Sorting and Searching

Instructor: Dmitri A. Gusev

Fall 2007

CS 113: Introduction to Computers

Lecture 14, December 4, 2007

Sorting

- A sort is an algorithm for ordering an array

  Unordered: -4, 3, 5, 0, 9, -1, 10, 2

  Ordered: -4, -1, 0, 2, 3, 5, 9, 10 (increasing order)

  Ordered: 10, 9, 5, 3, 2, 0, -1, -4 (decreasing order)

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

Shell Sort

1. Begin with a gap of g = Int (n/2).
2. Compare items 1 and 1+g, 2 and 2+g, …, n-g and n. Swap any pairs that are out of order.
3. Repeat Step 2 until no swaps are made for gap g.
4. Halve the value of g.
5. Repeat Steps 2, 3, and 4 until the value of g is 0.

Shell Sort: Example

-1, 4, 5, 6, -7, 0 (g=Int(6/2)=3)

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

-1, -7, 0, 6, 4, 5 (g=Int(3/2)=1)

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

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

-7, -1, 0, 4, 5, 6 (g=Int(1/2)=0, stop)

Split (pseudocode!)

```
left = first+1   ‘left points to the second element of the list
right = last     ‘right points to the last element of the list
Do While left<=right
    Increment left until list(left)>splitVal Or left>right
    Decrement right until list(right)<splitVal Or left>right
    If  left < right Then
            Swap list(left) And list(right)
    End If
Loop
Swap list(first) and list(right)
Split = right    ‘to return as a function value
```

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

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.

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: Flowchart