CSC 212
Data Structures and Abstractions
Fall 2015

Lecture 10: Elementary Sorts
Sorting
Total Order

binary relation such that for all $x, y, z$ in a set $X$:

- if $x \leq y$ and $y \leq x$ then $x = y$ (antisymmetry)
- if $x \leq y$ and $y \leq z$ then $x \leq z$ (transitivity)
- $x \leq y$ or $y \leq z$ (totality)
Elementary Sorts

Bubble Sort
Insertion Sort
Selection Sort
Bubble Sort

1) pass through the input array and:
   - **compare** each pair of adjacent items
   - **swap** them if they are in the wrong order

2) repeat step 1), n-1 times
   - can be optimized using a boolean flag
Bubble Sort

http://visualgo.net/sorting.html

Click on Bubble, then Exploration Mode, then Sort
void bubble(ul_int *A, ul_int n) {
    ul_int temp, i, j;
    // pass through the array n-1 times
    for (i = 0 ; i < (n-1) ; i ++) {
        // check for out-of-order pairs in the 'unsorted' half
        for (j = 0 ; j < (n-i-1) ; j ++) {
            // if A[j] and A[j+1] are out-of-order, swap them
            if (A[j] > A[j+1]) {
                temp = A[j+1];
                A[j] = temp;
            }
        }
    }
}
Analysis


$$(n-1) + (n-2) + (n-3) + \ldots + 1 = n(n-1)/2 = \mathcal{O}(n^2)$$

running time insensitive to input

Number of swaps?
Selection Sort

Array is divided into sorted (left) and unsorted (right) parts, and several scans are run from left to right

sorted part grows over time

In iteration $i$, find index $\text{min}$ of smallest element in the unsorted part

swap $A[i]$ and $A[\text{min}]$
Selection Sort

http://visualgo.net/sorting.html

Click on Select, then Exploration Mode, then Sort
void selection(ul_int *A, ul_int n) {
    ul_int i, j, min, temp;
    // grows the left part (sorted)
    for (i = 0; i < n; i++) {
        min = i;
        // find min in unsorted part
        for (j = i+1; j < n; j++) {
            if (A[j] < A[min]) {
                min = j;
            }
        }
        // swap A[i] and A[min]
        temp = A[i];
        A[i] = A[min];
        A[min] = temp;
    }
}
Analysis

Number of comparisons (A[j] < A[min])

\[(n-1) + (n-2) + (n-3) + \ldots + 1 = n(n-1)/2 = O(n^2)\]

running time insensitive to input

Number of swaps? \(O(n)\)
Insertion Sort

Array is divided into **sorted** (left) and **unsorted** (right) parts, and several scans are run from left to right. Sorted part grows over time.

In iteration $i$, swap $A[i]$ with each larger entry to its left. Inserts $A[i]$ (from unsorted) into the sorted part.
Insert Sort

http://visualgo.net/sorting.html

Click on Insert, then Exploration Mode, then Sort
void insertion(ul_int *A, ul_int n) {
    ul_int temp, i, j;
    // grows the left part (sorted)
    for (i = 0 ; i < n ; i ++) {
        temp = A[i];
        // inserts A[j] into the right place in sorted part
        for (j = i ; j > 0 && A[j-1] > temp ; j --) {
            A[j] = A[j-1];
        }
        A[j] = temp;
    }
}
Analysis

Best Case
number of comparisons \((A[j-1] > \text{temp})\) = \(O(n)\)
number of swaps = 0

Worst Case
number of comparisons \((A[j-1] > \text{temp})\) = \(O(n^2)\)
number of swaps = \(O(n^2)\)
Analysis

Average Case

number of comparisons \((A[j-1] > \text{temp})\) = \(O(n^2)\)
number of swaps = \(O(n^2)\)

Partially Sorted Arrays

array is partially sorted if the number of pairs that are out-of-order is \(O(n)\)
for partially sorted arrays InsertionSort runs in linear-time