QuickSort

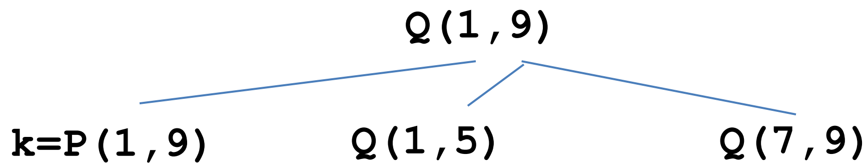
QuickSort

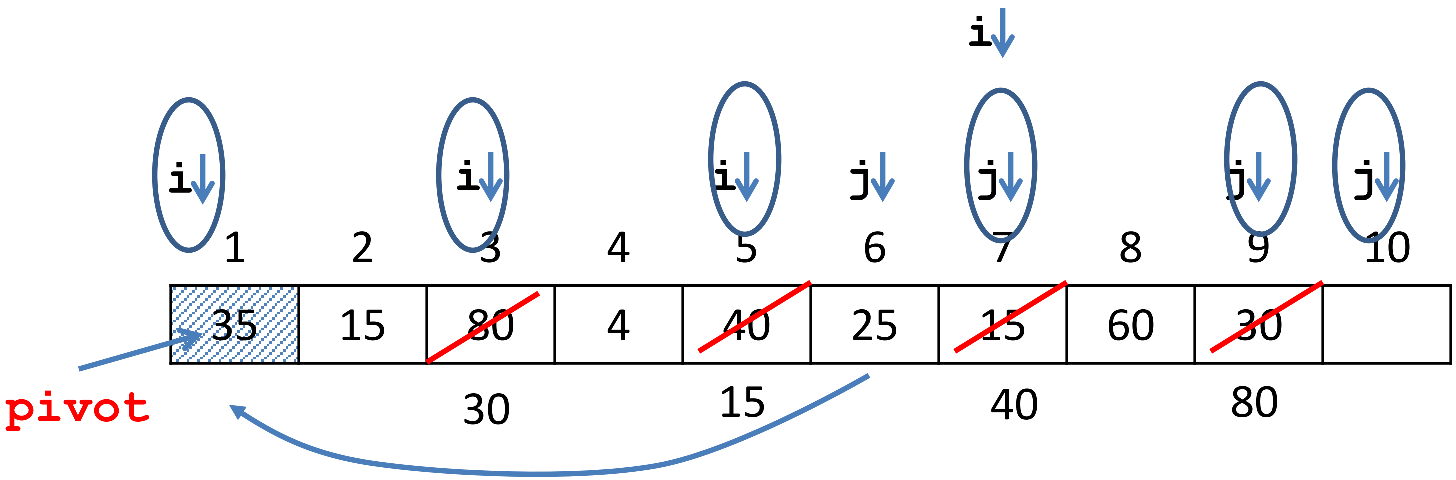
- Quicksort is a divide-and-conquer sorting algorithm.
- Partitioning is *static* in the case of Merge Sort algorithm. The list is divided into two near halves.
- In Quicksort, partitioning is dynamic and the sizes of the two lists are not be the same. It depends on the elements present in the list.

QuickSort

```
Algorithm QuickSort(a,p,q)
{
    if(p<q) then
    {
        j:=Partition(a,p,q);
        QuickSort(a,p,j-1);
        QuickSort(a,j+1,q);
    }
}
```

QuickSort(1,9)





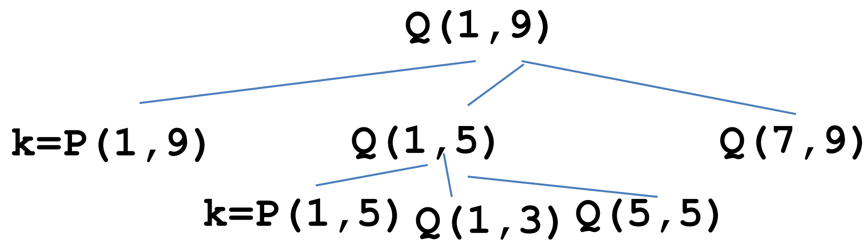
1	2	3	4	5	6	7	8	9	10
25	15	30	4	15	35	40	60	80	

```
repeat
{
    i:i+1;
}untl(a[i]>=v)
```

```
repeat
{
    j:j-1;
}untl(a[j]<=v)
```

```
if (i<j) then
     Swap(a,i,j);
```

QuickSort(1,5)



i		i	j	i j	j				
1	2	3	4	5	6	7	8	9	10
25	15	~~30~~	4	~~15~~	35	40	60	80	
		15		30					

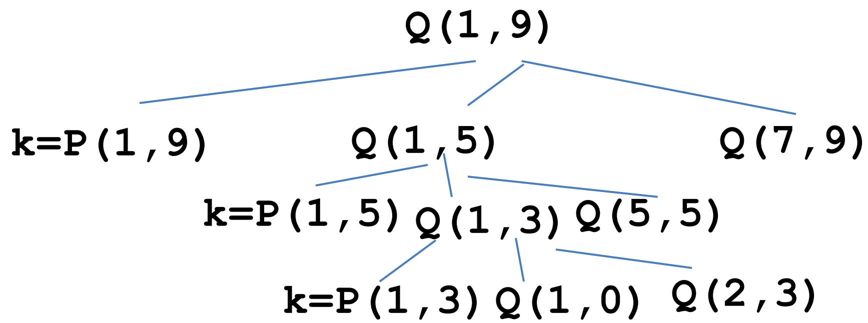
1	2	3	4	5	6	7	8	9	10
4	15	15	25	30	35	40	60	80	

```
repeat
{
   i:i+1;
}untl(a[i]>=v)
```

```
repeat
{
   j:j-1;
}untl(a[j]<=v)
```

```
if (i<j) then
    Swap(a,i,j);
```

QuickSort(1,3)



j

i i j

1	2	3	4	5	6	7	8	9	10
4	15	15	25	30	35	40	60	80	

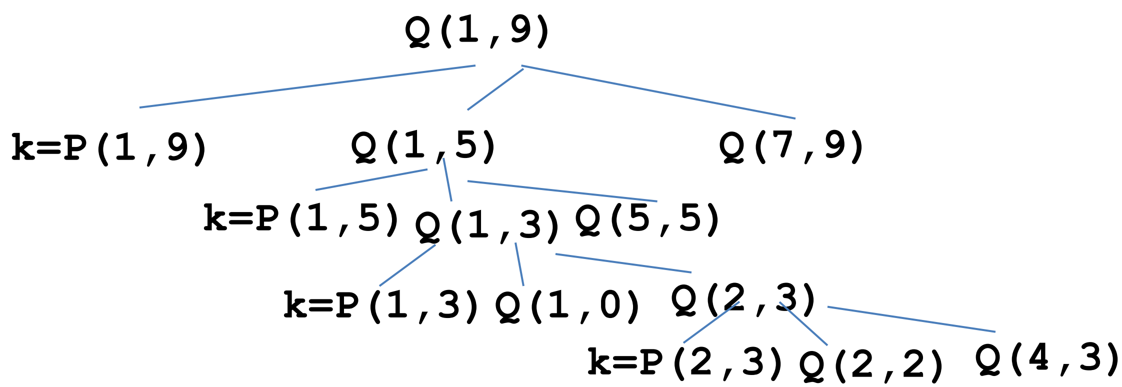
1	2	3	4	5	6	7	8	9	10
4	15	15	25	30	35	40	60	80	

```
repeat
{
    i:i+1;
}untl(a[i]>=v)
```

```
repeat
{
    j:j-1;
}untl(a[j]<=v)
```

```
if (i<j) then
    Swap(a,i,j);
```

QuickSort(2,3)



j

i i j

1	2	3	4	5	6	7	8	9	10
4	15	15	25	30	35	40	60	80	

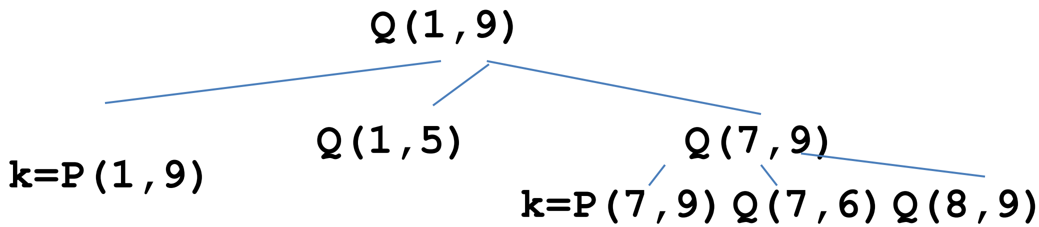
1	2	3	4	5	6	7	8	9	10
4	15	15	25	30	35	40	60	80	

```
repeat
{
    i:i+1;
}untl(a[i]>=v)
```

```
repeat
{
    j:j-1;
}untl(a[j]<=v)
```

```
if (i<j) then
    Swap(a,i,j);
```

QuickSort(7,9)



j i i j

1	2	3	4	5	6	7	8	9	10
4	15	15	25	30	35	40	60	80	

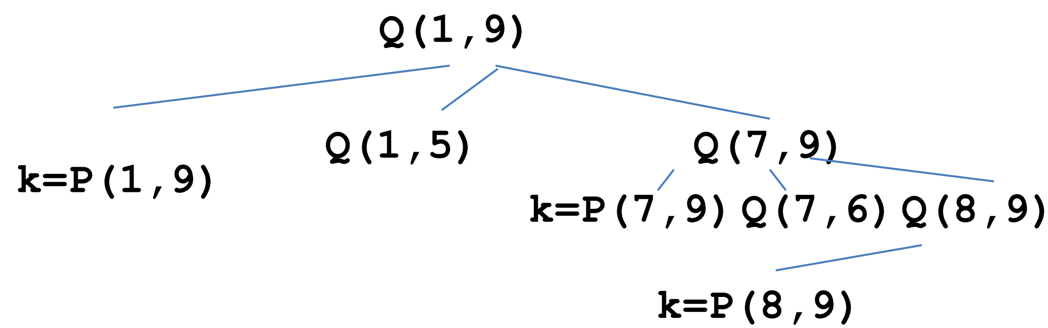
1	2	3	4	5	6	7	8	9	10
4	15	15	25	30	35	40	60	80	

```
repeat
{
    i:i+1;
}untl(a[i]>=v)
```

```
repeat
{
    j:j-1;
}untl(a[j]<=v)
```

```
if (i<j) then
    Swap(a,i,j);
```

QuickSort(8,9)



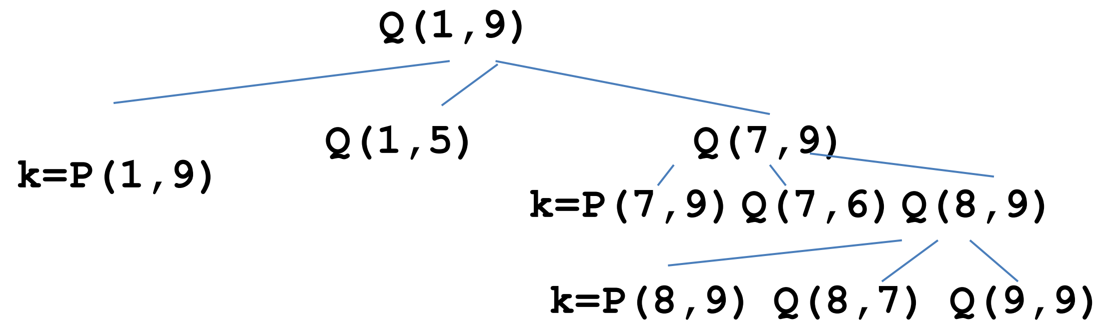
							i		j
1	2	3	4	5	6	7	8	9	10
4	15	15	25	30	35	40	60	80	

```
repeat
{
   i:i+1;
}untl(a[i]>=v)
```

```
repeat
{
   j:j-1;
}untl(a[j]<=v)
```

```
if (i<j) then
    Swap(a,i,j);
```

```
QuickSort(8,9)
```



							j	i	
1	2	3	4	5	6	7	8	9	10
4	15	15	25	30	35	40	60	80	

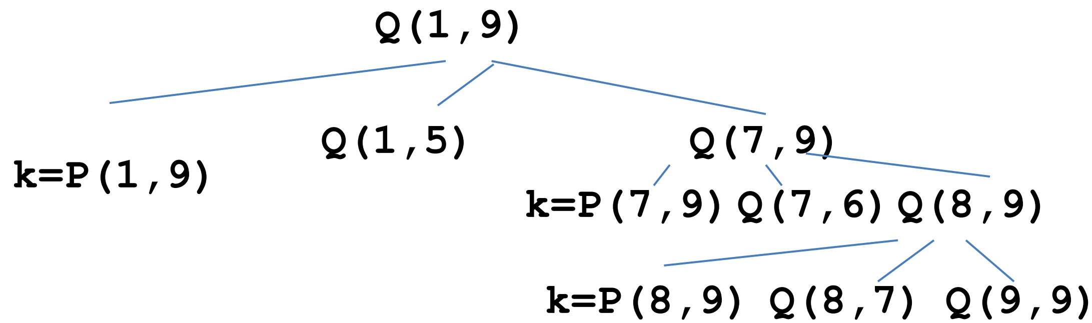
1	2	3	4	5	6	7	8	9	10
4	15	15	25	30	35	40	60	80	

```
repeat
{
    i:i+1;
}untl(a[i]>=v)
```

```
repeat
{
    j:j-1;
}untl(a[j]<=v)
```

```
if (i<j) then
    Swap(a,i,j);
```

QuickSort(8,9)

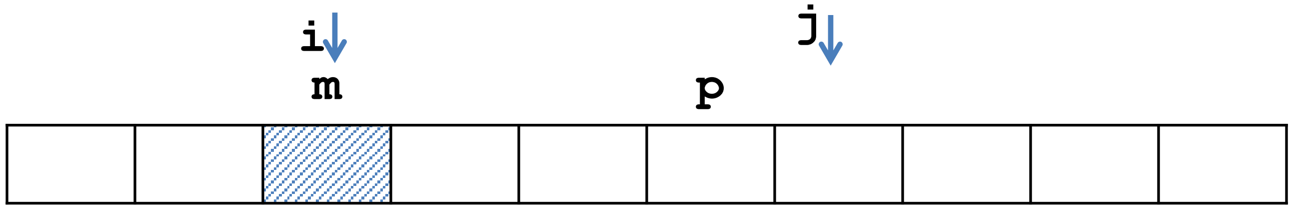


1	2	3	4	5	6	7	8	9	10
4	15	15	25	30	35	40	60	80	

```
repeat
{
    i:i+1;
}untl(a[i]>=v)
```

```
repeat
{
    j:j-1;
}untl(a[j]<=v)
```

```
if (i<j) then
    Swap(a,i,j);
```




```
Algorithm Partition(a,m,p)
{
    v:=a[m];i=m;j=p+1;
    repeat
    {
        repeat
        {
            i:i+1;
        }until(a[i]>=v);//stop at element >= pivot
        repeat
        {
            j:=j-1;
        }until(a[i]<=v);//stop at element <= pivot
        if (i<j) then swap(a,i,j); //swap if not crossed
    }until(i>=j)                   //i and j crossed over
    swap(a,m,j); //place the pivot in its position
    return j;
}
```

```
Algorithm QuickSort(a,p,q)
{
    if(p<q) then
    {
        j:=Partition(a,p,q);
        QuickSort(a,p,j-1);
        QuickSort(a,j+1,q);
    }
}
```

QuickSort-Analysis

- We find that in partitioning the index variables $i\ and\ j$ move by $n+1$ times together. Hence, the complexity of partitioning is in the order of $n+1$.
- The complexity of QuickSort can be formulated as:
- $T(n) = (n+1) + T(|L|) + T(|R|)$
- $|L|$ and $|R|$ may taken any value from $0 \ldots n-1$

QuickSort-Analysis

- $T(n) = (n + 1) + T(|L|) + T(|R|)$
- $|L|$ and $|R|$ may taken any value from $0 \dots n - 1$.
- The best case is $|L| = |R| = \frac{\mathrm{n}-1}{2}$
- Hence, $T(n) = (n + 1) + 2T((n - 1)/2))$
- The worst case is

QuickSort-Analysis

- $T(n) = (n + 1) + T(|L|) + T(|R|)$
- The worst case is $|L| = 0$ or $|R| = 0$
- Hence, $T(n) = (n + 1) + T(n - 1)$

QuickSort-Analysis

- Average case:Two sublists are formed with $k-1$ elements and $n-k$ elements, as k varies from *1 to n*

k	$\|L\|$	$\|R\|$
1	*0*	$n-1$
2	*1*	$n-2$
3	*2*	$n-3$
…	…	
n-1	*n-2*	1
n	*n-1*	0

$Average\ complexity\ is\ \frac{2}{n}(T(0)+T(1)+\cdots+T(n-1))$

QuickSort-Analysis

- *Best case analysis*
- The pivot element will divide the list into two halves.
- Two sublists are formed with $(\mathrm{n}-1)/2$ elements each.
- $T(n) = 2T((n-1)/2) + (n+1)$
- *This can be modeled as*

QuickSort-Analysis

- $T(n) = 2T\left(\frac{n}{2}\right) + \theta(n)$
- Hence, $T(n) = \theta(n \log n)$

QuickSort-Analysis

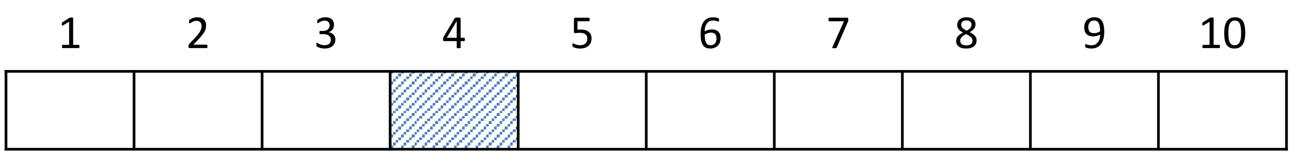
- *Worst case analysis*
- The pivot element will divide the list into one sublist of $n - 1\ elements$.
- $T(n) = T(n-1) + (n+1)$
- $T(n-1) = T(n-2) + (n)$
- $T(n-2) = T(n-3) + (n-1)$
- $\dots$
- $T(1) = T(0) + (1)$

QuickSort-Analysis

- $T(n) = \cancel{T(n-1)} + (n+1)$
- $\cancel{T(n-1)} = \cancel{T(n-2)} + (n)$
- $\cancel{T(n-2)} = \cancel{T(n-3)} + (n-1)$
- $\dots$
- $\cancel{T(1)} = T(0) + (1)$
- $T(n) = T(0) + 1 + 2 + \cdots + n + (n+1)$
- $= \frac{(n+1)(n+2)}{2} = \theta(n^2)$

QuickSort-Analysis

- *Average case analysis*
- The pivot element can be ith smallest element, $1 \leq k \leq n$



- Two sublists are formed with $k - 1$ elements and $n - k$ elements, as *k* varies from *1 to n*

QuickSort-Analysis

- Two sublists are formed with $k-1$ elements and $n-k$ elements, as *k* varies from *1 to n*

k	$T(k-1)$	$T(n-k)$
1	$T(0)$	$T(n-1)$
2	$T(1)$	$T(n-2)$
3	$T(2)$	$T(n-3)$
…		
n-2	$T(n-3)$	$T(2)$
n-1	$T(n-2)$	$T(1)$
n	$T(n-1)$	$T(0)$

QuickSort-Analysis

- $T(0) = T(1) = 0$
- $T(n)\ for\ average\ case\ is$
- $T(n) = n + 1 + \frac{1}{n}\sum_{1 \le k \le n}(T(k-1) + T(n-k))$
- After expansion
- $T(n) = (n+1) + \frac{2}{n}(T(0) + T(1) + \cdots + T(n-1))$

QuickSort-Analysis

- $T(n) = (n+1) + \frac{2}{n}(T(0) + T(1) + \cdots + T(n-1))$
- $Multiply\ by\ n$
- $nT(n) = n(n+1) + 2(T(0) + T(1) + \cdots + T(n-1))$
- $Substitute\ n-1\ for\ n$
- $(n-1)T(n-1) = n(n-1) + 2(T(0) + T(1) + \cdots + T(n-2))$
- $Subtract$
- $nT(n) - (n-1)T(n-1) = 2n + 2T(n-1)$

QuickSort-Analysis

- $nT(n) = 2n + 2T(n-1) + (n-1)T(n-1)$
- $nT(n) = 2n + \ (n+1)T(n-1)$
- $\frac{T(n)}{n+1} = \frac{T(n-1)}{n} + \frac{2}{n+1}$

$$= \frac{T(n-2)}{n-1} + \frac{2}{n} + \frac{2}{n+1}$$

$$= \frac{T(n-3)}{n-2} + \frac{2}{n-1} + \frac{2}{n} + \frac{2}{n+1}$$

QuickSort-Analysis

- $\frac{T(n)}{n+1} = \frac{T(n-3)}{n-2} + \frac{2}{n-1} + \frac{2}{n} + \frac{2}{n+1}$
- $\ldots$
- $= \frac{T(1)}{2} + \frac{2}{3} + \frac{2}{4} + \cdots + \frac{2}{n} + \frac{2}{n+1}$
- $= 2\sum_{3\le k\le n+1} \frac{1}{k}$
- $\le 2\int_2^{n+1} \frac{1}{k}\,dk \le 2\log(n+1) - 2\log 2 \le 2\log(n+1)$
- $T(n) \le 2(n+1)\log(n+1) = O(n\log n)$