Quicksort partitioning

```
private static int partition(Comparable[] a, int lo, int hi)
{  // Partition into a[lo..i-1], a[i], a[i+1..hi].
   int i = lo, j = hi+1;            // left and right scan indices
   Comparable v = a[lo];            // partitioning item
   while (true)
   {  // Scan right, scan left, check for scan complete, and exchange.
      while (less(a[++i], v)) if (i == hi) break;
      while (less(v, a[--j])) if (j == lo) break;
      if (i >= j) break;
      exch(a, i, j);
   }
   exch(a, lo, j);       // Put v = a[j] into position
   return j;             // with a[lo..j-1] <= a[j] <= a[j+1..hi].
}
```

This code partitions on the item `v` in `a[lo]`. The main loop exits when the scan indices `i` and `j` cross. Within the loop, we increment `i` while `a[i]` is less than `v` and decrement `j` while `a[j]` is greater than `v`, then do an exchange to maintain the invariant property that no entries to the left of `i` are greater than `v` and no entries to the right of `j` are smaller than `v`. Once the indices meet, we complete the partitioning by exchanging `a[lo]` with `a[j]` (thus leaving the partitioning value in `a[j]`).

	i	j	v	a[] 0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
initial values	0	16		K	R	A	T	E	L	E	P	U	I	M	Q	C	X	O	S
scan left, scan right	1	12		K	R	A	T	E	L	E	P	U	I	M	Q	C	X	O	S
exchange	1	12		K	C	A	T	E	L	E	P	U	I	M	Q	R	X	O	S
scan left, scan right	3	9		K	C	A	T	E	L	E	P	U	I	M	Q	R	X	O	S
exchange	3	9		K	C	A	I	E	L	E	P	U	T	M	Q	R	X	O	S
scan left, scan right	5	6		K	C	A	I	E	L	E	P	U	T	M	Q	R	X	O	S
exchange	5	6		K	C	A	I	E	E	L	P	U	T	M	Q	R	X	O	S
scan left, scan right	6	5		K	C	A	I	E	E	L	P	U	T	M	Q	R	X	O	S
final exchange	6	5		E	C	A	I	E	K	L	P	U	T	M	Q	R	X	O	S
result		5		E	C	A	I	E	K	L	P	U	T	M	Q	R	X	O	S

Partitioning trace (array contents before and after each exchange)