CS202 - Algorithm Analysis

Quick Sort

Aravind Mohan

Allegheny College

March 29, 2021

Discussion Based On ...

Sedgewick 2.3 Quick Sort

Quick Sort Algorithm

Strategy:

- **Divide:** partition array into 2 subarrays such that elements in the lower part $\leq$ elements in the higher part.
- **Conquer:** recursively sort the 2 subarrays.
- **Combine:** trivial since sorting is done in place.

Quick Sort Algorithm

Characteristics:

- sort almost in ”place”, i.e., does not require an additional array.
- **pivot** is generally chosen as the last element.
- very practical, average and best case sort performance $O(N \times log(N))$, with small constant factors.
- worst case running time is $O(N^2)$

Quick Sort Algorithm

Partitioning Procedure (linear)

Algorithm - Partition(A, p, r)

Input: an n-element un-sorted array A of integer values, a lower bound p of the array A, and a pivot r in the array A.

Output: an n-element sorted array A of integer values.

```
$i \leftarrow p - 1$
for j = p to r-1 do
    if A[j] $\leq$ A[r] then
        $i \leftarrow i + 1$
        swap A[i] and A[j]
    end if
end for
swap A[i+1] and A[r]
return i+1
```

Quick Sort Algorithm

QuickSort Procedure (linear)

Algorithm - QuickSort(A, p, r)

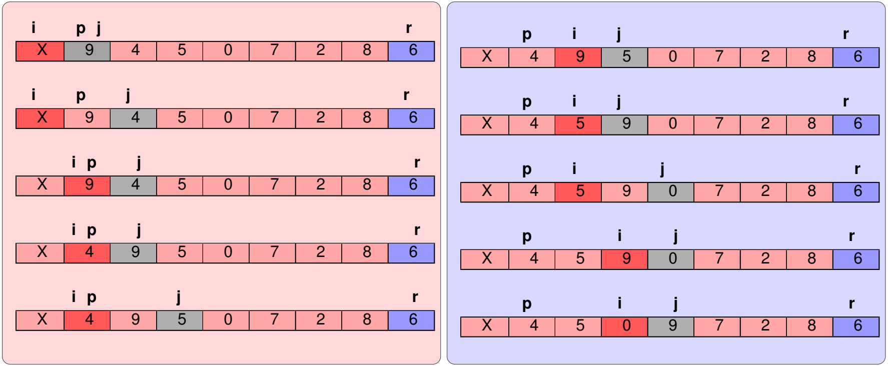
Input: an n-element un-sorted array A of integer values, a lower bound p of the array A, and a pivot r in the array A.

Output: an n-element sorted array A of integer values.

```
if p $<$ r then
    $q \leftarrow$ Partition(A,p,r)
    QuickSort(A, p, q-1)
    QuickSort(A, q+1, r)
end if
```

Partitioner Example

9	4	5	0	7	2	8	6



i	p j							r
X	9	4	5	0	7	2	8	6

i	p	j						r
X	9	4	5	0	7	2	8	6

	i p	j						r
X	9	4	5	0	7	2	8	6

	i p	j						r
X	4	9	5	0	7	2	8	6

	i p		j					r
X	4	9	5	0	7	2	8	6

	p	i	j					r
X	4	9	5	0	7	2	8	6

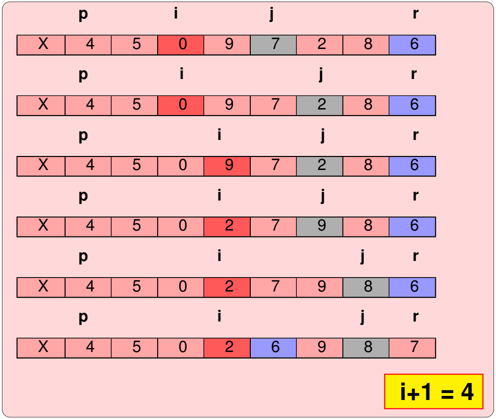
	p	i	j					r
X	4	5	9	0	7	2	8	6

	p	i		j				r
X	4	5	9	0	7	2	8	6

	p		i	j				r
X	4	5	9	0	7	2	8	6

	p		i	j				r
X	4	5	0	9	7	2	8	6

Partitioner Example

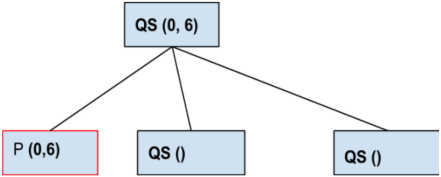


	p		i		j			r
X	4	5	0	9	7	2	8	6
	p		i			j		r
X	4	5	0	9	7	2	8	6
	p			i		j		r
X	4	5	0	9	7	2	8	6
	p			i		j		r
X	4	5	0	2	7	9	8	6
	p			i			j	r
X	4	5	0	2	7	9	8	6
	p			i			j	r
X	4	5	0	2	6	9	8	7

i+1 = 4

Quick Sort Example

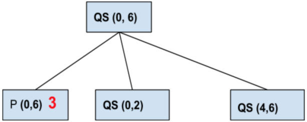
5	7	6	1	3	2	4



i	p j						r
X	5	7	6	1	3	2	4
i	p	j					r
X	5	7	6	1	3	2	4
i	p		j				r
X	5	7	6	1	3	2	4
i	p			j			r
X	5	7	6	1	3	2	4
	i p			j			r
X	5	7	6	1	3	2	4
	i p			j			r
X	1	7	6	5	3	2	4
	i p				j		r
X	1	7	6	5	3	2	4

Quick Sort Example

5	7	6	1	3	2	4



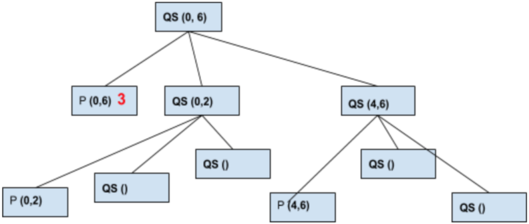
Level 1 complete

	p	i			j		r
X	1	7	6	5	3	2	4
	p	i			j		r
X	1	3	6	5	7	2	4
	p	i				j	r
X	1	3	6	5	7	2	4
	p		i			j	r
X	1	3	6	5	7	2	4
	p		i			j	r
X	1	3	2	5	7	6	4
	p		i			j	r
X	1	3	2	4	7	6	5

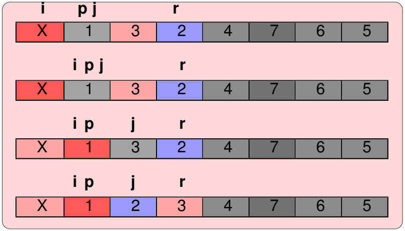
i+1 = 3

Quick Sort Example

1	3	2	4	7	6	5



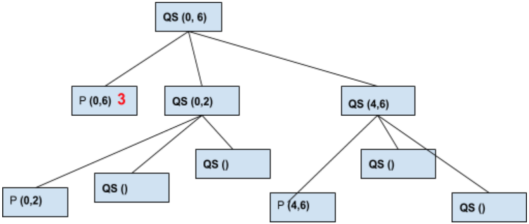
Find the partition P(0,2) & P(4,6)?



i	p j		r				
X	1	3	2	4	7	6	5
	i p j		r				
X	1	3	2	4	7	6	5
	i p	j	r				
X	1	3	2	4	7	6	5
	i p	j	r				
X	1	2	3	4	7	6	5

Quick Sort Example

1	2	3	4	7	6	5



Find the partition P(0,2) & P(4,6)?

				i	p j		r
X	1	2	3	4	7	6	5

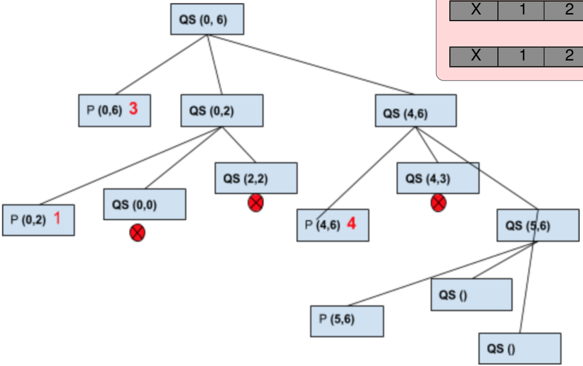
				i	p	j	r
X	1	2	3	4	7	6	5

				i	p	j	r
X	1	2	3	4	5	6	7

Quick Sort Example

1	2	3	4	5	6	7

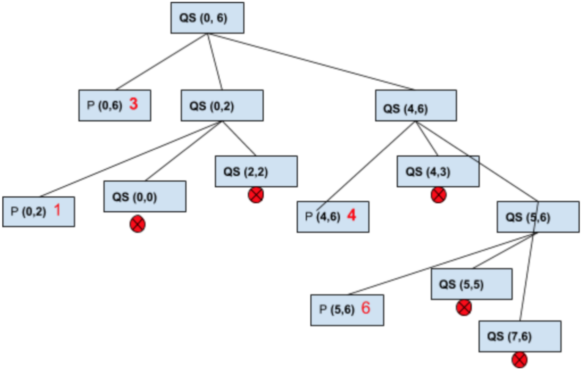
					i	p j	r
X	1	2	3	4	5	6	7
						i p j	r
X	1	2	3	4	5	6	7
						i p j	r
X	1	2	3	4	5	6	7



Find the partition P(5,6)?

Quick Sort Example

1	2	3	4	5	6	7



Level 3 complete

Quick Sort Algorithm - An analysis

Running Time:

- **Worst case:** $O(n^2)$
- **Best case:** $O(n \times log(n))$
- **Average case:** $O(n \times log(n))$

Quick Sort Split

- 1:9 split $O(n \times log(n))$
- 1:99 split $O(n \times log(n))$
- 1:999 split $O(n \times log(n))$
- 0:n split $O(n^2)$

Quick Sort Example (Analyze)

1	2	3	4	5	6	7

- Running time : O(n^2)

Quick Sort Example (Analyze)

7	6	5	4	3	2	1

- Running time : O(n^2)

Quick Sort Finishing Up

One question to think of is can we do a better job in selecting the pivot element? **Random position for [pivot] better split? - In Lab**

Reading Assignment

Sedgewick 2.3 Quick Sort

Questions?

Please ask if there are any Questions!