CLRS Notes

Classic ElegantLaTeX Template

Author: Haopeng Li

Institute: ElegantLaTeX Program

Date: July 27, 2022

ElegantLaTeX Program

Contents

Chapter 1 Analysis of Algorithms

Introduction

- *Insertion sort*
- *Asymptotic analysis*
- *Merge sort*
- *Recurrences*

Definition 1.1 (Algorithms)

The theoretical study of computer-program performance and resource usage. ♣

Note *Why study algorithms and performance?*

- *Algorithms help us to understand scalability.*
- *Performance often draws the line between what is feasible and what is impossible.*
- *Algorithmic mathematics provides a language for talking about program behavior.*
- *Performance is the currency of computing.*
- *The lessons of program performance generalize to other computing resources.*
- *Speed is fun!*

1.1 The problem of sorting

Problem 1.1(The problem of sorting)

- Input:sequence $< a_1, a_2, \cdots, a_n >$ of numbers.
- Output: permutation $< a'_1, a'_2, \cdots, a'_n >$ such that $a'_1 \le a'_2 \le \cdots \le a'_n$

1.2 Insertion Sort

```
Insertion-Sort(A,n)
  for j <- 2 to n
    do key <- A[j]
      i <- j - 1
      while i > 0 and A[i] > key
        do A[i+1] <- A[i]
        i<- i-1
      A[i+1] = key
```

1.3 Running time

- The running time depends on the input:an already sorted sequence is easier to sort.
- Parameterize the running time by the size of the input,since short sequences are easier to sort than long ones.
- Generally,we seek upper bounds on the running time,because everybody likes a guarantee.

1.3.1 Kinds of Analysis

Definition 1.2 (Worst-Case(usually))

$T(n)$ = maximum time of algorithm on any input of size n. ♣

Definition 1.3 (Average-Case(Sometimes))

- *$T(n)$ = expected time of algorithm over all inputs of size n.*
- *Need assumption of statistical distribution of inputs.* ♣

Definition 1.4 (Best-case: (bogus))

Cheat with a slow algorithm that works fast on some input. ♣

Note *What is insertion sort's worst-case time?*

It depends on the speed of our computer:

- relative speed (on the same machine),
- absolute speed (on different machines).

Note *BIG IDEA:*

1. Ignore machine-dependent constants.
2. look at the growth of $T(n)$ as $n \to \infty$
3. "Asymptotic Analysis"

1.3.2 Θ-Notation

Definition 1.5 (Θ-Notation)

$\Theta(g(n)) = \{f(n)$: *there exist positive constants* c_1, c_2, *and* n_0 *such that* $0 \le c_1 g(n) \le f(n) \le c_2 g(n)$ *for all* $n \ge n_0\}$ ♣

Note *Engineering:Drop low-order terms; Ignore leading constants.*

Example 1.1

$$3n^3 + 90n^2 - 5n + 6046 = \Theta\left(n^3\right)$$

1.3.3 Asymptotic performance

Note *When n gets large enough, a $\Theta\left(n^2\right)$ algorithm always beats a $\Theta\left(n^3\right)$ algorithm.*

Note

- *We shouldn't ignore asymptotically slower algorithms, however.*
- *Real-world design situations often call for a careful balancing of engineering objectives.*
- *Asymptotic analysis is a useful tool to help to structure our thinking.*

1.3.4 Insertion sort analysis

Worst case

Input reverse sorted

$$T(n) = \sum_{j=2}^{n} \Theta(j) = \Theta\left(n^2\right)$$

Average case

All permutations equally likely.

$$T(n) = \sum_{j=2}^{n} \Theta(j/2) = \Theta\left(n^2\right)$$

Note *Is insertion sort a fast sorting algorithm?*

- *Moderately so, for small n*
- *Not at all, for large n*

1.4 Merge Sort

```
Merge-Sort A[1..n]
  1. If n = 1,done
  2. Recurisively sort A[1...⌈n/2⌉] and A[⌈n/2⌉+1...n]
  3. Merge the 2 sorted lists.
```

Note *Key subroutine: MERGE*

1.4.1 Analyzing Merge Sort

Time $= \Theta(n)$ to merge a total of n elements (linear time).

$T(n)$	MERGE-SORT $A[1 \ldots n]$
$\Theta(1)$	1. If $n = 1$, done.
$2T(n/2)$	2. Recursively sort $A[1 \ldots \lceil n/2 \rceil]$ and $A[\lceil n/2 \rceil + 1 \ldots n]$.
$\Theta(n)$	3. "Merge" the 2 sorted lists

Note ***Sloppiness**: $2T(n/2)$ should be $T(\lceil n/2 \rceil) + T(\lfloor n/2 \rfloor)$, but it turns out not to matter asymptotically.*

1.4.2 Recurrence for merge sort

$$T(n) = \begin{cases} \Theta(1) \text{ if } n = 1 \\ 2T(n/2) + \Theta(n) \text{ if } n > 1 \end{cases}$$

Note

- *We shall usually omit stating the base case when $T(n) = \Theta(1)$ for sufficiently small n, but only when it has no effect on the asymptotic solution to the recurrence.*
- *CLRS and Lecture 2 provide several ways to find a good upper bound on $T(n)$.*

1.4.3 Recursion tree

Example 1.2 Solve $T(n) = 2T(n/2) + cn$, where $c > 0$ is constant.

1.4.4 Conclusions

- $\Theta(n \lg n)$ grows more slowly than $\Theta\left(n^2\right)$
- Therefore, merge sort asymptotically beats insertion sort in the worst case.
- In practice, merge sort beats insertion sort for $n > 30$ or so.