Quicksort Algorithm

I learned the concept of this sorting algorithm few days ago. I’ve decided to implement it without looking at the actual implementation.

The recursion took me a few days to implement since I had a hard time to think about how to recur code until a condition has been fulfilled. It turns out that I just have to nest the functions together and surround them with proper conditions to execute.

Although my code could work sometimes, it means that there is still something wrong with the recursion. Especially for the right recursion function calls.

Here are the samples of successful recursion and failed ones (green and red respectively):

Sample #1

[ 67, 23, 92, 18, 17, 83, 90, 94, 62, 86,  ] - initial

[ 17, 18, 23, 62, 83, 67, 86, 94, 92, 90,  ]

Sample # 2

[ 20, 40, 62, 6, 41, 47, 70, 76, 44, 60,  ] - initial

[ 6, 40, 20, 41, 44, 47, 60, 76, 62, 70,  ]

Sample #3

[ 57, 57, 59, 83, 12, 23, 54, 60, 59, 36, ] – initial

Pivot value is 59----

[ 12, 23, 36, 54, 57, 57, 59, 59, 60, 83, ]

Sample # 4

[ 4, 15, 19, 36, 94, 70, 55, 91, 21, 18, ] – initial

[ 4, 15, 18, 19, 21, 55, 70, 91, 94, 36, ]

I’m thinking of working out this samples based on the algorithmic concept tree I made in the xmind file so that I am able to follow through based on the code I wrote…

Some lists of flaws I’ve seen while deducting those samples:

When I look at Sample #3

[12, 23, 36, 83, 57, 57, 54, 60, 59, 59]

[23, 12, 36.. // quicksort left function () completed

[12, 23, 36.. // quicksort right function () completed

I just realised that there was something wrong with my code:

Wrong sectors of my algorithm

Partitioning Function():

while(j < lastPivot) {

**if**(j == lastPivot -1 ) {

QuicksortR\_L Function():

**if**(side == 'R') {

end = end - 1;

mainPointer = mainPointer + 1;

}**else** **if**(side == 'L') {

end = end - 1;

By ‘end’ variable I mean the pivot obtained from the previous function (Sorry for bad naming, I will habitually fix my incorrectness in the future XD)

To put it simply, when the previous pivot has been obtained, the **end variable** is subtracted by one in the quicksort function() and again in the partitioning function().

Which means, instead of

[12, 23, 36..

| new pivot

| previous pivot

I have this:

[12, 23, 36…

| new pivot

| previous pivot

That is the reason why when

[23, 12, 36..

| pivot returned from the partitioning function and then passed to quicksort right()

| previous pivot (prevEnd)

The quicksort right runs and thus, giving me a sorted array:

[12, 23, 36..

When actually, it can be solved if I correct the subtraction in the partitioning function and make necessary changes.. thus **solving the right problems and making things efficient**