Sorting data - When would we want to sort?

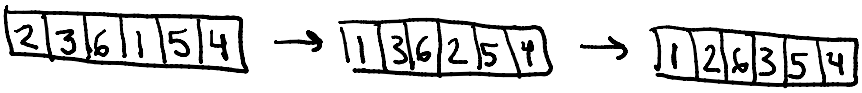
What are some potential characteristics of a sorting algorithm?

Group Work

Devise 2-3 algorithms to sort a list in $O(n^2)$ runtime.

- What is the worst case input for your algorithm
- Which of the above characteristics does the algorithm have? (If you need a hint - see below)

Selection Sort - iteratively find min in remaining unsorted list.



Which characteristics?

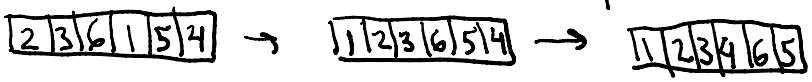
Worst case?

Argue correctness of output

Worst case:

Argue correctness of output

Insertion Sort – left part will be sorted and then iteratively insert next element into sorted part.



Which characteristics?

Worst case?

Argue correctness of output

Any other algorithms?

Quick Think!

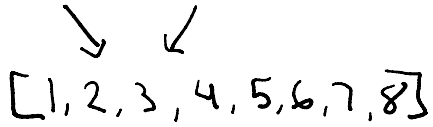
- Suppose you want to merge and sort k lists each with n/k elements that are already sorted. Show that this can be done in $O(kn)$ time
- Given a list, count the number of inversions in (an inversion is a pair of elements such that the larger element is to the left of the smaller) in $O(m+n)$ time.

Merge Sort Algorithm

Idea – break problem into smaller subproblems then merge

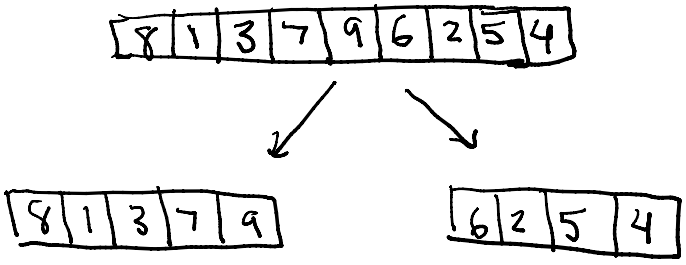
Algorithm then merge

[2, 3, 7, 8] [1, 4, 5, 6]



[1, 2, 3, 4, 5, 6, 7, 8]

Computation Tree



Divide-and-conquer

break problem into multiple subproblems and recurse, then combine solutions

What data structure to use? Lists/arrays → need to copy O(n) space

Linked lists → O(1) space

Group Work

- Argue correctness of output
- Analyze runtime
- Best case/worst case?