Algorithms and Heuristics

Instructor: Dmitri A. Gusev

Fall 2007

CS 210: Computing and Culture

Lecture 5, October 1, 2007

Linear Search

- Check one list item after another, in a predetermined order, to find out whether it is identical (equal) to the item we're searching for. If such an item is found, report its position on the list, otherwise return a negative value.

Selection Sort

- Repeatedly find the smallest element in the unsorted part of the list, swap it with the leftmost element of the unsorted part if needed, and reduce the unsorted part by 1.

Example:

-1, 4, 5, 6, -7, 0

-7, 4, 5, 6, -1, 0

-7, -1, 5, 6, 4, 0

-7, -1, 0, 6, 4, 5

-7, -1, 0, 4, 6, 5

-7, -1, 0, 4, 5, 6

Bubble Sort

Start on the right-hand side (“at the bottom”) and swap two adjacent elements if the element on the right is smaller than the one on the left. Have the “bubble” float to the leftmost position in the unsorted part of the list, then reduce the unsorted part. If no swaps occurred in a pass, stop.

Example:

-1, 4, 5, 6, -7, 0

-1, 4, 5, ***-7***, 6, 0

-1, 4, ***-7***, 5, 6, 0

-1, ***-7***, 4, 5, 6, 0

-7, -1, 4, 5, 6, 0

-7, -1, 4, 5, ***0***, 6

-7, -1, 4, ***0***, 5, 6

-7, -1, 0, 4, 5, 6

Quicksort

if (*there is more than one item in the list*)
{
  Select splitVal;
  Split the list so that
    (list[0]..list[splitPoint-1] ≤ splitVal) AND
    (list[splitPoint]=splitVal) AND
    (list[splitPoint+1]..list[listLength-1] > splitVal);
  Quicksort list[0]..list[splitPoint-1];
  Quicksort list[splitPoint+1]..list[listLength-1];
}

Split

Set left to first+1

Set right to last

Do

    Increment left until list[left]>splitVal OR left>right

    Decrement right until list[right]<splitVal OR left>right

    If (left < right)

        Swap list[left] and list[right]

While (left≤right)

Set splitPoint to right

Swap list[first] and list[right]

Quicksort

- Split the list “around” the leftmost element, then recursively split the sublists.

Example:

-1, ***4***, 5, 6, ***-7***, 0

-1, -7, 5, 6, 4, 0

-7, -1, 5, ***6***, 4, ***0***

-7, -1, 5, 0, 4, 6

-7, -1, 4, 0, 5, 6

-7, -1, 0, 4, 5, 6

Merge Sort

-1 4 5 6 -7 0 2 -2

-1, 4 5, 6 -7, 0 -2, 2

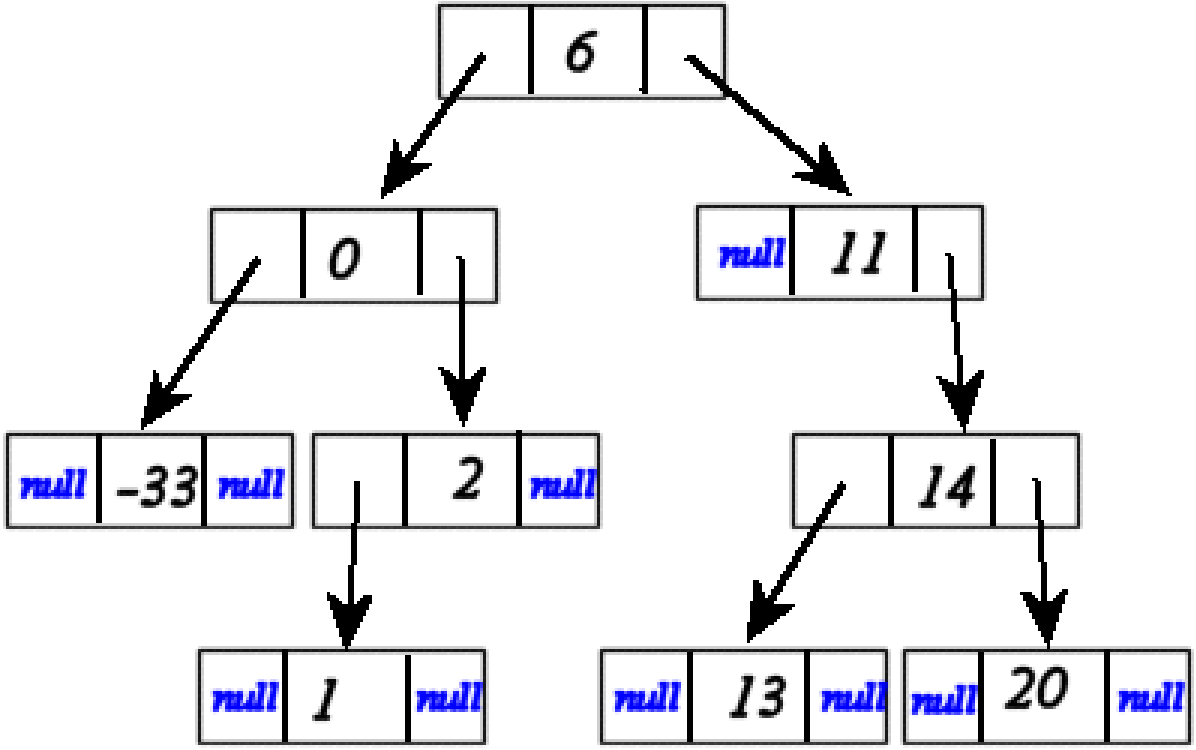
-1, 4, 5, 6 -7, -2, 0, 2

-7, -2, -1, 0, 2, 4, 5, 6

Binary Search

We search in a list that's **already sorted**. We compare the value of the middle element with the one that we're searching for. If they are equal, return true. If the value of the middle element is larger, we continue our search by examining the middle element of the half where the smaller values are located. Otherwise, check the middle element of the other half of the list. If the half to examine is an empty list (no elements left to check), return false.

Binary Search Trees



Traveling Salesman Problem

- If a salesman starts at point A, and if the distances between every pair of points are known, what is the shortest route which visits all points and returns to point A?

- Heuristic: A “rule of thumb” that tends to give a good solution. “Greedy algorithms.”

Computational Complexity

Big-O notation expresses computing time (~ the number of operations) as the term in a function that increases most rapidly relative to the size of a problem (N). (Can do *average*, *best case*, *worst case* analysis.)

O(1) is called *bounded time*. The amount of work is bounded by a constant.

$O(\log_2 N)$ is called *logarithmic time*. Example: Binary search.

O(N) is called *linear time*. Example: Sequential search.

$O(N \log_2 N)$ is called *N $\log_2 N$ time.* Applying a logarithmic algorithm N times.

$O(N^2)$ is called *quadratic time*. $O(N^3)$ is called *cubic time*.

Polynomial-time (Class P) algorithms: Algorithms whose complexity can be expressed as a polynomial in the size of the problem.

$O(2^N)$ is called *exponential time*.

O(N!) is called *factorial time*. The traveling salesman problem.

NP and NP-Complete Problems

- Class NP problems: Problems that can be solved in polynomial time with as many processors as desired.
- NP-complete problems: A class of problems within Class NP such that if a polynomial time solution with one processor can be found for any member of the class, such a solution exists for every member of the class.