Reading Note: Chapter 2 & 3

Xitong Yang

Dec.23 2015

1 Introduction

Chapter 2 & 3 mainly give an introduction about what algorithms are and how to analyze the efficiency of an algorithm. It is important to understand the concept of the "asymptotic notation".

2 Key Note

- Use **loop invariant** to prove the correctness of a program. Three steps: (1) Initialization: true prior to the first iteration of the loop; (2) Maintenance: if true before an iteration of the loop, remains true before the next iteration; (3) Termination: when the loop terminates, the invariant helps show that the algorithm is correct.
- Asymptotic Notation
  - $f(n) = \Theta(g(n))$: $g(n)$ is an asymptotically tight bound for $f(n)$, is like $a = b$. Formally, $\Theta(g(n)) = \{f(n)$:there **exists** positive constants c_1, c_2 and n_0 such that $0 \leq c_1 g(n) \leq f(n) \leq c_2 g(n)$ for all $n \geq n_0\}$
  - $f(n) = O(g(n))$: asymptotic upper bound, is like $a \geq b$. $O()$ bound on worst-case running time also applies to its running time on *every input (even input is already sorted)*.
  - $f(n) = \Omega(g(n))$, asymptotic lower bound, is like $a \leq b$.
  - $f(n) = o(g(n))$, asymptotic loose upper bound, is like $a > b$.
  - $f(n) = w(g(n))$, asymptotic loose lower bound, is like $a < b$.
  - Not all functions are asymptotically comparable.
- Some maths

3 Algorithms

- Insertion sort* ($\Theta(n^2)$)
- Merge sort* ($\Theta(n \log n)$)

* Sample codes implemented in *Codes* folder