FUNDAMENTAL ALGORITHMS

classmate
Date
Page

Homework - 2

Q1.

(a) $f = O(g)$; $f = o(g)$

(b) $f = \Theta(g)$; $f = O(g)$; $f = \Omega(g)$

(c) $f = \omega(g)$; $\Omega(g)$

(d) $f = O(g)$; $o(g)$

(e) $f = O(g)$; $o(g)$

Q2. (a) Given: $P(x) = a_0 + a_1 x + \ldots + a_{n-1} x^{n-1} + a_n x^n$

$$= \sum_{i=0}^{n} a_i x^i$$

Also, $P(x) = a_0 + a_1 x + \ldots + a_{n-2} x^{n-2} + b_{n-1} x^{n-1}$

$$\Rightarrow b_{n-1} = \frac{a_{n-1} x^{n-1} + a_n x^n}{x^{n-1}}$$

$$\Rightarrow \boxed{b_{n-1} = a_{n-1} + a_n x} \quad \text{— ①}$$

(b)

	Cost
Eval (A, n, c)	
if n == 0 :	-
return A[0]	-
B = A[n-1] + A[n]*c //using ①	2
A[n-1] = B	-
return Eval (A, n-1, c)	T(n-1)

classmate

Date
Page

(c) The recurrence equation for $T(n)$, running time of procedure EVAL(A, n, c) is as follows :- [Assuming cost of add^n = c ; " " " multi^n = c]

$$T(n) = T(n-1) + 2c$$ [cost of addition & multiplication]

$$= [T(n-2) + 2c] + 2c$$

$$= [T(n-3) + 2c] + 4c$$

$$\vdots$$

$$= [T(n-(n-1)) + 2c] + 2(n-2)c$$

$$= T(0) + 2(n)c$$

$$= 2nc$$ [Assuming $T(0) = 0$]

$$\therefore \boxed{T(n) = \Theta(n)}$$

(d) Assumption : $n = 2^l$

Prove : $P(x) = a_0 + a_1 x + \dots + a_n x^n$
$= P_0(x) + x^{n/2} P_1(x)$

Proof :

$$P(x) = a_0 + a_1 x + \dots + a_n x^n$$
$$= (a_0 + a_1 x + \dots a_{n/2} x^{n/2}) + x^{n/2}(a_{\frac{n}{2}+1} x + \dots a_n x^{n/2})$$
$$= P_0(x) + x^{n/2} P_1(x)$$

where $P_0(x) = [a_0 + a_1 x + \dots x_{n/2} x^{n/2}]$
& $P_1(x) = [a_{\frac{n}{2}+1} x + \dots a_n x^{n/2}]$

both polynomial of degree $\left(\frac{n}{2}\right)$

Scanned by CamScanner

Date
Page

Pseudo-Code

Assuming $A[0] = a_0$ & Array contains coefficients of $P(x)$ at position of index.

```
EVAL2(A, n, c)
    if n == 0:
        return A[0]
    return (EVAL2(A[0 ... n/2], n/2 + 1, c)
        + c^(n/2) EVAL2(A[n/2 + 1 .... n], n/2 - 1, c))
```

The recurrence relation for this would be as follows:

$$T(n) = \underbrace{2\left[T\left(\frac{n}{2}\right)\right]}_{\text{due to split in two halves of } \frac{n}{2}} + \underbrace{O(n)}_{\text{due to } c^{n/2}}$$

$T(n) = \Theta(n \log n)$ using Master's theorem

This function EVAL2 has a worse complexity than EVAL due to the fact that EVAL2 makes more recursive calls (2 instead of 1 in every function call).

2(e) In part "d", we assumed $c^{n/2}$ to have a time complexity of ~~oder~~ order $O(n)$.
However, with an alternate implemention of the power method we can reduce the time complexity.

Pseudo-Code

```
Power(base, exponent)
    if n==0:              # base case
        return 1
    if (n%2 ==0):
        temp = power(base,
```

```
Power (x, n):
    if n==0 :             # base case
        return 1
    if (n%2 ==0):         # n is even
        temp = power(x, n/2)
        return temp * temp
    else :                # n is odd
        return x * power (x, n-1)
```

Q3.(a) ~~f0~~

$$foo(n) = 2^{\log_3 n}$$

$\therefore$ it returns $foo(\frac{n}{3}) + foo(\frac{n}{3})$, when $n \neq 1$

$$= 2\, foo(\frac{n}{3})$$

(b) $$T(n) = 2T(\frac{n}{3}) + c$$

~~= a T~~

Let $a = 2$, $b = 3$ & $f(n) = c$ with $O(1)$ time complexity

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

$$= \Theta(n^{0.630})$$

(c) By changing line 5, the number of recursive calls will be halved and so will the memory (stack memory) used.

But ~~it~~ foo would still compute $2^{\log_3 n}$

And the time recurrence would be:

$$T(n) = T(\frac{n}{3}) + c$$

for $a = 1$, $b = 3$ & $f(n) = c = \Theta(n^{\log_3 1})$

~~$= \Theta(1)$~~

$\therefore T(n) = \Theta(n^{\log_3 1} \log n) = \Theta(\log n)$ [Master theorem]

2(e)

Q4. ~~MIN_MAX (A, i, j)~~ For A.length = n, function call: ~~MIN_MAX~~
(assuming 0-index) MIN_MAX(A, 0, n-1)

```
MIN_MAX (A, low, high)
    if low == high:                          # base case
        return [A[low], A[low]]              # no comparisons
    if high == low+1:                        # if A.length = 2
        if A[low] > A[high]                  # 1 comparison
            max = A[low]
            min = A[high]
        else
            max = A[high]
            min = A[low]
    mid = (low+high)/2                       # DIVIDE - CONQUER
    mml = MIN_MAX(A, low, mid)               # similar to BINARY SEARCH
    mmr = MIN_MAX(A, mid+1, high)

    if mml[0] < mmr[0]:                      # 1 compare
        min = mml[0]
    else:
        min = mmr[0]

    if mml[1] > mmr[1]:                      # 1 compare
        max = mml[1]
    else:
        max = mmr[1]

    return [min, max].
```

Q5

Using $\log n! = \Theta(n\log n - n)$

$$\Rightarrow \log\left({}^{n}C_{k}\right) = \log\left(\frac{n!}{(n-k)!\,k!}\right) = f(n,k)$$

$$= \log(n!) - \log((n-k)!) - \log(k!)$$

We know $0.1\log n < k \le \frac{n}{2}$

So for $k = n/2$ ~~i.e finding lower boun~~

$$f\left(n,\frac{n}{2}\right) = \log(n!) - \log\left(\left(\frac{n}{2}\right)!\right) - \log\left(\left(\frac{n}{2}\right)!\right)$$

$$= \log(n!) - 2\log\left(\left(\frac{n}{2}\right)!\right)$$

$$\Rightarrow f(n,k) = \Theta\left[\log(n!) - 2\log\left(\left(\frac{n}{2}\right)!\right)\right]$$

$$\Rightarrow f\left(n,\frac{n}{2}\right) = \Theta(n\log n - n) - \Theta\left(\frac{n}{2}\log\frac{n}{2} - \frac{n}{2}\right)$$

$$\Rightarrow f\left(n,\frac{n}{2}\right) = c_1(n\log n - n) - c_2\left(\frac{n}{2}\log\left(\frac{n}{2}\right) - \frac{n}{2}\right)$$

~~$\le c \log$~~

$$\Rightarrow f(n,k) \le c_1(n\log n - n) - c_2\left(\frac{n}{2}\log\frac{k}{2} - \frac{n}{2}\right)$$

$$\le c\log\left(\frac{n^n}{\left(\frac{n}{2}\right)^{n/2}}\right)$$

$$\le c\log\left(\frac{n}{n/2}\right)^{n/2}$$

$$\le c\,\frac{n}{2}\log\left(\frac{n}{n/2}\right)$$ where we replace $\frac{n}{2}$ with k

$$\Rightarrow f(n,k) \le c\, k \log\left(\frac{n}{k}\right).$$

Now for $k = 0.1 \log(n)$

$$\cancel{\log}\;f(n,k) = \Theta\Big((n\log n - n) - \big((n-k)\log(n-k) - n + k\big) - (k\log k - k)\Big)$$

$$= \Theta\left(\log \frac{n^n}{k^k (n-k)^{(n-k)}}\right)$$

~~fo~~ $f(n, 0.1\log(n))$

$$\ge c \log\left(\frac{n^n}{k^k (n-k)^{(n-k)}}\right)$$

$$\ge c \log\left(\frac{n^n}{k^k (n - 0.1\log(n))^{n-0.1\log n}}\right)$$

$$\ge c \log\left(\frac{n^n}{k^k\, n^{n-0.1\log n}}\right)$$

$$\ge c \log\left(\frac{n^n}{k^k\, n^{n-k}}\right)$$

$$\ge c \log\left(\frac{n^k}{k^k}\right)$$

$$\ge c \cdot k \log\left(\frac{n}{k}\right)$$

$$\therefore \boxed{f(n,k) = \Theta\left(k \log\left(\frac{n}{k}\right)\right)}$$