Algorithm Analysis

CSE/IT 122 ~ Algorithms and Data Structures

Theorem of Polynomial Orders

- ➔ Suppose $a_0, a_1, ..., a_n$ are real numbers and $a_n \neq 0$
  - Then $a_n x^n + a_{n-1} x^{n-1} + ... + a_1 x^1 + a_0$ is $O(x^s)$ for all integers $s \geq n$

Example: Polynomial Orders

- What is the Big-Oh of 1+2+...+n
  - 1+2+...+n = (n(n+1))/2 = $1/2n^2$ + 1/2n so by theorem it is $O(n^2)$ and $O(n^3)$, etc
- He theorem works for rational powers as well
  - The biggest term dominates and you can find multiple of the largest power that is greater than the sum of the terms of the polynomial

More Examples: Polynomial Orders

- Show $f(n) = n^2+2n+1$ is $O(n^2)$
  - $n^2+2n+1 < n^2+2n^2+n^2 < 4n^2$ for all $n > 1$
  - Can do better $n^2 > 2n+1$, $n > 2+1/n$, true for $n > 2$
  - So $f(n) < 2n^2$, for $c=2$, and $n_0=2$
- Show $7n^2$ is $O(n^3)$
  - $7 < n$ for $n > 7$ and multiplying both sides by n^2 we get
  - $7n^2 < n^3$
  - So $c=1$ and $n_0=7$
  - We can also show that $7n^2$ is $O(n^2)$
  - Upper bounds are not unique

Upper Bounds Of Functions Besides Power Functions

- Find an upper bound for $T(n) = n!$
  - $N! = 1*2*3*...*n \le n*n*n*...*n = n^n = O(n^n)$
- Logs
  - $y = \log_b x$ iff $b^y = x$
  - Taking logs of both sides doesn't change the sign of an inequality
  - $n! \le n^n$ Why?
  - So $\log n! \le \log n^n = n \log n$
  - Soo $\log n! = O(n \log n)$

Example: Remember Induction

- Show $2^n < n!$
  - By induction
    - Base Case: $2^4 < 4!$
    - Assume: $2^k < k!$
    - Show $2^{k+1} < (k+1)!$
    - Proof:
      - $2^{k+1} = 2^k * 2 < k! * 2$, by assumption
      - Now $(k+1)! = (k+1)k!$, and $2 < (k+1)$ for all $k \geq 4$

Example: Remember Induction

- Show $2^n > n^3$
  - By induction:
    - Base case: $2^{10} > 10^3$ (this is also a good approximation)
    - Assume: $2^k > k^3$
    - Show: $2^{k+1} > (k+1)^3$
    - Proof:
      - $2^{k+1} = 2^k * 2 > 2k^3$
      - Now, $(k+1)^3 = k^3+3k^2+3k+1$
      - So, have to show that $2k^3 > (k+1)^3 = k^3+3k^2+3k+1$
      - Which is really just showing that $k^3 > 3k^2+3k+1$, which is true by theorem on the order of polynomials

Combining Running Times

- Suppose $T_1(n)$ and $T_2(n)$ are running times of two program fragments.
  - $T_1(n) = O(f(n))$ and $T_2(n) = O(g(n))$
  - Then the running time of $T_1(n)+T_2(n) = O(max(f(n), g(n)))$
- Proof

For some constants c_1, c_2, n_1, n_2.

If $n \geq n_1$ then $T_1(n) \leq c_1 f(n)$ and $n \geq n_2$ then $T_2(n) \leq c_2 g(n)$.

Let $n_o = max(n_1, n_2)$.

If $n \geq n_0$ then $T_1(n) + T_2(n) \leq c_1 f(n) + c_2 g(n) \leq c_1 max(f(n), g(n)) + c_2 max(f(n), g(n)) =$
$(c_1 + c_2) max(f(n), g(n))$

Ordering The Functions

- We order the functions:
  - $\log n \le n \le n*\log n \le n^2 \le n^3 \le 2^n \le n! \le n^n$
  - $O(\log n) \le O(n) \le O(n*\log n) \le O(n^2) \le O(n^3) \le O(2^n) \le O(n!) \le O(n^n)$

A More Concrete View of Run Times

➔ There are 86400 seconds in a day, which to keep the math simple we will assume is approximately 10^5

Growth Rates

n f(n)	log n	n	n log n	n^2	2^n	n!
10	0.003ns	0.01ns	0.033ns	0.1ns	1ns	3.65ms
20	0.004ns	0.02ns	0.086ns	0.4ns	1ms	77years
30	0.005ns	0.03ns	0.147ns	0.9ns	1sec	8.4×10^{15}yrs
40	0.005ns	0.04ns	0.213ns	1.6ns	18.3min	--
50	0.006ns	0.05ns	0.282ns	2.5ns	13days	--
100	0.07	0.1ns	0.644ns	0.10ns	4×10^{13}yrs	--
1,000	0.010ns	1.00ns	9.966ns	1ms	--	--
10,000	0.013ns	10ns	130ns	100ms	--	--
100,000	0.017ns	0.10ms	1.67ms	10sec	--	--
1'000,000	0.020ns	1ms	19.93ms	16.7min	--	--
10'000,000	0.023ns	0.01sec	0.23ms	1.16days	--	--
100'000,000	0.027ns	0.10sec	2.66sec	115.7days	--	--
1,000'000,000	0.030ns	1sec	29.90sec	31.7 years	--	--

Back to Our RAM Model

- Generic one processor model, does one instruction after another, instructions can have different cost
- Basic model $T(n) \approx kC(n)$, where $C(n)$ is a count of the number of instructions
- Input size *n* affects runtime
- Assume you have a runtime of $T(n) = n^2$.
  - How much longer will the algorithm run if you double the input size n?
  - $$\frac{T(2n)}{T(n)} = \frac{(2n)^2}{n^2} = \frac{4n^2}{n^2} = 4$$

Back to Our RAM Model

- What about $T(n) = n^3$, and you double *n*
  - $$\frac{T(2n)}{T(n)} = \frac{(2n)^3}{n^2} = \frac{8n^3}{n^3} = 8$$
- What about $T(n) = 2^n$, and you double *n*
  - $$\frac{T(2n)}{T(n)} = \frac{(2)^{2n}}{2^n} = 2^{2n-n} = 2^n$$
- What about $t(n) = n!$, and you double *n*
  - $$\frac{T(2n)}{T(n)} = \frac{(2n)!}{n!} = 2n \cdot (2n-1) \cdots (n+1)$$

Orders of Growth Cost

- ➔ How long would it take to solve a problem with T(n), where $n = 10^6$
  - 10^6 operations ($1/10^{12}$ operations per second) = 10^{-6} seconds
  - $T(n^2)$, $n^2 = 10^{12}$, 10^{12} operations ($1/10^{12}$ operations per second) = 1 second
  - $T(n^3)$, $n^3 = 10^{18}$, 10^{18} operations ($1/10^{12}$ operations per second) = 10^6 seconds which is approximately 11.6 days
  - $T(2^n)$, with $n = 10^3$, $2^n = 10^{300}$, 10^{300} operations ($1/10^{12}$ operations per second) = 10^{318} seconds, so approximately 10^{313} years assuming 10^5 seconds in a year
    - Overall age of the universe $1.4\text{x}10^{10}$