Maths 182 HW 1

Ken DENG
UCLA ID# 205452321

Question 1

We let $f(n) = \log_b(n)$, WTS $f(n)$ is both $O(\log_2(n))$ and $\Omega(\log_2(n))$.

To show $f(n)$ is $O(\log_2(n))$: $f(n) = \log_b(n) = \frac{\log_2(n)}{\log_2(b)} = \frac{1}{\log_2(b)} \cdot \log_2(n)$, and notice that both $\frac{1}{\log_2(b)}$ and $\log_2(n)$ are positive, so we set $c = 1 + \frac{1}{\log_2(b)} > \frac{1}{\log_2(b)} > 0$.
Therefore, $f(n) = \frac{1}{\log_2(b)} \cdot \log_2(n) \leq (1 + \frac{1}{\log_2(b)}) \cdot \log_2(n) = c\log_2(n)$, which implies $f(n)$ is $O(\log_2(n))$.

To show $f(n)$ is $\Omega(\log_2(n))$: $f(n) = \log_b(n) = \frac{\log_2(n)}{\log_2(b)} = \frac{1}{\log_2(b)} \cdot \log_2(n)$, and notice that both $\frac{1}{\log_2(b)}$ and $\log_2(n)$ are positive, so we set $c' = \frac{1}{2 \cdot \log_2(b)} < \frac{1}{\log_2(b)}$, and we have $c' > 0$.
Therefore, $f(n) = \frac{1}{\log_2(b)} \cdot \log_2(n) \geq (\frac{1}{2 \cdot \log_2(b)}) \cdot \log_2(n) = c'\log_2(n)$, which implies $f(n)$ is $\Omega(\log_2(n))$.

Combine them together, we have $f = O(\log_2(n))$ and $f = \Omega(\log_2(n))$, so $f = \Theta(\log_2(n))$.

Question 2

$a(n) = n^2 + n3^n = \Theta(n3^n)$
$b(n) = \log(\log n) + 8log(n) = \Theta(\log n)$
$c(n) = \sqrt{n}\log^3 n + 3n\log n = \Theta(n\log n)$
$d(n) = n^2 \log n - 112n^2 + 23 = \Theta(n^2 \log n)$
$e(n) = \sqrt{n} + n\log^{100} n = \Theta(n\log^{100} n)$
$f(n) = \frac{n}{logn+1} = \Theta(\frac{n}{\log(n)})$
$g(n) = n\log(n^2) = \Theta(n\log n)$
$h(n) = n^{1/\log n} = \Theta(1)$
$i(n) = \sqrt{7^n} - n^{10} = \Theta((\sqrt{7})^n)$
$j(n) = 10^{10} = \Theta(1)$
$k(n) = n\log(2^n + n^3) = \Theta(n^2)$
$l(n) = \max(\sqrt{n^2 + 5n}, n^{2/3} + 1000)\ \Theta(n)$

Rank by increasing order:
$h = j < b < f < l < c = g < e < k < d < i < a$.
Note there are two pairs with the same order of growth: h with j and c with g.

Question 3

We let $f(N)$ be the number of digits we have to write down.
claim: $f(N) = \Theta(N \log N)$.

Then we need to show the lower bound is $c_1(\log(N))$ and upper bound is $c_2(\log(N))$, for some constant $c_1, c_2 > 0$.

For upper bound, we see that the largest number, N, has at most $(\log_{10} N) + 1$ digits, so every number from 1 to N has at most $(\log_{10} N) + 1$ digits. In total, it is $N((\log_{10} N) + 1)$ digits. Notice when N is large enough, $N((\log_{10} N) + 1) \leq N((\log_{10} N) + (\log_{10} N)) \leq 2N(\log_{10} N) = \frac{2}{\log 10} \cdot N \log N$. Let c_1 be $\frac{2}{\log 10}$, we have $c_1 N \log N$ is an upper bound of $f(N)$. Namely, $f(N) = O(N \log N)$.

For lower bound, we see that every number from $\lfloor \frac{N}{2} \rfloor$ to N has at least $\log N$ digits. In total, it is $\lceil \frac{N}{2} \rceil \log N$ digits. Notice when N is large enough, $\lceil \frac{N}{2} \rceil \log N \geq \frac{N-1}{2} \log N \geq \frac{N - \frac{N}{2}}{2} \log N = \frac{1}{4} N \log N$. Let c_2 be $\frac{1}{4}$, we have $c_2 N log N$ is a lower bound of $f(N)$. Namely, $f(N) = \Omega(N \log N)$.

Now, since $f(N)$ is bounded below by $c_1 N \log N$ and bounded above by $c_2 N \log N$, we have $f(N)$ has a tight bound as $f(N) = \Theta(N \log N)$.

Question 4

```
for d = 1,...,N do ---------------------------------- O(N) iterations    #1
    Print(On the d-th day of Christmas my true love gave to me); - O(1)   #2
    for k = d,...,1 do  ---------------- d iterations = O(N) iterations   #3
        if k=1then ------------------------------------------------ O(1)   #4
            Print(A partridge in a pear tree); ------------------- O(1)   #5
        else ------------------------------------------------------ O(1)   #6
            Print(k golden rings); ------------------------------ O(1)   #7
        end
    end
end
```

Claim: the big-O bound is $O(N^2 \log N)$.
We prove the upper bound:
The inner loop has d iterations, and we have $d \leq N$, so the inner loop has $O(N)$ iterations. For each iteration, it at most print $O(\log N + c)$ times characters as in line #7, as the variable k can have at most $O(\log N)$ digits and others are just constant numbers of characters. So the each inner loop prints $O(\log N + c) = O(\log N)$ characters.
Then the outer loop also $O(N)$ iterations. For each iteration, it prints at most $c' + \log N + 1$ times

characters as in line #2, as the amount of characters except for d is a constant c', and d has at most $logN+1$ digits (characters) as $d=N$. So in total, line #2 prints $O(c'+\log N+1)=O(\log N)$ in each iteration. Thus, each outer loop prints $O(\log N)+O(N)\cdot O(\log N)=O(N\log N)$ characters at most. (Here, O(N) is the maximum number of iterations of inner loops can be executed inside each outer loop). And since there is $O(N)$ iterations of outer loop, the overall maximum number of characters to be printed is $O(N)\cdot O(N\log N)=O(N^2\log N)$.

Question 5

5a:
We set a constant natural number i. if the egg breaks after released from the i-th floor, then we have to test all the floors from $1-$st to $(i-1)$-th floor one by one with the other egg. If an egg is not broken after released from the i-th floor, then we do not have to test for all the 1st to i-th floors, and we test for the $2i$-th floor. If it breaks on the $2i$-th floor, we use test all the floors from $(i+1)$-th to $(2i)$-th floor one by one with the other egg. if it does not break on the $2i$-th floor, we move on to the $3i$-th floor, and so forth. In general, on the $k\cdot i$-th floor, where k is some natural number, if an egg breaks, we test all the floors from $((k-1)\cdot i+1)$th to $(k\cdot i)$-th floor one by one with the other egg. And we let i be the smallest natural number with $i\geq\sqrt{N}$. Notice that $i\leq\sqrt{N}+1$, and we have to try at most $i+i=2i\leq 2(\sqrt{N}+1)=O(\sqrt{N})$ times. (The first i indicates the maximum number of floors the first egg have to try, and the second i indicates the maximum number of floors the second egg have to try from $(k-1)i+1$-th floor to ki-th floor.)

5b:
Observe that once the first egg is broken, we then have only one egg left, and the issue becomes using only 1 egg to find the threshold, where we can only use linear probing to test floors from the highest level not broken to the level broken. We set the maximum amount of floors the second egg has to test as j. Then for the first egg, to ensure that the second egg will only be tested for j times at most, it can only skip j levels at most from m-th floor to $(m+j+1)$-th floor. We let the maximum amount of floors the first egg has to test be k, then we must have $k\cdot j\geq N$. By the inequality of arithmetic and geometric means, we have the total steps $k+j\geq 2\sqrt{kj}\geq 2\sqrt{N}=\Omega(\sqrt{N})$ in the worst case.

Or we can also use recursion. (Idea by TA) Then we should have $cT^2\geq\binom{T}{2}+\binom{T}{1}+1=\frac{T(T-1)}{2}+T+1\geq n$. So $T\geq\sqrt{\frac{n}{c}}=O(\sqrt{N})$. Thus, $T=\Omega(\sqrt{N})$.

5c:
We set $x\in\mathbb{N}$ such that x is the smallest number where $x\geq\sqrt[k]{N}$. Then we use the first egg to test for $(1x^{k-1})$-th, $(2x^{k-1})$-th, ... ,$(x\cdot x^{k-1})$-th floor.

Once it breaks on (h_1x^{k-1})-th floor, we use the second egg to test for
$(h_1x^{k-1}+1x^{k-2})$-th, $(h_1x^{k-1}+2x^{k-2})$-th, ... , $(h_1x^{k-1}+x\cdot x^{k-2})$-th floor, and so forth.

For the g-th egg, we will be use it to test for

$(h_1x^{k-1} + h_2x^{k-2} + ... + h_{g-1}x^{k-g+1} + 1x^{k-g})$-th, $(h_1x^{k-1} + h_2x^{k-2} + ... + h_{g-1}x^{k-g+1} + 2x^{k-g})$-th, ... , $(h_1x^{k-1} + h_2x^{k-2} + ... + h_{g-1}x^{k-g+1} + x \cdot x^{k-g})$-th floor.

So each egg is tested for at most x times, in total, it's $kx = O(x) = O(1 + \sqrt[k]{N}) = O(\sqrt[k]{N})$ times.