**Today**
- Quiz 4
- Brief intro to Merge and Quick Sort

**Assignments**
- HW 4 out
Merge Sort

The basic idea (note: recursive “divide and conquer” approach)

- divide list into two halves
- sort each half (recursive step)
- merge the sorted halves into sorted list
The merge sort algorithm (psuedocode):

Note: start, mid, and end are array indexes

MergeSort(T array[], int start, int end)
1. if start < end
2. mid = (start + end) / 2
3. mergesort(array, start, mid) # recursive step
4. mergesort(array, mid + 1, end) # recursive step
5. merge(array, start, mid, end) # merge sorted sublists

Merge(T array[], int start, int mid, int end)
1. T tmp[(end - start) + 1] # tmp array size n
2. first1 := start
3. first2 := mid + 1
4. i := 0
5. while first1 <= mid and first2 <= end do # merge into tmp
6. if array[first1] < array[first2] then
7.   tmp[i++] = array[first1++]
8. else
9.   tmp[i++] = array[first2++]
10. while first1 <= mid do # copy rest
11.   tmp[i++] = array[first1++]
12. while first2 <= mid do # copy rest
13.   tmp[i++] = array[first2++]
14. for i = 0 to (end - start) do # copy to array
15.   array[start + i] = tmp[i]

For HW-4, combine both steps into one function
Merge Sort for HW-4:

- Both array (ArraySeq) and linked list (LinkedSeq) implementations
- Can get started now on the array-based implementation (in ArraySeq)

For LinkedSeq ...

- We’ll do merge sort “in place” by splicing and reattaching
- Linked lists are nice for this ...
- More on Tuesday ...
Quick Sort

The basic idea (also a recursive “divide and conquer” approach)

- pick a “pivot” element (e.g., first element in list)
- put values smaller than pivot on left
- put values larger than pivot on right
- put pivot value in the middle
- sort the left and right halves (using quick sort)
The quick sort algorithm (psuedocode):

```c
quick_sort(T array[], int start, int end) # start, end indexes
1. if start < end then
2. pivot_val := array[start]
3. end_p1 := start # end index, start part
4. for i = start + 1 to end do # partition
5. if array[i] < pivot_val then
6.   end_p1 := end_p1 + 1
7.   swap(array[i], array[end_p1])
8. swap(array[start], array[end_p1]) # move pivot
9. quick_sort(array, start, end_p1 - 1) # recursive step
10. quick_sort(array, end_p1 + 1, end) # recursive step
```

Quick sort for HW-4

- Both an array (via `ArraySeq`) and linked list (via `LinkedSeq`) implementation

- Can get started now on the array-based implementation (in `ArraySeq`)

For `LinkedSeq` ...

- We’ll again do quick sort “in place” by splicing and reattaching

- Linked lists are nice for this ...

- More on Tuesday ...