Selection sort

- Easiest and most intuitive sorting method
- Idea: find the smallest item in the list and place that item first. Then, find the second smallest item in the list and place that item second. ...
- Coolness factor: Each time through, you are guaranteed of placing one item into its correct position in the list.

Selection sort algorithm

Start at first item in list.

While there are more items in the list,

  find the smallest item below the first item.
  swap first item with smallest item.
  reset the “first” item to be the next item in the list.
Selection sort performance

- **Best case:** list is in order
  - number of comparisons $\sim N^2$ (N is size of list)
- **Worst case:** every item is out of order.
  - number of comparisons $\sim N^2$
  - number of swaps = 0
- **Average case:** looks a lot like the worst case, performance-wise.

Bubble sort

- Everyone’s favorite sorting algorithm (although it's the slowest!)
- **Idea:** go through the list, compare side-by-side items, and swap items that are out of order with each other
- **Coolness factor:**
  - same as selection sort, PLUS
  - once the list is sorted, the algorithm stops!
    - (not so with selection sort, which will always go through the entire list)

Bubble sort algorithm

*Start at first item in list.  
While the list is not sorted,  
compare the first item to the second item.  
if the first item is greater than the second item, swap.  
repeat for (2nd and 3rd), (3rd and 4th), etc.  
if no swaps were made, stop.  SORTED.  
else, “fix” the last item in the list (it’s in place already).*

Bubble sort performance

- **Best case:** list is in order already
  - number of comparisons = N
  - number of swaps = 0
- **Worst case:** list is in reverse order
  - number of comparisons = $N^2$
  - number of swaps = $N^2$
- **Average case:** looks like the worst case
Quicksort

- One of the fastest sorting algorithms (hence the name!)
- Idea: Each time through, select an item in the list, and move higher items to the right and lower items to the left. Split the list and repeat.
- Coolness factor: it's fast!

Quicksort algorithm

Set some item on list = “pivot”

While the list is not sorted,

search the list from the beginning until you find an item that's greater than the pivot.

search the list from the end until you find an item that's less than the pivot.

swap these two items.

repeat until there are no more items to compare

move the pivot so that it's at the separation point.

split the list at the pivot point, and repeat for each sublist.

Quicksort performance

- Worst case: list is already sorted or in descending order
  - number of comparisons = number of swaps ~ N*N
- Average case: split list in half each time, but do N comparisons on each sublist
  - number of comparisons = number of swaps ~ N*\log_2 N

Merge sort

- Idea: Keep splitting the list in half, sort the sublists, and then recombine
- Coolness factor: The list actually sorts itself!
Merge sort algorithm

While the list is not sorted,
divide the list in half.
if a half contains one item, it is sorted. combine it with its neighboring one-item list.
else, divide the list again.

Merge sort performance

- Best case: list is already sorted
  - number of comparisons = number of swaps = N log₂N
- Worst case/average case: anything else
  - number of comparisons = number of swaps = N log₂N

Summary of sort algorithm performance

- Selection sort has the fewest number of swaps
- Quicksort and merge sort have the fewest number of comparisons
- Merge sort takes the longest, time-wise, because we have to keep copying the array (the rest of the algorithms “sort in place”)