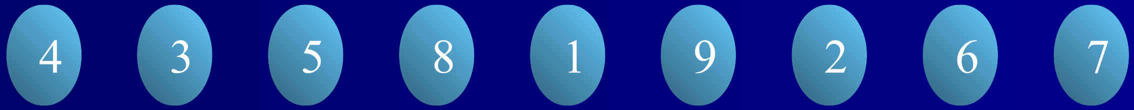
Data Structures and Algorithms (4)
CS C363 / IS C 363

Sorting

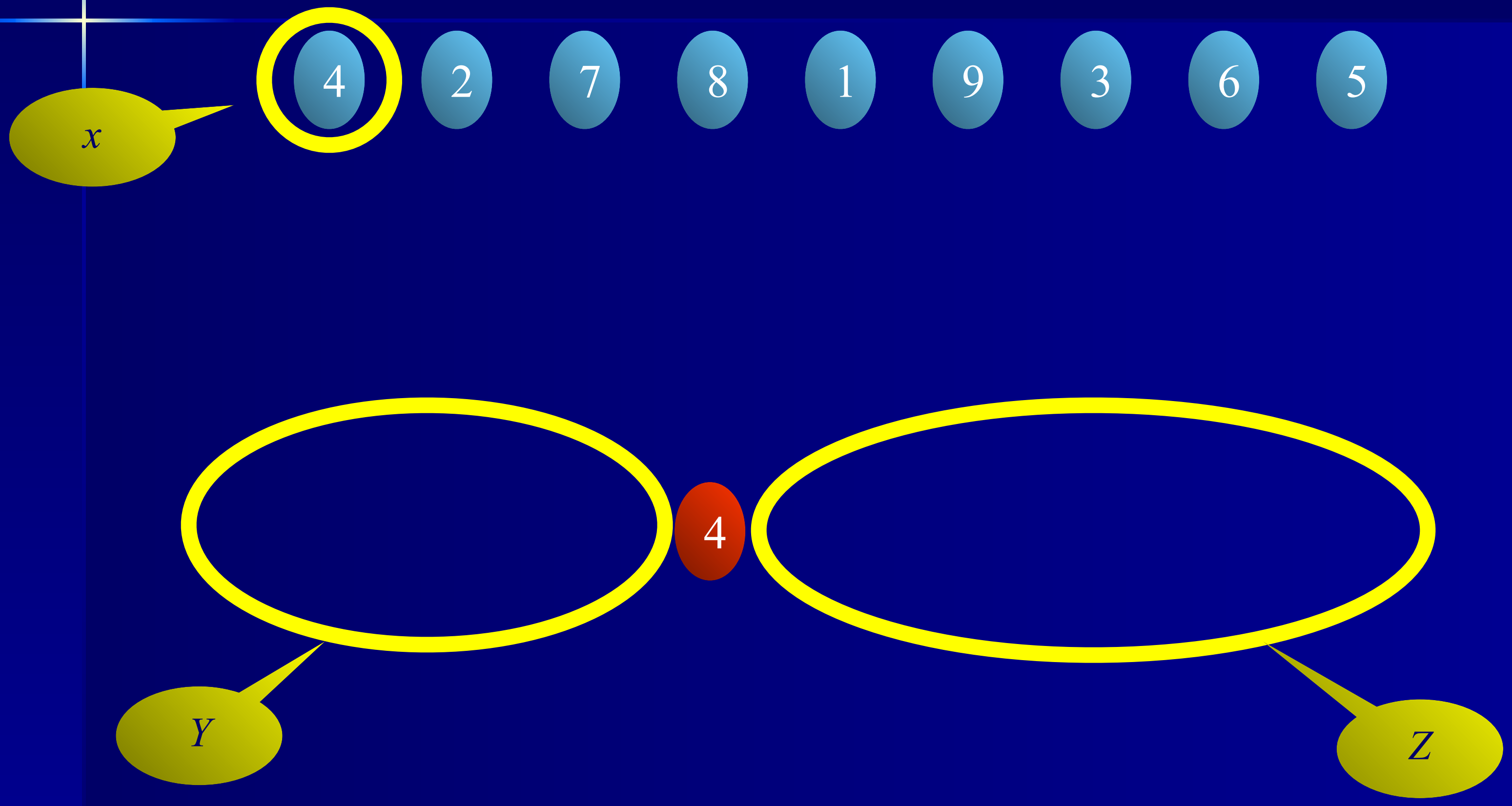
- Input: n distinct numbers $x_1, x_2, \ldots, x_n$.
- Output: a sorted list of $x_1, x_2, \ldots, x_n$.



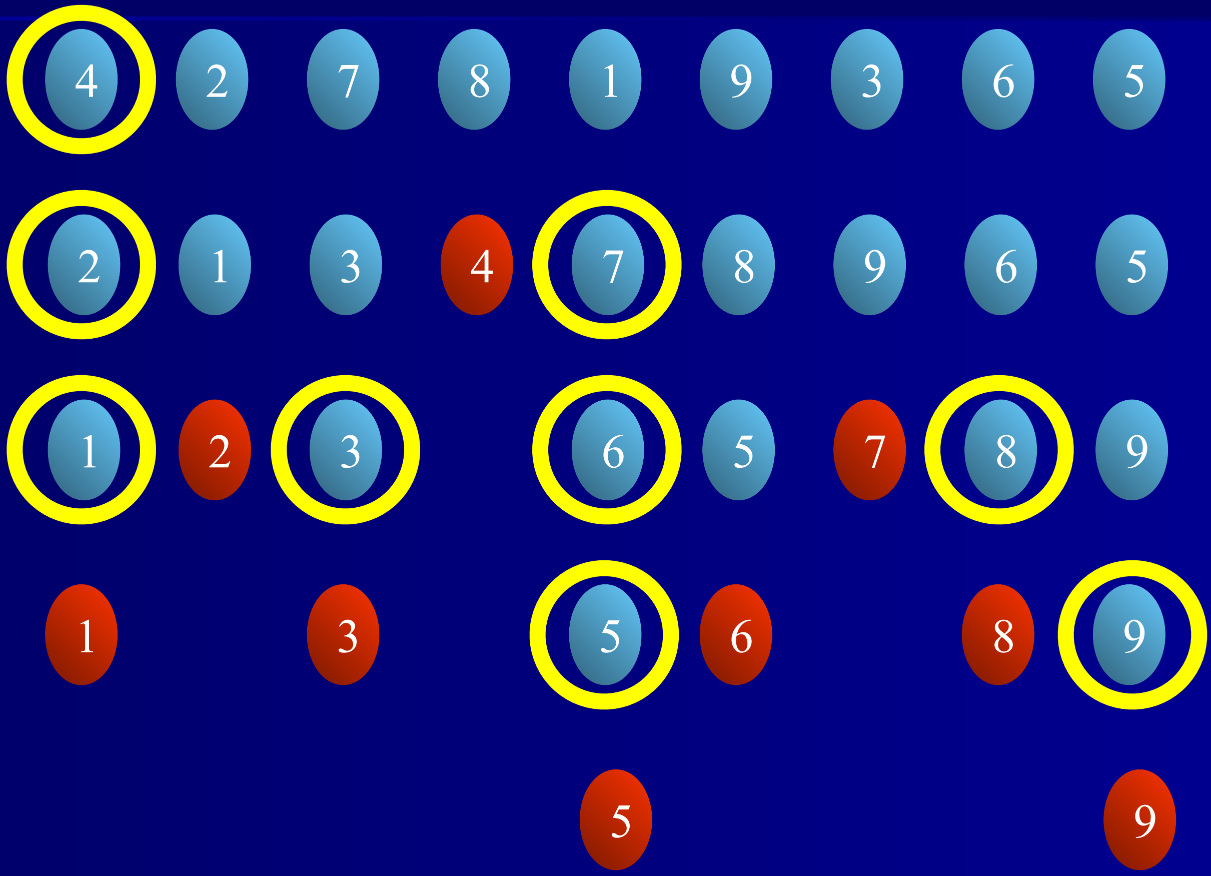
Quick sort: divide & conquer

```
algor hm qu cksor (X ) f
    f X s emp y  hen
      re urn;
    selec  a number x;
    le  Y = f y 2 X j y < xg;
    le  Z = f z 2 X j z > xg;
    call qu cksor (Y );
    pr n  x;
    call qu cksor (Z );
g
```

x divides X into Y and Z.



The whole process

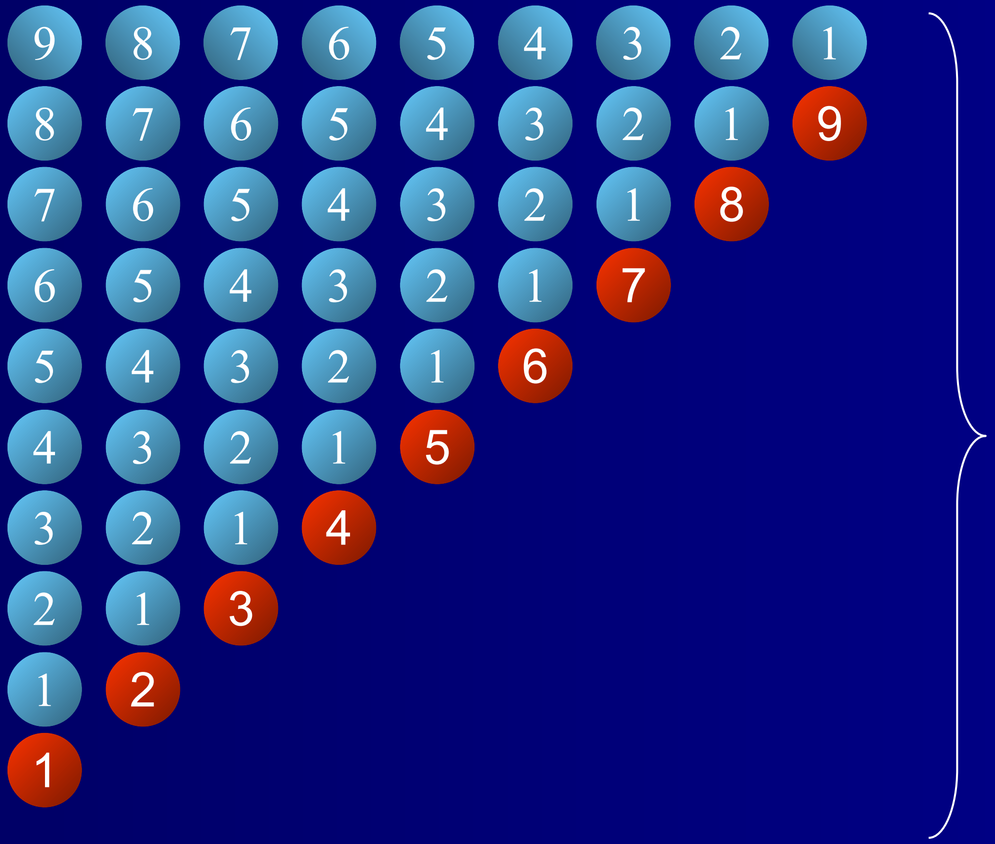


Efficiency depends on …

```
algorithm quicksort (X ) f
   if X is empty then
      return;
   select a number x;
   let Y = f y 2 X j y < xg;
   le Z = f z 2 X j z > xg;
   call quicksor (Y );
   prin x;
   call qu cksor (Z);
g
```

Critical step

A bad case

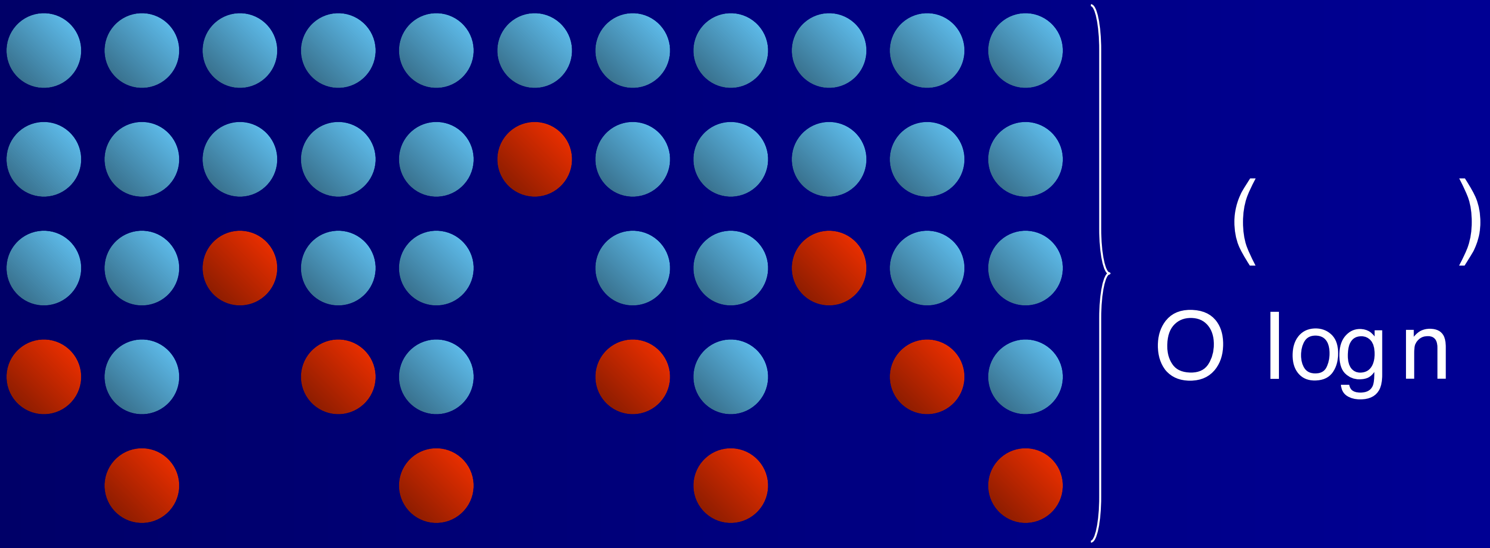

9 8 7 6 5 4 3 2 1
8 7 6 5 4 3 2 1 9
7 6 5 4 3 2 1 8
6 5 4 3 2 1 7
5 4 3 2 1 6
4 3 2 1 5
3 2 1 4
2 1 3
1 2
1

( )

- n

A good case

- Each x divides X evenly into Y and Z, i.e., when $-1 \cdot |Y| - |Z| \cdot 1$.
- The running time is $O(n \log n)$.



Quick Sort, an implementation

```
quicksort( void *a, int low, int high )
    {
    int pivot;
    /* Termination condition! */
    if ( high > low )
      {
      pivot = partition( a, low, high );
      quicksort( a, low, pivot-1 );
      quicksort( a, pivot+1, high );
      }
    }
```

Divide

Conquer

Quick Sort - Partition

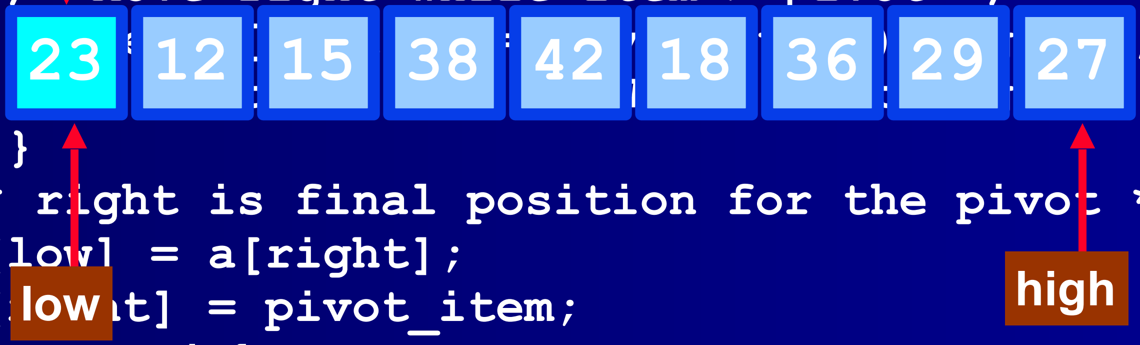
```
int partition( int *a, int low, int high ) {
   int left, right;
   int pivot_item;
   pivot_item = a[low];
   pivot = left = low;
   right = high;
   while ( left < right ) {
     /* Move left while item < pivot */
     while( a[left] <= pivot_item ) left++;
     /* Move right while item > pivot */
     while( a[right] >= pivot_item ) right--;
     if ( left < right ) SWAP(a,left,right);
     }
   /* right is final position for the pivot */
   a[low] = a[right];
   a[right] = pivot_item;
   return right;
   }
```

Quicksort - Partition

```
int partition( int *a, int low, int high ) {
    int left, right;
    int pivot_item;
    pivot_item = a[low];
    pivot = left = low;
    right = high;
    while ( left < right ) {
      /* Move left while item < pivot */
      while( a[left] <= pivot_item ) left++;
      /* Move right while item > pivot */
                                              ;
      }
    /* right is final position for the pivot */
    a[low] = a[right];
    a[right] = pivot_item;
    return right;
    }
```

This example uses `int`'s to keep things simple!

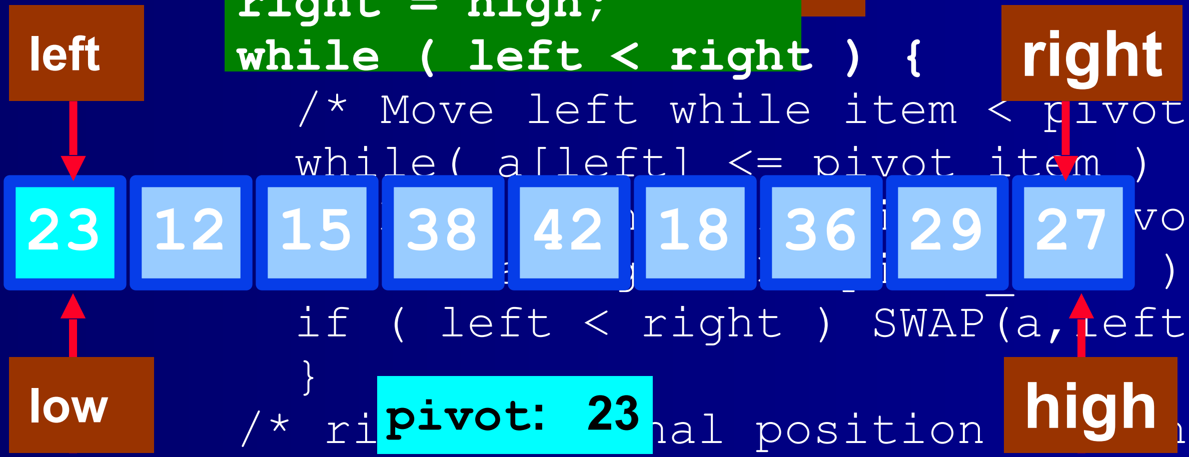
Any item will do as the pivot, choose the leftmost one!



Quick Sort - Partition

```
int partition( int *a, int low, int high ) {
   int left, right;
   int pivot_item;
   pivot_item = a[low];
   pivot = left = low;
   right = high;
   while ( left < right ) {
     /* Move left while item < pivot */
     while( a[left] <= pivot_item ) left++;
                                             vot */
                                              ) right--;
     if ( left < right ) SWAP(a,left,right);
     }
   /* ri          nal position          e pivot */
   a[low] = a[right];
   a[right] = pivot_item;
   return right;
   }
```

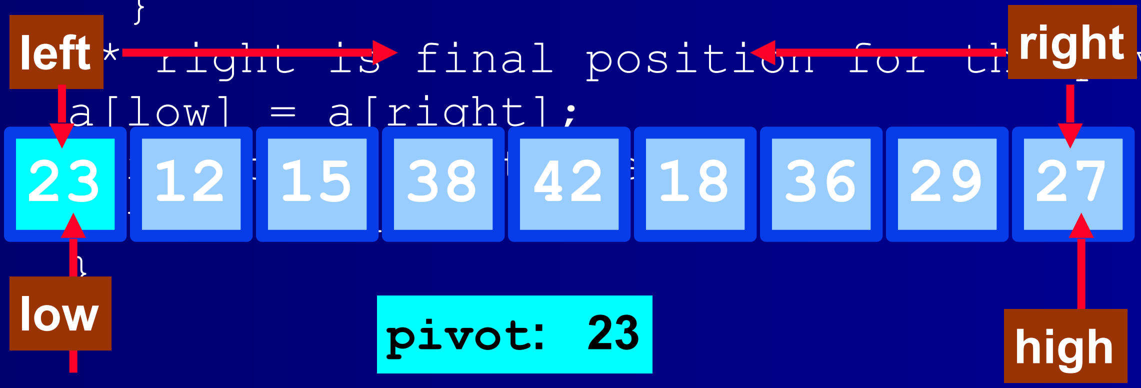
Set left and right markers



Quick Sort - Partition

```
int partition( int *a, int low, int high ) {
    int left, right;
    int pivot_item;
    pivot_item = a[low];
    pivot = left = low;
    right = high;
    while ( left < right ) {
      /* Move left while item < pivot */
      while( a[left] <= pivot_item ) left++;
      /* Move right while item > pivot */
      while( a[right] >= pivot_item ) right--;
      if ( left < right ) SWAP(a,left,right);
      }
    /* right is final position for the pivot */
    a[low] = a[right];
```

Move the markers until they cross over

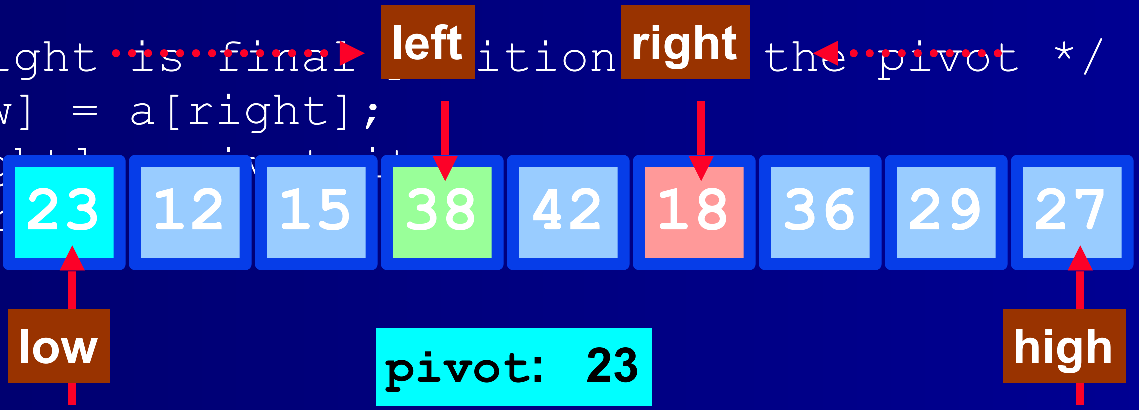


Quick Sort - Partition

```
int partition( int *a, int low, int high ) {
    int left, right;
    int pivot_item;
    pivot_item = a[low];
    pivot = left = low;
    right = high;
    while ( left < right ) {
      /* Move left while item < pivot */
      while( a[left] <= pivot_item ) left++;
      /* Move right while item > pivot */
      while( a[right] >= pivot_item ) right--;
      if ( left < right ) SWAP(a,left,right);
      }
    /* right is final      ition      the pivot */
    a[low] = a[right];
    a[rig
    retur
    }
```

Move the left pointer while it points to items <= pivot

Move right similarly

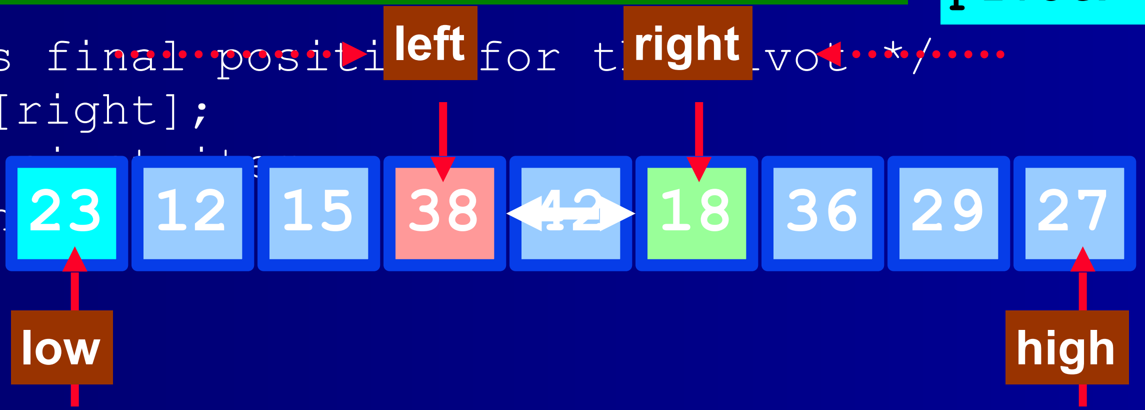


Quick Sort - Partition

```
int partition( int *a, int low, int high ) {
    int left, right;
    int pivot_item;
    pivot_item = a[low];
    pivot = left = low;
    right = high;
    while ( left < right ) {
      /* Move left while item < pivot */
      while( a[left] <= pivot_item ) left++;
      /* Move right while item > pivot */
      while( a[right] >= pivot_item ) right--;
      if ( left < right ) SWAP(a,left,right);
      }
    /* right is final positi  for t      vot */
    a[low] = a[right];
    a[right] =
    return righ
    }
```

Swap the two items on the wrong side of the pivot

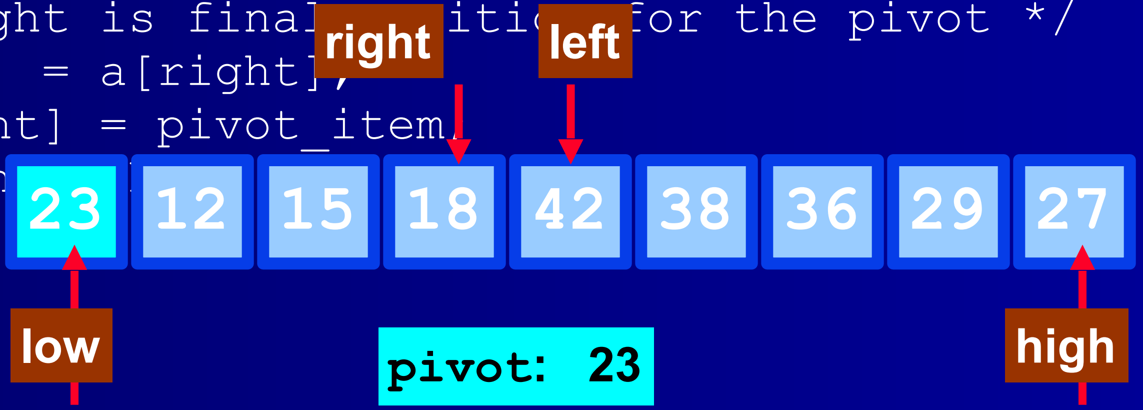
pivot: 23



Quick Sort - Partition

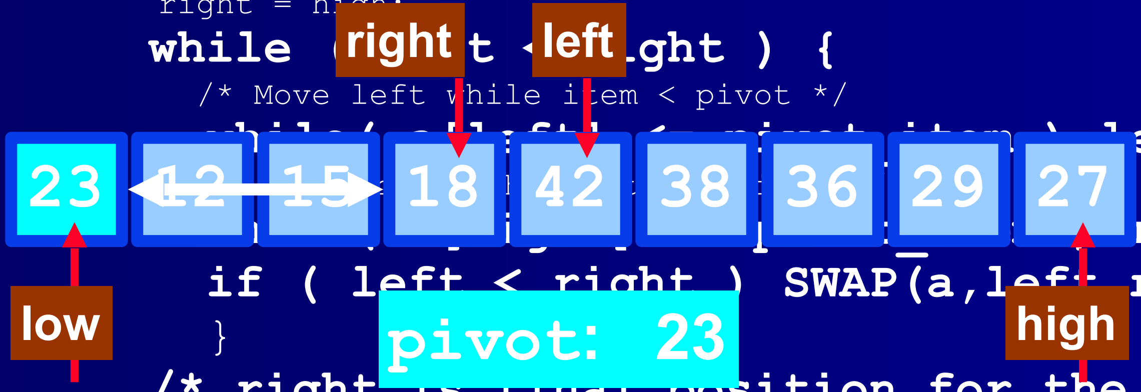
```
int partition( int *a, int low, int high ) {
    int left, right;
    int pivot_item;
    pivot_item = a[low];
    pivot = left = low;
    right = high;
    while ( left < right ) {
      /* Move left while item < pivot */
      while( a[left] <= pivot_item ) left++;
      /* Move right while item > pivot */
      while( a[right] >= pivot_item ) right--;
      if ( left < right ) SWAP(a,left,right);
      }
    /* right is fina     itio     for the pivot */
    a[low] = a[right];
    a[right] = pivot_item;
    return
    }
```

left and right have swapped over, so stop



Quick Sort - Partition

```
int partition( int *a, int low, int high ) {
    int left, right;
    int pivot_item;
    pivot_item = a[low];
    pivot = left = low;
    right = high;
    while ( left < right ) {
      /* Move left while item < pivot */
      while( a[left] <= pivot_item ) left++;
      /* Move right while item > pivot */
      while( a[right] > pivot_item ) right--;
      if ( left < right ) SWAP(a,left,right);
      }
    /* right is final position for the pivot */
    a[low] = a[right];
    a[right] = pivot_item;
    return right;
    }
```

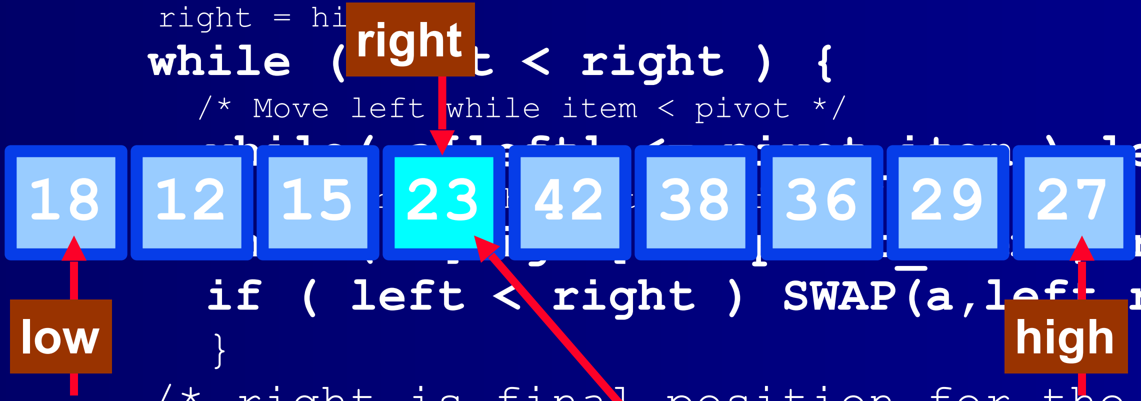


Finally, swap the pivot and right

Quick Sort - Partition

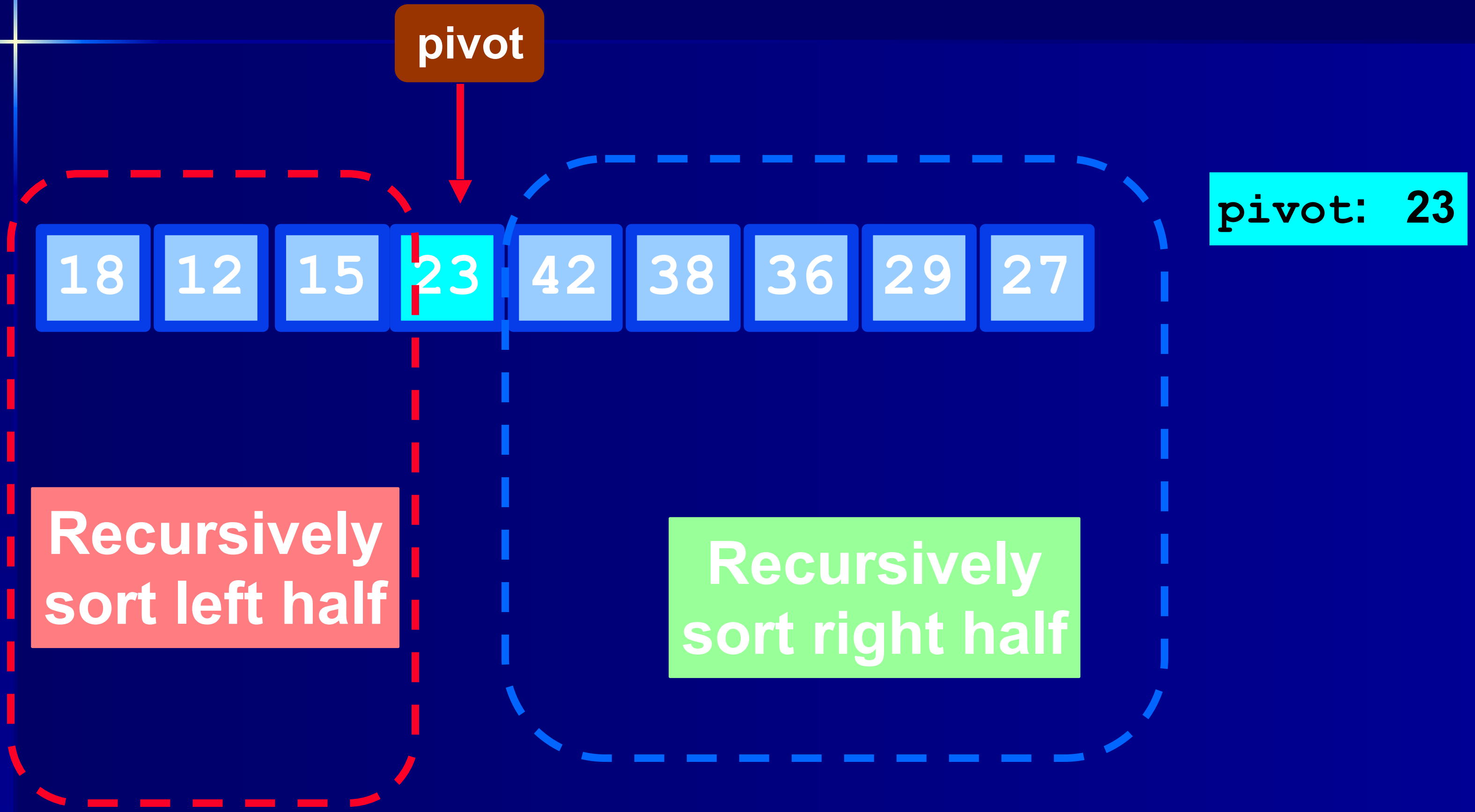
```
int partition( int *a, int low, int high ) {
    int left, right;
    int pivot_item;
    pivot_item = a[low];
    pivot = left = low;
    right = hi
    while ( t < right ) {
      /* Move left while item < pivot */
                                          left++;
                                          right--;
      if ( left < right ) SWAP(a,le   right);
      }
    /* right is final position for the pivot */
    a[low] = a[right];
    a[right] = pivot_
    return right;
    }
```

pivot: 23



Return the position of the pivot

Quick Sort - Conquer


pivot
18
12
15
23
42
38
36
29
27
pivot: 23
Recursively sort left half
Recursively sort right half