Quick Sort - 1

01 July 2022 20:09

- divide & conquer } ⇒ class of problems.
- works on concept of $\underbrace{\text{Partitioning}}_{\text{heart of quick sort}}$

1 8 3 9 4 5 7

① choose a pivot element → last elem.

| Every elem < 7 | | 7 | | Every elem ≥ 7 |

| 1 | 3 | 4 | 5 | | 7 | | 8 | 9 |

Q. 9 4 3 2 6 8 1 7 → Pivot

< 7 Pivot > 7

| 4 3 2 6 1 | | 7 | | 9 8 | → Pivot

? . 1 | 4 3 2 6 | ↓ Pivot 7 ? 8 9

1 | 4 3 2 | 6 ? 7 8 9

↓ Pivot

1 ? 2 | 4 3 | 6 7 8 9

? 3 4

1 2 3 4 6 7 8 9

| 8 7 6 1 0 9 2 |

↓ Pivot

| 1 0 | | 2 | | 8 7 6 9 |

0 1 2 | 8 7 6 | 9 |

0 1 2 . 6 | 8 7 | 9

| 0 1 2 6 - 7 8 9 |

Psudo code :-

low high

Pivot

quickSort(arr, low, high){

if (low ≥ high){

return;

}

let Pi = Partition (arr, low, high); }

quick Sort (arr, low, Pi-1);

quickSort (arr, Pi+1, end)

}

i

low high

2 12 3 11 1 6 8 10

j L> Pivot

Before i Pointer all elements be

less than pivot elem.

let pivot = arr[high]
for(let j = low; j < high; j++) {
if (arr[j] < pivot) {
i++;
swap(arr[i], arr[j]);
}

low = high

low i high
2 3 1 6 8 11 12 10
↳ pivot
j

arr[j] < pivot

< pivot | > pivot | pivot

swap(arr[i+1], pivot);

low i high
2 3 1 6 8 11 12 10
↳ pivot
j

0 - 1 => -1
low => i - 1 => -1

2 3 1 6 8 10 12 11

2 < 10
↳ i++
swap()

i
2 3 9 11 1 6 8 10
low high
j
let pi = arr[high]

low & high => marking regions
pi => not a pointer where to store.

i => low - 1 // low
j => moving from low to high
↳ low

low
i
2 12 3 11 1 6 8 10
j pivot
high

12 > 10 → no steps => move on } j++

low
i
2 12 3 11 1 6 8 10
i pivot
j high

arr[j] < pivot } i++

3 < 10 } true

Swap(arr[i], arr[j])

low i high

2 12 3 11 1 6 8 10

j pivot

low i j

2 3 12 11 1 6 8 10

j high

low i

2 3 12 11 1 6 8 10

j high

11 < 10 } false - j++

1 < 10 } true i++ swap( )

low j i

2 3 12 11 1 6 8 10

j high

i

2 3 1 11 12 6 8 10

j

6 < 10 true

j i

2 3 1 11 12 6 8 10

j

i++ swap

i

2 3 1 6 12 11 8 10

j

8 < 10

2 3 1 6 | 12 11 8 10

low

high

2 3 1 6 8 - 10 12 11

< 10

0 8 4 3 6 2 9 1 7

_07_QuickSort.js > QuickSort > pi

```
function QuickSort(arr, low, high){
    // base case
    if(low >= high){
        return;
    }
    let pi = partition(arr, low, high);
    QuickSort(arr, low, pi-1);
    QuickSort(arr, pi + 1, high);
}
function partition(arr, low, high){
    let pivot = arr[high];
    let i = low - 1;
    for(let j = low; j < high; j++){
        if(arr[j] < pivot){
            i++;
            [arr[j],arr[i]] = [arr[i], arr[j]];
        }
    }
    // put pivot element at the correct position
    [arr[i + 1], arr[high]] = [arr[high], arr[i + 1]];
    return i + 1;
}
let arr = [4,2,1,5,6,8];
QuickSort(arr,0,arr.length-1);
console.log(arr);
```

debug

0 5 6 9 1 10 12 15

pivot

5 6 9 1 10 12 | 15

5 6 9 1 10 | 12 | 15

n length

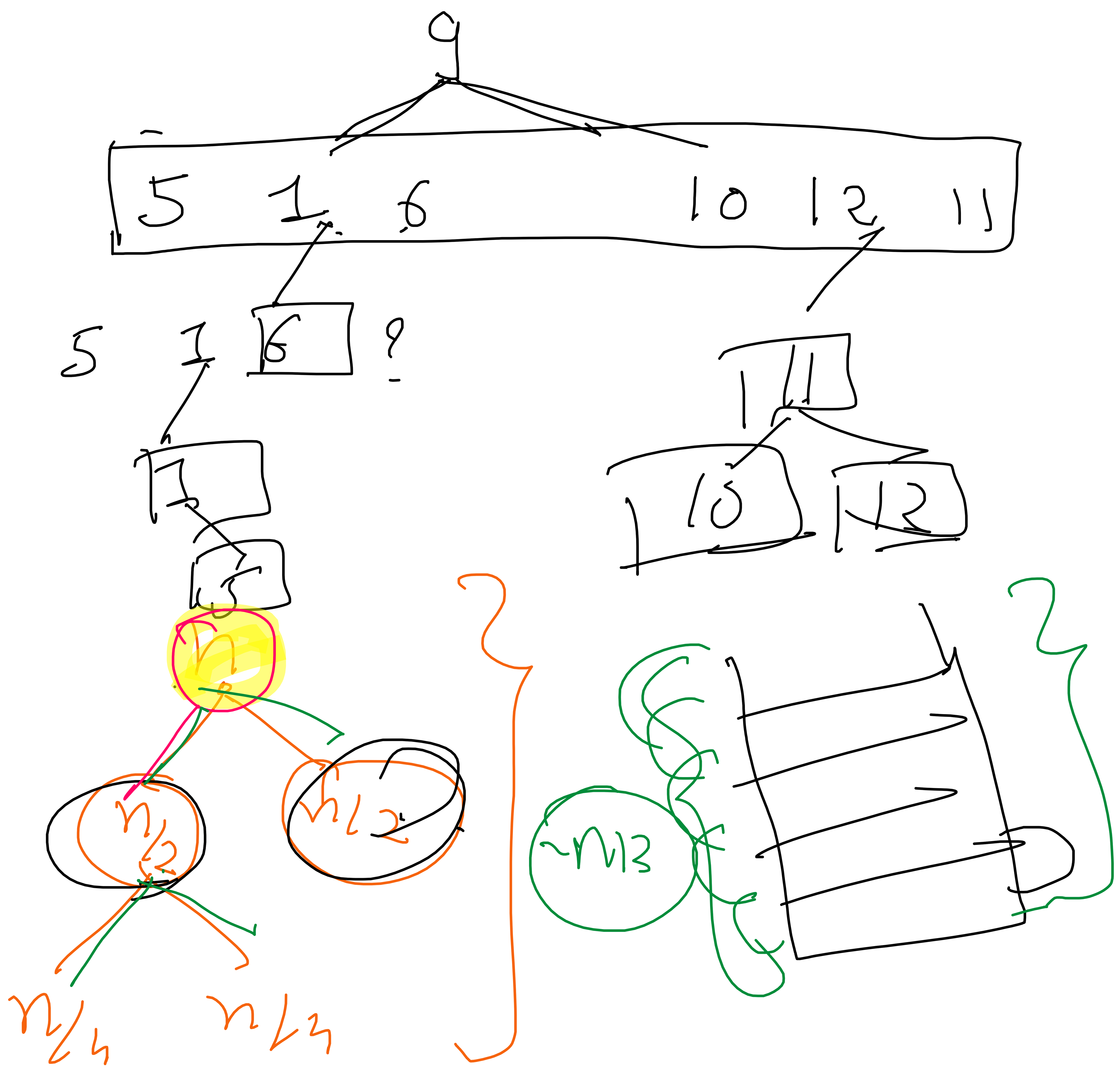
n layers

$$T(n) = T(n-1) + T(1) + O(n)$$

Partitions

$$\boxed{T(n) = O(n^2)}$$

5 10 1 12 6 11 $\boxed{9}$



$$T(n) = T\left(\frac{n}{2}\right) + T\left(\frac{n}{2}\right) + O(n)$$

$$= 2T\left(\frac{n}{2}\right) + O(n)$$

$$\vdots$$

$$\boxed{O(n \log n)}$$

Time Complexity

Best - $O(n \log n)$
Worst - $O(n^2)$
Avg - $O(n \log n)$

Space Complexity :- $\boxed{O(\log n)}$

$\boxed{n \} \Rightarrow n/2}$

$\boxed{\log_2 25} \Rightarrow 5$

$\boxed{\frac{25}{5}}$

$\boxed{\text{Stable}}$

Merge
Best - $O(n \log n)$
Worst - $O(n \log n)$
Avg - $O(n \log n)$

SC - $O(n)$

SC → extra space - $\boxed{O(n)}$

Quick
Best - $O(n \log n)$
Avg - $O(n \log n)$
Worst - $O(n^2)$

S C → $O(\log n)$

R-C

extra space
$O(1)$

$\boxed{\text{Quick Select}}$

Merge / Quick

1 2 3 10_a 5 10_b 8 60

1 2 3 5 8 10a 10b 60

19

[illegible]