【数据结构】Day1

Class	Advanced Data Structures
Date	@December 1, 2021
Material	
Series Number	
Summary	

【Ch2】算法分析

2.1 数学基础

估计算法资源消耗所需分析是一个理论问题，因此需要一套正式的系统构架，从下面数学定义开始：

1. 如果存在正常数c和n_0使得$N >= n_0$时$T(N) <= cf(N)$，则记为$T(N) = O(f(N))$
2. 如果存在正常数c和n_0使得$N >= n_0$时$T(N) >= cg(N)$，则记为$T(N) = \Omega(g(N))$
3. $T(N) = \Theta(h(N))$当且仅当$T(N) = O(h(N))$且$T(N) = \Omega(h(N))$

这些定义的目的是要在函数间建立一种相对的级别。给定两个函数，通常存在一些点，在这些点上一个函数的值小于另一个函数的值。因此，像$f(N) < g(N)$这样的声明是没有什么意义的。

所以，我们比较它们的相对增长率(relative rate of growth)。

1. 如果我们用传统的不等式来计算增长率，那么第一个定义是说T(N)的增长率小于等于f(N)的增长率。
2. 第二个定义$T(N) = \Omega(g(n))$是说T(N)的增长率大于等于g(N)的增长率。
3. 第三个定义$T(N) = \Omega(h(N))$是说T(N)的增长率等于h(N)的增长率

当我们说$T(N) = O(f(N))$时，我们是在保证函数T(N)是在以不快与f(N)的速度增长；因此f(N)是T(N)的一个上界(upper bound)

于此同时，$f(N) = \Omega(T(N))$意味着T(N)是f(N)的一个下界(lower bound)

了解定义后，我们需要掌握的重要结论为：

1. 法则一：

   如果$T_1(N) = O(f(N))$且$T_2(N) = O(g(N))$，那么

   1. $T_1(N) + T_2(N) = max(O(f(N)),\ O(g(N)))$
   2. $T_1(N) * T_2(N) = O(f(N) * g(N))$

2. 法则二：

   如果T(N)是一个k次多项式，则$T(N) = \Theta(N^k)$

3. 法则三：

   对任意常数k，$log_k(N) = O(N)$。它告诉我们对数增长的非常缓慢

注意：将常熟或低阶放进大O是不好的习惯。不要写成$T(N) = O(2N^2)$，正确的形式应该是$T(N) = O(N^2)$

2.3 要分析的问题

影响程序运行时间的主要因素为使用的算法以及对该算法的输入。

典型的情况是：输入的大小是主要的考虑方面。我们定义两个函数$T_{avg}(N)$和$T_{worst}(N)$分别为输入为N时，算法所花费的平均运行时间和最坏情况下的运行时间。显然，$T_{avg}(N) <= T_{worst}(N)$。

一般来说，所需要的量是最坏情况下的运行时间。其原因是它对所有的输入提供了一个界限，包括特别坏的输入，而平均情况分析不提供这样的界。另一个原因是平均情况的界计算起来通常要困难得多。

注意：我们在分析算法时忽略从计算机磁盘中读取数据的时间