CSCI 241

Scott Wehrwein

Quick Sort: Algorithm

Goals

Thoroughly understand the mechanism of quicksort.

Be able to execute quicksort on paper.

Be prepared to implement quicksort and its partition helper method.

Mergesort vs Quicksort

```
/** mergesort A[st..end]*/
mergeSort(A, st, end):
  if (small):
    return

  mid = (end+st)/2

  mergeSort(A,st, mid)
  mergeSort(A,mid,end)

  merge(A, st, mid, end)
```

Divide (Quick: the "real work" happens here)

Conquer

Combine (Merge: the "real work" happens here)

Mergesort vs Quicksort

```
/** mergesort A[st..end]*/
mergeSort(A, st, end):
  if (small):
    return

  mid = (end+st)/2

  mergeSort(A,st, mid)
  mergeSort(A,mid,end)

  merge(A, st, mid, end)
```

Divide

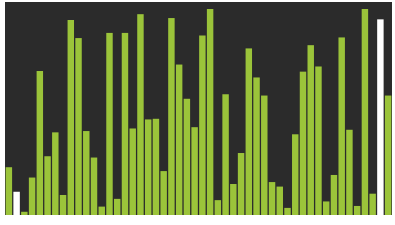
Conquer

Combine

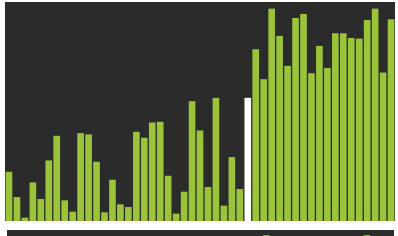
```
/** quicksort A[st..end]*/
quickSort(A, st, end):
  if (small):
    return

  mid = partition(A,st,end)

  quickSort(A,st,mid)
  quickSort(A,mid+1,end)

  # (nothing to do!)
```

Quicksort: Algorithm

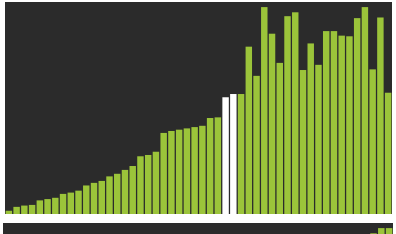
Unsorted:



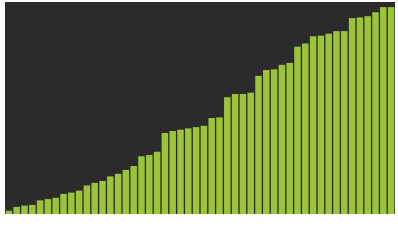
Small things left
big things right:



Sort left things:



Sort right things:



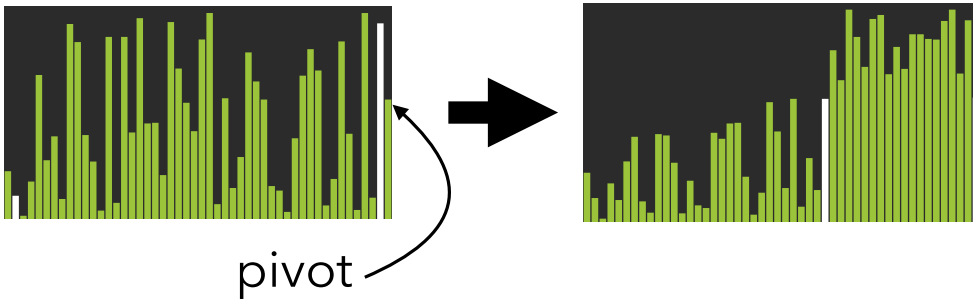
```
/** quicksort A[st..end]*/
quickSort(A, st, end):
  if (small):
    return
```

← `mid = partition(A,st,end)`

← `quickSort(A,st,mid)`

← `quickSort(A,mid+1, end)`

Quicksort: Partition



partition: choose a pivot, then:

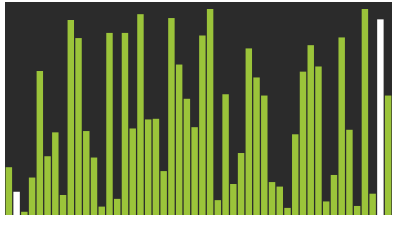
- elements <= pivot go to its left
- elements > pivot to the right

What pivot should we choose?

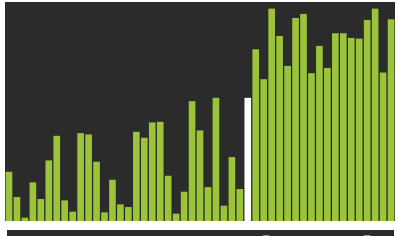
- First, middle, or last
- Median of first, middle, and last

Quicksort: Algorithm

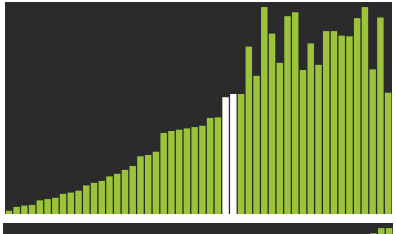
Unsorted:



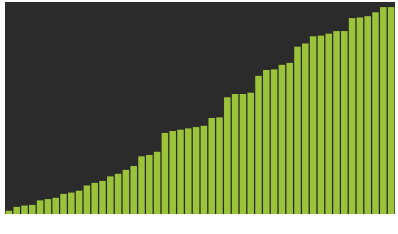
Small things left
big things right:



Sort left things:



Sort right things:



```
/** quicksort A[st..end]*/
quickSort(A, st, end):
  if (small):
    return
```

← `mid = partition(A,st,end)`

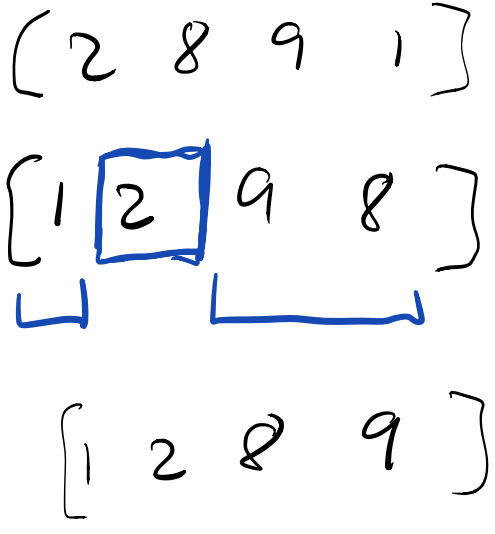
← `quickSort(A,st,mid)`

← `quickSort(A,mid+1, end)`

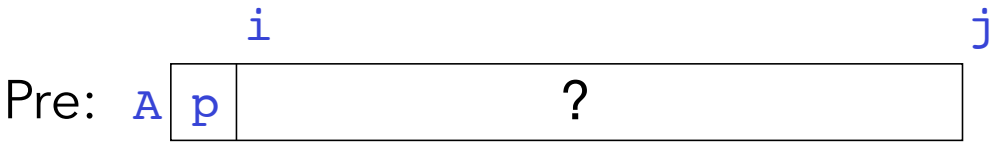
Tiny Example

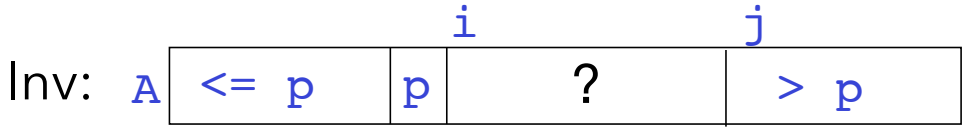
`quickSort([2 8 9 1])`

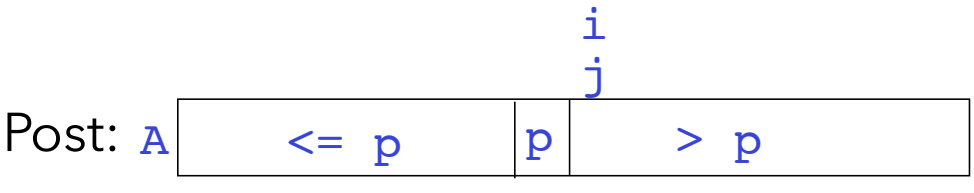
For now: choose the first element as the pivot.



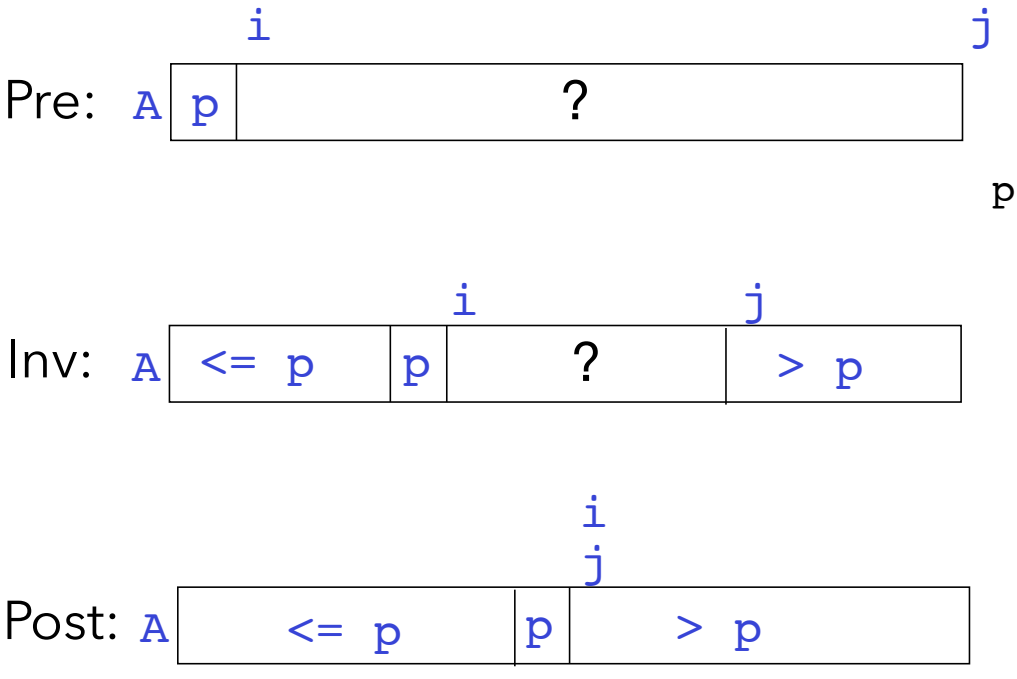
Quicksort: Partition







Quicksort: Partition




```
partition(A, start, end)
  initialize i, j
  choose pivot
  swap pivot to A[0]
  while [?] section != []
    # process A[i]:
    if <= p:
      grow <= p section
    else:
      grow > p section
```

Note: this is just one of several ways to implement partition.