Sorting algorithms

Textbook: Weiss Chapter 7.1

Byoungyoung Lee

https://compsec.snu.ac.kr

byoungyoung@snu.ac.kr

Lecture slides were prepared based on Douglas W. Harder’s notes (**dwharder@alumni.uwaterloo.ca).**

Outline

In this topic, we will introduce sorting, including:

- Definitions
- Assumptions
- *In-place* sorting
- Sorting techniques and strategies
- Overview of run times

Lower bound on run times

Define inversions and use this as a measure of *unsortedness*

Definition

Sorting is the process of:

- Taking a list of objects (e.g., numbers),

$$(a_0, a_1, \ldots, a_{n-1})$$

, and then returning an reordering

$$(a'_0, a'_1, \ldots, a'_{n-1}) \text{ such that } a'_0 \le a'_1 \le \cdots \le a'_{n-1}$$

The conversion of an Abstract List into **an Abstract Sorted List**

Assumption

In these topics, we will assume that:

- Arrays are used for both input and output,
- We will focus on sorting objects for a specific, single key
  - We don’t cover sorting objects with multiple keys

In-place and Out-of-place Sorting

In-place sorting algorithms are performed in-place, that is, with the allocation of at most $\Theta(1)$ additional memory

Out-of-place sorting algorithms require the allocation of second array of equal size

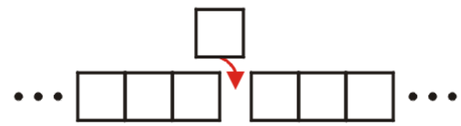
- Requires $\Theta(n)$ additional memory

We will prefer in-place sorting algorithms

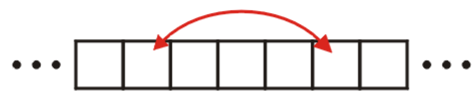
Classifications

The operations of a sorting algorithm are based on the actions performed:

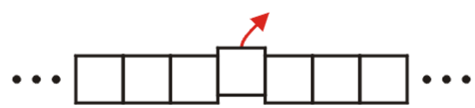
- Insertion



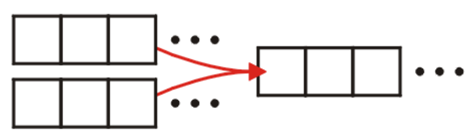
- Exchanging



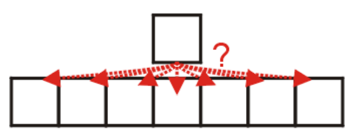
- Selection



- Merging



- Distribution



Run-time

The runtime of the sorting algorithms we will look at fall into one of three categories:

$$\mathbf{\Theta}(n) \quad \mathbf{\Theta}(n \ln(n)) \quad \mathbf{O}(n^2)$$

We will examine average- and worst-case scenarios for each algorithm

The runtime may change significantly based on the scenario

Sorting Algorithms

1) **traditional $O(n^2)$ sorting** algorithms:
   – Insertion sort

2) **faster $\Theta(n \ln(n))$ sorting** algorithms:
   – Heap sort, Quicksort, and Merge sort

3) **linear-time $O(n)$ sorting** algorithms
   – Bucket sort and Radix sort
   – We must make assumptions about the data

Summary

Introduction to sorting, including:

- Assumptions
- In-place sorting ($\mathbf{O}(1)$ additional memory)
- Sorting techniques
  - insertion, exchanging, selection, merging, distribution
- Run-time classification: $\mathbf{O}(n)$ $\mathbf{O}(n \ln(n))$ $\mathbf{O}(n^2)$

Overview of proof that a general sorting algorithm must be $\mathbf{\Omega}(n \ln(n))$

References

Wikipedia, http://en.wikipedia.org/wiki/Sorting_algorithm
http://en.wikipedia.org/wiki/Sorting_algorithm#Inefficient.2Fhumorous_sorts

[1] Donald E. Knuth, *The Art of Computer Programming, Volume 3: Sorting and Searching*, 2nd Ed., Addison Wesley, 1998, §5.1, 2, 3.

[2] Cormen, Leiserson, and Rivest, *Introduction to Algorithms*, McGraw Hill, 1990, p.137-9 and §9.1.

[3] Weiss, *Data Structures and Algorithm Analysis in C++*, *3rd Ed.*, Addison Wesley, §7.1, p.261-2.

[4] Gruber, Holzer, and Ruepp, *Sorting the Slow Way: An Analysis of Perversely Awful Randomized Sorting Algorithms*, 4th International Conference on Fun with Algorithms, Castiglioncello, Italy, 2007.