Computational Complexity

Charles Zhao

September 30, 2016

1 Introduction

The *computational complexity* of an algorithm is a measure of its efficiency. It is the amount of time (*time complexity*) or space (*space complexity*) an algorithm takes to run as a function of the size of the input.

2 Asymptotic Computational Complexity

Asymptotic computational complexity is the most common way we estimate the complexity of algorithms. There are two important simplifications asymptotics use: if an algorithm's exact complexity is $f(n)$ where n is the input size, then (1) if $f(n)$ is a sum of several terms, only the term with the largest order is considered, and (2) if $f(n)$ is a product of several factors, any constants are omitted. Usually when we talk about an algorithm's complexity, we are referring to the upper bound of the algorithm's asymptotic computational complexity (denoted by big O notation). However, other types of asymptotic estimates include the lower bound (denoted by big omega notation) and a function that is both the upper and lower bound (denoted by big theta notation). Here are their formal definitions:

- $f(n) = O(g(n))$ if $f(n) \leq c \cdot g(n) \; \forall \; n > n_0 \wedge c > 0$
- $f(n) = \Omega(g(n))$ if $f(n) \geq c \cdot g(n) \; \forall \; n > n_0 \wedge c > 0$
- $f(n) = \theta(g(n))$ if $f(n) \leq c_1 \cdot g(n) \wedge f(n) \geq c_2 \cdot g(n) \; \forall \; n > n_0 \wedge c_1 > 0 \wedge c_2 > 0$

3 Common Complexities

- $O(1)$ - constant
- $O(\log n)$ - logarithmic
- $O(n)$ - linear
- $O(n^2)$ - quadratic
- $O(n^3)$ - cubic
- $O(2^n)$ - exponential

4 Example

What is the time complexity of `RabinKarp`?

Algorithm 1 Rabin-Karp

```
function RabinKarp(string s[1..n], stringSet searchStrs, int m):
    set hashedSearchStrs = emptySet
    foreach string in searchStrs
        insert hash(string[1..m]) into hashedSearchStrs
    hashedSub = hash(s[1..m])
    for i = 1 to n-m+1
        if hashedSub in hashedSearchStrs and s[i..i+m-1] in searchStrs
            return i
        hashedSub = hash(s[i+1..i+m])
    return not found

function hash(string key, int M)
    hash = 0
    for int j = 0 to M-1
        h = (R * h + key[j])
    return h
```

5 Contests Cheat Sheet

In USACO, for each test case, you are given 1 second for C++ and 2 seconds for Java. Your programs are run on machines that do approximately 10^8 operations per second. Based on the input size bounds given to you, here are around the complexities your programs should be:

- $N \leq 10 : O(N!)$
- $N \leq 25 : O(2^N)$
- $N \leq 50 : O(N^4)$
- $N \leq 500 : O(N^3)$
- $N \leq 5000 : O(N^2)$
- $N \leq 100000 : O(N \log N)$
- $N \leq 1000000 : O(N)$