Sorting

Sorting

Arrange a list in a well-defined order

VERY common need

VERY important to pick a good sort algorithm

Interesting to explore sort algorithms

Selection Sort

Divides list into sorted and unsorted

Find smallest in unsorted list

put at front of unsorted list
and change split of sorted/unsorted

repeat

Selection Sort

```
//go thru each item in unsorted list  (unsl)
for int unsl = 0  to  unsl < n-1
    minPos = unsl            //assume in is at unsl
    for i = unsl+1 to i < n      //remainder of list
        if  a [ i ] < a[ minPos ]
            minPos = i
    if  minPos != unsl
        swap   a[ unsl ]  and a [minPos ]
```


Time Complexity

What is the expected complexity of Selection Sort

big Oh notation

(n-1)+(n-2)+(n-3)…+2+1 = n (n-1) / 2

Can still be effective for small lists

What is the worst-case beginning order?

Is there a best-case beginning order?

Insertion Sort

Builds final order one item at a time

Efficient for small data sets : $O(n^2)$

Better in practice than Selection Sort

Insertion Sort

Starting List: 3, 4, 2, 7, 1, 5

For each item i in list 0-n

find it's correct position in the earlier list < i

3*42715*	*trivial start: first item is by definition in the right location*
34*2715*	*4 compares with 3 - also in correct location*
342*715*	*2 compares to 4 and 3 and needs to go in front*
2347*15*	*7 is OK*
23471*5*	*1 must go in front*
123475	*5 must go before 7*
123457	

Insertion Sort

```
//Insertion sort algorithm
for i  = 1 to length(A)-1
    j = i
    while j > 0 and  A[j-1] > A[j]
        swap A[j] and  A[j-1]
        j = j -1
```


Shell Sort

Sorting pairs that are separated by a gap

Gap shrinks for each iteration

Example: Using a gap sequence: 5,3,1

0	1	2	3	4	5	6	7	8	
12	1	4	7	6	0	3	8	5	
0	1	4	5	6	12	3	8	7	5-sort
		4			7			12	3-sort
									1-sort

Shell Sort

Shell Sort is a series of Insertion Sorts on sub-lists of the full list

- Insertion Sort works very well with partially sorted lists and small lists
- 1-Sort finishes to make any last swaps
- 1-Sort is equivalent to Insertion Sort

Choosing a gap sequence is important

"Gonnet and Baeza-Yates observed that Shellsort makes the fewest comparisons on average when the ratios of successive gaps are roughly equal to 2.2.[11] This is why their sequence with ratio 2.2 and Tokuda's sequence with ratio 2.25 prove efficient. However, it is not known why this is so. Sedgewick recommends to use gaps that have low greatest common divisors or are pairwise coprime.[14]" - *Wikipedia ShellSort entry*

Time complexity is more complicated by less than $O(n^2)$

Merge Sort

Divide and conquer algorithm

Divide lists into smaller and smaller lists

- Smallest (two item) sort it
- Now combine as (4 item); sort it
- Combine as (8 item); sort it

Each larger list is partially sorted

Small lists are quick to sort; Partial sorted lists are quick to sort

Radix sort

see animation

Quick Sort

"King" of Sorts

Divide and conquer also

- Pick a pivot value
- Partition the list so that < pivot on left; > pivot on right
  put pivot value in between them
- Now repeat for each side (left and right - leaving out
  original pivot)
- Repeating until lists divide into 0 or 1 (sorted)