Algorithm Midterm

2012 fall

By Chun-An Chen

1.(20%) (1) (10%) Give formal definitions of $\Theta(g(n)), O(g(n))$, and $\Omega(g(n))$. (2) (10%) Prove or disprove: Can any two functions be compared using asymptotic notation ?

(a)

$\Theta(g(n))$ = { f(n) : there exist positive constants c_1, c_2 and n_0 such that $0 \le c_1\, g(n) \le f(n) \le c_2\, g(n)$ for all $n \ge n_0$ }.

$O(g(n))$ = { f(n) : there exist positive constants c and n_0 such that $0 \le f(n) \le c\, g(n)$ for all $n \ge n_0$ }.

$\Omega(g(n))$ = { f(n) : there exist positive constants c and n_0 such that $0 \le c\, g(n) \le f(n)$ for all $n \ge n_0$ }.

(b)

$f(n) = n, \quad g(n) = n^{1+\sin n}$

(1+sin n) oscillates between 0 and 2.

$\therefore$ this case is neither $f(n) = O(g(n))$ nor $f(n) = \Omega(g(n))$.

2.(10%) Use the master Theorem to solve $T(n) = 2T(\frac{n}{2}) + n\lg n$.

$f(n) = \Theta(\ n^{\log_b a} \lg^k n\ )$, where $k \geq 0$
($f(n)$ is within a polylog factor of $n^{\log_b a}$, but not smaller)

Solution: $T(n) = \Theta(\ n^{\log_b a} \lg^{k+1} n)$

$f(n) = n \lg n = \Theta(\ n \lg n\ ) = \Theta(\ n^{\log_b a} \lg^k n\ )$, where $a=b=2$, $k=1$

$\Rightarrow T(n) = \Theta(\ n \lg^2 n)$

3.(10%) What are the minimum and maximum number of elements in a heap of height h ?

Since a heap is an almost-complete binary tree (complete at all levels except possibly the lowest),

at most $2^{h+1} - 1$ elements (if it is complete) and
at least 2^h elements (if the lowest level has just 1 element and the other levels are complete)

4.(20%) Show that any comparison sort algorithm requires $\Omega(n \lg n)$ comparisons in the worst case.

Proof From the preceding discussion, it suffices to determine the height of a decision tree in which each permutation appears as a reachable leaf. Consider a decision tree of height h with l reachable leaves corresponding to a comparison sort on n elements. Because each of the $n!$ permutations of the input appears as some leaf, we have $n! \le l$. Since a binary tree of height h has no more than 2^h leaves, we have

$$n! \le l \le 2^h$$

which, by taking logarithms, implies

$h \ge \lg(n!)$ (since the lg function is monotonically increasing)

$= \Omega(n \lg n)$ (by equation (3.18)).

5.(15%) Describe an algorithm that, given n integers in the range 0 to k, preprocesses its input and then answers any query about how many of the n integers fall into in a range $[a...b]$ in O(1) time. Your algorithm should use $\Theta(n+k)$ preprocessing time.

Compute the C array as is done in counting sort. The number of integers in the range $[a\,..\,b]$ is $C[b] - C[a-1]$, where we interpret $C[-1]$ as 0.

	1	2	3	4	5	6	7	8	9	10	11	12
A	1	3	5	7	9	8	6	2	6	1	7	3

	0	1	2	3	4	5	6	7	8	9
C	0	2	1	2	0	1	2	2	1	1

↓

	0	1	2	3	4	5	6	7	8	9
C	0	2	3	5	5	6	8	10	11	12

6.(15%) Present the quicksort algorithm.

```
QUICKSORT(A, p, r)
if p < r
   then q ← PARTITION(A, p, r)
        QUICKSORT(A, p, q - 1)
        QUICKSORT(A, q + 1, r)

PARTITION(A, p, r)
x ← A[r]
i ← p - 1
for j ← p to r - 1
     do if A[j] ≤ x
           then i ← i + 1
                exchange A[i] ↔ A[j]
exchange A[i + 1] ↔ A[r]
return i + 1
```

7.(10%) Express the function $\frac{n^3}{1000} - 1000n^2 - 100n + 3$ in terms of Θ-notation.

$$c_1 n^3 = \frac{1}{2000} n^3 \le \frac{n^3}{1000} - 1000n^2 - 100n + 3 \le n^3 = c_2 n^3$$

$$\Rightarrow \frac{n^3}{1000} - 1000n^2 - 100n + 3 = \Theta(n^3)$$