Recursive sorting

The best known strategy for sorting a list in a classical computer is called “divide-and-conquer”

* Divide the list into two (or more) parts – it’s best if the two parts are roughly the same size
* Sort each part
* Put them back together in order

Quicksort – to sort a list in increasing order:

* Take a list
* Take the first element of the list – call this the “pivot”
* Of the remaining elements (i.e. excluding the pivot):
	+ Put all elements less than or equal to the pivot in one list (low list)
	+ Put all elements greater than the pivot in another list (high list)
* Sort the low list (recursive call)
* Sort the high list (recursive)
* Put the list back together:
	+ (low list) (pivot) (high list)
* Stopping case: if the size of the list is 1 or 0 - it’s already sorted so you can just return the same list

Example:

3, 5, 2, 4, 1 – sort in increasing order using quicksort

1. Pivot? 3
	1. Low list: 2, 1
	2. High list: 5, 4
2. Sort the low list (recursively)
	1. Low list: 2, 1
	2. Pivot? 2
	3. Low low list: 1
	4. Low high list: empty
	5. Sort the low list (recursively)
		1. 1 : stopping case: return 1
	6. Sort the low high list (recursively)
		1. (empty) : return an empty list
	7. Put the low list back together: 1, 2
	8. Return: 1,2
3. Sort the high list
	1. High list: 5,4
	2. Pivot: 5
	3. High low list: 4
	4. High high list: (empty)
	5. Sort the high low list (recursively)
		1. Sort 4 : return 4
	6. Sort the high high list (recursively)
		1. (empty) : return (empty)
	7. Put the high list back together: 4, 5
	8. Return 4, 5
4. Low list: 1,2; pivot 3; high list: 4, 5
5. Put it all together: 1, 2, 3, 4, 5

Efficiency of quicksort: the number of operations needed is roughly n log n

For naïve sort algorithms, n^2

Imagine you are sorting a list with 1,000,000 elements

Naïve sort: 1,000,000^2 = 1 trillion

Quick sort: 1,000,000 \* log (1,000,000) ~= 10 million