		DATE
0		PAGE No.
(1)	Algorithm in Place Partition (arr, s.e)	012343
		1P = [1,612,4,3,5]
	phot Layr tes	[5,6,2,4,3,1],0,5
	Pos & s	[116,2,413,5]
Sel sel	, for 1 = 5 → e-1 do	
	if arr [i] < pivot then	4
	if (il==Pos) then	1 1/26, 20 43,5
	temp Earr [1]	[155, 2, 4, 3, 6], 1,5
	OTY CPOSIC femp	
	Pas +4	6. 29
5, 9	temp Garr [POS]	[1,5,2,4,3,6] (>
	arrepose pivot	113,2,4,5,6]1,4
	ary [e] = temp	1/ = 3
	return Pos	[] [1930294.596] [] [1942239596] 193
		[19294939596]
	input = 61/15/12/1/3/13	[1,2,3,4,5,6]
	KET MILLSON TO SELECT	
	[1,6,2,1,3,5]	
		576
	and the second	base with condition.
	Tyloly18/81	base condition condition.

	一方: 10 1 5 1 5 1 5 1 5 1 5 1 5 1 5 1 5 1 5
2.	Enter Committee The Treatment of the Committee of the Com
(a)	
0	[5,1,4,3,6,2,7,1,3] (n=9)
	Andread Rivers
	# O elements = 3×9 × 7
	The sements of the se
	Part = 6 Pinch = 1 1
	75=2 / 1171=7 (1)
	25=6 21=0,3
	Rvot=4 7 Rvot=3 7 Rvot=6 7 Pivot=2 7 74=3 (V >3=4 4V >6=1 (X 78=6 6V
	(4:5) (3:3) (6:7) (3:3)
	19-5
	D 4 2 Dini - 1 2 2 2
	Pivot = 7 Pivot = 1 Pivot = 3 7
	57 E
	27=8] 21=0] (3=3)
	total Pivots = 9
	Pirots with right and left elements & 7 = 5
	Good Pivot = 55 / 519 7 12
6	You, At least half of a are good pivots.
27	

2000		
(3)		
	Good case is when pivot divides the array into reasonably balanced arrays halves.	
	Best case is when pivot divides the array into two equal halves.	
	Good case is not the base case in Quicksort	
	Good case is not the base case in Quicksort but they do have similar Time Complexity.	
	lets take an Array with nelements and Suppose to the pivot divides the Array into equal halves in every levels.	
	n work = partitiona = n	
	$\frac{1}{12} \frac{1}{12} \frac$	
	1 1 mm work = 10 + 10 + 10 + 10 + 10 + 10 + 10 + 10	
	[Nu] [Nu] [Nu] Work = "4 + "4 + "4 + "4 + "4 + "4 + "4 + "4	
	- 151F13	
	Time Complexity = # Offevels * work	
	Time Complexity = # Oflevels * work Ulogn *Uh)	
	$= O(n\log_2 n)$	
	01/9	

```
class Solution {
  public static int quickSortedIndex(int[] arr, int s, int e){
    int pivot = arr[e];
    int pos = s;
    for(int i=s;i<e;i++){
      if(arr[i] < pivot){</pre>
        if(pos!=i){}
          int temp = arr[pos];
          arr[pos] = arr[i];
          arr[i] = temp;
        pos++;
    arr[e] = arr[pos];
    arr[pos] = pivot;
    return pos;
 }
  public int findKthLargest(int[] nums, int k) {
    int position = nums.length - k;
    int start = 0;
    int end = nums.length - 1;
    while(start < end){ // element in position or index is the of the correct order
      int cursor = quickSortedIndex(nums,start,end);
      if(cursor==position){
        break;
      if(position < cursor){</pre>
        end = cursor - 1;
      }else{
        start = cursor + 1;
      }
    return nums[position];
```