

For Friday

- Read Weiss, chapter 4, section 4
- Program 2 due
- Homework:
 - Quicksort homework described on Blackboard

Programming Assignment 2

- Any 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

Bucket Sort

- Also known as bin sort
- Radix sort

Binary Search Trees

- An addition to our possible dictionary implementations
- We're interested in operations:
 - Search
 - Insert
 - Delete
 - Output in ascending order of keys
 - FindMin
 - FindMax

Definition

- Binary tree
- Each element has an associated key
- Keys are not duplicated
- For any node, its left child (if any) has a key smaller than its key and its right child (if any) has a key larger than its key

Binary Search Tree Operations

- Search
 - Time Complexity
- Insert
 - Time Complexity
- Deletion
 - Of a leaf
 - Of a node with one non-empty subtree
 - Of a node with two non-empty subtrees
 - Time Complexity
- Output Ordered List
- FindMin
- Find Max