Algorithm

HW#01

Weijun Zhu.

1. $g(n) = n^2(\sin(n)+1)$; $f(n) = n$

20

$\because -1 \leq \sin(n) \leq 1$ $\therefore 0 \leq \sin(n)+1 < 1$

First, let's check $f(n) = O_*(g(n))$

$\Rightarrow f(n) \leq C \cdot g(n)$.

$\Rightarrow n \leq C \cdot n^2(\sin(n)+1)$

$\Rightarrow 1 \leq C \cdot n(\sin(n)+1)$.

Assume that $C=1$, We have $1 \leq n \cdot (\sin(n)+1)$

If $n = k \cdot \frac{\pi}{2}$, $k = 1, 5, 9 \ldots$, we will get $\sin(n) = 1$.

$\Rightarrow 1 \leq 2n$, and there are infinite n to make $2n \geq 1$

So we get $f(n) \leq C \cdot g(n)$ for infinitely values n.

Last, let's check $g(n) = O_*(f(n))$

$\Rightarrow g(n) \leq C \cdot f(n)$.

$\Rightarrow n^2(\sin(n)+1) \leq C \cdot n$

$\Rightarrow n(\sin(n)+1) \leq C$.

We still assume that $C=1$, We have $n \cdot (\sin(n)+1) \leq 1$

If $n = k \cdot \frac{\pi}{2}$, $k = 3, 7, 11 \ldots$, we will get $\sin(n) = -1$

$\Rightarrow n \cdot 0 \leq 1$, and there are infinitely n to make $0 \cdot n \leq 1$

So we get $g(n) \leq C \cdot f(n)$ for infinitely values n.

Both A and B are correct.

2. Assume that $\begin{cases} f(n)=2n \\ g(n)=n \end{cases}$, and $f(n)=O(g(n))$

20

If $2^{f(n)}=O(2^{g(n)})$, we get $2^{2n}=O(2^n)$

$\Rightarrow 2^{2n} \leq c \cdot 2^n$.

$\Rightarrow c \geq 2^n$.

But we can not find a positive constant which greater than all of 2^n.

So $2^{f(n)} \neq O(2^{g(n)})$

By the Theorem, $f(n)=\Theta(g(n))$ if and only if $f(n)=O(g(n))$ and $f=\Omega(g(n))$.

So we can not get $2^{f(n)}=\Theta(2^{g(n)})$ because $2^{f(n)} \neq O(2^{g(n)})$

Also, if $f(n)=\Theta(g(n))$,

We will have $0 \leq c_1 \cdot g(n) \leq f(n) \leq c_2 \cdot g(n)$

We can not find a constant s.t. $2^{f(n)} \leq c_2 2^{g(n)}$ from previous proof.

For $0 \leq c_1 \cdot g(n) \leq f(n)$.

$\Rightarrow c_1 \cdot 2^{g(n)} \leq 2^{f(n)}$ & $g(n)=n$ $f(n)=2n$

$\Rightarrow c_1 \cdot 2^n \leq 2^{2n}$

$\Rightarrow c_1 \leq 2^n$

But we can not find a positive constant c_1 which smaller than all of 2^n

So $2^{f(n)} \neq \Theta(2^{g(n)})$

3(a). $0.9^n < 1 \approx \frac{n}{2n+1} \approx \sin(n) < \log n \approx \log n + \frac{\log n}{n} < n^{\frac{1}{3}} < \lfloor n^2 \rfloor \approx \lfloor n \rfloor^2 < n^3$

18

$< n^{\log n} < 3^{\sqrt{n}} < 1.0001^n < 3^n < 2^{3\ln n} < n! < n^n$

(b). a and b both greater than 1.

If we want $\log_a(n) = \Theta(\log_b(n))$, we will prove

$$0 \le C_1 \log_b(n) \le \log_a(n) \le C_2 \log_b(n)$$

let $\log_a b = x$, and we have $a^x = b$

$$a^x = b \Rightarrow a^{\log_a b} = b$$

$$\Rightarrow (a^{\log_a b})^{\log_b n} = b^{\log_b n} = n$$

$$\Rightarrow \log_a b \cdot \log_b n = \log_a n$$

$\therefore$ If we choose $C = \log_a b$, we will have $C \cdot \log_b n = \log_a n$

If we choose $C_1 = C_2 = C$, we will get $\log_a n = \Theta(\log_b n)$

$P(X) = P_0 + X(P_1 + X(P_2 + X(P_3 \cdots)))$

4.(a). Let's define the horner function: horner(P, X), and $P = [P_0, P_1, P_2 \ldots P_n]$ and $X = \alpha$.

20

We have pseudocode:

```
If length(P) == 1:
    return P[1]
else:
    P(X) = P[1] + X·horner(P[2:n+1], X)
    return P(X).
```

(b). $f(n) = \begin{cases} C_1 = 1 & : n = 1 \\ f(n-1) + C_2 & : n > 1 \end{cases}$ Define C_1 & C_2 are positive constant

(c). Let's consider $f(n) = f(n-1) + C_2$.

$= (f(n-2) + C_2) + C_2$

$= (f(n-3) + C_2) + 2C_2$

$\vdots$

$= f(1) + (n-1)C_2$

$= C_2 + (n-1)C_2 = C_2 n$

By definition of Θ: $f(n) = C_2 n$ & $g(n) = n$

There exist two positive constant C_3 & C_4 s.t.

$C_3 n \leq C_2 n \leq C_4 n$

$\Rightarrow C_3 \leq C_2 \leq C_4$

$\Rightarrow \Theta(n)$

By definition of O: $f(n) = C_2 n$ & $O(n)$.

$\Rightarrow C_2 \cdot n \leq C \cdot n$

$\Rightarrow C \geq C_2$

$\Rightarrow O(n)$.

5(a). $L = [a_1, a_2, \dots, a_n]$ and $a_1 \le a_2 \le \dots \le a_n$

20

x is new item

Define the Binarysearch function: binarysearch(L, x)

Code:
```
mid = floor(length(L)/2)   # floor is floor function "⌊x⌋"
   if L[mid] == x:
       return mid
   if length(L) == 1:
       if L[0] > x:
           return "x is not in the L"
       if L[0] < x:
           return "x is not in the L"
   if L[mid] < x:
       return mid + binarysearch(L[mid:n], x)
   if L[mid] x:
       return binarysearch(L[1:mid], x).
```

(b). $f(n) = \begin{cases} f(\frac{n}{2}) + C_2 & ; n > 1 \\ C_1 = 1 & ; n = 1 \end{cases}$

$f(n) = f(\frac{n}{2}) + C_2$. Suppose $n = 2^k$, and $k = \log_2 n$

$$
\begin{aligned}
f(2^k) &= f(2^{k-1}) + C_2 \\
&= f(2^{k-2}) + 2C_2 \\
&= f(2^{k-3}) + 3C_2 \\
&\vdots \\
&= f(1) + kC_2
\end{aligned}
$$

$$
\begin{aligned}
\Rightarrow f(n) &= f(1) + \log_2 n \cdot C_2 \\
&= C_2 \cdot \log_2 n
\end{aligned}
$$

By definition of Θ: We have $f(n) = C_2 \cdot \log_2 n$ $\quad g(n) = \log_2 n$

There exists two positive constants C_3 & C_4.

$$0 \le C_3 \cdot \log_2 n \le C_2 \cdot \log_2 n \le C_4 \cdot \log_2 n.$$

$$\Rightarrow C_3 \le C_2 \le C_4$$

So we have $\Theta(\log_2 n)$