CSCI203

Week 2 – Lecture B

More on Complexity

- Last week we looked at the idea of *Complexity Classes*.
- These included:
  - Constant: the difficulty of the problem is independent of its size;
  - Logarithmic: the difficulty of the problems grows slowly as the problem size increases;
  - Linear: the difficulty increases at the same rate as the problem size grows;
  - Linearithmic: grows as $n \log n$;
  - Quadratic: grows as n^2;
  - Exponential: Grows as 2^n;
  - Factorial: grows as $n!$

Order of Complexity

- A very useful way to talk about complexity is to use the concept of “order of”.
- This idea allows us to directly compare complexity as well as allowing us to simplify the way in which we represent complexity.
- There are three standard order of complexity measures in common use:
  - Big Oh;
  - Omega;
  - Theta.

Big Oh notation

- Consider the following function:
  - $t(n) = 12n^2 + 17/3\ n + 7/5$
- For $n \geq 1$, $t(n) \leq 12n^2 + 17/3\ n^2 + 7/5\ n^2 = 286/15n^2$
  - This is a constant (286/15) times a simple function n^2
- We say that $t(n)$ is in the *order of* n^2.
- We can write $t(n) \in O(n^2)$; $t(n)$ is in "big Oh" of n^2
- More formally, $O(\ f(n)\ )$ is the set of all functions t such that $t(n) \leq cf(n)$, for some positive real constant c for all $n \geq n_0$

O(f(n))

- Note that O(f(n)) is the set of all functions that behave in the same general way:
  - They all grow at a rate that is *no faster than* $f(n)$.
- We say that O(f(n)) is the set of all functions *bounded above* by $f(n)$.

Asymptotic Notation

- It is legitimate to talk about the order of $f(n)$ even if $f(n)$ is badly behaved below n_0
- Some texts use the notation $t(n) = O(f(n))$ which, given that $O(f(n))$ is a set, is strictly incorrect
- The maximum rule: $O(f(n)+g(n)) = O(\max(f(n),g(n)))$
  - E.g. $O(n^3 + n^2 + n \log n) = O(\max(n^3, n^2, n \log n)) = O(n^3)$

Comparing functions

- We can use the following results to determine the relationship between two functions, *f*(*n*) and *g*(*n*).
  - If $\lim_{n\to\infty} \frac{f(n)}{g(n)} \in R^{+}$ then f(n) $\in$ O(g(n)) and g(n) $\in$ O(f(n))
  - If $\lim_{n\to\infty} \frac{f(n)}{g(n)} = 0$ then f(n) $\in$ O(g(n)) but g(n) $\notin$ O(f(n))
  - If $\lim_{n\to\infty} \frac{f(n)}{g(n)} = +\infty$ then f(n) $\notin$ O(g(n)) but g(n) $\in$ O(f(n))

Other Asymptotic Notation: Omega

- O(f(n)) is the set of all functions that are *less than* the limiting function.
- This is not always a very useful measure.
- An alternative is Omega notation.
- A function t(n) is in Omega of f(n) if, for sufficiently large values of n, t(n) is less than some constant times f(n).
- We say that Ω(f(n)) is the set of all functions *bounded below* by f(n).

Relating O and Ω

- If we have two functions $f(n)$ and $g(n)$, the following is true:
  - $f(n) \in \Omega(g(n))$ **`iff`** $g(n) \in O(f(n))$
- That is to say that $f(n)$ is bounded below by $g(n)$ if and only if $g(n)$ is bounded above by $f(n)$.
- Both of O and Ω are *loose* bounds. A function can be in one of these and yet grow much more slowly (or quickly) than the bounding function.
  - E.g. $n^2 \in O(n!)$ and $n! \in \Omega\, n^2$.
- A third notation allows us to create a tighter bound for some functions.

Theta, a tighter bound.

- Sometimes, we can find a function $f(n)$ so that:
  - $t(n) \in O(f(n))$
  - $t(n) \in \Omega(f(n))$
- That is, the same function is both an upper bound and a lower bound.
- In this case we say that $t(n) \in \Theta(f(n))$

Bounds and Algorithms

- We can make use of all three of these bounds in analyzing algorithms, as follows:
  - $O(f(n))$, the upper bound, is related to the *worst-case* behavior of the algorithm.
  - $\Omega(f(n))$, the lower bound, is related to the *best-case* behavior.
  - $\Theta(f(n))$ is related to the *average* or *typical-case* behavior.
- In this subject, we will usually be interested in the typical-case behavior of an algorithm although we will sometimes examine best- (and worst-) case behaviors.