CSCI 5454: PS1

Robert Werthman

1.

Let's say these algorithms solve an array sorting problem.

- Let algorithm A be bubblesort with a worst-case runtime of n^2.
- Let algorithm B be mergesort with a worst-case runtime of $n * log(n)$.
- Let C be the newly designed sorting algorithm with a worst-case runtime of $h(n)$.

In this case, $O(min(f(n), g(n)))$ will become $O(n * log(n))$ because it is the smaller of the two runtimes.
If $h(n)$ is $log(n)$ then $h(n)$ achieves the running time $O(min(f(n), g(n)))$ because $log(n)$ does not grow faster than $n * log(n)$ and is therefore bounded above by it.

Yes, you can achieve a running time exactly $min(f(n), g(n))$. Algorithm C would need to be designed in such a way that its running was equal to $min(f(n), g(n))$.

2.

Proposition/Claim: For any real constants a and b, where $b > 0$, the asymptotic relation $(n+a)^b = \Theta(n^b)$ is true.

Theorem: The asymptotic relation $(n+a)^b = \Theta(n^b)$ is true iff:

- There exists positive constants c_1, c_2, n_0 s uch that $0 \le c_1(n^b) \le (n+a)^b \le c_2(n^b)$ for all $n \ge n_0$.

In order to prove the proposition above we must find some constants c_1, c_2, n_0 to satisfy the above bulleted sentence.

Proof:
First we want to find the floor and ceiling of $n + a$ so we can create an inequality similar to the one in the theorem above.

1. If $|a| \le n$ then we can say that $n + a \le n + |a| \le 2n$ (Ceiling of $n + a$).
2. If $|a| \le \frac{1}{2}n$ then we can say that $n + a \ge n - |a| \ge \frac{1}{2}n$ (Floor of $n + a$).

Now if $2|a| \le n$ then we can combine the floor and ceilings into an compound inequality that holds true :

$$0 \le \frac{1}{2}n \le n + a \le 2n$$

The only thing missing from this new equation is a power of b. Raising the new equation to a power of b gives:

$$0 \le (\frac{1}{2}n)^b \le (n+a)^b \le (2n)^b \Rightarrow 0 \le (\frac{1}{2})^b n^b \le (n+a)^b \le (2)^b n^b$$

Extracting the constants c_1, c_2, n_0 from this equation yields $c_1 = (\frac{1}{2})^b$, $c_2 = 2^b$, and $n_0 = 2|a|$ since $n \ge 2|a|$. These represent one solution.

3.

$f(n) = \Omega g(n)$ means that for all values to the right of some n_0 the value of $f(n)$ is on or above $cg(n)$.

$n!$	e^n	$(\frac{3}{2})^n$	$(lg\,n)!$	n^2	$n\,lg\,n$	$lg(n!)$	n	$(\sqrt{2})^{lg\,n}$	2^{lg*n}	$n^{1/lg\,n}$	1

Equivalence Classes

$lg(n!) = \Theta(n\,lg\,n)$
$n^{1/lg\,n} = \Theta(1)$

4.

a.

$T(n) = T(n-1) + n$, $T(1) = 1$
I will a recurrence tree to solve this recurrence relation.

$$\begin{array}{c} n \\ | \\ n-1 \\ | \\ n-2 \\ | \\ 2 \\ | \\ \vdots \\ | \\ 1 \end{array}$$

The height of the tree is n and the cost at the root starts at n and decreases by 1 each level in the tree.
This means that the total cost of the tree is n.
So $T(n) = O(cost * depth) = O(n^2)$.

b.

$T(n) = 2T(n/2) + n^3$, $T(1) = 1$
I will use the master method to solve this recurrence relation.
$a = 2, b = 2, f(n) = n^3$
so $n^{\log_b a} = n^{\log_2 2} = n$
This tells us that the first 2 rules of the master theorem do not apply.

1. $f(n) \neq O(n^{1-\epsilon})$

2. $f(n) \neq \Theta(n)$

This leaves the 3rd rule of the master theorem as the solution.

3. $f(n) = n^3 = \Omega(n^{1+\epsilon})$ if $\epsilon = 1$. And $2f(n/2) \leq cf(n) \Rightarrow 2(n/2)^3 \leq cn^3$ if $c = \frac{1}{2}$ and $n \geq 1$.

Therefore, $T(n) = \Theta(n^3)$.

5.