The left array consists of a single element, hence it is considered sorted

Take right array since it doesn't contain a single element

[9, 7] [pivot - 9]

Now send all small elements to right side

→ [7, 9]

update in the array

5, 6, 7, 9.

↳ Now update this in the main array.

1, 2, 3, 4, 5, 6, 7, 9

4, 6, 2, 5, 7, 9, 1, 3

low pointer ↑ high pointer ↓

pivot = low

Epiphany

Appointments

8 ~~Pseudocode~~

9 ~~quicksort(~~

10 To place the pivot in ~~ints~~ its correct place.

11 ⓘ ↑ ⓙ
4, 6, 2, 5, 7, 9, 7, 9, 1, 3
↓ ↓
low high

12

13 Take i & place @ leftmost part of arr

14 Take j & place @ rightmost part of arr.

15 Now start →
is 4 > pivot - (-ve) (false).

16 is 6 greater than pivot, pivot<6 - +ve

17 ⓘ ⓙ
4, 6, 2, 5, 7, 9, 1, 3

18 i pointer stops @ 6
Now find the smaller element than pivot from right, using the j pointer

19 (pivot > j)

20 Now we swap 6 & 3
4, 3, 2, 5, 7, 9, 1, 6

February

M	T	W	T	F	S	S	M	T	W	T	F	S	S
							1	2	3	4	5	6	7	8
9	10	11	12	13	14	15	16	17	18	19	20	21	22
23	24	25	26	27	28								

January
2015

007-358

7 Wednesday

wk - 2

Appointments

8

9

10

11

12

13

14

15

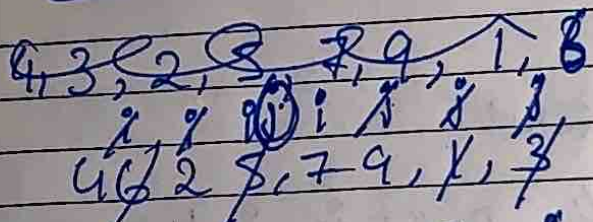
16

17

18

19

20



since j crosses i, the function stops

January

M	T	W	T	F	S	S	M	T	W	T	F	S	S
			1	2	3	4	5	6	7	8	9	10	11
12	13	14	15	16	17	18	19	20	21	22	23	24	25
26	27	28	29	30	31								

8 January 2015

Thursday 008-357 wk - 2

Appointments

Pseudocode

```
qs(arr, low, high)
{   if(low >= high)
        return;

    pivot = arr[low]

partition = func(arr, low, high)
```

finds the pivot, places @ it correct place & tells where it has placed

```
qs(arr, low, partition-1);
qs(arr, partition+1, high)
}
```

```
int func(arr, low, high)
{ pivot = arr[low];
  i = low;
  j = high;
  while (i<j) {
  while (arr[i] <= arr[pivot] && i <= high)
  {i++;}
  while (arr[j] > pivot && j >= low)
  {j--;}
  if(i<j)
  {swap (arr[i], arr[j]);
```

February

M	T	W	T	F	S	S
					1	2
3	4	5	6	7	8	9
10	11	12	13	14	15	16
17	18	19	20	21	22	23
24	25	26	27	28		