Selection sort

Strategy

Place the largest item in its correct place

Place the next largest item in its correct place, and so on

Selection sort is O(n^2) - always

Does not depend on initial arrangement of the data

* Swaps things when ordering
* Swaps lowest in list even if its first

Buble sort

Compare adjacent elements and exchange them if they are out of order

repeating this process wil eventually sort the array

Analysis worst case: o(n^2)

best case: o(n)

Insertion sort

Partition the array into two regions: sorted and unsorted

Take each item from the unsorted region and insert it into its correct order in the sorted region

Analysis

Worst case: o(n^2)

-individually takes items 1 by 1 and orders them as they occur( kinda like shifting things up and down)

-swapping things would require a temp variable to put the item changed into temp variable to be able to put back in after desired spot is changed where variable was.

Mergesort

A recursive sorting algorithm

Strategy

Divide an array into halves

Sort each half

Merge the sorted halves into one sorted array

Analysis

O(n\* log\_2 n) - always

Requires additional memory

Quicksort

A divide-and-conquer algorithm

Strategy

Partition an array into items that are less than the pivot and those that are greater than or equal to the pivot

Sort the left section

Sort the right section

Worst case: O(n^2)

Array is already sorted and smallest item is chosen as pivot

Average case: merge see above

Usually extemely fast

Radix sort

Radix sort(integers)

Treats each data element as a character string

Strat

See example

Sorting right digit in number and then to left each time