Sorting Algorithms

Bryce Boe

2012/08/13

CS32, Summer 2012 B

Overview

- Bubble Sort
- Insertion Sort
- Selection Sort
- Merge sort
- Heapsort
- Quicksort
- Project 1 Part 2

Bubble Sort

- Mini lecture by Tyralyn Tran
- Quick recap
  - Until done, bubble largest elements to the *right* side
  - Worst case: $O(n^2)$
  - Best case: O(n) – requires optimization

Insertion Sort

- Mini lecture by Shanen Cross
- Quick recap
  - *Left* side is sorted: continuously select the *left* most unsorted element and move it left until it is in the proper location
  - Worst case: $O(n^2)$
  - Best case: O(n)

Selection Sort

- Mini lecture by Wei Guo
- Quick recap:
  - *Right* side is sorted: Until done, find the largest unsorted element, swap it into its final location (*right* most unsorted position)
  - Worst Case: $O(n^2)$
  - Best Case: $O(n^2)$
  - Benefits: Requires fewer swaps

Merge Sort

- Mini lecture by Grant Ball
- Quick recap:
  - Recursively* divide into smaller and smaller chunks until down to 1 element. Merge two already sorted lists into a larger sorted list. Repeat until all the data has been merged
  - Divide and conquer algorithm
  - Best/Worst Case: O(n*log(n))
    - Requires log(n) merge steps of n amount of data

*recursively only represents the top-down approach, can also be done bottom up

Heapsort

- Mini lecture by Crystal Cheung
- Quick recap:
  - Copy data into a min-heap. The heap will guarantee the smallest item is always at the top. Remove all the items from the heap to get them in sorted order.
  - Best/worst case: O(n*log(n))

Quicksort

- Select a **pivot**
- Move the pivot into its final position such that
  - All elements to the left are less than the pivot
  - All elements to the right are greater than the pivot
- Recursively run quicksort on the left side of the pivot
- Recursively run quicksort on the right side of the pivot

Quicksort

- Divide and conquer algorithm
- Average case: O(n*log(n))
  - Assuming pivots will somewhat evenly divide the data then there are log(n) pivot steps that reorganize **n** items
- Worst case: $O(n^2)$
  - Occurs when pivot selection does not partition data such as always selecting the left most element as the pivot in already sorted data

Project 1 Part 2

- Implementation of the following sort algorithms:
  - Bubble sort
  - Insertion sort
  - Selection sort
  - Merge sort
- Expected to write your own test cases
  - You will only have 6 submissions, and receive no diff feedback

In-place algorithms

- Algorithms that require a constant amount of extra memory to operate. That is, the amount of memory they require is not a function of the input size
- In-place sorts:
  - Bubble sort, insertion sort, selection sort, heapsort
- Not in-place sorts (by default):
  - Merge sort, quicksort

Stable sorting algorithms

- Items that compare equally such as (0, foo) and (0, bar) if we only compare the first item in the structure, will still be in the same relative order after sorting
- Stable sort algorithms:
  - Insertion sort, mergesort (usually)

For Tomorrow

- Jigsaw exercise for section 2 of the Reader, "Thinking Object-Oriented"
  - Three expert groups, one per chapter
  - You are to become a master, or expert of the section you are assigned
  - You will meet first with other experts of the same group to agree upon the key information
  - Then you will share/receive knowledge with/from members of other groups