

CS062

DATA STRUCTURES AND ADVANCED PROGRAMMING

15: Quicksort

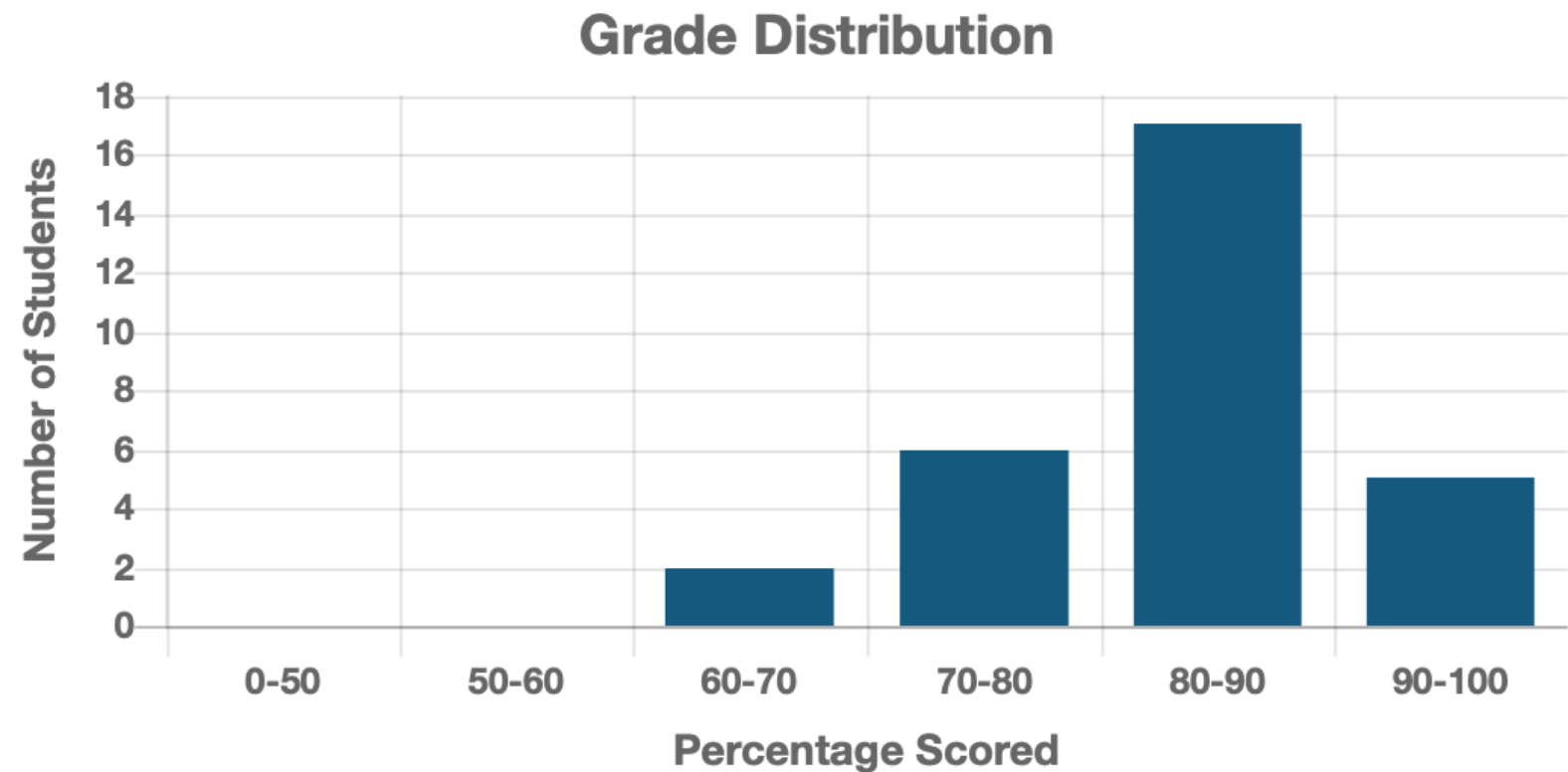


Tom Yeh
he/him/his

Lecture 14: Quicksort

- ▶ Midterm
- ▶ Quiz 4
- ▶ Quicksort

Midterm 1 Stats

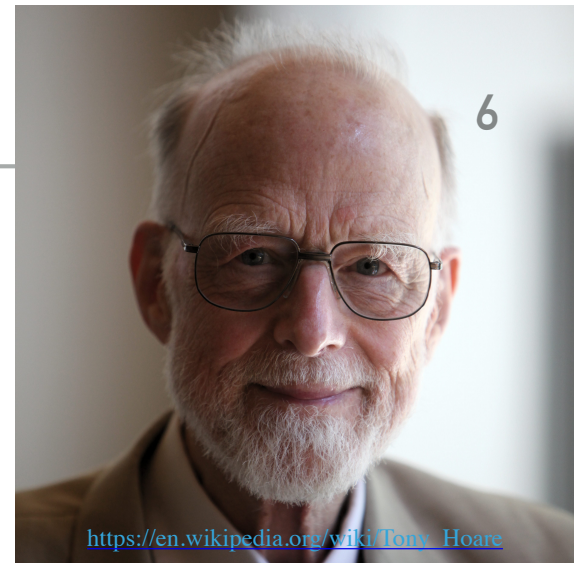


Average (mean) grade	74.37
Median grade	76.50
Standard deviation	7.01
Lowest grade	58.00
Highest grade	86.00
Total graded	30

Midterm Solutions!

Lecture 15: Quicksort

- ▶ Quiz 4
- ▶ Visualize sorting - <https://visualgo.net/en>
 - ▶ Selection + Insertion sort the same
 - ▶ Merge sort and quick sort slightly different
- ▶ Quicksort
- ▶



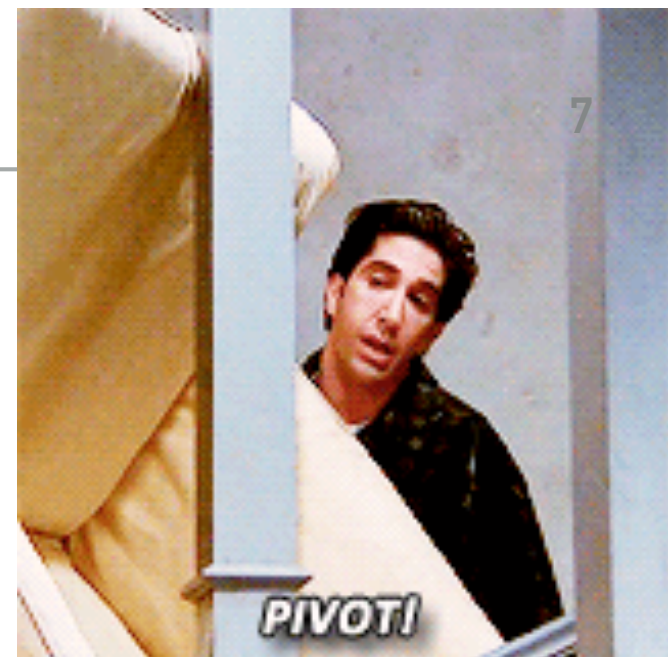
Basics

- ▶ Divide-and-conquer method
- ▶ Invented by [Sir Tony Hoare](#) in 1959.
 - ▶ Wanted to sort Russian words before looking them up in dictionary.
 - ▶ Came up with quicksort but did not know how to implement it.
 - ▶ Learned Algol 60 and recursion and implemented it.
 - ▶ Won the 1980 Turing Award (also invented null - and regretted it).
- ▶ [Bob Sedgewick](#) (author of our textbook) refined and analyzed many versions of quicksort.

QUICKSORT

Algorithm sketch

- ▶ **Shuffle** the array.
- ▶ **Partition** so that, for some pivot j :
 - ▶ Entry $a[j]$ is in place.
 - ▶ There is no larger entry to the left of j .
 - ▶ No smaller entry to the right of j .
- ▶ **Sort** each subarray recursively.



input	Q	U	I	C	K	S	O	R	T	E	X	A	M	P	L	E
shuffle	K	R	A	T	E	L	E	P	U	I	M	Q	C	X	O	S
partition	E	C	A	I	E	K	L	P	U	T	M	Q	R	X	O	S
sort left	A	C	E	E	I	K	L	P	U	T	M	Q	R	X	O	S
sort right	A	C	E	E	I	K	L	M	O	P	Q	R	S	T	U	X
result	A	C	E	E	I	K	L	M	O	P	Q	R	S	T	U	X

Quicksort overview

Quicksort Code

```
// quicksort the subarray from a[lo] to a[hi]
private static void sort(Comparable[] a, int lo, int hi) {
    if (hi <= lo) return;
    int j = partition(a, lo, hi);
    sort(a, lo, j-1);
    sort(a, j+1, hi);
}

/**
 * Rearranges the array in ascending order, using the natural order.
 * @param a the array to be sorted
 */
public static void sort(Comparable[] a) {
    StdRandom.shuffle(a);
    sort(a, 0, a.length - 1);
}
```

Hoare's partition scheme (faster than Lomuto's you will see in CS140)

- ▶ Partition the subarray $a[lo..hi]$ so that $a[lo..j-1] \leq a[j] \leq a[j+1..hi]$
- ▶ Start with pivot at lo , pointer i at lo and pointer j at $hi+1$.
- ▶ Repeat the following until pointers i and j cross:
 - ▶ Scan i from left to right as long as $a[lo] > a[i]$.
 - ▶ (i.e. stop when you find a key that is greater than (or equal to) the pivot)
 - ▶ Scan j from right to left as long as $a[lo] < a[j]$.
 - ▶ (i.e. stop when you find a key that is less than (or equal to) the pivot)
 - ▶ Exchange $a[i]$ with $a[j]$.
- ▶ Exchange $a[lo]$ and $a[j]$. Return j .

Partition Code

```
// partition the subarray a[lo..hi] so that a[lo..j-1] <= a[j] <= a[j+1..hi] and return the index j.
private static int partition(Comparable[] a, int lo, int hi) {
    int i = lo;
    int j = hi + 1;
    Comparable v = a[lo];
    while (true) {

        // find item greater than (or equal to) v to swap
        while (less(a[++i], v)) {
            if (i == hi) break;
        }

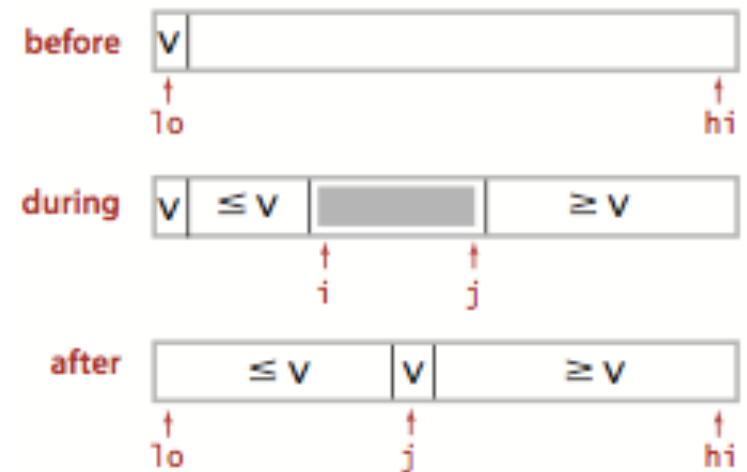
        // find item smaller than (or equal to) v to swap
        while (less(v, a[--j])) {
            if (j == lo) break; // redundant since a[lo] acts as sentinel
        }

        // check if pointers cross
        if (i >= j) break;

        exch(a, i, j);
    }

    // put partitioning item v at a[j]
    exch(a, lo, j);

    // now, a[lo .. j-1] <= a[j] <= a[j+1 .. hi]
    return j;
}
```



Quicksort partitioning overview

Quicksort Example - Sort [Q,U,I,C,K,S,O,R,T,E,X,A,M,P,L,E]

```
public static void sort(Comparable[] a) {  
    StdRandom.shuffle(a);  
    sort(a, 0, a.length - 1);  
}
```

```
// quicksort the subarray from a[lo] to a[hi]  
private static void sort(Comparable[] a, int lo, int hi) {  
    if (hi <= lo) return;  
    int j = partition(a, lo, hi);  
    sort(a, lo, j-1);  
    sort(a, j+1, hi);  
}
```

Shuffling resulted to:

K,R,A,T,E,L,E,P,U,I,M,Q,C,X,O,S

Call partition with lo=0, hi=15

K R A T E L E P U I M Q C X O S

0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
---	---	---	---	---	---	---	---	---	---	----	----	----	----	----	----

lo

hi

Partition Example, lo=0, hi=15

K	R	A	T	E	L	E	P	U	I	M	Q	C	X	O	S
0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
lo														hi	
i															j

```
// partition the subarray a[lo..hi] so that a[lo..j-1] <= a[j] <= a[j+1..hi] and return the index j.  
private static int partition(Comparable[] a, int lo, int hi) {  
    int i = lo;  
    int j = hi + 1;  
    Comparable v = a[lo];
```

Partition Example, lo=0, hi=15

K	R	A	T	E	L	E	P	U	I	M	Q	C	X	O	S
0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
lo															hi
i															j

```
// find greater than (or equal to) v to swap
while (less(a[++i], v)) {
    if (i == hi) break;
}
```

Partition Example, lo=0, hi=15

K	R	A	T	E	L	E	P	U	I	M	Q	C	X	O	S
0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
lo															hi
i															j

```
// find item smaller than (or equal to) v to swap
while (less(v, a[--j])) {
    if (j == lo) break; // redundant since a[lo] acts as sentinel
}
```

Partition Example, lo=0, hi=15

K	R	A	T	E	L	E	P	U	I	M	Q	C	X	O	S
0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
lo															hi
i														j	

```
// find item smaller than (or equal to) v to swap
while (less(v, a[--j])) {
    if (j == lo) break; // redundant since a[lo] acts as sentinel
}
```

Partition Example, lo=0, hi=15

K	R	A	T	E	L	E	P	U	I	M	Q	C	X	O	S
0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
lo															hi
i													j		

```
// find item smaller than (or equal to) v to swap
while (less(v, a[--j])) {
    if (j == lo) break; // redundant since a[lo] acts as sentinel
}
```

Partition Example, lo=0, hi=15

K	R	A	T	E	L	E	P	U	I	M	Q	C	X	O	S
0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
lo															hi
i												j			

```
// find item smaller than (or equal to) v to swap
while (less(v, a[--j])) {
    if (j == lo) break; // redundant since a[lo] acts as sentinel
}
```

Partition Example, lo=0, hi=15

K	R	A	T	E	L	E	P	U	I	M	Q	C	X	O	S
0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
lo															hi
i												j			

```
exch(a, i, j);
```

Partition Example, lo=0, hi=15

K	C	A	T	E	L	E	P	U	I	M	Q	R	X	O	S
0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
lo															hi
i												j			

```
exch(a, i, j);
```

Partition Example, lo=0, hi=15

K	C	A	T	E	L	E	P	U	I	M	Q	R	X	O	S
0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
lo															hi
	i										j				

```
// find greater than (or equal to) v to swap
while (less(a[++i], v)) {
    if (i == hi) break;
}
```

Partition Example, lo=0, hi=15

K	C	A	T	E	L	E	P	U	I	M	Q	R	X	O	S
0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
lo															hi
		i									j				

```
// find greater than (or equal to) v to swap
while (less(a[++i], v)) {
    if (i == hi) break;
}
```

Partition Example, lo=0, hi=15

K	C	A	T	E	L	E	P	U	I	M	Q	R	X	O	S
0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
lo															hi
		i								j					

```
// find item smaller than (or equal to) v to swap
while (less(v, a[--j])) {
    if (j == lo) break; // redundant since a[lo] acts as sentinel
}
```

Partition Example, lo=0, hi=15

K	C	A	T	E	L	E	P	U	I	M	Q	R	X	O	S
0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
lo															hi
		i						j							

```
// find item smaller than (or equal to) v to swap
while (less(v, a[--j])) {
    if (j == lo) break; // redundant since a[lo] acts as sentinel
}
```

Partition Example, lo=0, hi=15

K	C	A	T	E	L	E	P	U	I	M	Q	R	X	O	S
0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
lo															hi
		i						j							

```
// find item smaller than (or equal to) v to swap
while (less(v, a[--j])) {
    if (j == lo) break; // redundant since a[lo] acts as sentinel
}
```

Partition Example, lo=0, hi=15

K	C	A	T	E	L	E	P	U	I	M	Q	R	X	O	S
0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
lo															hi
		i						j							

```
exch(a, i, j);
```

Partition Example, lo=0, hi=15

K	C	A	I	E	L	E	P	U	T	M	Q	R	X	O	S
0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
lo															hi
			i						j						

```
exch(a, i, j);
```

Partition Example, lo=0, hi=15

K	C	A	I	E	L	E	P	U	T	M	Q	R	X	O	S
0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
lo															hi
			i					j							

```
// find greater than (or equal to) v to swap
while (less(a[++i], v)) {
    if (i == hi) break;
}
```

Partition Example, lo=0, hi=15

K	C	A	I	E	L	E	P	U	T	M	Q	R	X	O	S
0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
lo															hi
					i				j						

```
// find greater than (or equal to) v to swap
while (less(a[++i], v)) {
    if (i == hi) break;
}
```

Partition Example, lo=0, hi=15

K	C	A	I	E	L	E	P	U	T	M	Q	R	X	O	S
0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
lo															hi
					i			j							

```
// find greater than (or equal to) v to swap
while (less(v, a[--j])) {
    if (j == lo) break; // redundant since a[lo] acts as sentinel
}
```

Partition Example, lo=0, hi=15

K	C	A	I	E	L	E	P	U	T	M	Q	R	X	O	S
0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
lo															hi
					i		j								

```
// find greater than (or equal to) v to swap
while (less(v, a[--j])) {
    if (j == lo) break; // redundant since a[lo] acts as sentinel
}
```

Partition Example, lo=0, hi=15

K	C	A	I	E	L	E	P	U	T	M	Q	R	X	O	S
0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
lo															hi
					i	j									

```
// find greater than (or equal to) v to swap
while (less(v, a[--j])) {
    if (j == lo) break; // redundant since a[lo] acts as sentinel
}
```

Partition Example, lo=0, hi=15

K	C	A	I	E	L	E	P	U	T	M	Q	R	X	O	S
0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
lo															hi
			i	j											

```
exch(a, i, j);
```

Partition Example, lo=0, hi=15

K	C	A	I	E	E	L	P	U	T	M	Q	R	X	O	S
0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
lo															hi
			i	j											

```
exch(a, i, j);
```

Partition Example, lo=0, hi=15

K	C	A	I	E	E	L	P	U	T	M	Q	R	X	O	S
0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
lo															hi
						i,j									

```
// find greater than (or equal to) v to swap
while (less(a[++i], v)) {
    if (i == hi) break;
}
```

Partition Example, lo=0, hi=15

K	C	A	I	E	E	L	P	U	T	M	Q	R	X	O	S
0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
lo															hi
			j	i											

```
// find item smaller than (or equal to) v to swap
while (less(v, a[--j])) {
    if (j == lo) break; // redundant since a[lo] acts as sentinel
}
```

Partition Example, lo=0, hi=15

K	C	A	I	E	E	L	P	U	T	M	Q	R	X	O	S
0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
lo															hi
				j		i									

```
// check if pointers cross  
if (i >= j) break;
```

Partition Example, lo=0, hi=15

K	C	A	I	E	E	L	P	U	T	M	Q	R	X	O	S
0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
lo															hi
			j	i											

```
// put partitioning item v at a[j]
exch(a, lo, j);

// now, a[lo .. j-1] <= a[j] <= a[j+1 .. hi]
return j;
```

Partition Example, lo=0, hi=15

E	C	A	I	E	K	L	P	U	T	M	Q	R	X	O	S
0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
lo															hi
					j	i									

```
// put partitioning item v at a[j]
exch(a, lo, j);

// now, a[lo .. j-1] <= a[j] <= a[j+1 .. hi]
return j;
```

Partition for lo=0, hi=15 is complete. j=5

Call sort recursively on left subarray with lo=0, hi =4

```
private static void sort(Comparable[] a,
    int lo, int hi) {
    if (hi <= lo) return;
    int j = partition(a, lo, hi);
    sort(a, lo, j-1);
    sort(a, j+1, hi);
}
```

QUICKSORT

Quicksort Trace

```
private static void sort(Comparable[] a, int lo, int hi) {  
    if (hi <= lo) return;  
    int j = partition(a, lo, hi);  
    sort(a, lo, j-1);  
    sort(a, j+1, hi);  
}
```

initial values random shuffle	lo	j	hi	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
				Q	U	I	C	K	S	O	R	T	E	X	A	M	P	L	E
				K	R	A	T	E	L	E	P	U	I	M	Q	C	X	O	S
	0	5	15	E	C	A	I	E	K	L	P	U	T	M	Q	R	X	O	S
	0	3	4	E	C	A	E	I	K	L	P	U	T	M	Q	R	X	O	S
	0	2	2	A	C	E	E	I	K	L	P	U	T	M	Q	R	X	O	S
	0	0	1	A	C	E	E	I	K	L	P	U	T	M	Q	R	X	O	S
	1		1	A	C	E	E	I	K	L	P	U	T	M	Q	R	X	O	S
	4		4	A	C	E	E	I	K	L	P	U	T	M	Q	R	X	O	S
	6	6	15	A	C	E	E	I	K	L	P	U	T	M	Q	R	X	O	S
	7	9	15	A	C	E	E	I	K	L	M	O	P	T	Q	R	X	U	S
	7	7	8	A	C	E	E	I	K	L	M	O	P	T	Q	R	X	U	S
	8		8	A	C	E	E	I	K	L	M	O	P	T	Q	R	X	U	S
	10	13	15	A	C	E	E	I	K	L	M	O	P	S	Q	R	T	U	X
	10	12	12	A	C	E	E	I	K	L	M	O	P	R	Q	S	T	U	X
	10	11	11	A	C	E	E	I	K	L	M	O	P	Q	R	S	T	U	X
	10		10	A	C	E	E	I	K	L	M	O	P	Q	R	S	T	U	X
	14	14	15	A	C	E	E	I	K	L	M	O	P	Q	R	S	T	U	X
	15		15	A	C	E	E	I	K	L	M	O	P	Q	R	S	T	U	X
no partition for subarrays of size 1				A	C	E	E	I	K	L	M	O	P	Q	R	S	T	U	X
result				A	C	E	E	I	K	L	M	O	P	Q	R	S	T	U	X

Partition Example, lo=0, hi=4

E	C	A	I	E	K	L	P	U	T	M	Q	R	X	O	S
0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
lo				hi											
i					j										

```
// partition the subarray a[lo..hi] so that a[lo..j-1] <= a[j] <= a[j+1..hi] and return the index j.  
private static int partition(Comparable[] a, int lo, int hi) {  
    int i = lo;  
    int j = hi + 1;  
    Comparable v = a[lo];
```

Partition Example, lo=0, hi=4

E	C	A	I	E	K	L	P	U	T	M	Q	R	X	O	S
0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
lo				hi											
i					j										

```
// find greater than (or equal to) v to swap
while (less(a[++i], v)) {
    if (i == hi) break;
}
```

Partition Example, lo=0, hi=4

E	C	A	I	E	K	L	P	U	T	M	Q	R	X	O	S
0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
lo				hi											
		i			j										

```
// find greater than (or equal to) v to swap
while (less(a[++i], v)) {
    if (i == hi) break;
}
```

Partition Example, lo=0, hi=4

E	C	A	I	E	K	L	P	U	T	M	Q	R	X	O	S
0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
lo				hi											
			i			j									

```
// find greater than (or equal to) v to swap
while (less(a[++i], v)) {
    if (i == hi) break;
}
```

Partition Example, lo=0, hi=4

E	C	A	I	E	K	L	P	U	T	M	Q	R	X	O	S
0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
lo				hi											
			i	j											

```
// find item smaller than (or equal to) v to swap
while (less(v, a[--j])) {
    if (j == lo) break; // redundant since a[lo] acts as sentinel
}
```

Partition Example, lo=0, hi=4

E	C	A	I	E	K	L	P	U	T	M	Q	R	X	O	S
0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
lo				hi											
			i	j											

```
exch(a, i, j);
```

Partition Example, lo=0, hi=4

E	C	A	E	I	K	L	P	U	T	M	Q	R	X	O	S
0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
lo				hi											
			i	j											

```
exch(a, i, j);
```

Partition Example, lo=0, hi=4

E	C	A	E	I	K	L	P	U	T	M	Q	R	X	O	S
0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
lo				hi											
				i,j											

```
// find greater than (or equal to) v to swap
while (less(a[++i], v)) {
    if (i == hi) break;
}
```

Partition Example, lo=0, hi=4

E	C	A	E	I	K	L	P	U	T	M	Q	R	X	O	S
0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
lo				hi											
			j	i											

```
// find item smaller than (or equal to) v to swap
while (less(v, a[--j])) {
    if (j == lo) break; // redundant since a[lo] acts as sentinel
}
```

Partition Example, lo=0, hi=4

E	C	A	E	I	K	L	P	U	T	M	Q	R	X	O	S
0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
lo				hi											
			j	i											

```
// check if pointers cross  
if (i >= j) break;
```

Partition Example, lo=0, hi=4

E	C	A	E	I	K	L	P	U	T	M	Q	R	X	O	S
0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
lo				hi											
			j	i											

```
// put partitioning item v at a[j]
exch(a, lo, j);

// now, a[lo .. j-1] <= a[j] <= a[j+1 .. hi]
return j;
```

Partition Example, lo=0, hi=4

E	C	A	E	I	K	L	P	U	T	M	Q	R	X	O	S
0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
lo				hi											
			j	i											

```
// put partitioning item v at a[j]
exch(a, lo, j);

// now, a[lo .. j-1] <= a[j] <= a[j+1 .. hi]
return j;
```

Partition for lo=0, hi=4 is complete. j=3

Call sort recursively on left subarray with lo=0, hi =2

```
private static void sort(Comparable[] a,
    int lo, int hi) {
    if (hi <= lo) return;
    int j = partition(a, lo, hi);
    sort(a, lo, j-1);
    sort(a, j+1, hi);
}
```

QUICKSORT

Quicksort Trace

```
private static void sort(Comparable[] a, int lo, int hi) {  
    if (hi <= lo) return;  
    int j = partition(a, lo, hi);  
    sort(a, lo, j-1);  
    sort(a, j+1, hi);  
}
```

initial values random shuffle	lo	j	hi	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
				Q	U	I	C	K	S	O	R	T	E	X	A	M	P	L	E
				K	R	A	T	E	L	E	P	U	I	M	Q	C	X	O	S
	0	5	15	E	C	A	T	E	K	L	P	U	T	M	Q	R	X	O	S
	0	3	4	E	C	A	E	I	K	L	P	U	T	M	Q	R	X	O	S
	0	2	2	A	C	E	E	I	K	L	P	U	T	M	Q	R	X	O	S
	0	0	1	A	C	E	E	I	K	L	P	U	T	M	Q	R	X	O	S
no partition for subarrays of size 1	1		1	A	C	E	E	I	K	L	P	U	T	M	Q	R	X	O	S
	4		4	A	C	E	E	I	K	L	P	U	T	M	Q	R	X	O	S
	6	6	15	A	C	E	E	I	K	L	P	U	T	M	Q	R	X	O	S
	7	9	15	A	C	E	E	I	K	L	M	O	P	T	Q	R	X	U	S
	7	7	8	A	C	E	E	I	K	L	M	O	P	T	Q	R	X	U	S
	8		8	A	C	E	E	I	K	L	M	O	P	T	Q	R	X	U	S
	10	13	15	A	C	E	E	I	K	L	M	O	P	S	Q	R	T	U	X
	10	12	12	A	C	E	E	I	K	L	M	O	P	R	Q	S	T	U	X
	10	11	11	A	C	E	E	I	K	L	M	O	P	Q	R	S	T	U	X
	10		10	A	C	E	E	I	K	L	M	O	P	Q	R	S	T	U	X
	14	14	15	A	C	E	E	I	K	L	M	O	P	Q	R	S	T	U	X
	15		15	A	C	E	E	I	K	L	M	O	P	Q	R	S	T	U	X
result				A	C	E	E	I	K	L	M	O	P	Q	R	S	T	U	X

Partition Example, lo=0, hi=2

E	C	A	E	I	K	L	P	U	T	M	Q	R	X	O	S
0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
lo		hi													
i			j												

```
// partition the subarray a[lo..hi] so that a[lo..j-1] <= a[j] <= a[j+1..hi] and return the index j.  
private static int partition(Comparable[] a, int lo, int hi) {  
    int i = lo;  
    int j = hi + 1;  
    Comparable v = a[lo];
```

Partition Example, lo=0, hi=2

E	C	A	E	I	K	L	P	U	T	M	Q	R	X	O	S
0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
lo		hi													
i		j													

```
// find greater than (or equal to) v to swap
while (less(a[++i], v)) {
    if (i == hi) break;
}
```

Partition Example, lo=0, hi=2

E	C	A	E	I	K	L	P	U	T	M	Q	R	X	O	S
0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
lo		hi													
		i	j												

```
// find greater than (or equal to) v to swap
while (less(a[++i], v)) {
    if (i == hi) break;
}
```

Partition Example, lo=0, hi=2

E	C	A	E	I	K	L	P	U	T	M	Q	R	X	O	S
0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
lo		hi													
		i,j													

```
// find item smaller than (or equal to) v to swap
while (less(v, a[--j])) {
    if (j == lo) break; // redundant since a[lo] acts as sentinel
}
```

Partition Example, lo=0, hi=2

E	C	A	E	I	K	L	P	U	T	M	Q	R	X	O	S
0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
lo		hi													
		i,j													

```
// check if pointers cross  
if (i >= j) break;
```

Partition Example, lo=0, hi=2

E	C	A	E	I	K	L	P	U	T	M	Q	R	X	O	S
0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
lo		hi													
		i,j													

```
// put partitioning item v at a[j]
exch(a, lo, j);

// now, a[lo .. j-1] <= a[j] <= a[j+1 .. hi]
return j;
```

Partition Example, lo=0, hi=2

A	C	E	E	I	K	L	P	U	T	M	Q	R	X	O	S
0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
lo		hi													
		i,j													

```
// put partitioning item v at a[j]
exch(a, lo, j);

// now, a[lo .. j-1] <= a[j] <= a[j+1 .. hi]
return j;
```

Partition for lo=0, hi=2 is complete. j=2

Call sort recursively on left subarray with lo=0, hi =1

```
private static void sort(Comparable[] a,
    int lo, int hi) {
    if (hi <= lo) return;
    int j = partition(a, lo, hi);
    sort(a, lo, j-1);
    sort(a, j+1, hi);
}
```

QUICKSORT

Quicksort Trace

```
private static void sort(Comparable[] a, int lo, int hi) {  
    if (hi <= lo) return;  
    int j = partition(a, lo, hi);  
    sort(a, lo, j-1);  
    sort(a, j+1, hi);  
}
```

	lo	j	hi	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
initial values				Q	U	I	C	K	S	O	R	T	E	X	A	M	P	L	E
random shuffle				K	R	A	T	E	L	E	P	U	I	M	Q	C	X	O	S
	0	5	15	E	C	A	I	E	K	L	P	U	T	M	Q	R	X	O	S
	0	3	4	E	C	A	E	I	K	L	P	U	T	M	Q	R	X	O	S
	0	2	2	A	C	E	E	I	K	L	P	U	T	M	Q	R	X	O	S
	0	0	1	A	C	E	E	I	K	L	P	U	T	M	Q	R	X	O	S
	1		1	A	C	E	E	I	K	L	P	U	T	M	Q	R	X	O	S
	4		4	A	C	E	E	I	K	L	P	U	T	M	Q	R	X	O	S
	6	6	15	A	C	E	E	I	K	L	P	U	T	M	Q	R	X	O	S
	7	9	15	A	C	E	E	I	K	L	M	O	P	T	Q	R	X	U	S
	7	7	8	A	C	E	E	I	K	L	M	O	P	T	Q	R	X	U	S
	8		8	A	C	E	E	I	K	L	M	O	P	T	Q	R	X	U	S
	10	13	15	A	C	E	E	I	K	L	M	O	P	S	Q	R	T	U	X
	10	12	12	A	C	E	E	I	K	L	M	O	P	R	Q	S	T	U	X
	10	11	11	A	C	E	E	I	K	L	M	O	P	Q	R	S	T	U	X
	10		10	A	C	E	E	I	K	L	M	O	P	Q	R	S	T	U	X
	14	14	15	A	C	E	E	I	K	L	M	O	P	Q	R	S	T	U	X
	15		15	A	C	E	E	I	K	L	M	O	P	Q	R	S	T	U	X
result				A	C	E	E	I	K	L	M	O	P	Q	R	S	T	U	X

no partition
for subarrays
of size 1

Partition Example, lo=0, hi=1

A	C	E	E	I	K	L	P	U	T	M	Q	R	X	O	S
0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
lo	hi														
i		j													

```
// partition the subarray a[lo..hi] so that a[lo..j-1] <= a[j] <= a[j+1..hi] and return the index j.  
private static int partition(Comparable[] a, int lo, int hi) {  
    int i = lo;  
    int j = hi + 1;  
    Comparable v = a[lo];
```

Partition Example, lo=0, hi=1

A	C	E	E	I	K	L	P	U	T	M	Q	R	X	O	S
0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
lo	hi														
	i	j													

```
// find greater than (or equal to) v to swap
while (less(a[++i], v)) {
    if (i == hi) break;
}
```

Partition Example, lo=0, hi=1

A	C	E	E	I	K	L	P	U	T	M	Q	R	X	O	S
0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
lo	hi														
	i,j														

```
// find item smaller than (or equal to) v to swap
while (less(v, a[--j])) {
    if (j == lo) break; // redundant since a[lo] acts as sentinel
}
```

Partition Example, lo=0, hi=1

A	C	E	E	I	K	L	P	U	T	M	Q	R	X	O	S
0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
lo	hi														
j	i														

```
// find item smaller than (or equal to) v to swap
while (less(v, a[--j])) {
    if (j == lo) break; // redundant since a[lo] acts as sentinel
}
```

Partition Example, lo=0, hi=1

A	C	E	E	I	K	L	P	U	T	M	Q	R	X	O	S
0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
lo	hi														
j	i														

```
// check if pointers cross  
if (i >= j) break;
```

Partition Example, lo=0, hi=1

A	C	E	E	I	K	L	P	U	T	M	Q	R	X	O	S
0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
lo	hi														
j	i														

```
// put partitioning item v at a[j]
exch(a, lo, j);

// now, a[lo .. j-1] <= a[j] <= a[j+1 .. hi]
return j;
```

Partition Example, lo=0, hi=1

A	C	E	E	I	K	L	P	U	T	M	Q	R	X	O	S
0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
lo	hi														
j	i														

```
// put partitioning item v at a[j]
exch(a, lo, j);

// now, a[lo .. j-1] <= a[j] <= a[j+1 .. hi]
return j;
```

Partition for lo=0, hi=1 is complete. j=0

No partition for subarrays of size 1.

Call sort recursively on right subarray with lo=6, hi =15

```
private static void sort(Comparable[] a, int lo, int hi){

    if (hi <= lo) return;
    int j = partition(a, lo, hi);
    sort(a, lo, j-1);
    sort(a, j+1, hi);
}
```

QUICKSORT

Quicksort Trace

```
private static void sort(Comparable[] a, int lo, int hi) {  
    if (hi <= lo) return;  
    int j = partition(a, lo, hi);  
    sort(a, lo, j-1);  
    sort(a, j+1, hi);  
}
```

			lo	j	hi	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
initial values						Q	U	I	C	K	S	O	R	T	E	X	A	M	P	L	E
random shuffle						K	R	A	T	E	L	E	P	U	I	M	Q	C	X	O	S
			0	5	15	E	C	A	I	E	K	L	P	U	T	M	Q	R	X	O	S
			0	3	4	E	C	A	E	I	K	L	P	U	T	M	Q	R	X	O	S
			0	2	2	A	C	E	E	I	K	L	P	U	T	M	Q	R	X	O	S
			0	0	1	A	C	E	E	I	K	L	P	U	T	M	Q	R	X	O	S
			1		1	A	C	E	E	I	K	L	P	U	T	M	Q	R	X	O	S
			4		4	A	C	E	E	I	K	L	P	U	T	M	Q	R	X	O	S
			6	6	15	A	C	E	E	I	K	L	P	U	T	M	Q	R	X	O	S
			7	9	15	A	C	E	E	I	K	L	M	O	P	T	Q	R	X	U	S
			7	7	8	A	C	E	E	I	K	L	M	O	P	T	Q	R	X	U	S
			8		8	A	C	E	E	I	K	L	M	O	P	T	Q	R	X	U	S
			10	13	15	A	C	E	E	I	K	L	M	O	P	S	Q	R	T	U	X
			10	12	12	A	C	E	E	I	K	L	M	O	P	R	Q	S	T	U	X
			10	11	11	A	C	E	E	I	K	L	M	O	P	Q	R	S	T	U	X
			10		10	A	C	E	E	I	K	L	M	O	P	Q	R	S	T	U	X
			14	14	15	A	C	E	E	I	K	L	M	O	P	Q	R	S	T	U	X
			15		15	A	C	E	E	I	K	L	M	O	P	Q	R	S	T	U	X
result						A	C	E	E	I	K	L	M	O	P	Q	R	S	T	U	X

no partition
for subarrays
of size 1

Partition Example, lo=6, hi=15

A	C	E	E	I	K	L	P	U	T	M	Q	R	X	O	S	
0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	
						lo								hi		
						i									j	

```
// partition the subarray a[lo..hi] so that a[lo..j-1] <= a[j] <= a[j+1..hi] and return the index j.  
private static int partition(Comparable[] a, int lo, int hi) {  
    int i = lo;  
    int j = hi + 1;  
    Comparable v = a[lo];
```

Partition Example, lo=6, hi=15

A	C	E	E	I	K	L	P	U	T	M	Q	R	X	O	S		
0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15		
						lo								hi			
								i								j	

```
// find greater than (or equal to) v to swap
while (less(a[++i], v)) {
    if (i == hi) break;
}
```

Partition Example, lo=6, hi=15

A	C	E	E	I	K	L	P	U	T	M	Q	R	X	O	S
0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
						lo								hi	
							i								j

```
// find item smaller than (or equal to) v to swap
while (less(v, a[--j])) {
    if (j == lo) break; // redundant since a[lo] acts as sentinel
}
```

Partition Example, lo=6, hi=15

A	C	E	E	I	K	L	P	U	T	M	Q	R	X	O	S
0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
						lo								hi	
								i						j	

```
// find item smaller than (or equal to) v to swap
while (less(v, a[--j])) {
    if (j == lo) break; // redundant since a[lo] acts as sentinel
}
```

Partition Example, lo=6, hi=15

A	C	E	E	I	K	L	P	U	T	M	Q	R	X	O	S		
0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15		
						lo								hi			
								i						j			

```
// find item smaller than (or equal to) v to swap
while (less(v, a[--j])) {
    if (j == lo) break; // redundant since a[lo] acts as sentinel
}
```

Partition Example, lo=6, hi=15

A	C	E	E	I	K	L	P	U	T	M	Q	R	X	O	S
0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
						lo								hi	
								i				j			

```
// find item smaller than (or equal to) v to swap
while (less(v, a[--j])) {
    if (j == lo) break; // redundant since a[lo] acts as sentinel
}
```

Partition Example, lo=6, hi=15

A	C	E	E	I	K	L	P	U	T	M	Q	R	X	O	S
0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
						lo									hi
							i				j				

```
// find item smaller than (or equal to) v to swap
while (less(v, a[--j])) {
    if (j == lo) break; // redundant since a[lo] acts as sentinel
}
```

Partition Example, lo=6, hi=15

A	C	E	E	I	K	L	P	U	T	M	Q	R	X	O	S	
0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	
						lo								hi		
							i			j						

```
// find item smaller than (or equal to) v to swap
while (less(v, a[--j])) {
    if (j == lo) break; // redundant since a[lo] acts as sentinel
}
```

Partition Example, lo=6, hi=15

A	C	E	E	I	K	L	P	U	T	M	Q	R	X	O	S
0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
						lo									hi
						i	j								

```
// find item smaller than (or equal to) v to swap
while (less(v, a[--j])) {
    if (j == lo) break; // redundant since a[lo] acts as sentinel
}
```

Partition Example, lo=6, hi=15

A	C	E	E	I	K	L	P	U	T	M	Q	R	X	O	S		
0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15		
						lo										hi	
						i		j									

```
// find item smaller than (or equal to) v to swap
while (less(v, a[--j])) {
    if (j == lo) break; // redundant since a[lo] acts as sentinel
}
```

Partition Example, lo=6, hi=15

A	C	E	E	I	K	L	P	U	T	M	Q	R	X	O	S
0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
						lo									hi
						i,j									

```
// find item smaller than (or equal to) v to swap
while (less(v, a[--j])) {
    if (j == lo) break; // redundant since a[lo] acts as sentinel
}
```

Partition Example, lo=6, hi=15

A	C	E	E	I	K	L	P	U	T	M	Q	R	X	O	S
0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
						lo									hi
					j	i									

```
// find item smaller than (or equal to) v to swap
while (less(v, a[--j])) {
    if (j == lo) break; // redundant since a[lo] acts as sentinel
}
```

Partition Example, lo=6, hi=15

A	C	E	E	I	K	L	P	U	T	M	Q	R	X	O	S
0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
						lo									hi
						j	i								

```
// check if pointers cross  
if (i >= j) break;
```

Partition Example, lo=6, hi=15

A	C	E	E	I	K	L	P	U	T	M	Q	R	X	O	S
0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
						lo									hi
						j	i								

```
// put partitioning item v at a[j]
exch(a, lo, j);

// now, a[lo .. j-1] <= a[j] <= a[j+1 .. hi]
return j;
```

Partition Example, lo=6, hi=15

A	C	E	E	I	K	L	P	U	T	M	Q	R	X	O	S
0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
						lo									hi
						j	i								

```
// put partitioning item v at a[j]
exch(a, lo, j);

// now, a[lo .. j-1] <= a[j] <= a[j+1 .. hi]
return j;
```

Partition for lo=6, hi=15 is complete. j=6

No partition for subarrays of size 1.

Call sort recursively on right subarray with lo=7, hi =15

```
private static void sort(Comparable[] a, int lo, int hi){

    if (hi <= lo) return;
    int j = partition(a, lo, hi);
    sort(a, lo, j-1);
    sort(a, j+1, hi);
}
```

QUICKSORT

```
private static void sort(Comparable[] a, int lo, int hi) {  
    if (hi <= lo) return;  
    int j = partition(a, lo, hi);  
    sort(a, lo, j-1);  
    sort(a, j+1, hi);  
}
```

Quicksort Trace

	lo	j	hi	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
initial values				Q	U	I	C	K	S	O	R	T	E	X	A	M	P	L	E
random shuffle				K	R	A	T	E	L	E	P	U	I	M	Q	C	X	O	S
	0	5	15	E	C	A	I	E	K	L	P	U	T	M	Q	R	X	O	S
	0	3	4	E	C	A	E	I	K	L	P	U	T	M	Q	R	X	O	S
	0	2	2	A	C	E	E	I	K	L	P	U	T	M	Q	R	X	O	S
	0	0	1	A	C	E	E	I	K	L	P	U	T	M	Q	R	X	O	S
	1		1	A	C	E	E	I	K	L	P	U	T	M	Q	R	X	O	S
	4		4	A	C	E	E	I	K	L	P	U	T	M	Q	R	X	O	S
	6	6	15	A	C	E	E	I	K	L	P	U	T	M	Q	R	X	O	S
	7	9	15	A	C	E	E	I	K	L	M	O	P	T	Q	R	X	U	S
	7	7	8	A	C	E	E	I	K	L	M	O	P	T	Q	R	X	U	S
	8		8	A	C	E	E	I	K	L	M	O	P	T	Q	R	X	U	S
	10	13	15	A	C	E	E	I	K	L	M	O	P	S	Q	R	T	U	X
	10	12	12	A	C	E	E	I	K	L	M	O	P	R	Q	S	T	U	X
	10	11	11	A	C	E	E	I	K	L	M	O	P	Q	R	S	T	U	X
	10		10	A	C	E	E	I	K	L	M	O	P	Q	R	S	T	U	X
	14	14	15	A	C	E	E	I	K	L	M	O	P	Q	R	S	T	U	X
	15		15	A	C	E	E	I	K	L	M	O	P	Q	R	S	T	U	X
result				A	C	E	E	I	K	L	M	O	P	Q	R	S	T	U	X

no partition
for subarrays
of size 1

Partition Example, lo=7, hi=15

A	C	E	E	I	K	L	P	U	T	M	Q	R	X	O	S	
0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	
							lo								hi	
								i								j

```
// partition the subarray a[lo..hi] so that a[lo..j-1] <= a[j] <= a[j+1..hi] and return the index j.  
private static int partition(Comparable[] a, int lo, int hi) {  
    int i = lo;  
    int j = hi + 1;  
    Comparable v = a[lo];
```

Partition Example, lo=7, hi=15

A	C	E	E	I	K	L	P	U	T	M	Q	R	X	O	S	
0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	
							lo								hi	
								i						j		

```
// find greater than (or equal to) v to swap
while (less(a[++i], v)) {
    if (i == hi) break;
}
```

Partition Example, lo=7, hi=15

A	C	E	E	I	K	L	P	U	T	M	Q	R	X	O	S
0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
							lo								hi
								i						j	

```
// find item smaller than (or equal to) v to swap
while (less(v, a[--j])) {
    if (j == lo) break; // redundant since a[lo] acts as sentinel
}
```

Partition Example, lo=7, hi=15

A	C	E	E	I	K	L	P	U	T	M	Q	R	X	O	S
0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
							lo								hi
								i					j		

```
// find item smaller than (or equal to) v to swap
while (less(v, a[--j])) {
    if (j == lo) break; // redundant since a[lo] acts as sentinel
}
```

Partition Example, lo=7, hi=15

A	C	E	E	I	K	L	P	U	T	M	Q	R	X	O	S
0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
							lo								hi
								i					j		

```
exch(a, i, j);
```

Partition Example, lo=7, hi=15

A	C	E	E	I	K	L	P	O	T	M	Q	R	X	U	S
0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
							lo								hi
								i					j		

```
exch(a, i, j);
```

Partition Example, lo=7, hi=15

A	C	E	E	I	K	L	P	O	T	M	Q	R	X	U	S
0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
							lo								hi
								i						j	

```
// find greater than (or equal to) v to swap
while (less(a[++i], v)) {
    if (i == hi) break;
}
```

Partition Example, lo=7, hi=15

A	C	E	E	I	K	L	P	O	T	M	Q	R	X	U	S
0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
							lo								hi
										i			j		

```
// find item smaller than (or equal to) v to swap
while (less(v, a[--j])) {
    if (j == lo) break; // redundant since a[lo] acts as sentinel
}
```

Partition Example, lo=7, hi=15

A	C	E	E	I	K	L	P	O	T	M	Q	R	X	U	S
0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
							lo								hi
								i				j			

```
// find item smaller than (or equal to) v to swap
while (less(v, a[--j])) {
    if (j == lo) break; // redundant since a[lo] acts as sentinel
}
```

Partition Example, lo=7, hi=15

A	C	E	E	I	K	L	P	O	T	M	Q	R	X	U	S
0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
							lo								hi
								i		j					

```
// find item smaller than (or equal to) v to swap
while (less(v, a[--j])) {
    if (j == lo) break; // redundant since a[lo] acts as sentinel
}
```

Partition Example, lo=7, hi=15

A	C	E	E	I	K	L	P	O	T	M	Q	R	X	U	S	
0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	
							lo									hi
								i		j						

```
// find item smaller than (or equal to) v to swap
while (less(v, a[--j])) {
    if (j == lo) break; // redundant since a[lo] acts as sentinel
}
```

Partition Example, lo=7, hi=15

A	C	E	E	I	K	L	P	O	T	M	Q	R	X	U	S
0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
							lo								hi
							i		j						

```
exch(a, i, j);
```

Partition Example, lo=7, hi=15

A	C	E	E	I	K	L	P	O	M	T	Q	R	X	U	S
0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
							lo								hi
								i		j					

```
exch(a, i, j);
```

Partition Example, lo=7, hi=15

A	C	E	E	I	K	L	P	O	M	T	Q	R	X	U	S
0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
							lo								hi
								i,j							

```
// find greater than (or equal to) v to swap
while (less(a[++i], v)) {
    if (i == hi) break;
}
```

Partition Example, lo=7, hi=15

A	C	E	E	I	K	L	P	O	M	T	Q	R	X	U	S
0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
							lo								hi
								j		i					

```
// find item smaller than (or equal to) v to swap
while (less(v, a[--j])) {
    if (j == lo) break; // redundant since a[lo] acts as sentinel
}
```

Partition Example, lo=7, hi=15

A	C	E	E	I	K	L	P	O	M	T	Q	R	X	U	S
0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
							lo								hi
								j		i					

```
// check if pointers cross  
if (i >= j) break;
```

Partition Example, lo=7, hi=15

A	C	E	E	I	K	L	P	O	M	T	Q	R	X	U	S
0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
							lo								hi
							j		i						

```
// put partitioning item v at a[j]
exch(a, lo, j);

// now, a[lo .. j-1] <= a[j] <= a[j+1 .. hi]
return j;
```

Partition Example, lo=7, hi=15

A	C	E	E	I	K	L	M	O	P	T	Q	R	X	U	S		
0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15		
							lo									hi	
								j		i							

```
// put partitioning item v at a[j]
exch(a, lo, j);

// now, a[lo .. j-1] <= a[j] <= a[j+1 .. hi]
return j;
```

Partition for lo=6, hi=15 is complete. j=9

Call sort recursively on left subarray with lo=7, hi = 8

```
private static void sort(Comparable[] a, int lo, int hi){

    if (hi <= lo) return;
    int j = partition(a, lo, hi);
    sort(a, lo, j-1);
    sort(a, j+1, hi);

}
```

QUICKSORT

Quicksort Trace

```
private static void sort(Comparable[] a, int lo, int hi) {  
    if (hi <= lo) return;  
    int j = partition(a, lo, hi);  
    sort(a, lo, j-1);  
    sort(a, j+1, hi);  
}
```

			lo	j	hi	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
initial values						Q	U	I	C	K	S	O	R	T	E	X	A	M	P	L	E
random shuffle						K	R	A	T	E	L	E	P	U	I	M	Q	C	X	O	S
			0	5	15	E	C	A	I	E	K	L	P	U	T	M	Q	R	X	O	S
			0	3	4	E	C	A	E	I	K	L	P	U	T	M	Q	R	X	O	S
			0	2	2	A	C	E	E	I	K	L	P	U	T	M	Q	R	X	O	S
			0	0	1	A	C	E	E	I	K	L	P	U	T	M	Q	R	X	O	S
			1		1	A	C	E	E	I	K	L	P	U	T	M	Q	R	X	O	S
			4		4	A	C	E	E	I	K	L	P	U	T	M	Q	R	X	O	S
			6	6	15	A	C	E	E	I	K	L	P	U	T	M	Q	R	X	O	S
no partition for subarrays of size 1			7	9	15	A	C	E	E	I	K	L	M	O	P	T	Q	R	X	U	S
			7	7	8	A	C	E	E	I	K	L	M	O	P	T	Q	R	X	U	S
			8		8	A	C	E	E	I	K	L	M	O	P	T	Q	R	X	U	S
			10	13	15	A	C	E	E	I	K	L	M	O	P	S	Q	R	T	U	X
			10	12	12	A	C	E	E	I	K	L	M	O	P	R	Q	S	T	U	X
			10	11	11	A	C	E	E	I	K	L	M	O	P	Q	R	S	T	U	X
			10		10	A	C	E	E	I	K	L	M	O	P	Q	R	S	T	U	X
			14	14	15	A	C	E	E	I	K	L	M	O	P	Q	R	S	T	U	X
			15		15	A	C	E	E	I	K	L	M	O	P	Q	R	S	T	U	X
result						A	C	E	E	I	K	L	M	O	P	Q	R	S	T	U	X

Partition Example, lo=7, hi=8

A	C	E	E	I	K	L	M	O	P	T	Q	R	X	U	S
0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
								lo	hi						
						i		j							

```
// partition the subarray a[lo..hi] so that a[lo..j-1] <= a[j] <= a[j+1..hi] and return the index j.  
private static int partition(Comparable[] a, int lo, int hi) {  
    int i = lo;  
    int j = hi + 1;  
    Comparable v = a[lo];
```

Partition Example, lo=7, hi=8

A	C	E	E	I	K	L	M	O	P	T	Q	R	X	U	S
0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
							lo	hi							
								i	j						

```
// find greater than (or equal to) v to swap
while (less(a[++i], v)) {
    if (i == hi) break;
}
```

Partition Example, lo=7, hi=8

A	C	E	E	I	K	L	M	O	P	T	Q	R	X	U	S
0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
							lo	hi							
								i,j							

```
// find item smaller than (or equal to) v to swap
while (less(v, a[--j])) {
    if (j == lo) break; // redundant since a[lo] acts as sentinel
}
```

Partition Example, lo=7, hi=8

A	C	E	E	I	K	L	M	O	P	T	Q	R	X	U	S
0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
							lo	hi							
							j	i							

```
// find item smaller than (or equal to) v to swap
while (less(v, a[--j])) {
    if (j == lo) break; // redundant since a[lo] acts as sentinel
}
```

Partition Example, lo=7, hi=8

A	C	E	E	I	K	L	M	O	P	T	Q	R	X	U	S
0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
							lo	hi							
							j	i							

```
// check if pointers cross  
if (i >= j) break;
```

Partition Example, lo=7, hi=8

A	C	E	E	I	K	L	M	O	P	T	Q	R	X	U	S
0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
							lo	hi							
							j	i							

```
// put partitioning item v at a[j]
exch(a, lo, j);

// now, a[lo .. j-1] <= a[j] <= a[j+1 .. hi]
return j;
```

Partition Example, lo=7, hi=8

A	C	E	E	I	K	L	M	O	P	T	Q	R	X	U	S
0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
							lo	hi							
							j	i							

```
// put partitioning item v at a[j]
exch(a, lo, j);

// now, a[lo .. j-1] <= a[j] <= a[j+1 .. hi]
return j;
```

Partition for lo=7, hi=8 is complete. j=7

No partition for subarrays of size 1.

Call sort recursively on right subarray with lo=10, hi = 15

```
private static void sort(Comparable[] a, int lo, int hi){

    if (hi <= lo) return;
    int j = partition(a, lo, hi);
    sort(a, lo, j-1);
    sort(a, j+1, hi);
}
```

QUICKSORT

```
private static void sort(Comparable[] a, int lo, int hi) {  
    if (hi <= lo) return;  
    int j = partition(a, lo, hi);  
    sort(a, lo, j-1);  
    sort(a, j+1, hi);  
}
```

Quicksort Trace

			lo	j	hi	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
initial values						Q	U	I	C	K	S	O	R	T	E	X	A	M	P	L	E
random shuffle						K	R	A	T	E	L	E	P	U	I	M	Q	C	X	O	S
			0	5	15	E	C	A	I	E	K	L	P	U	T	M	Q	R	X	O	S
			0	3	4	E	C	A	E	I	K	L	P	U	T	M	Q	R	X	O	S
			0	2	2	A	C	E	E	I	K	L	P	U	T	M	Q	R	X	O	S
			0	0	1	A	C	E	E	I	K	L	P	U	T	M	Q	R	X	O	S
			1		1	A	C	E	E	I	K	L	P	U	T	M	Q	R	X	O	S
			4		4	A	C	E	E	I	K	L	P	U	T	M	Q	R	X	O	S
			6	6	15	A	C	E	E	I	K	L	P	U	T	M	Q	R	X	O	S
			7	9	15	A	C	E	E	I	K	L	M	O	P	T	Q	R	X	U	S
no partition for subarrays of size 1			7	7	8	A	C	E	E	I	K	L	M	O	P	T	Q	R	X	U	S
			8		8	A	C	E	E	I	K	L	M	O	P	T	Q	R	X	U	S
			10	13	15	A	C	E	E	I	K	L	M	O	P	S	Q	R	T	U	X
			10	12	12	A	C	E	E	I	K	L	M	O	P	R	Q	S	T	U	X
			10	11	11	A	C	E	E	I	K	L	M	O	P	Q	R	S	T	U	X
			10		10	A	C	E	E	I	K	L	M	O	P	Q	R	S	T	U	X
			14	14	15	A	C	E	E	I	K	L	M	O	P	Q	R	S	T	U	X
			15		15	A	C	E	E	I	K	L	M	O	P	Q	R	S	T	U	X
result						A	C	E	E	I	K	L	M	O	P	Q	R	S	T	U	X

Partition Example, lo=10, hi=15

A	C	E	E	I	K	L	M	O	P	T	Q	R	X	U	S	
0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	
										lo						hi
										i				j		

```
// partition the subarray a[lo..hi] so that a[lo..j-1] <= a[j] <= a[j+1..hi] and return the index j.  
private static int partition(Comparable[] a, int lo, int hi) {  
    int i = lo;  
    int j = hi + 1;  
    Comparable v = a[lo];
```

Partition Example, lo=10, hi=15

A	C	E	E	I	K	L	M	O	P	T	Q	R	X	U	S	
0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	
										lo						hi
										i				j		

```
// find greater than (or equal to) v to swap
while (less(a[++i], v)) {
    if (i == hi) break;
}
```

Partition Example, lo=10, hi=15

A	C	E	E	I	K	L	M	O	P	T	Q	R	X	U	S	
0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	
										lo						hi
												i				j

```
// find greater than (or equal to) v to swap
while (less(a[++i], v)) {
    if (i == hi) break;
}
```

Partition Example, lo=10, hi=15

A	C	E	E	I	K	L	M	O	P	T	Q	R	X	U	S	
0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	
										lo						hi
													i			j

```
// find greater than (or equal to) v to swap
while (less(a[++i], v)) {
    if (i == hi) break;
}
```

Partition Example, lo=10, hi=15

A	C	E	E	I	K	L	M	O	P	T	Q	R	X	U	S	
0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	
										lo						hi
												i		j		

```
// find item smaller than (or equal to) v to swap
while (less(v, a[--j])) {
    if (j == lo) break; // redundant since a[lo] acts as sentinel
}
```

Partition Example, lo=10, hi=15

A	C	E	E	I	K	L	M	O	P	T	Q	R	X	U	S	
0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	
										lo						hi
												i		j		

```
exch(a, i, j);
```

Partition Example, lo=10, hi=15

A	C	E	E	I	K	L	M	O	P	T	Q	R	S	U	X	
0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	
										lo						hi
												i		j		

```
exch(a, i, j);
```

Partition Example, lo=10, hi=15

A	C	E	E	I	K	L	M	O	P	T	Q	R	S	U	X	
0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	
										lo						hi
														i	j	

```
// find greater than (or equal to) v to swap
while (less(a[++i], v)) {
    if (i == hi) break;
}
```

Partition Example, lo=10, hi=15

A	C	E	E	I	K	L	M	O	P	T	Q	R	S	U	X	
0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	
										lo						hi
														i,j		

```
// find item smaller than (or equal to) v to swap
while (less(v, a[--j])) {
    if (j == lo) break; // redundant since a[lo] acts as sentinel
}
```

Partition Example, lo=10, hi=15

A	C	E	E	I	K	L	M	O	P	T	Q	R	S	U	X	
0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	
										lo						hi
												j		i		

```
// find item smaller than (or equal to) v to swap
while (less(v, a[--j])) {
    if (j == lo) break; // redundant since a[lo] acts as sentinel
}
```

Partition Example, lo=10, hi=15

A	C	E	E	I	K	L	M	O	P	T	Q	R	S	U	X	
0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	
										lo						hi
												j		i		

```
// check if pointers cross  
if (i >= j) break;
```

Partition Example, lo=10, hi=15

A	C	E	E	I	K	L	M	O	P	T	Q	R	S	U	X	
0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	
										lo						hi
												j		i		

```
// put partitioning item v at a[j]
exch(a, lo, j);

// now, a[lo .. j-1] <= a[j] <= a[j+1 .. hi]
return j;
```

Partition Example, lo=10, hi=15

A	C	E	E	I	K	L	M	O	P	S	Q	R	T	U	X	
0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	
										lo						hi
												j		i		

```
// put partitioning item v at a[j]
exch(a, lo, j);

// now, a[lo .. j-1] <= a[j] <= a[j+1 .. hi]
return j;
```

Partition for lo=6, hi=15 is complete. j=13

Call sort recursively on left subarray with lo=10, hi = 12

```
private static void sort(Comparable[] a, int lo, int hi) {

    if (hi <= lo) return;
    int j = partition(a, lo, hi);
    sort(a, lo, j-1);
    sort(a, j+1, hi);
}
```

QUICKSORT

Quicksort Trace

```
private static void sort(Comparable[] a, int lo, int hi) {  
    if (hi <= lo) return;  
    int j = partition(a, lo, hi);  
    sort(a, lo, j-1);  
    sort(a, j+1, hi);  
}
```

			lo	j	hi	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
initial values						Q	U	I	C	K	S	O	R	T	E	X	A	M	P	L	E
random shuffle						K	R	A	T	E	L	E	P	U	I	M	Q	C	X	O	S
			0	5	15	E	C	A	I	E	K	L	P	U	T	M	Q	R	X	O	S
			0	3	4	E	C	A	E	I	K	L	P	U	T	M	Q	R	X	O	S
			0	2	2	A	C	E	E	I	K	L	P	U	T	M	Q	R	X	O	S
			0	0	1	A	C	E	E	I	K	L	P	U	T	M	Q	R	X	O	S
			1		1	A	C	E	E	I	K	L	P	U	T	M	Q	R	X	O	S
			4		4	A	C	E	E	I	K	L	P	U	T	M	Q	R	X	O	S
			6	6	15	A	C	E	E	I	K	L	P	U	T	M	Q	R	X	O	S
			7	9	15	A	C	E	E	I	K	L	M	O	P	T	Q	R	X	U	S
			7	7	8	A	C	E	E	I	K	L	M	O	P	T	Q	R	X	U	S
			8		8	A	C	E	E	I	K	L	M	O	P	T	Q	R	X	U	S
no partition for subarrays of size 1			10	13	15	A	C	E	E	I	K	L	M	O	P	S	Q	R	T	U	X
			10	12	12	A	C	E	E	I	K	L	M	O	P	R	Q	S	T	U	X
			10	11	11	A	C	E	E	I	K	L	M	O	P	Q	R	S	T	U	X
			10		10	A	C	E	E	I	K	L	M	O	P	Q	R	S	T	U	X
			14	14	15	A	C	E	E	I	K	L	M	O	P	Q	R	S	T	U	X
			15		15	A	C	E	E	I	K	L	M	O	P	Q	R	S	T	U	X
						A	C	E	E	I	K	L	M	O	P	Q	R	S	T	U	X
result						A	C	E	E	I	K	L	M	O	P	Q	R	S	T	U	X

Partition Example, lo=10, hi=12

A	C	E	E	I	K	L	M	O	P	S	Q	R	T	U	X
0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
										lo	hi				
										i	j				

```
// partition the subarray a[lo..hi] so that a[lo..j-1] <= a[j] <= a[j+1..hi] and return the index j.
private static int partition(Comparable[] a, int lo, int hi) {
    int i = lo;
    int j = hi + 1;
    Comparable v = a[lo];
```

Partition Example, lo=10, hi=12

A	C	E	E	I	K	L	M	O	P	S	Q	R	T	U	X
0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
										lo		hi			
											i		j		

```
// find greater than (or equal to) v to swap
while (less(a[++i], v)) {
    if (i == hi) break;
}
```

Partition Example, lo=10, hi=12

A	C	E	E	I	K	L	M	O	P	S	Q	R	T	U	X
0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
										lo					
												i	j		

```
// find greater than (or equal to) v to swap
while (less(a[++i], v)) {
    if (i == hi) break;
}
```

Partition Example, lo=10, hi=12

A	C	E	E	I	K	L	M	O	P	S	Q	R	T	U	X
0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
										lo		hi			
												i,j			

```
// find item smaller than (or equal to) v to swap
while (less(v, a[--j])) {
    if (j == lo) break; // redundant since a[lo] acts as sentinel
}
```

Partition Example, lo=10, hi=12

A	C	E	E	I	K	L	M	O	P	S	Q	R	T	U	X
0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
										lo		hi			
												i,j			

```
// check if pointers cross  
if (i >= j) break;
```

Partition Example, lo=10, hi=12

A	C	E	E	I	K	L	M	O	P	S	Q	R	T	U	X
0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
										lo		hi			
												i,j			

```
// put partitioning item v at a[j]
exch(a, lo, j);

// now, a[lo .. j-1] <= a[j] <= a[j+1 .. hi]
return j;
```

Partition Example, lo=10, hi=12

A	C	E	E	I	K	L	M	O	P	R	Q	S	T	U	X
0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
										lo		hi			
												i,j			

```
// put partitioning item v at a[j]
exch(a, lo, j);

// now, a[lo .. j-1] <= a[j] <= a[j+1 .. hi]
return j;
```

Partition for lo=10, hi=12 is complete. j=12

Call sort recursively on left subarray with lo=10, hi = 11

```
private static void sort(Comparable[] a, int lo, int hi) {

    if (hi <= lo) return;
    int j = partition(a, lo, hi);
    sort(a, lo, j-1);
    sort(a, j+1, hi);
}
```

QUICKSORT

Quicksort Trace

```
private static void sort(Comparable[] a, int lo, int hi) {  
    if (hi <= lo) return;  
    int j = partition(a, lo, hi);  
    sort(a, lo, j-1);  
    sort(a, j+1, hi);  
}
```

			lo	j	hi	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
initial values						Q	U	I	C	K	S	O	R	T	E	X	A	M	P	L	E
random shuffle						K	R	A	T	E	L	E	P	U	I	M	Q	C	X	O	S
			0	5	15	E	C	A	I	E	K	L	P	U	T	M	Q	R	X	O	S
			0	3	4	E	C	A	E	I	K	L	P	U	T	M	Q	R	X	O	S
			0	2	2	A	C	E	E	I	K	L	P	U	T	M	Q	R	X	O	S
			0	0	1	A	C	E	E	I	K	L	P	U	T	M	Q	R	X	O	S
			1		1	A	C	E	E	I	K	L	P	U	T	M	Q	R	X	O	S
			4		4	A	C	E	E	I	K	L	P	U	T	M	Q	R	X	O	S
			6	6	15	A	C	E	E	I	K	L	P	U	T	M	Q	R	X	O	S
			7	9	15	A	C	E	E	I	K	L	M	O	P	T	Q	R	X	U	S
			7	7	8	A	C	E	E	I	K	L	M	O	P	T	Q	R	X	U	S
			8		8	A	C	E	E	I	K	L	M	O	P	T	Q	R	X	U	S
			10	13	15	A	C	E	E	I	K	L	M	O	P	S	Q	R	T	U	X
			10	12	12	A	C	E	E	I	K	L	M	O	P	R	Q	S	T	U	X
			10	11	11	A	C	E	E	I	K	L	M	O	P	Q	R	S	T	U	X
			10		10	A	C	E	E	I	K	L	M	O	P	Q	R	S	T	U	X
			14	14	15	A	C	E	E	I	K	L	M	O	P	Q	R	S	T	U	X
			15		15	A	C	E	E	I	K	L	M	O	P	Q	R	S	T	U	X
result						A	C	E	E	I	K	L	M	O	P	Q	R	S	T	U	X

Partition Example, lo=10, hi=11

A	C	E	E	I	K	L	M	O	P	R	Q	S	T	U	X
0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
										lo	hi				
										i	j				

```
// partition the subarray a[lo..hi] so that a[lo..j-1] <= a[j] <= a[j+1..hi] and return the index j.  
private static int partition(Comparable[] a, int lo, int hi) {  
    int i = lo;  
    int j = hi + 1;  
    Comparable v = a[lo];
```

Partition Example, lo=10, hi=11

A	C	E	E	I	K	L	M	O	P	R	Q	S	T	U	X
0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
										lo	hi				
											i	j			

```
// find greater than (or equal to) v to swap
while (less(a[++i], v)) {
    if (i == hi) break;
}
```

Partition Example, lo=10, hi=11

A	C	E	E	I	K	L	M	O	P	R	Q	S	T	U	X
0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
										lo	hi				
										i,j					

```
// find item smaller than (or equal to) v to swap
while (less(v, a[--j])) {
    if (j == lo) break; // redundant since a[lo] acts as sentinel
}
```

Partition Example, lo=10, hi=11

A	C	E	E	I	K	L	M	O	P	R	Q	S	T	U	X
0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
										lo	hi				
										i,j					

```
// check if pointers cross  
if (i >= j) break;
```

Partition Example, lo=10, hi=11

A	C	E	E	I	K	L	M	O	P	R	Q	S	T	U	X
0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
										lo	hi				
										i,j					

```
// put partitioning item v at a[j]
exch(a, lo, j);

// now, a[lo .. j-1] <= a[j] <= a[j+1 .. hi]
return j;
```

Partition Example, lo=10, hi=11

A	C	E	E	I	K	L	M	O	P	Q	R	S	T	U	X
0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
										lo	hi				
										i,j					

```
// put partitioning item v at a[j]
exch(a, lo, j);

// now, a[lo .. j-1] <= a[j] <= a[j+1 .. hi]
return j;
```

Partition for lo=10, hi=11 is complete. j=11

No partition for subarrays of size 1.

Call sort recursively on right subarray with lo=14, hi = 15

```
private static void sort(Comparable[] a, int lo, int hi){

    if (hi <= lo) return;
    int j = partition(a, lo, hi);
    sort(a, lo, j-1);
    sort(a, j+1, hi);
}
```

QUICKSORT

Quicksort Trace

```
private static void sort(Comparable[] a, int lo, int hi) {  
    if (hi <= lo) return;  
    int j = partition(a, lo, hi);  
    sort(a, lo, j-1);  
    sort(a, j+1, hi);  
}
```

			lo	j	hi	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
initial values						Q	U	I	C	K	S	O	R	T	E	X	A	M	P	L	E
random shuffle						K	R	A	T	E	L	E	P	U	I	M	Q	C	X	O	S
			0	5	15	E	C	A	I	E	K	L	P	U	T	M	Q	R	X	O	S
			0	3	4	E	C	A	E	I	K	L	P	U	T	M	Q	R	X	O	S
			0	2	2	A	C	E	E	I	K	L	P	U	T	M	Q	R	X	O	S
			0	0	1	A	C	E	E	I	K	L	P	U	T	M	Q	R	X	O	S
			1		1	A	C	E	E	I	K	L	P	U	T	M	Q	R	X	O	S
			4		4	A	C	E	E	I	K	L	P	U	T	M	Q	R	X	O	S
			6	6	15	A	C	E	E	I	K	L	P	U	T	M	Q	R	X	O	S
			7	9	15	A	C	E	E	I	K	L	M	O	P	T	Q	R	X	U	S
			7	7	8	A	C	E	E	I	K	L	M	O	P	T	Q	R	X	U	S
			8		8	A	C	E	E	I	K	L	M	O	P	T	Q	R	X	U	S
			10	13	15	A	C	E	E	I	K	L	M	O	P	S	Q	R	T	U	X
			10	12	12	A	C	E	E	I	K	L	M	O	P	R	Q	S	T	U	X
			10	11	11	A	C	E	E	I	K	L	M	O	P	Q	R	S	T	U	X
			10		10	A	C	E	E	I	K	L	M	O	P	Q	R	S	T	U	X
			14	14	15	A	C	E	E	I	K	L	M	O	P	Q	R	S	T	U	X
			15		15	A	C	E	E	I	K	L	M	O	P	Q	R	S	T	U	X
result						A	C	E	E	I	K	L	M	O	P	Q	R	S	T	U	X

Partition Example, lo=14, hi=15

A	C	E	E	I	K	L	M	O	P	Q	R	S	T	U	X
0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
													lo	hi	
													i	j	

```
// partition the subarray a[lo..hi] so that a[lo..j-1] <= a[j] <= a[j+1..hi] and return the index j.
private static int partition(Comparable[] a, int lo, int hi) {
    int i = lo;
    int j = hi + 1;
    Comparable v = a[lo];
```

Partition Example, lo=14, hi=15

A	C	E	E	I	K	L	M	O	P	Q	R	S	T	U	X
0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
														lo	hi
														i	j

```
// find greater than (or equal to) v to swap
while (less(a[++i], v)) {
    if (i == hi) break;
}
```

Partition Example, lo=14, hi=15

A	C	E	E	I	K	L	M	O	P	Q	R	S	T	U	X
0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
														lo	hi
														i,j	

```
// find item smaller than (or equal to) v to swap
while (less(v, a[--j])) {
    if (j == lo) break; // redundant since a[lo] acts as sentinel
}
```

Partition Example, lo=14, hi=15

A	C	E	E	I	K	L	M	O	P	Q	R	S	T	U	X
0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
													lo	hi	
													j	i	

```
// find item smaller than (or equal to) v to swap
while (less(v, a[--j])) {
    if (j == lo) break; // redundant since a[lo] acts as sentinel
}
```

Partition Example, lo=14, hi=15

A	C	E	E	I	K	L	M	O	P	Q	R	S	T	U	X
0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
													lo	hi	
													j	i	

```
// put partitioning item v at a[j]
exch(a, lo, j);

// now, a[lo .. j-1] <= a[j] <= a[j+1 .. hi]
return j;
```

Partition Example, lo=14, hi=15

A	C	E	E	I	K	L	M	O	P	Q	R	S	T	U	X
0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
														lo	hi
														j	i

```
// put partitioning item v at a[j]
exch(a, lo, j);

// now, a[lo .. j-1] <= a[j] <= a[j+1 .. hi]
return j;
```

Partition for lo=14, hi=15 is complete. j=14

No partition for subarrays of size 1.

Sorting complete

```
private static void sort(Comparable[] a, int lo, int hi) {

    if (hi <= lo) return;
    int j = partition(a, lo, hi);
    sort(a, lo, j-1);
    sort(a, j+1, hi);
}
```

QUICKSORT

```
private static void sort(Comparable[] a, int lo, int hi) {  
    if (hi <= lo) return;  
    int j = partition(a, lo, hi);  
    sort(a, lo, j-1);  
    sort(a, j+1, hi);  
}
```

Quicksort Trace

	lo	j	hi	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
initial values				Q	U	I	C	K	S	O	R	T	E	X	A	M	P	L	E
				K	R	A	T	E	L	E	P	U	I	M	Q	C	X	O	S
random shuffle	0	5	15	E	C	A	I	E	K	L	P	U	T	M	Q	R	X	O	S
	0	3	4	E	C	A	E	I	K	L	P	U	T	M	Q	R	X	O	S
	0	2	2	A	C	E	E	I	K	L	P	U	T	M	Q	R	X	O	S
	0	0	1	A	C	E	E	I	K	L	P	U	T	M	Q	R	X	O	S
	1		1	A	C	E	E	I	K	L	P	U	T	M	Q	R	X	O	S
	4		4	A	C	E	E	I	K	L	P	U	T	M	Q	R	X	O	S
	6	6	15	A	C	E	E	I	K	L	P	U	T	M	Q	R	X	O	S
	7	9	15	A	C	E	E	I	K	L	M	O	P	T	Q	R	X	U	S
	7	7	8	A	C	E	E	I	K	L	M	O	P	T	Q	R	X	U	S
	8		8	A	C	E	E	I	K	L	M	O	P	T	Q	R	X	U	S
	10	13	15	A	C	E	E	I	K	L	M	O	P	S	Q	R	T	U	X
	10	12	12	A	C	E	E	I	K	L	M	O	P	R	Q	S	T	U	X
	10	11	11	A	C	E	E	I	K	L	M	O	P	Q	R	S	T	U	X
	10		10	A	C	E	E	I	K	L	M	O	P	Q	R	S	T	U	X
	14	14	15	A	C	E	E	I	K	L	M	O	P	Q	R	S	T	U	X
	15		15	A	C	E	E	I	K	L	M	O	P	Q	R	S	T	U	X
result				A	C	E	E	I	K	L	M	O	P	Q	R	S	T	U	X

Great algorithms are better than good ones

- ▶ Your laptop executes 10^8 comparisons per second
- ▶ A supercomputer executes 10^{12} comparisons per second

	Insertion sort			Mergesort			Quicksort		
Computer	Thousand inputs	Million inputs	Billion inputs	Thousand inputs	Million inputs	Billion inputs	Thousand inputs	Million inputs	Billion inputs
Home	Instant	2 hours	300 years	instant	1 sec	15 min	Instant	0.5 sec	10 min
Supercomputer	Instant	1 second	1 week	instant	instant	instant	instant	instant	Instant

Quicksort analysis: best case

- ▶ Quicksort divides everything exactly in half.
- ▶ Similar to merge sort.
- ▶ Number of compares is $\sim n \log n$.

Quicksort analysis: worst case

- ▶ Data are already sorted or we pick the smallest or largest key as pivot.
- ▶ Number of compares is $\sim n^2$ - quadratic!
- ▶ Extremely unlikely (less likely than the probably that your computer is struck by lightning) if we shuffle and our shuffling is not broken.

Things to remember about quick sort

- ▶ 39% more compares than merge sort but in practice it is faster because it does not move data much.
 - ▶ If good implementation, even in sorted arrays it can be linearithmic. If not, we end up with quadratic.
- ▶ $O(n \log n)$ average, $O(n^2)$ worst, in practice faster than mergesort.
- ▶ In-place sorting.
- ▶ **Not** stable.

Quicksort practical improvements

- ▶ Use insertion sort for small subarrays.
- ▶ Best choice of pivot is the median of a small sample.
- ▶ For years, Java used quicksort for collections of primitives and mergesort for collections of objects due to stability.
 - ▶ Has moved to dual-pivot quick sort (Yaroslavskiy, Bentley, and Bloch, 2009) and timsort (Peters, 1993), respectively.

Sorting: the story so far

Which Sort	In place	Stable	Best	Average	Worst	Remarks
Selection	X		$O(n^2)$	$O(n^2)$	$O(n^2)$	n exchanges
Insertion	X	X	$O(n)$	$O(n^2)$	$O(n^2)$	Use for small arrays or partially ordered
Merge		X	$O(n \log n)$	$O(n \log n)$	$O(n \log n)$	Guaranteed performance; stable
Quick	X		$O(n \log n)$	$O(n \log n)$	$O(n^2)$	$n \log n$ probabilistic guarantee; fastest in practice

Sorting based on comparisons

- ▶ All sorting algorithms we have seen so far and we will see in this class are compare-based.
- ▶ No compare-based sorting algorithm can sort n elements in less than $O(n \log n)$ time in the worst case.
 - ▶ Proof and proper notation in CS140.

Lecture 14: Quicksort

- ▶ This week's assignment
- ▶ Quicksort

Readings:

- ▶ Textbook:
 - ▶ Chapter 2.3 (Pages 288-296)
- ▶ Website:
 - ▶ Quicksort: <https://algs4.cs.princeton.edu/23quicksort/>
 - ▶ Code: <https://algs4.cs.princeton.edu/23quicksort/Quick.java.html>

Practice Problems:

- ▶ 2.3.1-2.3.4