

# For Friday

- Read Weiss, chapter 7, section 8-10
- Homework:
  - Weiss, chapter 7, exercise 15

# Exam 1

- Monday
- Covers material through mergesort

# Programming Assignment 2

- Any questions?

# Mergesort

- What's the concept?

# Mergesort

```
void MergeSort(int arr[], int temp[], int left,
               int right)
{
    if (left < right) {
        int center = (left + right) / 2;
        // sort left half
        MergeSort(arr, temp, left, center);
        // sort right half
        MergeSort(arr, temp, center+1, right);
        // merge left and right halves
        Merge(arr, temp, left, center+1, right);
    }
}
```

```
void Merge(int arr[], int temp[], int curLeft,
           int curRight, int endRight)
{
    int endLeft = curRight - 1;
    int curTemp = curLeft;
    int numElems = endRight - curLeft + 1;
    // Main loop
    while (curLeft <= endLeft && curRight <= endRight)
        if ( arr[curLeft] <= arr[curRight])
            temp[curTemp++] = arr[curLeft++];
        else
            temp[curTemp++] = arr[curRight++];
    while (curLeft <= endLeft) // finish left
        temp[curTemp++] = arr[curLeft++];
    while (curRight <= endRight) // finish right
        temp[curTemp++] = arr[curRight++];
    // copy back to arr
    for (int i = 0; i < numElems; i++, endRight--)
        arr[endRight] = temp[endRight];
}
```

# Performance of Mergesort

# Bottom-up Mergesort

Review questions?

# Quicksort

- Basic concept:
  - We're going to select a pivot
  - We're going to swap items into either smaller or large than the pivot, giving us two portions (partitions) of the array
  - We're going to sort each resulting partition using quicksort

```
void quicksort(int a[], int left, int right)
{ int i, j, v, temp;
  if (right - left > 0) { // at least two items in the partition
    v = a[right]; //v is the pivot
    i = left - 1; // 1 to the left of the beginning
    j = right;    // 1 to the right of where search starts
    while (true) { // infinite loop
      while (a[++i] < v); // pre-increment i until a[i] is >= the pivot
      while (a[--j] > v); // pre-decrement j until a[j] is <= the pivot
      if (i >= j) break; //if i and j have crossed -- get out of the loop
      temp = a[i]; // otherwise, swap a[i] and a[j]
      a[i] = a[j];
      a[j] = temp;
    }
    // i and j have crossed, so swap a[i] and the pivot
    a[right] = a[i];
    a[i] = v;
    // the pivot is now in place at i
    // now call quicksort on the two partitions
    quicksort(a, left, i-1); // left partition
    quicksort(a, i+1, right); // right partition
  }
}
```

# Performance of Quicksort

# Improving Quicksort

- Median of 3
- Cutoffs