Fundamentals of Computing: Lecture 20

Piyush P Kurur
Office no: 224
Dept. of Comp. Sci. and Engg.
IIT Kanpur

September 14, 2009

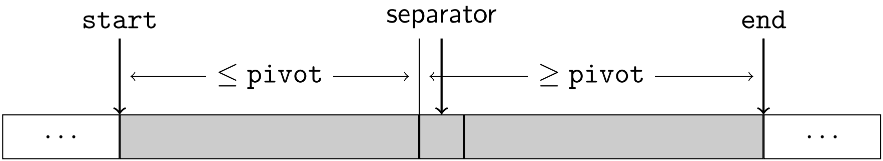
Quick sort

- Choose a pivot element x
- Divide the rest of elements into two groups A and B such that
    - A consists of all elements less than x.
    - B consists of all elements greater than or equal to x.
- Sort A and B recursively and then the arange them in the order A, x, B.

The function partition

```
int partition(int a[], int start, int end, int pivotIndex)
```

- Takes as input an array slice {a[start], . . . , a[end - 1]},
- Chooses pivot = a[pivotIndex] as the pivot element and,
- Rearranges the elements and returns seperator such that
    - The slice {a[start], . . . a[sperator-1]} contains elements less than or equal to pivot and,
    - {a[seperator], . . . , a[end - 1]} contains elements greater than equal to pivot.



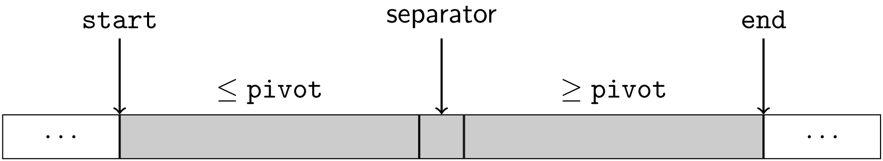
The Quick sort algorithm

```
void qsort(int a[], int start, int end);
```

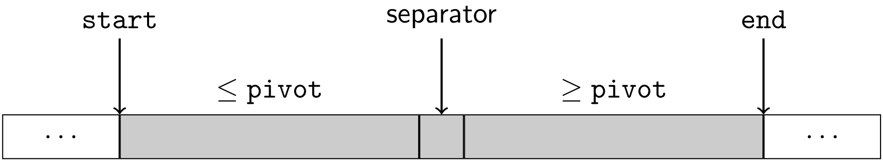
- Takes as input the array slice {a[start],...,a[end -1]}.
- Rearranges the input in sorted order.

```
void qsort(int *a , int start, int end)
{
  int seperator;
  if( start >= end - 1) return;

  seperator = partition(a, start, end, start);

  qsort(a, start, seperator);
  qsort(a, seperator , end);
}
```

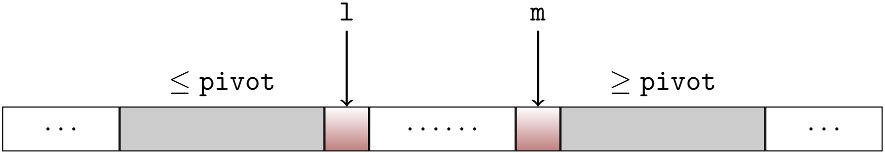
Designing the function `partition`



Designing the function `partition`

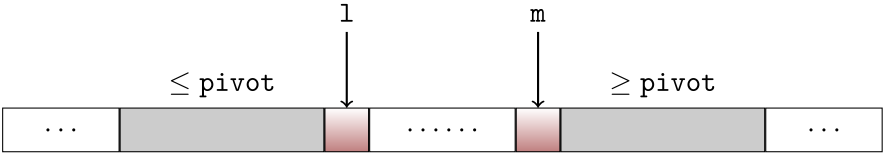


The invariant



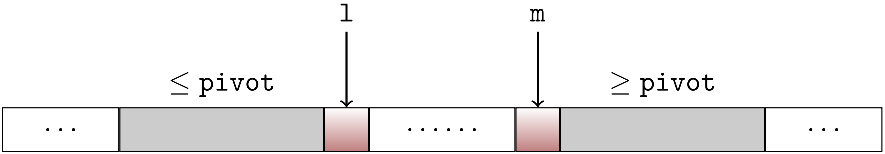
The invariant

- Elements `a[start]` to `a[l-1]` are less than or equal to `pivot` and
- Elements `a[m+1]` to `a[end - 1]` are greater than or equal to `pivot`.

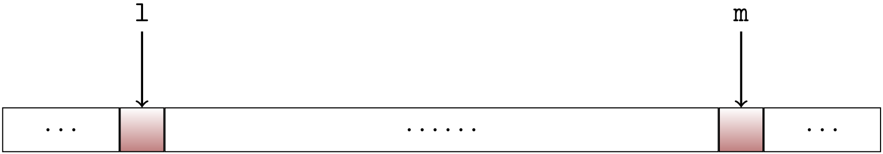


The invariant

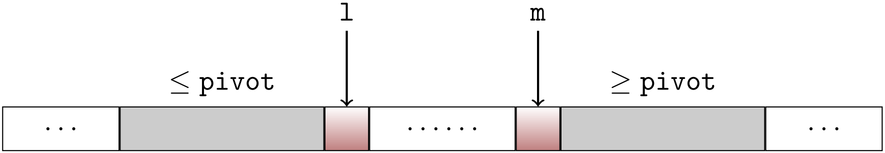
- Elements `a[start]` to `a[l-1]` are less than or equal to `pivot` and
- Elements `a[m+1]` to `a[end - 1]` are greater than or equal to `pivot`.



To begin with



If `a[l] <= pivot`



l

m

$\leq$ pivot

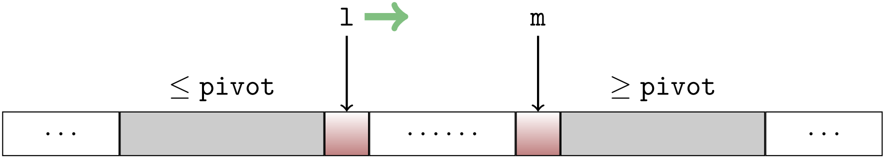
$\geq$ pivot

. . .

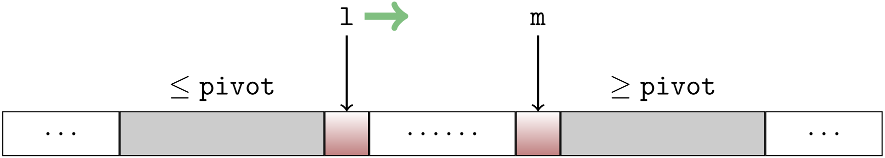
. . . . . .

. . .

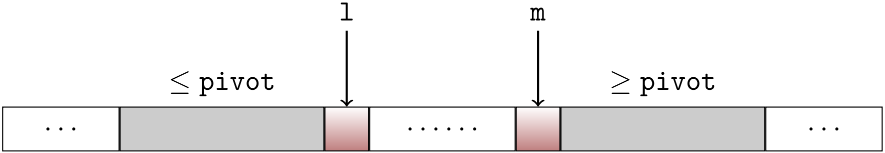
If `a[l] <= pivot`



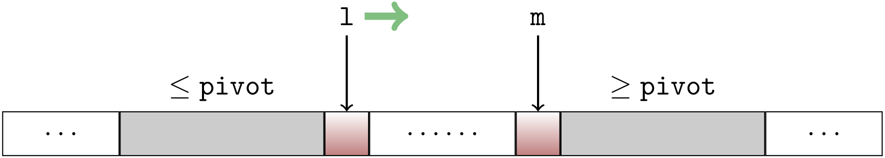
If `a[l] <= pivot`



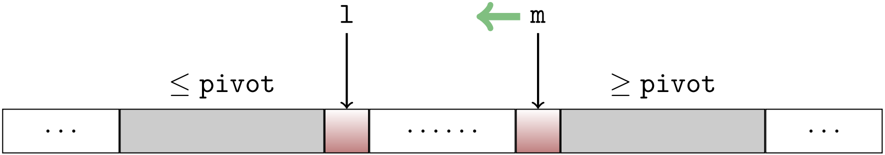
Similarly if `a[m] >= pivot`



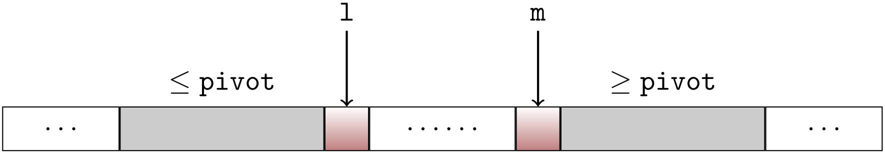
If a[l] <= pivot



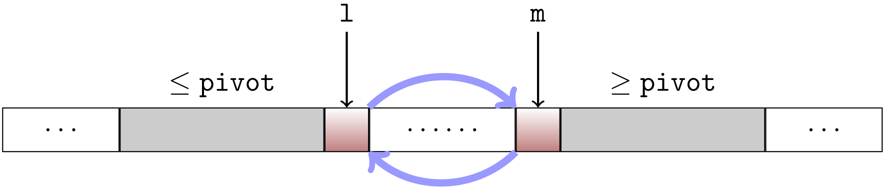
Similarly if a[m] >= pivot



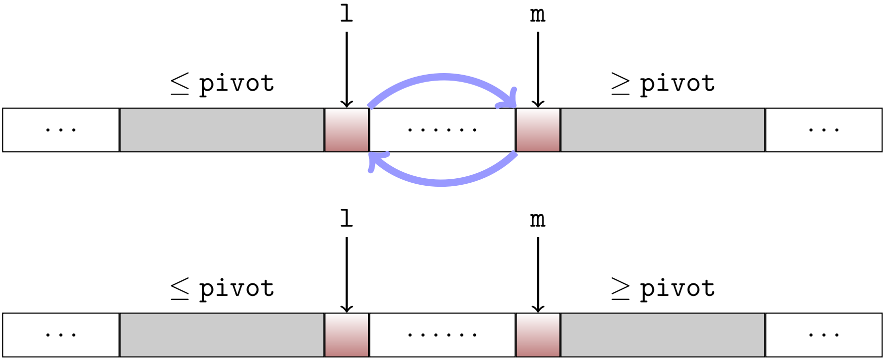
What when a[l] > pivot and a[m] < pivot



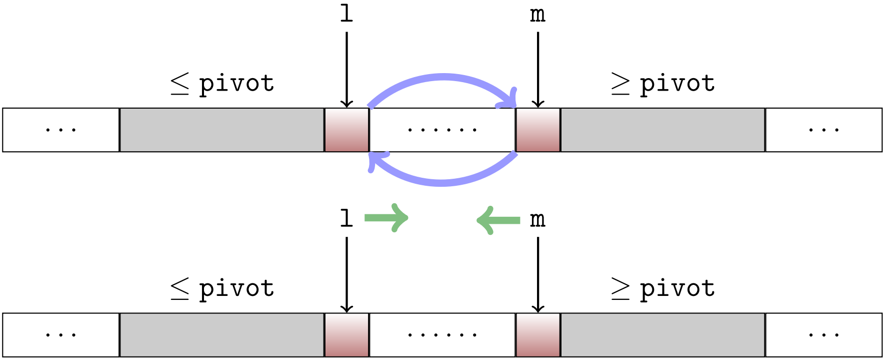
What when a[l] > pivot and a[m] < pivot



What when a[l] > pivot and a[m] < pivot



What when a[l] > pivot and a[m] < pivot



The function partition contd

```
int partition(int *a, int start, int end, int pivotIndex)
{
  int l = start;
  int m = end - 1;
  int pivot = a[pivotIndex];

  while(l < m )
  {
    if( a[l] <= pivot ) {l++; continue;}
    if( a[m] >= pivot ) {m--; continue;}

    /* Here a[l] > pivot && a[m] < pivot */

    swap(a, l, m);
  }
  return l;
}
```