Amber Harding

Analysis of Algorithms
Homework 1

Arthur Nunes-Harwitt

1. Rank the following functions by order of growth. Further, partition the list into equivalence classes such that functions $f(n)$ and $g(n)$ are in the same class if and only if $f(n) \in \Theta(g(n))$. (See page 58 in CLRS for a definition of $\lg^*(n)$.)

$\ln(\ln(n))$	$\lg^*(n)$	$n2^n$	$n^{\lg(lg(n))}$	$\ln(n)$	1 p
$2^{\lg(n)}$	$(\lg(n))^{\lg(n)}$	e^n	$4^{\lg(n)}$	$(n + 1)!$	$\lg(n)$
$(\frac{3}{2})^n$	n^3 √	$(\lg(n))^2$	$\lg(n!)$	2^{2^n}	$n^{1/\lg(n)}$
$\lg^*(\lg(n))$	$2^{2\lg(n)}$	n √	2^n	$n\lg(n)$	$2^{2^{n+1}}$
$\lg(\lg^*(n))$	$2^{\lg^*(n)}$	$(2)^{\lg(n)}$	n^2	$n!$	$(\lg(n))!$

$2^{2^{n+1}}$	>	2^{2^n}	>	$(n + 1)!$	>	$n!$	>	e^n	>	$n2^n$	>
2^n	>	$(\frac{3}{2})^n$	>	$(\lg(n))^{\lg(n)}$	>	$n^{\lg(\lg(n))}$	>	$(\lg(n))!$	>	$(\lg(n))!$	>
n^3	>	n^2	>	$\lg(n!)$	>	$n\lg(n)$	>	$(2)^{\lg(n)}$	>	n	>
$\sqrt{n}$	>	$2^{\sqrt{2lgn}}$	>	$lg^2 n$	>	$(\lg(n))^2$	>	ln(n)	>	$\sqrt{lgn}$	>
ln(ln(n))	>	$4^{\lg(n)}$	>	$2^{lg^* n}$	>	$\lg^*(\lg(n))$	>	$\lg(\lg^*(n))$	>	$\lg^*(n)$	>

$n^{1/\lg(n)}$	>	1	>								

2. Rank the following functions of x by order of growth. Note: the constants a,b,c,k are all greater than one. $\sqrt{k}$ ~~x~~, a^x, x^c,

$a^x > x^c > \sqrt[k]{X} >$ $\log_b(x)$

3. (a) Using the class definition of O, prove that $n = O(n^2)$.

f(n) = n, g(n) = n^2

n < n^2* c

When C >= 1

n < n^2* 1 = True

(b) Using the class definition of O, prove that $n^2 = O(n^2)$.

f(n) = n^2, g(n) = n^2

n^2< n^2 * C

When C >= 2

n^2< n^2* 2 = True

(c) Using the class definition of O, prove that $3n^2 + 5n = O(n^2)$.

f(n) = $3n^2 + 5n$, $g(n)$ = n^2

$3n^2 + 5n$ <= n^2* C

When C >= 8 and n >= 1

3 + 5 <= 1 * 8

True

4a. Given that $\sum_{k=2}^{n} 1/k \leq \ln(n) - \ln(1)$, using the class definition of O, prove that $H_n \in O(\ln(n))$.

For K = 2, N = 2

½ <= ln(2) - 0

0.5 <= 0.693 = True

4b. Given that $\sum_{k=2}^{n} 1/k \geq \ln(n+1) - \ln(2)$, using the class definition of Ω, prove that $H_n \in \Omega(\ln(n))$.

K = 2, N = 2

½ >= ln(3) - ln(2)

0.5 > 0.405

5. (project) Recall the definition of the Fibonacci numbers.

$$F_0 = 0$$
$$F_1 = 1$$
$$F_n = F_{n-1} + F_{n-2}$$

Write a recursive function fib that implements the above recurrence. What is the smallest *n* such that you notice fib running slowly?

N > 11 Function begins to run slower. At N = 30 function runs for over 8 seconds.

6. Consider the following recurrence.

$$f(0; a, b) = a$$
$$f(1; a, b) = b$$
$$f(n; a, b) = f(n-1; b, a+b)$$

(a) Prove using mathematical induction that for any $n \in \mathbb{N}$ if $n > 1$ then $f(n; a, b) = f(n-1; a, b) + f(n-2; a, b)$.

For $n = 2$

$$f(2; a, b) = f(1; a, b) + f(0; a, b)$$

$$f(1; a, a+b) = f(1; a, b) + f(0; a, b)(\because f(n; a, b) = f(n-1; a, b))$$

$$a + b = a + b\,(\because f(0; a, b) = a, f(1; a, b) = b)$$

Therefore True for $n = 2$

Assume true for $n = k$

$f(k; a, b) = f(k-1; a, b) + f(k-2; a, b)$ = True

Prove true for $n = k + 1$

$$f(k+1; a, b) = f(k; a, b) + f(k-1; a, b)$$

$$f(k; a, b) + f(k-1; a, b) = f(k; a, b) + f(k-1; a, b)$$

Also true for $n = k + 1$

Therefore $f(n; a, b) = f(n-1; a, b) + f(n-2; a, b)$ = True

(b) Prove using the strong form of induction that for any $n \in N$, $F_n = f(n; 0, 1)$. You should use the previous result in your proof.

$F_1 = f(1; 0, 1) = 1$

$F_2 = f(2; 0, 1) = f(1; 0, 1) + f(0; 0, 1) = 1 + 0 = 1$

$F_3 = f(3; 0, 1) = f(2; 0, 1) + f(1; 0, 1) = 1 + 0 = 1$

$F_4 = f(4; 0, 1) = f(3; 0, 1) + f(2; 0, 1) = 1 + 1 = 2$

$F_k = F_{k-1} +_{k-2}$

$r > 1$ so that $F_n \geq r^n$

$F_n \geq r^{n-2}$

let $P(n)$ denotes $F_n \geq r^{n-2}$

$P(1)$ is true while $r^{1-2} = r^{-1} \leq 1$ and $P(2)$ is also true because $F_2 = 1$ and $r^{2-2} = r^0 = 1$

$F_{n+1} = F_n + F_{n-1}$

$F_{n+1} \geq r^{n-2} + r^{n-3} \rightarrow 2$

$F_{n+1 \geq} r^{n-3}(r + 1) \rightarrow 3$

$r^2 = r + 1$

$F_{n+1} \geq r_{n-3}(r + 1) = r^{n-3} r^2 = r^{n-1}$

7. (project) Write a recursive function fibItHelper that takes three arguments, *n*, *a*, and *b*; it should implement the recurrence *f*. Then write a function fibIt that calls fibItHelper initializing *a* to 0 and *b* to 1. Does fibIt also run slowly on the value of *n* that you found made fib run slowly?

No fibIt is much faster than the fib function.