CS330: Introduction To Algorithm **Spring 2014**

Lecture 2: Asymptotic Analysis

TF: Xianrui Meng *Scribes: Raman Bahdanouski, Ellie Vial*

Exercise 1

Suppose you have algorithms with the five running times listed below.How much slower do each of these algorithms get when you (a) double the input size, or (b) increase the input size by one?

1. n^2;
2. n^3;
3. $100n^2$;
4. $nlog(n)$;
5. 2^n;

Solution

For doubling the input:

1. $f(n) = n^2; f'(2n) = 4n^2$.;So the running time of the algorithm gets 4 times slower.
2. $f(n) = n^3; f'(2n) = 8n^3$.;So the running time of the algorithm gets 8 times slower.
3. $f(n) = 100n^2; f'(2n) = 400n^2$.;So the running time of the algorithm gets 4 times slower.
4. $f(n) = nlog(n); f'(2n) = 2n*log(2n) = 2n*(log(n)+log(2)) = 2n*log(n)+2n*log(2)$.;So the running time of the algorithm increases by $2n*log(n) + 2n*log(2)$.
5. $f(n) = 2^n; f'(2n) = 2^2n; f(2n)-f(n) = 2^2n-2^n = (2^n)*(2^n-1)$;So the running time of the algorithm gets slower by the factor of $(2^n - 1)$.

For increasing the input by one:

1. $f(n) = n^2; f'(n + 1) = n^2 + 2n + 1$.; So the running time of the algorithm increases by $2n + 1$.
2. $f(n) = n^3; f'(n+1) = n^3+3n^2+3n+1$.; So the running time of the algorithm increases by $3n^2+3n+1$.
3. $f(n) = 100n^2; f'(n + 1) = 100(n^2 + 2n + 1)$.; So the running time of the algorithm increases by $100(2n + 1)$.
4. $f(n) = nlog(n); f(n + 1) = (n + 1) * log(n + 1); f(n + 1) - f(n) = (n + 1) * log(n + 1) - nlog(n) = n * log(n + 1) + log(n + 1) - nlog(n) = log(n + 1) + n * (log(n + 1) - log(n))$; So the running time increases by $log(n + 1) + n * (log(n + 1) - log(n))$.
5. $f(n) = 2^n; f'(n + 1) = 2^{n+1} = 2 * 2^n$; So the running time of the algorithm doubles.

Exercise 3

Take the following list of functions and arrange them in ascending order of growth rate. That is, if function g(n) immediately follows function f(n) in your list, it should be the case that f(n) is O(g(n)).

- $f_1(n) = n^{2.5}$
- $f_2(n) = \sqrt{2n}$
- $f_3(n) = n + 10$
- $f_4(n) = 10^n$
- $f_5(n) = 100^n$
- $f_6(n) = n^2 log(n)$

Solution

We can start approaching this problem by putting f_4 and f_5 at end of the list, because these functions are exponential and will grow the fastest. $f_4 < f_5$ because $10 < 100$. Other four functions are polynomial and will grower slower then exponential. We can represent f_1 and f_2 as: $n^{2.5} = n^2 * \sqrt{2n}$; and $\sqrt{2n} = 2n^{0.5}$. Now, we can say that out of all polynomial functions f_2 will be the slowest because it has the smallest degree. Morover, and will be bounded by f_3 because it has a higher degree of 1. Furthermore, f_1 and f_6 will be beween exponential f_4 and f_5 and polynomial f_2 and f_3, because polynomial functions grow slower and both f_4 and f_5 and have the highest degree of 2 out of all other polynomial functions. And f_6 will be bounded by f_1 because $f_6 = n^2 log(n)$ and $f_1 = n^2\sqrt{2n}$ and $log(n) = O(\sqrt{2n})$. Therefore the final order will be: $f_2(n) < f_3(n) < f_6(n) < f_1(n) < f_4(n) < f_5(n)$

Exercise 5

Assume you have functions f and g such that $f(n)$ is $O(g(n))$. For each of the following statemenets, decided whether you think it is true or false and give a proof or counterexample.

1. $log\ f(n)$ is $O(log\ g(n))$.
2. $2^{f(n)}$ is $O(2^{g(n)})$.
3. $f(n)^2$ is $O(g(n)^2)$.

Solution

1. **False.** Disprove by counterexample.

   Suppose $f(n) = 2$ and $g(n) = 1$. This holds under the assumtion stated that $f(n)$ is $O(g(n))$ for all n, then $2 < C1$ for any constant $C > 2$. However, $log\ f(n)$ is not in $O(log\ g(n))$ in the case that $f(n) = 2$ and $g(n) = 1$ since $log\ 2 > C\ * \log(1) = 1 > C * 0$ for any n and any constant C.

   To make the statement hold, assume there exists n_1 such that $g(n) > 2$ for all $n > n_1$. By this assumption, there exists some n_0 for all $n_1 > n_0$, such that $f(n) < C * g(n)$ for some constant C. Thus, when $n > max(n_0, n_1)$, then $\log f(n) < log\ C + log\ g(n) < log\ C * log\ O(g(n))$, which means that $log\ f(n)) = O(log\ g(n))$.

2. **False.** Disprove by counterexample.

   Suppose $f(n) = 2n$ and $g(n) = n$. This holds under the assumtion that $f(n)$ is $O(g(n))$ for all n, $2n < Cn$ for any constant $C > 3$. However, 2^{2n} is not in $O(2^n)$ in the case if $f(n) = 2n$ and $g(n) = n$ because $2^{2n} >> C * 2^n \Rightarrow 1 > C * 0$ for any $n > 0$ and any constant C. Thus $2^{f(n)} > O(2^{g(n)})$

3. **True.**

   $\exists$ some n_0 $\forall n > n_0$, s.t. $f(n) < C * g(n)$ for some constant C,

   Therefore, for any positive n, $f(n)^2 < (C * g(n))^2$ because $f(n)^2 < C^2 * g(n)^2$ for all $C > 0$.