RADFORD UNIVERSITY

ITEC 120

Lecture 16
Sorting

Review

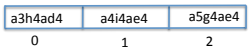
- Arrays
  - Passing to functions
- Comments

Objectives

- Motivate sorting
- Bubble
- Selection sort

Motivation

- Huge data set (3 million) stored with parallel arrays
- Given customer ID, find it in an array



Benefits of sorting

- Faster search access
- Statistical queries
  - Largest
  - Smallest
  - Range of values
- Report generation

Sorting numbers

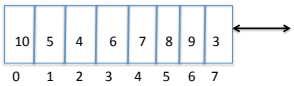
- Given this list, how would you sort it?

  12,4,9,10,3,1,2,9

- Find the largest, smallest, or ?
- Where do you move it

Bubble sort

- Starting value
- Compare it against its neighbor
- Swap if "unsorted'
- Repeat until you get to end of array (or non-swap case)



????

Repeating

- Current algorithm only happens one time
- Need to perform process for each index in the array
- How can you implement this?

Nesting

Reduce the size of checking (biggest is already bubbled up)

```
for (int i=1; i<=array.length; i++)
{
    for (int j=0; j<array.length-i; j++)
    {
        if (array[j] > array[j+1])
        {
            int temp=array[j];
            array[j] = array[j+1];
            array[j+1] = temp;
        }
    }
}
```


Pro/Con

- Pro
  - Simple algorithm
  - Works on any array
- Con
  - Bubble up N times for each number (n^2)
  - Has to look at every neighbor and lots of swaps
- There are better ways

Finding minimum

```
int[] array = new array[100];
//Code to initialize the array with data
int min=0;
for (int i=1; i<array.length; i++)
{
    if (array[min] > array[i])
        min=i;
}
```

How many operations could possibly happen?
Where is the smallest number in a list?

Selection sort

- Find minimum in array
- Swap with value in the first element
- Second element becomes first element
- Repeat until end

Code

```
for (int i=0; i<array.length; i++)
{
   int min = i;
   for (int j=i+1; j<array.length; j++)
   {
      if (array[j] < array[min])
         min=j;
   }
   if (min != i)
   {
      int temp = array[min];
      array[min] = array[i];
      array[i] = temp;
   }
}
```


Pro/Con

- Pro
  - Simple algorithm
  - Reduced number of swaps
- Con
  - Scan through array N times to find min
  - Scan through each # in array for the above
  - n^2 complexity

Complexity Analysis

- Picking which algorithm fits
  - Searching, sorting, etc…
- CPU usage
- Storage usage
- Best case, Average case, Worst case

Best case

- Bubble sort
  - Array of sorted numbers (n^2 comparisons)
- Selection sort
  - Array of sorted numbers (n^2 comparisons)
- In an ideal world, there are no costs to choices you make

Average/ Worst case

Bubble: 9K records in 18 seconds
Selection: 9k records in 12 seconds

- Bubble sort
  - Swap n^2 numbers
- Selection
  - Find the minimum n times
  - Swap n times

Summary

- Sorting
  - Bubble
  - Selection
- Complexity analysis
  - Best
  - Average
  - Worst