CSC207.01 2014S, Class 28: Quicksort

Overview

- Preliminaries.
  - Upcoming work.
  - Questions.
- A quick introduction to Quicksort.
- Key ideas from Quicksort.

Preliminaries

Upcoming Work

- Today's lab writeup: Exercise 4 (implement Partition)
- Reading for Monday: A List ADT and Array-based lists (forthcoming)
- NO HOMEWORK THIS WEEK!

Admin

- Have fun with Earnest!
- EB is the note taker today.

Aspects of QuickSort

- O(n log(n)) is best and average case
- O(n squared) is the worst case scenario (if the pivot is the first or last element each time)
- Still, generally faster than other O(n log(n)) algorithms

So what is the pivot?

- It's the point from which we divide and conquer

Is QuickSort stable?

- No, and the ways to make it stable are pretty darn inefficient. [Well, more complicated than inefficient.]

More importantly, it's memory efficient.

- We don’t have to make another array when sorting an array!
- This means it sorts in-place!

Practice:

Let’s organize this array:

```
[slots are numbered 0 â 11]
lb = 0; ub = 12
a|l|p|h|a|b|e|t|i|c|a|l [we pick b as our pivot]

lb = 1; ub = 12
b|l|p|h|a|a|e|t|i|c|a|l
```

If an element is > b we lower ub, if element is < b we raise lb

We keep sorting, get to

```
a|a|a|b|h|e|t|i|c|p|l|l
ub == lb == 3
```

So we recurse over the first four elements and the rest of the list, and the list is eventually sorted!