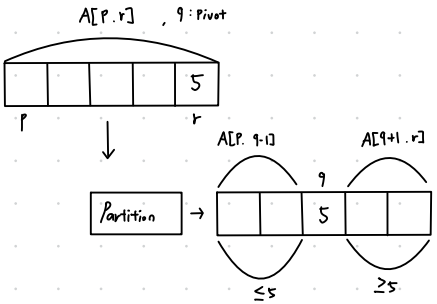
Quicksort

- Key method: Partition



Algorithm

```
QuickSort(A,p,r)
  if p<r then
    q:= Partition(A,p,r)
    Quicksort(A,p,q-1)
    Quicksort(A,q+1,r)
  fi
```

```
Partition(A,p,r)
  x,i := A[r], p-1
  for j := p to r-1 do
    if A[j] ≤ x then
      i := i+1
      A[i] ↔ A[j]
    fi
  od j
  A[i+1] ↔ A[r]
  return i+1
```

Example (Sorting elements) ✱Check A[j] ≤ Pivot

Initially: p r

2	5	8	3	9	4	1	7	10	6
i	j								

Note: Pivot(x) = 6 : A[r]

2	5	8	3	9	4	1	7	10	6
	i	j							

2	5	8	3	9	4	1	7	10	6
		i	j						

2	5	8	3	9	4	1	7	10	6
		i		j					

2	5	3	8	9	4	1	7	10	6
			i		j				

2	5	3	8	9	4	1	7	10	6
			i			j			

2	5	3	4	9	8	1	7	10	6
				i			j		

2	5	3	4	1	8	9	7	10	6
					i			j	

2	5	3	4	1	8	9	7	10	6
					i				j

2	5	3	4	1	8	9	7	10	6
					i				j

2	5	3	4	1	6	9	7	10	8
					i				j

Time Complexity

- Worst Case : Always get completely unbalanced partition

$$T(n) = T(n-1) + \Theta(n)$$

$$= \sum_{k=1}^{n} \Theta(k)$$

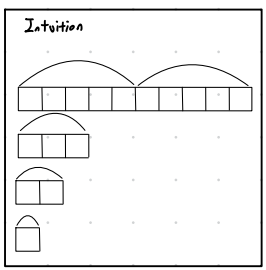
$$= \Theta\left(\sum_{k=1}^{n} k\right) = \Theta(n^2)$$

★ Worst case happens on sorted list

- Best case : Always get perfectly balanced partition

$$T(n) \leq 2\,T\left(\frac{n}{2}\right) + \Theta(n) = \Theta(n \cdot \lg n)$$

Why log n? : $n \to \frac{n}{2} \to \frac{n}{2^2} \to \cdots \to 1$ (roughly log n steps)



Randomized Version

: Solves worst case of original Quicksort by choosing random pivot value, instead of choosing P[r]

Algorithm

```
Randomized-Partition(A,p,r)
    i := Random(p,r)
    A[r] ↔ A[i]
    Partition(A,p,r)

Randomized-Quicksort(A,p,r)
    if p<r then
        q := Randomized-Partition(A,p,r)
        Randomized-Quicksort(A,p,q-1)
        Randomized-Quicksort(A,q+1,r)
    fi
```

Time Complexity : $\Theta(n \log n)$