

* If it returns a negative number, a should come before b
* If it returns a positive number, a should come after b
* If it returns 0, a and b are the same as far as the sort is concerned

*Example:*

function numberCompare(num1, num2) {

return num1 – num2;

}

function comparebyLen(str1, str2) {

return str1.length – str2.length;

}

[6, 4, 15, 10].sort(numberCompare); // [4, 6, 10, 15]

[“Steele”, “Colt”, “Data Structures”, “Algorithm”].sort(compaebyLen); // [“Colt”, “Steele”, .. ]

**Bubble Sort**

*Definition*: a sorting algorithm where the largest values bubble up to the top (swap if found bigger values)

*Graph:*

[5, 3, 4, 1, 2] => [3, 5, 4, 1, 2] => [3, 4, 5, 1, 2] => [3, 4, 1, 5, 2] => [3, 4, 1, 2, 5]. Then continue again until all is sorted

*Pseudocode:*

* Start looping with a variable called i from the end of the array towards the beginning
* Start an inner loop with a variable called j from the beginning until i – 1
* If arr[j] > arr[j+1], swap those two values
* Return the sorted array

*Solution:*

function bubbleSort(arr) {

var noSwaps;

for (var i = arr.length, i > 0; i--) {

noSwaps = true;

for (j = 0; j < i; j++) {

if (arr[j] > arr[j+1]) {

// SWAP!

var temp = arr[j];

arr[j] = arr[j+1];

arr[j+1] = temp;

noSwaps = false;

}

}

}

return arr;

}

*Big O:* O(n^2) because we have nested loop but it will be O(n) in best case if we have noSwaps variable

**Selection Sort**

*Definition*: Similar to bubble sort, but instead of first placing large values into sorted position, it places small values into sorted position (going through and select smallest element then put it at the beginning)

*Graph:*

[5, 3, 4, 1, 2] => [5, 3, 4, 1, 2] => [5, 3, 4, 1, 2] => [5, 3, 4, 1, 2] => [1, 3, 4, 5, 2]. Then continue when all is sorted.

*Pseudocode:*

* Store the first element as the smallest value you’ve seen so far
  + Ex: [19, 5, 17, 9] // 19 is assumed to be smallest
* Compare this item to the next item in the array until you find a smaller number
* If a smaller number is found, designate that smaller number to be the new “minimum” and continue until the end of the array
* If the “minimum” is not the value (index) you initially began with, swap the two values

*Solution:*

function selectionSort(arr) {

for (var i = 0; i < arr.length; i++) {

var min = i;

for (var j = i + 1; j < arr.length; j++) {

if (arr[j] < arr[min]) {

min = j;

}

}

if (i !== min) {

var temp = arr[i];

arr[i] = arr[min];

arr[min] = temp;

]

}

return arr;

}  
*Big O:* O(n^2) because we have nested loop

**Insertion Sort**

*Definition:* Builds up the sort by gradually creating a larger left half which is always sorted

*Graph:*

[5, 3, 4, 1, 2] => [3, 5, 4, 1, 2] => [3, 4, 5, 1, 2] => [1, 3, 4, 5, 2] => [1, 2, 3, 4, 5]

*Pseudocode:*

* Start by picking the second element in the array
* Now compare the second element with the one before it and swap it if necessary
* Continue to the next element and if it is in the correct order, iterate through the sorted portion (i.e, the left side) to place the element in the correct place

*Solution:*

function insertionSort(arr) {

for (var i = 1; i < arr.length; i++) {

var current = arr[i];

for (var j = i – 1; j >= 0 && arr[j] > current; j--) {

arr[j+1] = arr[j]

}

arr[j+1] = current;

}

return arr;

}

*Big O*: O(n^2) because we have nested for loop, best used if data is almost sorted

**Big O Sorting Algorithms**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Algorithm** | **Time Complexity (Best)** | **Time Complexity (Average)** | **Time Complexity (Worst)** | **Space Complexity** |
| Bubble Sort | O(n) | O(n^2) | O(n^2) | O(1) |
| Insertion Sort | O(n) | O(n^2) | O(n^2) | O(1) |
| Selection Sort | O(n^2) | O(n^2) | O(n^2) | O(1) |