Algorithm Analysis Homework 2 21700034 곽영혜

1. (8points) Use the master theorem to give tight asymptotic bounds for the following recurrences.

a. $T(n) = T(\frac{7}{8}n) + n$

b. $T(n) = T(\frac{9}{3}n) + n^2$

c. $T(n) = 9 \cdot T(\frac{n}{3}) + n^2$

d. $T(n) = 3 \cdot T(\frac{n}{2}) + n$

a.

$a=1,\ b=\frac{8}{7},\ f(n)=n$

$n^{\log_{\frac{8}{7}} 1} = n^0 = 1$

$f(n) > n^{\log_b a}$, $1 f(\frac{7}{8}n) \le c \cdot f(n)$ $(c<1)$

$\Rightarrow \underline{T(n) = \Theta(f(n)) = \Theta(n)}$

b.

$a=1,\ b=\frac{3}{9}=\frac{1}{3},\ f(n)=n^2=n^{0+2}$

$n^{\log_{\frac{1}{3}} 1} = n^0 = 1$ $\quad f(n) = n^{\log_b a+\varepsilon}$

$f(n) > n^{\log_b a}$ $1 f(\frac{n}{3}) \le c \cdot f(n)$ $(c<1)$

$\Rightarrow \underline{T(n) = \Theta(f(n)) = \Theta(n^2)}$

c.

$a=9,\ b=3,\ f(n)=n^2$

$n^{\log_3 9} = n^{\log_3 3^2} = n^2 \Rightarrow \Theta(n^2)$

$f(n) = \Theta(n^{\log_3 9})$

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

d.

$a=3,\ b=2,\ f(n)=n$

$n^{\log_2 3}$

$f(n) = O(n^{\log_2 3-\varepsilon})$ $(\varepsilon=1)$

$\Rightarrow \underline{T(n) = \Theta(n^{\log_2 3})}$

2. (4points) Write a recursive algorithm (in pseudo-code) to calculate $g(m,n)$ for $m>0$ and $n>0$, where

$$g(m,n) = \begin{cases} g(m-1,n) + g(m,n-1) & \text{if } m>1 \text{ and } n>1 \\ 2 & \text{if } m=1 \text{ or } n=1 \end{cases}$$

```
g(m,n)
    If   m=1   or   n=1   then
            then return 2
    else
            then return g(m-1,n) + g(m,n-1)
```

3. ~~(6points) Repeat problem2. At this time, write a dynamic programming algorithm (iterative) instead of recursive one.~~

~~g(m,n)~~

3. (6 points) Repeat problem 2. At this time, write a dynamic programming algorithm (iterative) instead of recursive one.

```
g(m,n)
    for i ← 2 to m
        do C[i,1] ← 2;
    for j ← 1 to n
        do C[1,j] ← 2;
    for i ← 2 to m
        do for j ← 2 to n
            do C[i,j] ← C[i-1,j] + C[i,j-1];
    return C[m,n];
```

4. (3 points) Fill the table below using the dynamic programming algorithm in problem 3.

C[i,j]	j=1	j=2	j=3	j=4
i=1	2	2	2	2
i=2	2	→ 4	→ 6	→ 8
i=3	2	→ 6	→ 12	→ 20
i=4	2	→ 8	→ 20	→ 40

i=1 or j=1

⇒ C[1,j] or C[i,1] = 2

C[2,2] = C[1,2] + C[2,1]

= 2 + 2 = 4

C[2,3] = C[1,3] + C[2,2]

= 2 + 4 = 6

⋮

5. (2 points) What is time Complexity of algorithm in problem 3.

if m=1 ⇒ n = 1, 2, 3, ⋯ N (n>0) N−1

if n=1 ⇒ m = 1, 2, 3, ⋯ N (m>0) N−1

if m>1 ⇒ n = 1, 2, 3, ⋯ N (N−1)

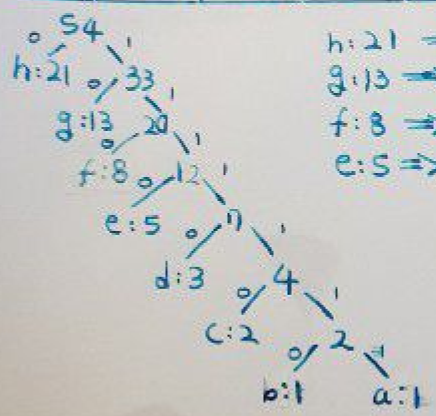
if n>1 ⇒ m = 1, 2, 3, ⋯ N (N−1)

Since m=1 or n=1 there's only one value (2)

time complexity ⇒ $\Omega(2^n)$

6. (5 points) What is an optimal Huffman Code for the following set of frequencies, based on the first 8 Fibonacci Numbers?

Character	a	b	c	d	e	f	g	h
Frequency	1	1	2	3	5	8	13	21



h:21 ⇒ 0
g:13 ⇒ 10
f:8 ⇒ 110
e:5 ⇒ 1110
d:3 ⇒ 11110
c:2 ⇒ 111110
b:1 ⇒ 1111110
a:1 ⇒ 1111111

7. (2 points) Can you generalize your answer in problem 6 to find the optimal code when the frequencies are the first n Fibonacci numbers?

8th Number Code = 0 (h:21) Num of 1: 0
7th " " = 10 (g:13) Num of 1: 1
6th " " = 110 (f:8) Num of 1: 2

⋮ ⋮

3rd " " = 111110 (c:2) Num of 1: 5
2nd " " = 1111110 (b:1) Num of 1: 6
1st " " = 1111111 (a:1) Num of 1: 7

the first N Fibonacci numbers ⇒ $\underbrace{11111\cdots1}_{N}$