CSCI 241

Lecture 5
Recursive Sorting:
Mergesort and Quicksort

Announcements

- First programming assignment (A1) out today.
- Mentor hours: 4-7pm, CF162/164

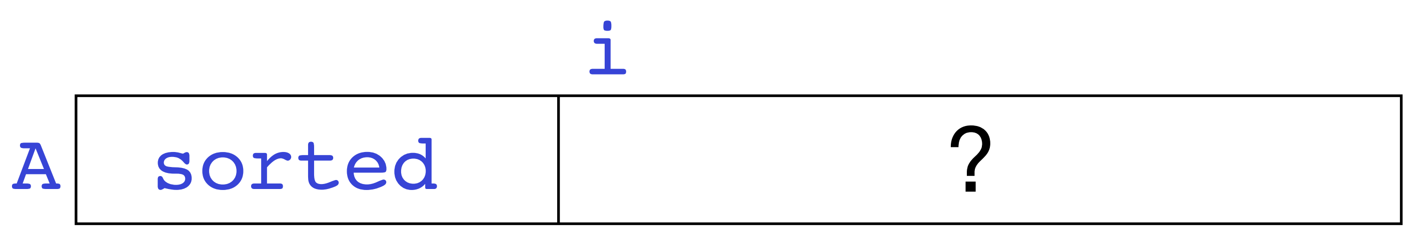
Goals:

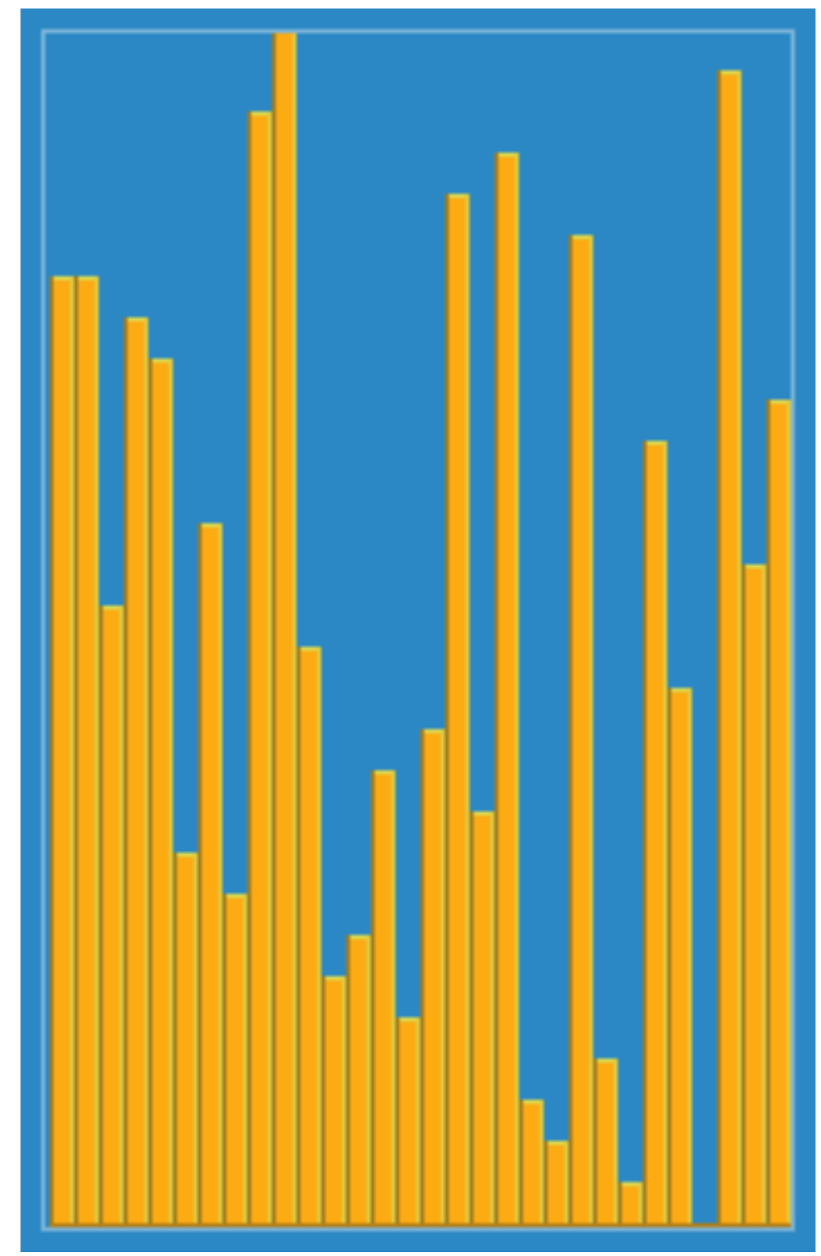
- Know the generic steps of a divide-and-conquer algorithm.
- Thoroughly understand the mechanism of mergesort and quicksort.
- Be prepared to implement **merge** and **partition** helper methods.

Incremental Algorithms

solve a problem a little bit at a time.

Natural programming mechanism: loops





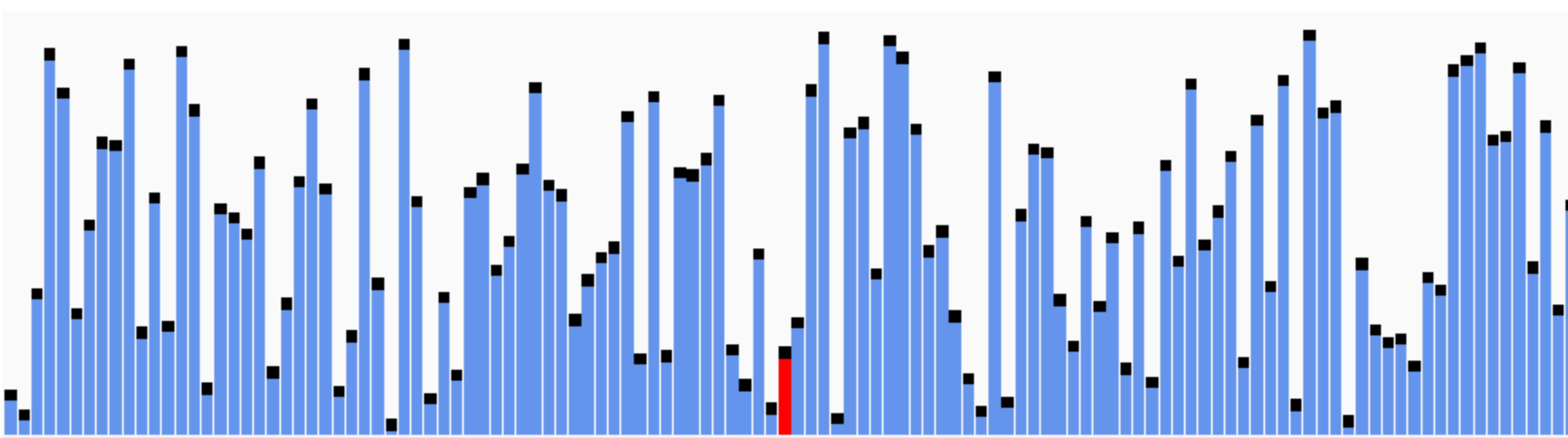
insertion sort

Divide-and-Conquer Algorithms

solve a problem by breaking it into smaller problems.

(easier!)

Natural programming mechanism: recursion



https://upload.wikimedia.org/wikipedia/commons/f/fe/Quicksort.gif

Divide-and-Conquer Algorithms

solve a problem by breaking it into smaller problems.

Natural programming mechanism: recursion

Three generic steps:

1. **Divide (into sub-problems)**
2. **Conquer (the sub-problems)**
3. **Combine (into a solution to the original problem)**

Divide-and-Conquer Algorithms

solve a problem by breaking it into smaller problems.

Natural programming mechanism: recursion

Three generic steps:

1. **Divide (into sub-problems)**
2. **Conquer (the sub-problems)**
3. **Combine (into a solution to the original problem)**

Why are we talking about divide-and-conquer, I thought we were learning how to sort things?

An example of Divide-and-Conquer

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

  mergeSort(A,start,mid)
  mergeSort(A,mid, end)

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

1. **Divide**
2. **Conquer**
3. **Combine**

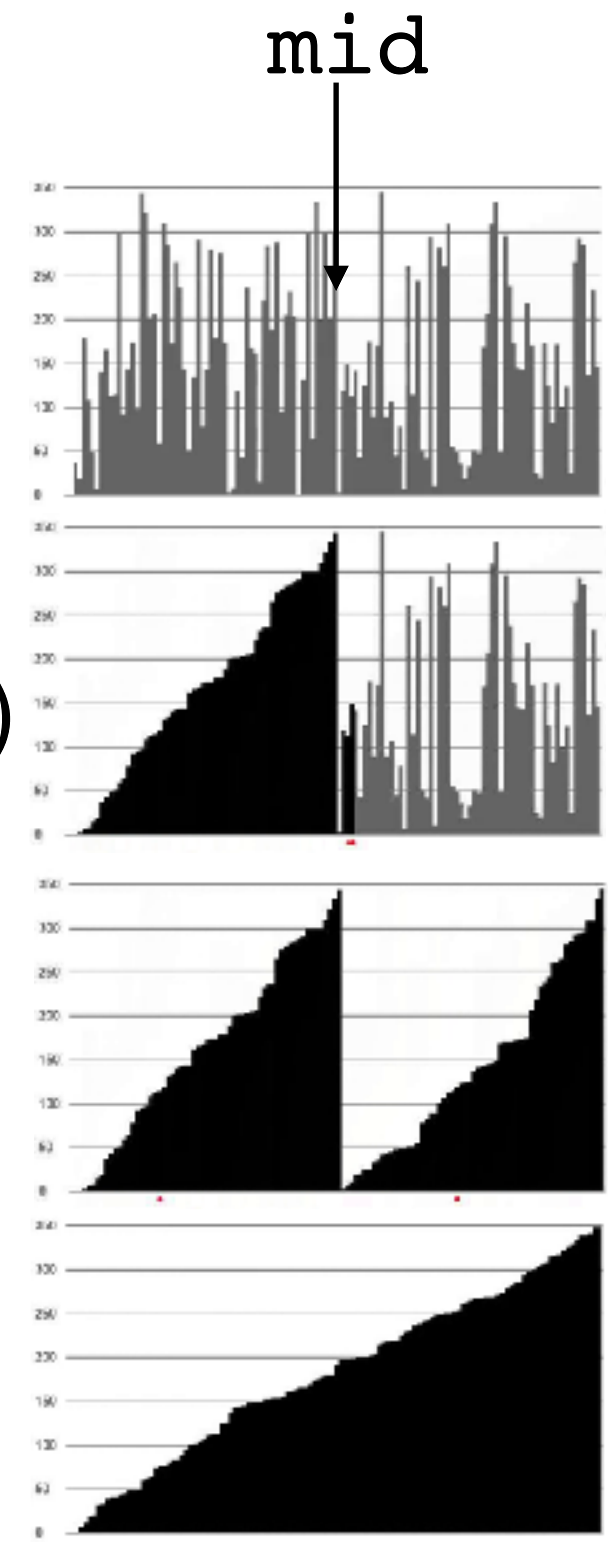
```
/** sort A[start..end] using mergesort */
mergeSort(A, start, end):
  if (A.length < 2):
    return
  mid = (end-start)/2

  mergeSort(A,start,mid)

  mergeSort(A,mid, end)

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

Divide

Conquer (left)

Conquer (right)

Combine



1. Spec

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

2. Base case

Divide

3. Progress

```
  mergeSort(A,start,mid)
```

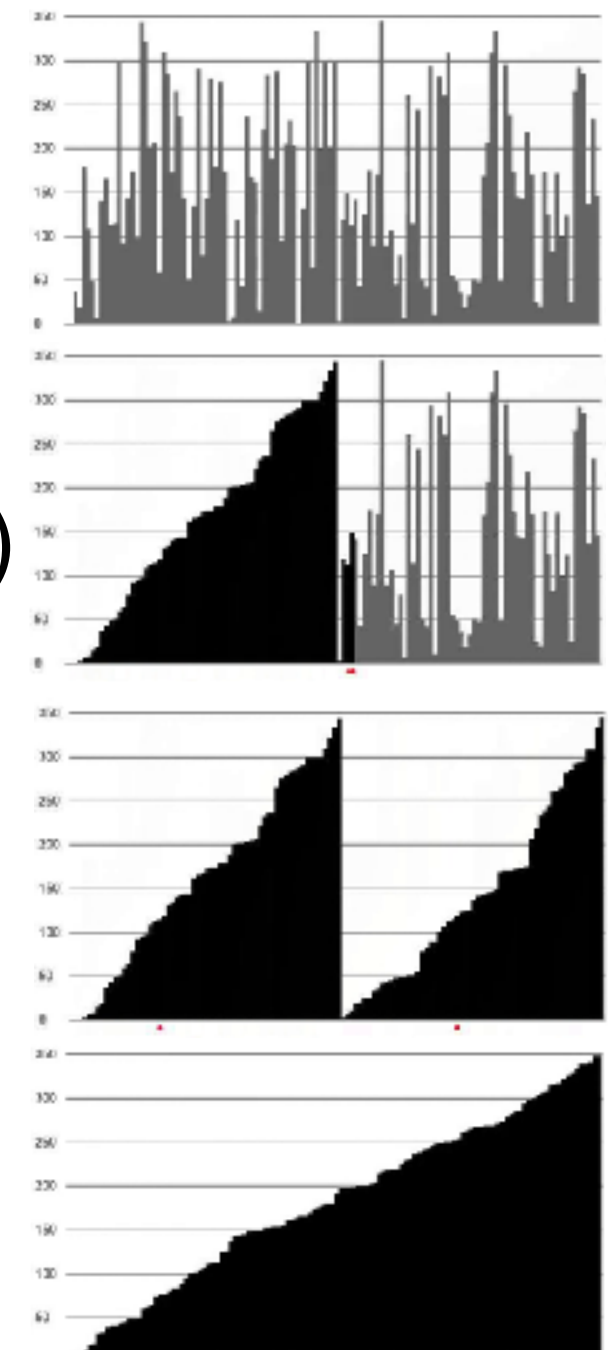
Conquer (left)

```
  mergeSort(A,mid, end)
```

Conquer (right)

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

Combine



1. Spec

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

2. Base case

Divide

3. Progress

```
  sort A[start..mid]
```

Conquer (left)

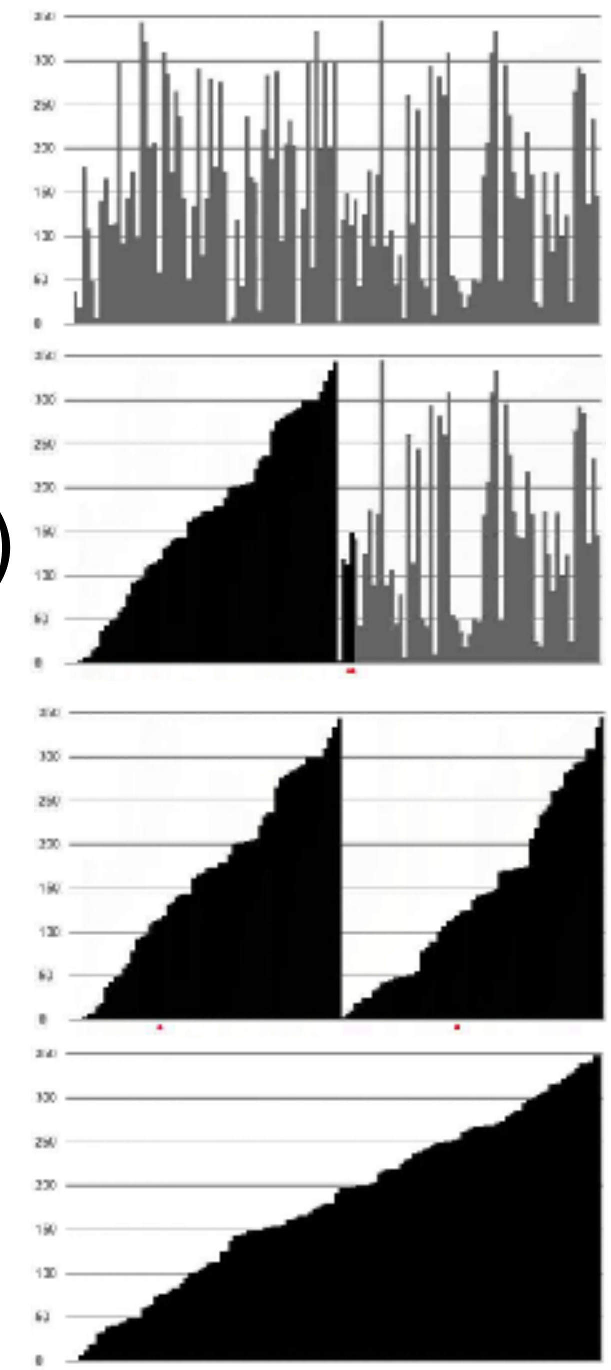
4. Replace recursive calls with spec

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

Conquer (right)

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

Combine



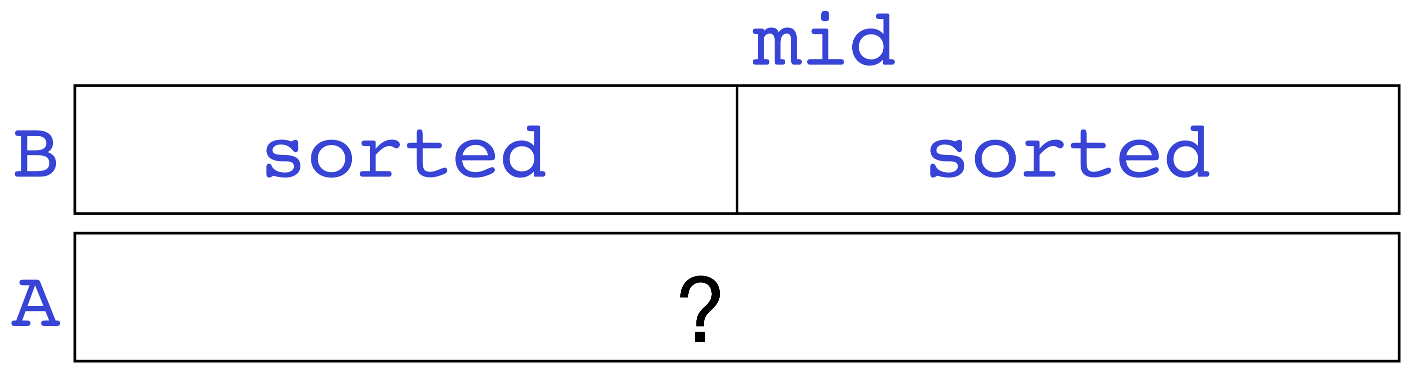
Merge Step

- Merge two halves, each of which is **sorted**.

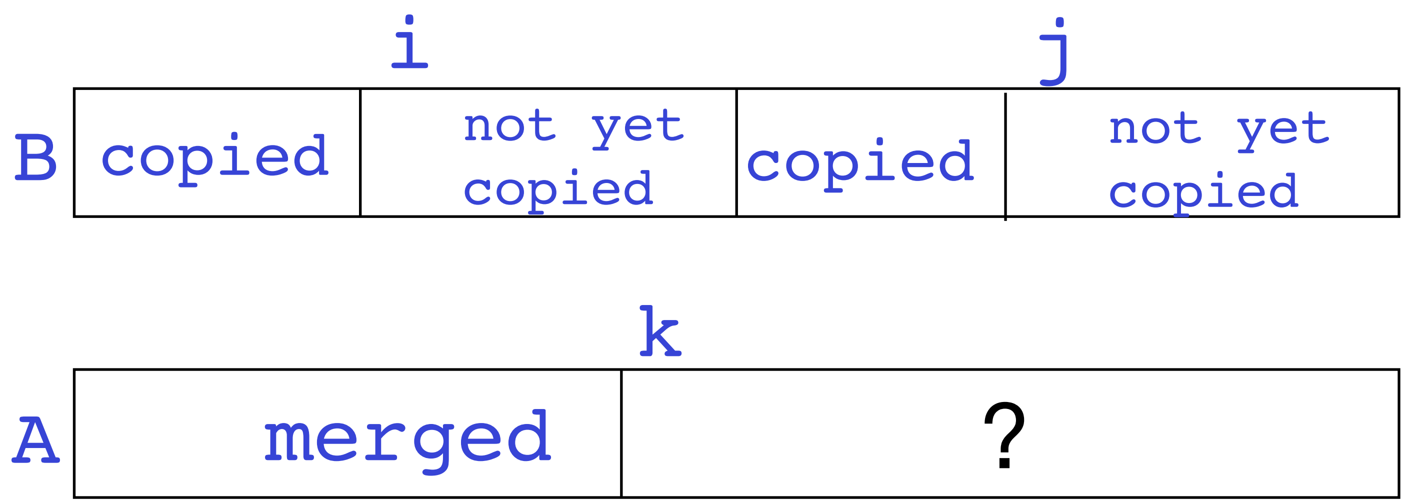
https://facultyweb.cs.wwu.edu/~wehrwes/courses/csci241_18f/img/merge.gif

Merge step: Loop Invariant

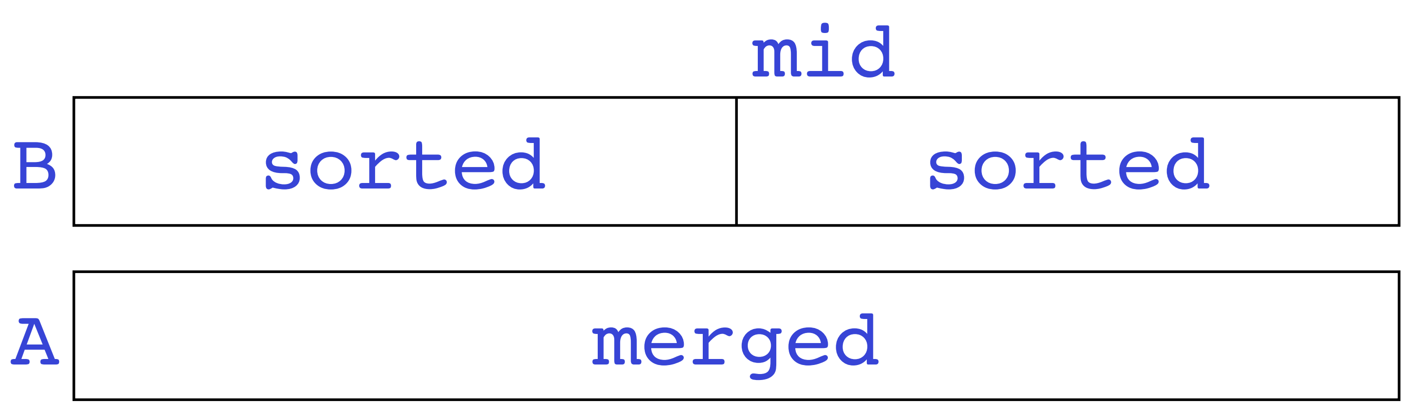
Precondition



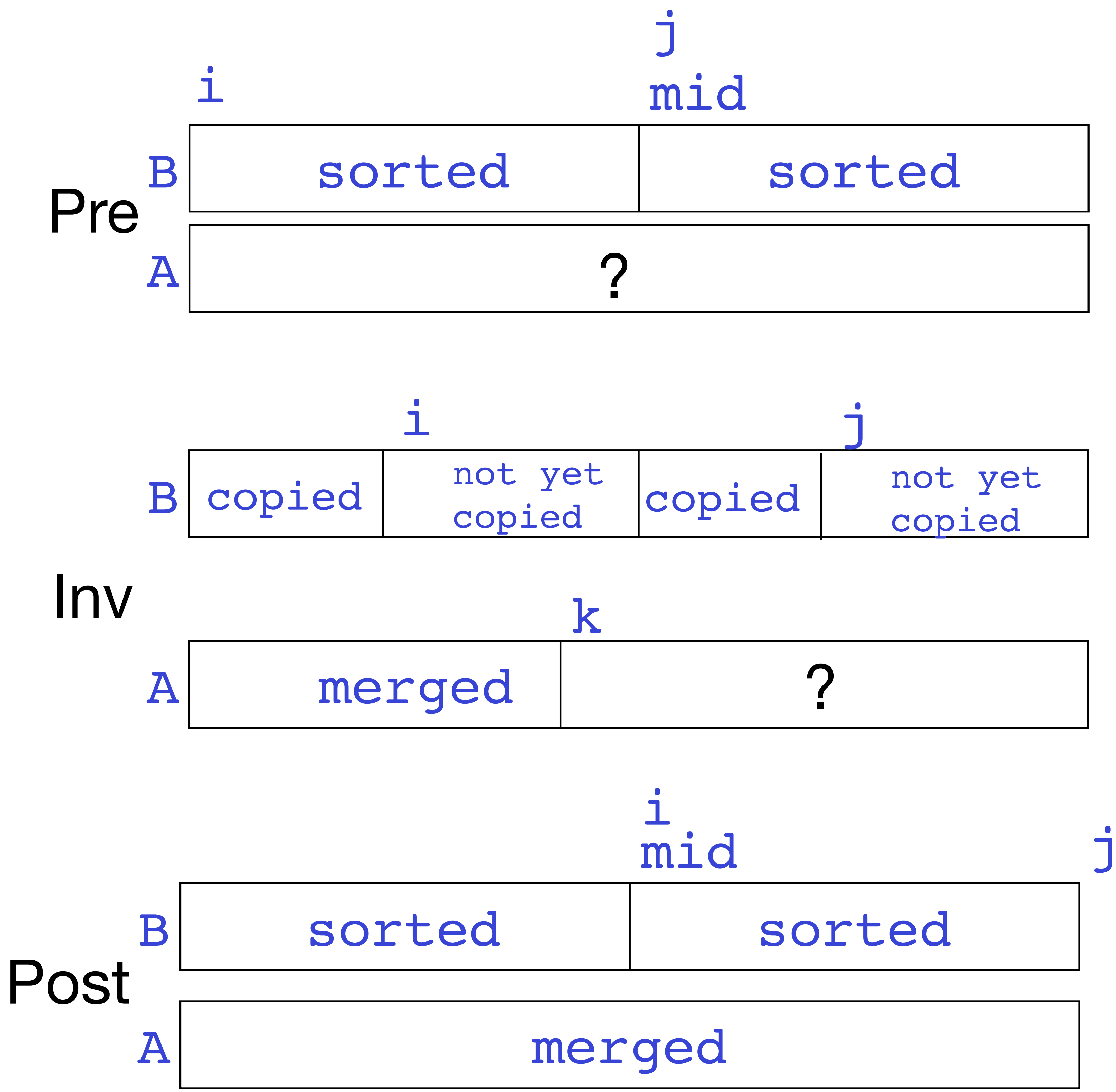
Invariant



Postcondition



Merge step




```
merge(A, start, mid, end):
  B = a deep copy of A
  i = start
  j = mid
  k = 0
  while i < mid and j < end:
    if B[i] < B[j]:
      A[k] = B[i]
      i++
    else:
      A[k] = B[j]
      j++
    k++

  while i < mid:
    A[k] = B[i]
    i++, k++

  while j < end:
    A[k] = B[j]
    j++, k++
```

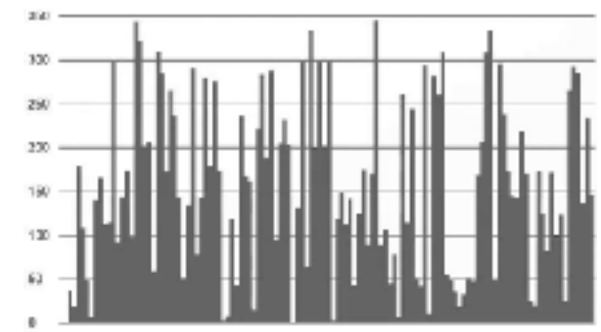
Smaller thing goes first

Ran out of things in one list or the other

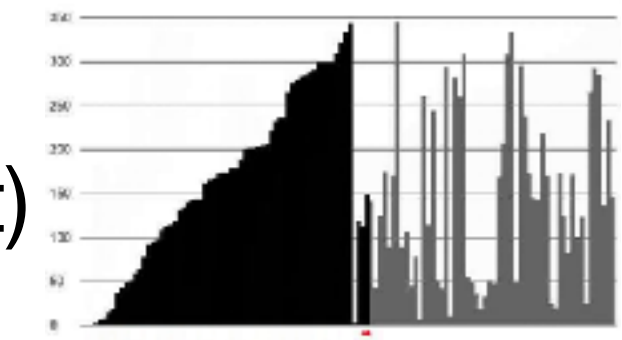
Copy remaining things from nonempty half

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

  mergeSort(A,start,mid)

  mergeSort(A,mid, end)

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

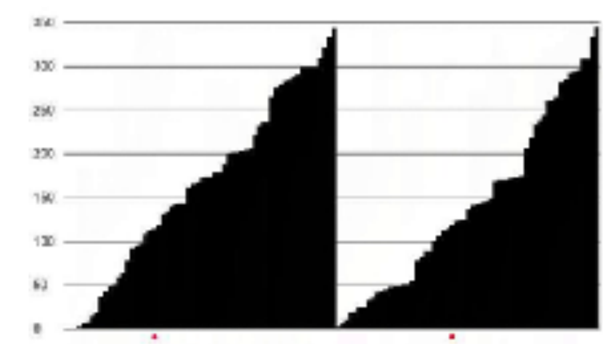
Divide



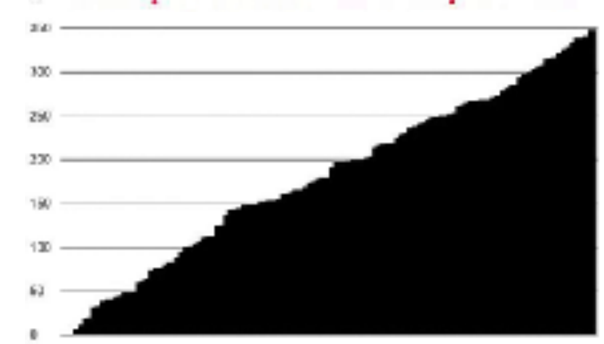
Conquer (left)



Conquer (right)



Combine



https://visualgo.net/bn/sorting

Merge step

ABCD:

What's the runtime of the merge step?

A. O(1)
B. O(n)
C. $O(n^2)$
D. $O(n^3)$

```
merge(A, start, mid, end):
  B = a deep copy of A
  i = start
  j = mid
  k = 0
  while i < mid and j < end:
    if B[i] < B[j]:
      A[k] = B[i]
      i++
    else:
      A[k] = B[j]
      j++
    k++

  while i < mid:
    A[k] = B[i]
    i++, k++

  while j < end:
    A[k] = B[j]
    j++, k++
```

Smaller thing goes first

Ran out of things in one list or the other

Copy remaining things from nonempty half

MergeSort: Runtime

```
/** sort A[start..end] using mergesort */
mergeSort(A, start, end):
  if (A.length < 2):                          O(1)
    return
  mid = (end-start)/2                         O(1)


  mergeSort(A,start,mid)                      O(huh?)


  mergeSort(A,mid, end)                       O(huh?)


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

How many times can we divide n by 2 before we hit 1?

$$n/2^x = 1$$
$$n = 2^x$$
$$x = \log_2 n$$