CSCI 241

Scott Wehrwein

Merge Sort: Algorithm

Goals

Thoroughly understand the mechanism of mergesort.

Be able to execute mergesort on paper.

Understanding Mergesort

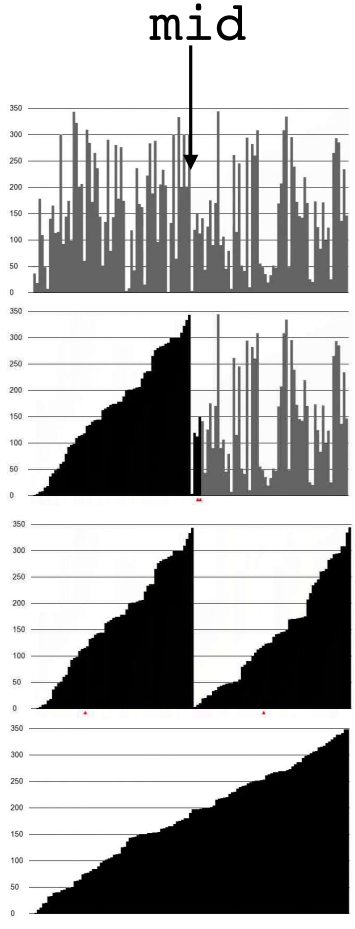
```
/** sort A[start..end] using mergesort */
mergeSort(A, start, end):
  if (end-start < 2):
    return
  mid = (end+start)/2

  mergeSort(A,start,mid)

  mergeSort(A,mid,end)

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

Divide

Conquer (left)

Conquer (right)

Combine



Understanding Mergesort

```
/** sort A[start..end] using mergesort */
mergeSort(A, start, end):
  if (end-start < 2):
    return
  mid = (end+start)/2

  mergeSort(A,start,mid)

  mergeSort(A,mid,end)

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

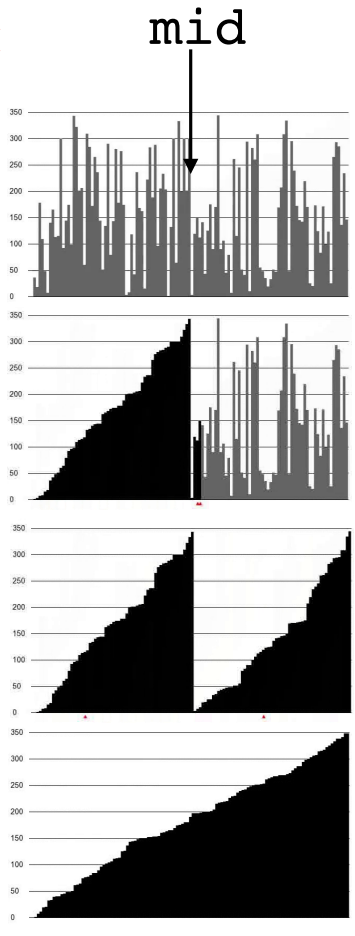
1. Spec

Divide

Conquer (left)

Conquer (right)

Combine



Understanding Mergesort

```
/** sort A[start..end] using mergesort */
mergeSort(A, start, end):
  if (end-start < 2):
    return
  mid = (end+start)/2

  mergeSort(A,start,mid)

  mergeSort(A,mid,end)

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

1. Spec

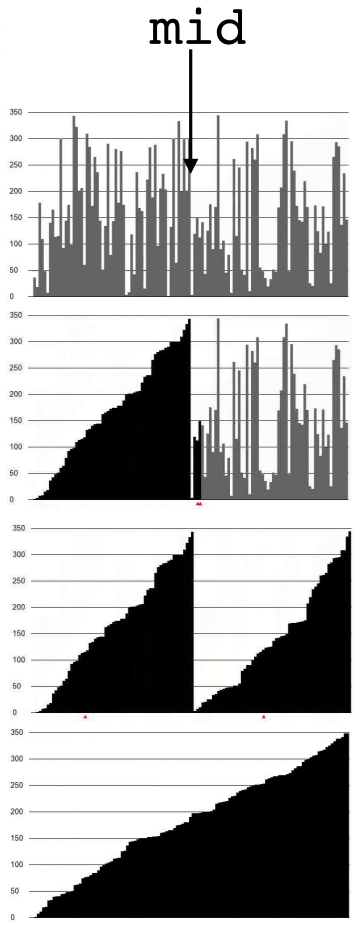
2. Base case

Divide

Conquer (left)

Conquer (right)

Combine



Understanding Mergesort

```
/** sort A[start..end] using mergesort */
mergeSort(A, start, end):
  if (end-start < 2):
    return
  mid = (end+start)/2

  mergeSort(A,start,mid)

  mergeSort(A,mid,end)

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

1. Spec

2. Base case

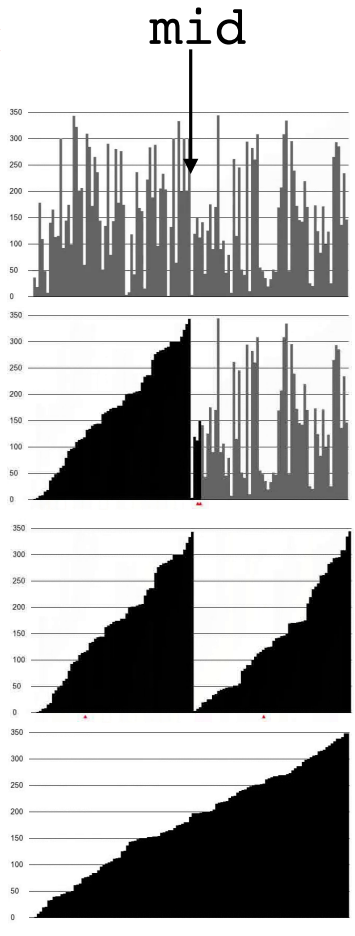
3. Progress

Divide

Conquer (left)

Conquer (right)

Combine



Understanding Mergesort

```
/** sort A[start..end] using mergesort */
mergeSort(A, start, end):
  if (end-start < 2):
    return
  mid = (end+start)/2

  /* sort A[start..mid] */

  /* sort A[mid..end]
```

1. Spec

2. Base case

Divide

Conquer (left)

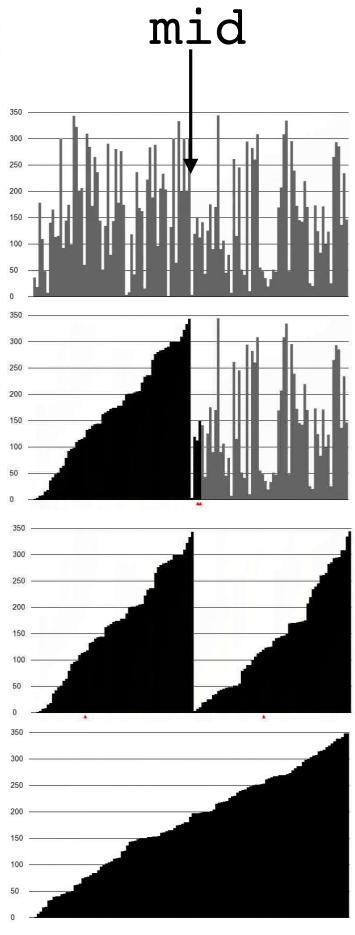
3. Progress

Conquer (right)

4. Replace recursive calls with spec

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

Combine



Merging two sorted arrays

Tiny Example

```
mergeSort([2 8 9 1])
```