QuickSort

Jose Chavez

December 27, 2021

Abstract

We cover implementations of the QuickSort algorithm in JavaScript and Python.

[1]¡++¿

Jose Dev

I think this would be nice side by side with an html elemement that visitors could simply copy and paste from.

Discussion

The general idea of the algorithm is that we recursively do the following: select a pivot point (an element of the array) which we take to the be right most element of arr and partition so that everything to the of the index containing the value of the pivot point has *value* less than the original pivot's value and everything with index greater than the index of the new pivot's place has value greater than that of the original pivot.

Functions, Variables and Keywords

Primary

1. **pivot**: the element used to partition the array;

[1] On this article's organization:

2. **partitionAboutAPivot**: This is a function that will

3. **wallPointer/wP** the pointer to the index of the place in the array that will eventually divide the array according to the elements less than the pivot and those greater than the pivot;

Secondary

1. **pointers**:

2. **Partition**

The Code

```
function quickSort(items) {
    return quickSortHelper(items, 0, items.length - 1);
}

function quickSortHelper(items, left, right) {
    var index;
    if (items.length > 1) {
        index = partition(items, left, right);

        if (left < index - 1) {
            quickSortHelper(items, left, index - 1);
        }

        if (index < right) {
            quickSortHelper(items, index, right);
        }
    }
    return items;
}

function partition(array, left, right) {
    var pivot = array[Math.floor((right + left) / 2)];
```

```
23      while (left <= right) {
24          while (pivot > array[left]) {
25              left++;
26          }
27          while (pivot < array[right]) {
28              right--;
29          }
30          if (left <= right) {
31              var temp = array[left];
32              array[left] = array[right];
33              array[right] = temp;
34              left++;
35              right--;
36          }
37      }
38      return left;
39  }
```

Hints

Here's a list of questions and/or hints that might help jog the mind when attempting to devise a solution yourself.

1. Q: When does **wP** move?;
2. Q: Does **wP** ever move by more than one place at a time? ;
3. Q: How many loops are in the partition function? ;
4. Q: What is the return type of the partition function?

¡++¿