CS310: ALGORITHMS AND DATA STRUCTURES

Running Time Analysis

- $T(n) = T(\frac{1}{2} n) + T(\frac{1}{2} n) + Kn$

- $T(n) = 2\,T(\frac{1}{2} n) + Kn$

Recurrence Equation

How to solve recurrence equations?

Recursion Tree Method

Running Time Complexity of Merge Sort is $\Theta(n \log n)$

Deriving Running Time Expressions

- Assuming constant costs for various elementary steps
- Ignoring constant coefficients
- Drop lower order terms
- Obtain Asymptotic notation
- Then we track the growth of T(n) as $n \longrightarrow \infty$
- Analysis applies regardless of the machine (all the platform dependent parameters have been removed)

Efficient Algorithms

- An algorithm that is asymptotically most efficient (among multiple algorithms for the same problem) will run fastest for very large input sizes but **NOT necessarily** for small input sizes

- Example: MergeSort versus Insertion Sort

Asymptotic Notations

- There are many types of asymptotic notations
  - O: the big-oh
  - Ω: the big-omega
  - o: the little-oh
  - ω: the little-omega

- n >= 0, T(n) >= 0, n (domain) is continuous

Theta Notation: Formal Definition

- When we say $T(n)$ is $\Theta(f(n))$, we mean that $T(n)$ is sandwiched between two constant multiples of $f(n)$ i.e. $c_1 f(n) <= T(n) <= c_2 f(n)$ for $c_1, c_2 > 0$
- This claim applies when n is sufficiently large, i.e. some $n >= n_0$
- Example: $1/2n^2 + 3$