HW1

UNI: yb2356 Name: Yang Bai

1

f	g	O	o	Ω	ω	Θ
$\log^2 n$	$25 \log n$	No	no	yes	yes	No
$\sqrt{\log n}$	$(\log \log n)^4$	no	no	yes	yes	No
$3n \log n$	$n \log 3n$	yes	no	yes	no	yes
$n^{3/5}$	$\sqrt{n} \log n$	No	no	yes	yes	No
$\sqrt{n} + \log n$	$2\sqrt{n}$	yes	no	yes	no	yes
$n^2 2^n$	3^n	yes	yes	no	no	No
$\sqrt{n} 2^n$	$2^{n/2+\log n}$	No	no	yes	yes	No
$n \log 3n$	$\frac{n^2}{\log n}$	yes	yes	no	no	No
$n!$	2^n	No	no	yes	yes	No
$\log n!$	$\log n^n$	yes	yes	no	no	No

2.

$$f(n) = \frac{1-\lambda^{n+1}}{1-\lambda} \qquad (\lambda \neq 1)$$

(1) If $0 < \lambda < 1$:

$0 < 1-\lambda < 1$, so $f(n) > 1-\lambda^{n+1}$ for all $n \geq 1$

$0 < \lambda^{n+1} < 1$, so $f(n) < \frac{1}{1-\lambda}$, for all $n \geq 1$

$$(1-\lambda^{n+1}) \cdot 1 \leq f(n) \leq \frac{1}{1-\lambda} \cdot 1$$

Thus $f(n) = \Theta(1), if\ 0 < \lambda < 1.$ $c_1 = (1-\lambda^{n+1}), c_2 = \frac{1}{1-\lambda}, n_0 = 1$

(2) If $\lambda = 1$:

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

$f(n) \geq 1 \cdot n$ for all $n \geq 1$, $c_1 = 1$
$f(n) \leq 2 \cdot n$ for all $n \geq 1$, $c_2 = 2$
Thus $f(n) = \Theta(n), if\ \lambda = 1$

(3) If $\lambda > 1$:

Let $c_1 = \frac{1}{2}$, $f(n) \geq \frac{\lambda^n}{2}$ for all $n \geq 1$.

Proof:

$$\frac{\lambda^{n+1}-1}{\lambda-1} \geq \frac{\lambda^n}{2}$$

$$\lambda^{n+1} - 1 \geq \frac{\lambda^{n+1}}{2} - \frac{\lambda^n}{2}$$

$$\frac{\lambda^n}{2}(\lambda + 1) \geq 1$$

if $\lambda > 1$, the above inequality holds.
Let $c_2 = \lambda^2$, $f(n) \leq \lambda^{n+2}$ for all $n \geq 1$.
Proof:

$$\frac{\lambda^{n+1}-1}{\lambda-1} \leq \lambda^{n+2}$$

$$\lambda^{n+1}(\lambda^2 - \lambda - 1) + 1 \geq 0$$

if $\lambda > 1$, the above inequality holds.
Thus, $f(n) = \Theta(\lambda^n), if\ \lambda > 1$

3.

(1) Let $f_1(n) = \frac{1}{n}$, $f_1(2n) = \frac{1}{2n}$.

For all $n \geq 1$, $f_1(2n) \leq 1 \cdot f_1(n)$. So $f_1(2n) = O(f_1(n))$.

(2) Let $f_1(n) = 2^n$, $f_1(2n) = 2^{2n}$

For all $n \geq 1$, $f_1(n) < 1 \cdot f_1(2n)$. So $f_1(2n)$ is not $O(f_1(n))$.

(3) If $f = o(g)$, for any $c > 0$, there exist $n0$ that, for $n \geq n0, f(n) < g(n)$.

Pick a number c1 (c1 $>$ 0), we can find n1 that , for $n \geq n1, f(n) \leq g(n)$

$$so\ f = O(g)$$

4.

(a)

$$T(n) = n + 1 + \frac{n(n+1)}{2} + \sum_{k=1}^{n} \frac{(k-1)k}{2} + \frac{(n-1)n}{2}$$

$$= \frac{5}{4}n^2 + \frac{5}{4}n + 1 + \frac{n(n+1)(2n+1)}{12}$$

$$= \frac{1}{6}n^3 + \frac{3}{2}n^2 + \frac{4}{3}n + 1$$

Suppose $f(n) = n^3$. For $n \geq 1, T(n) \leq 5n^3$. So $T(n) = O(n^3)$.

(b) For $n \geq 1, T(n) \geq \frac{1}{6}n^3$. So $T(n) = \Omega(n^2)$.

Hence $T(n) = \Theta(n^2)$.

(c)

pseudocode:

```
for i=1,2,3…,n do
        B[i, i] = A[i];
        for j = i+1, i+2,…,n do
              B[i, j] = B[i, j-1] + A[j];
        End for
End for
```

$$T'(n) = n + 1 + n + \frac{n(n+1)}{2} + \frac{n(n-1)}{2}$$

$$= n^2 + 2n + 1$$

$$T'(n) = O(n^2)$$

$$n^2 = o(n^3)$$

5.

We can use master's theorem for the first three T(n).

$$T(n) = \sum_{i=0}^{log\ _b n} a^i c(\frac{n}{b^i})^k = cn^k \sum_{i=0}^{log\ _b n} (\frac{a}{b^k})^i$$

(1) a=4, b=2, c=1, k=3.

$$T(n) = 2n^3\left(1 - \frac{1}{n}\right) = 2n^3 - 2n^2$$
$$T(n) \le 2n^3\ for\ all\ n \ge 0\ ;$$
$$T(n) \ge \frac{n^3}{2}\ for\ all\ n \ge 2\ ;$$

so $T(n) = \Theta(n^3)$

(2) a=8, b=2, c=1, k=2. a> b^k

$$T(n) = n^2(n-1)$$
$$T(n) \le n^3\ for\ all\ n \ge 0\ ;$$
$$T(n) \ge \frac{n^3}{2}\ for\ all\ n \ge 2\ ;$$

so $T(n) = \Theta(n^3)$

(3) a=6, b=3, c=1, k=1. a> b^k

$$T(n) = n\left(2^{\log_3 n} - 1\right) = n\left[(3^{\log_2 3})^{\log_3 n} - 1\right] = n(n^{\log_2 3} - 1)$$
$$= n^{\log_2 3+1} - n = n^{\log_2 6} - n$$
$$T(n) \le n^{\log_2 6}\ for\ all\ n \ge 0\ ;$$
$$T(n) \ge \frac{n^{\log_2 6}}{2}\ for\ all\ n \ge 2\ ;$$

so $T(n) = \Theta(n^{\log_2 6})$

(4) Suppose $n = 2^m,\ m = \log_2 n$

$$T(2^m) = T\left(2^{\frac{m}{2}}\right) + 1$$

Set $S(m) = T(2^m)$. The new equation is:

$$S(m) = S\left(\frac{m}{2}\right) + 1$$

We can use Master's Theorem for this new S(m).
a=1, b=2, k=0.   a=b^k
So $S(m) = \Theta(\lg m),\ T(n) = T(2^m) = S(m) = \Theta(\lg m) = \Theta(\log_2 \log_2 n)$

6.

(1) It's easy to calculate that $F6 = 8, F7 = 13$ and that $\mathrm{F6} \geq 2^{\frac{6}{2}}, F_7 \geq 2^{\frac{7}{2}}$.

Then we need to prove that if $F_n \geq 2^{\frac{n}{2}}$ *and* $F_{n+1} \geq 2^{\frac{1+n}{2}}$, $F_{n+2} \geq 2^{\frac{n+2}{2}}$ holds.

$$F_{n+2} = F_{n+1} + F_n \geq 2^{\frac{1+n}{2}} + 2^{\frac{n}{2}}$$

$$F_{n+2} \geq 2^n$$

$n \geq 6,\ so\ n \geq \frac{n+2}{2},\ \ F_{n+2} \geq 2^{\frac{n+2}{2}}$

Thus, $F_n \geq 2^{\frac{n}{2}}$ holds for all $\mathrm{n} \geq 6$.

(2)

- Pseudocode(recursive)

```
Function Fibonacci(n )
If n ==0 return 0;
If n==1 return 1;
Return Fn = Fibonacci(n-1) + Fibonacci (n-2)
```

Running time:

$$\begin{aligned} T(n) &= T(n-1) + T(n-2) = T(n-2) + T(n-3) + T(n-3) + T(n-4) \\ &= T(n-3) + T(n-4) + T(n-4) + T(n-5) + T(n-4) + T(n-5) \\ &\quad + T(n-5) + T(n-6) \\ &= \cdots\cdots\cdots\cdots\cdots\cdots\cdots\cdots\cdots\cdots\cdots\cdots\cdots\cdots\cdots\cdots\cdots\cdots\cdots\cdots\cdots\cdots\cdots\cdots \end{aligned}$$

If n is odd, $T(n) = 2^{\frac{n+1}{2}+1} - 3$; if n is even, $T(n) = 2^{\frac{n}{2}+1} - 1$.

So $T(n) = \Omega\left(2^{\frac{n}{2}}\right)\ (n \geq 1)$.

- pseudocode (non-recursive)

```
Function Fibonacci(n )
a= 0;
b= 1;
For i=2 to n
   c = a+ b
   a = b
   b = c
End for
Return c
```

Running time:

$$T(n) = 2 + n + 3 * (n - 1) = 4n - 1$$
$$T(n) = \Theta(n)$$

- pseudocode (using matrix)

$$\begin{pmatrix} F(n) & F(n+1) \\ F(n+1) & F(n+2) \end{pmatrix} = \begin{pmatrix} 0 & 1 \\ 1 & 1 \end{pmatrix}^n$$

```
// calculate the product of two 2D matrix
Function Matrix_multiply(M[2][2], P[2][2])
A[0][0] = M[0][0]*P[0][0]+ M[0][1]*P[1][0]
A[0][1] = M[0][0]*P[1][0]+ M[0][1]*P[1][1]
A[1][0] = M[1][0]*P[0][0]+ M[1][1]*P[1][0]
A[1][1] = M[1][0]*P[1][0]+ M[1][1]*P[1][1]
Return A

// calculate the power of matrix, recursively
Function Matrix_power(M[2][2], n )
If n==1    return M
Temp = Matrix_power(M, n/2)
Temp = Matrix_multiply(Temp,Temp)
If n is odd, Temp = Matrix_multiply(Temp,M)
Return Temp

Function Fibonacci(n )
If n ==0     return 0
M = [0,1; 1,1]
A = Matrix_power(M, n)
Return A[1][1]
```

Running time:

$$T(n) = T\left(\frac{n}{2}\right) + O(1)$$

So $T(n) = O(\log n)$

(3)

$$\mathrm{F(n)} = \frac{\sqrt{5}}{5}\left(\left(\frac{1+\sqrt{5}}{2}\right)^n - \left(\frac{1-\sqrt{5}}{2}\right)^n\right)$$

F(n) has $\lceil \log_2 F(n) + 1 \rceil$ bits. So adding $F(n-1)$ and $F(n-2)$ costs $\lceil \log_2 F(n-1) + 1 \rceil$ time.

New running time:

- recursive: only have adding of two integers

$$T(n) = T(n-1) + T(n-2) + \lceil \log_2 F(n-1) + 1 \rceil$$

$$= T(n-2) + T(n-3) + \lceil \log_2 F(n-2) + 1 \rceil + T(n-3) + T(n-4) + \lceil \log_2 F(n-3) + 1 \rceil + \lceil \log_2 F(n-1) + 1 \rceil$$

$$= \cdots\cdots\cdots\cdots\cdots\cdots\cdots\cdots\cdots\cdots\cdots\cdots\cdots\cdots\cdots\cdots\cdots\cdots\cdots\cdots\cdots\cdots\cdots\cdots\cdots\cdots$$

$$= \sum_{k=1}^{n} F(k) \lceil \log_2 F(n-k) + 1 \rceil$$

So $T(n) = \mathrm{O}\left(\mathrm{n} \cdot \left(\frac{1+\sqrt{5}}{2}\right)^n\right)$ $(n \geq 1)$.

- Non-recursive: (only has adding of two integers)

$$T(n) = \sum_{k=1}^{n-1} \lceil \log_2 F(k) + 1 \rceil \leq \sum_{k=1}^{n-1} \left\lceil n \cdot \log_2 \left(\frac{1+\sqrt{5}}{2}\right) + 1 \right\rceil$$

$$T(n) = O(n^2)$$

- Matrix: (has adding and multiplication of two integers)

$$T(n) = T\left(\frac{n}{2}\right) + O(\lceil \log_2 F(n) + 1 \rceil^2) + O(\lceil \log_2 F(n) + 1 \rceil) + O(1)$$

$$= O\left(\sum_{k=1}^{\log_2 n} \left[\log_2 F\left(\frac{n}{2^k}\right)\right]^2\right) = O\left(\sum_{k=0}^{\log_2 n} \left(\frac{n}{2^k}\right)^2\right)$$

$$= O(n^2)$$