COMP2119 Introduction to data structures and algorithms
Assignment 1
Recursive Equations and Mathematical Induction
Due Date: Feb 22, 2020 5:00pm

Q1 Recursive Equations [25%]

Solve the following recurrence relations and give a Θ bound for each of them.

(1) T(n) = T(n-1) + 1; T(0) = 0. **[5%]**

Sol:

T(n) = T(n-1) + 1
= T(n-2) + 1 + 1
= T(n-3) + 1 + 1 + 1
= T(0) + n
= n

So, T(n) = n where n is a natural number.

The Θ bound is Θ(n).

(2) T(n) = T(n-1) + n; T(0) = 0. **[5%]**

Sol:

T(n) = T(n-1) + n
= (T(n-2) + n-1) + n
= (T(n-3) + n-2) + n-1 + n
= (T(0) + 1) + 2 + … + n-1 + n
= $\frac{n(n+1)}{2}$

So, T(n) = $\frac{n(n+1)}{2}$ where n is a natural number.

The Θ bound is $\Theta(n^2)$.

(3) T(n) = 4T(n/4) + 1; T(l) = 0. (You may assume that n is a multiple of 4.) **[5%]**

Sol:

T(n) = 4T(n/4) + 1
= $4(4T(\frac{n}{4^2}) + 1) + 1$
= $4^2T(\frac{n}{4^2}) + 4^1 + 4^0$
= $4^2(4T(\frac{n}{4^3}) + 1) + 4^1 + 4^0$
= $4^3T(\frac{n}{4^3}) + 4^2 + 4^1 + 4^0$
= $4^{\log_4 n - 1}(4T(1) + 1) + 4^{\log_4 n - 2} + \cdots + 4^1 + 4^0$
= $4^{\log_4 n}T(1) + 4^{\log_4 n - 1} + 4^{\log_4 n - 2} + \cdots + 4^1 + 4^0$
= $\frac{1-4^{\log_4 n}}{1-4}$
= $\frac{1-n}{1-4}$

So, $T(n) = \begin{cases} 0 & n = 1 \\ \frac{1-n}{1-4} & n = 4^m \text{ and m is a positive number} \end{cases}$

The Θ bound is $\Theta(n)$.

(4) $T(n) = 2T(n/2) + n$; $T(l) = 0$. (You may assume that n is a multiple of 2.) **[5%]**

Sol:

$T(n) = 2T(n/2) + n$

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

$= 2^2T(\frac{n}{2^2}) + n + n$

$= 2^2(2T\left(\frac{n}{2^3}\right) + \frac{n}{2^2}) + n + n$

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

$= 2^{\log_2 n - 1}\left(2T(1) + \frac{n}{2^{\log_2 n - 1}}\right) + (\log_2 n - 1)n$

$= 2^{\log_2 n}T(1) + n + (\log_2 n - 1)n$

$= n\log_2 n$

So, $T(n) = n\log_2 n$ asuming $n = 2^m$ where m is a natural number.

The Θ bound is $\Theta(n\log_2 n)$ assuming

(5) $T(n) = \begin{cases} 1 & n = 1 \\ 1 & n = 2 \\ T(\sqrt{n}) + 1 & n > 2 \end{cases}$ (You may assume that n is a power of 2.) **[5%]**

Sol:

$T(n) = T(\sqrt{n}) + 1$

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

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

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

$= (T\left(n^{\frac{1}{2^3}}\right) + 1) + 1 + 1$

$= T\left(n^{\frac{1}{2^3}}\right) + 1 + 1 + 1$

$= \left(T\left(n^{\frac{1}{2^{\log_2\log_2 n}}}\right) + 1\right) + (\log_2\log_2 n - 1)$

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

$= T(2) + \log_2\log_2 n$

$= 1 + \log_2\log_2 n$

So, $T(n) = \begin{cases} 1 & n = 1 \text{ or } n = 2 \\ 1 + \log_2\log_2 n & \text{assuming } n = 2^{2^m} \text{ where m is a positive number} \end{cases}$

The Θ bound is $\Theta(\log_2\log_2 n)$.

Q2 Recursive Algorithms [25%]

(1) Fill in the line of the following Python code for calculating the factorial of a number. **[5%]**

```
def fact(num):
    if num == 0:
        return 1
    else:
        return __num_* fact(num-1)__
```

(2) What will be the output of the following Python code? **[5%]**

```
def test(i,j):
    if i==0:
        return j
    else:
        return test(i-1,i+j)
print (test(4,7))
```

Answer: 17

Recursive processes of test(4,7) are as follow:

test(4，7)

test(3，4+7)

test(2，3+11)

test(1，2+14)

test(0，1+16)

(3) What will be the output of the following Python code? **[5%]**

```
l = []
def convert(b):
    if b==0 :
        return 1
    dig = b%2
    l.append(dig)
    convert(b//2)
convert(6)
l.reverse()
for i in l:
    print(i, end=' ')
```

Answer: 1 1 0

Recursive processes of convert(6) are as follow:

convert(6) => l.append(0)

convert(3) => l.append(1)

convert(1) => l.append(1)

convert(0) => return 1
After reversing, l will be [1, 1, 0]

(4) List the function(s) that is/are tail recursive. **[5%]**

```
def a(n):
    if n == 0:
        return 0
    else:
        return n*a(n-1)
def b(n, tot):
    if n==0:
        return tot
    else:
        return b(n-2, tot-2)
```

Answer: a(n) is not tail function. b(n, tot) is tail function.

(5) What will be the output of the following Python code? **[5%]**

```
def a(n):
    if n == 0:
        return 0
    elif n == 1:
        return 1
    else:
        return a(n-1)+a(n-2)
for i in range(0,4):
    print(a(i), end=" ")
```

Answer: 0 1 1 2
We know,
a(0) => 0
a(1) => 1
a(2) => 0+1
a(3) => 0+1+1

Q3 Mathematical Induction [25%]

Prove or disprove the claims below.
(1) For every positive integer $n \geq 4$, $2^n < n!$. **[5%]**
(2) For any natural number n, $n^3 - n$ is divisible by 3. **[10%]**
(3) For any $n \geq 0$, F_{3n} is even. (F is Fibonacci numbers) **[10%]**

Answer:

(1) :

[Proof] : We'll use induction on n where $n \geq 4$.

[Base 1] : Assuming n=4. $2^4 < 4!$ is true.

[Induction]: Suppose that $2^k < k!$ is true, for $k \geq 4$.

We need to prove $2^{k+1} < (k+1)!$ is true.

$2^k < k!$

$2^k *(k+1) < k!*(k+1)$

$2^k *(k+1) < (k+1)!$

$2^k *2 < (k+1)!$

$2^{k+1} < (k+1)!$

Therefore, $2^n < n!$ holds, for every integer $n \geqslant 4$.

(2) :

[Proof] : We'll use induction on n where n is natural number.

[Base 1] : Assuming n=0. $0^3 - 0$ could be divided by 3.

[Base 2] : Assuming n=1. $1^3 - 1$ could be divided by 3.

[Induction]: Suppose that $k^3 - k$ could be divided by 3, for natural number k.

We need to prove $(k+1)^3 - (k+1)$ could be divided by 3.

$(k+1)^3 - (k+1)$

$=(k^3-k) + 3(k^2+k)$

k^3-k and $3(k^2+k)$ can be divided by 3, so $(k^3 - k) + 3(k^2 + k)$ can be divided by 3. Then, $(k+1)^3 - (k+1)$ can be divided by 3.

Therefore, for any natural number n, $n^3 - n$ is divisible by 3.

(3) :

[Proof] : We'll use induction on n where $n \geq 0$.

[Base 1] : Assuming n=0. $F_0 = 0$, so F_3 is even.

[Base 2] : Assuming n=1. $F_3 = 1+1+2 = 4$, so F_3 is even.

[Induction]: Suppose that F_{3n} is even, for $k \geq 0$.

We need to prove F_{3n} is even.

$F_{3(n+1)}$

$= F_{3n+2} + F_{3n+1}$

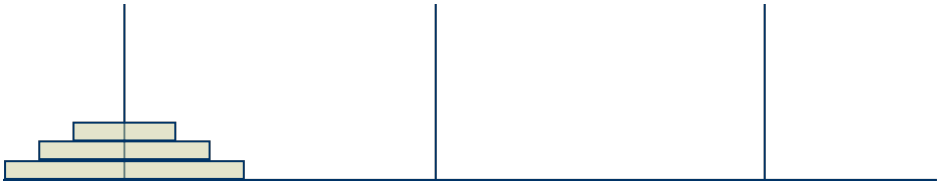
$= F_{3n} + 2F_{3n+1}$

F_{3n} and $2F_{3n+1}$ are even, so $F_{3n} + 2F_{3n+1}$ is even. Then, $F_{3(n+1)}$ is even.

Therefore, for any $n \geq 0$, F_{3n} is even. (F is Fibonacci numbers).

Q4 Recursive Algorithms and Induction [25%]

Given 3 pegs (A, B and C) and n disks of different sizes placed in order of size on one peg, transfer the disks from the original peg (A) to another peg (C) with the constraints that (1) each disk is on a peg. (2) no disk is ever on a smaller disk. (3) only one disk at a time is moved.



[Claim] : A function M is defined, which has 4 parameters, respectively the number of disks n, the original peg o, the destination peg d, and the intermediate peg t. This function can be written as M(n, o, d, t), which means moving n disks from o to d in the help of t. According to the problem, the claim is that M(n, o, d, t) is true for all positive integers n.

[Proof] : We'll use induction on n.
To simplify the description, disk k means the disk has size k.

[Base 1] : Assuming n=1. Disk 1 can be moved from A to C directly. M(1, A, C, B) is true.

[Base 2] : Assuming n=2.
(1): Please fill peg name to prove M(2, A, C, B) is true. **[5%]**
Step 1 : Move Disk 1 from A to _B_.
Step 2 : Move Disk 2 from A to _C_.
Step 3 : Disk 1 can be moved from B to C.
(2) Please prove M(2, B, C, A) and M(2, A, B, C) is true. **[5%]**
Answer:
Similar to the process of M(2, A, C, B), M(2, B, C, A) could be done as follow:
Step 1 : Move Disk 1 from B to A.
Step 2 : Move Disk 2 from B to C.
Step 3 : Disk 1 can be moved from A to C.

M(2, A, B, C) could be done as follow:
Step 1 : Move Disk 1 from A to C.
Step 2 : Move Disk 2 from A to B.
Step 3 : Disk 1 can be moved from C to B.
So, M(2, B, C, A) and M(2, A, B, C) are true.

[Induction]: Suppose that M(k, A, C, B) is true, for some positive integer k.
(3) If M(k, A, C, B) is true, is M(k, A, B, C) is true? Is M(k, B, C, A) is true? **[5%]**
Answer:
M(k, A, C, B) is true, which means we can move k disks from the original peg to the destination peg in the help of intermediate peg. So, M(k, B, A, C) is true if M(k, A, C, B) are true.

(4) Please fill peg name to prove M(k+1, A, C, B) is true. **[5%]**

Step 1: M(k, __A__, __B__, __C__).
Step 2: Move disk k+1 from A to C.
Step 4: M(k, __B__, __C__, __A__)

(4) What's the lower bound of the number of moves the M(n, A, C, B) take? **[5%]**
[Refer to slide 35 of chapter 2. What is the least number of moves required to solve the problem (lower bound)?]
Answer: Let S(n) be the steps moves of M(n, A, C, B). We know recursive equation

$$S(n) = \begin{cases} 1 & n = 1 \\ 2S(n-1) + 1 & n > 1 \end{cases}$$

Solving the recursive equation, we can get $S(n) = 2^n - 1$, which is the necessary number of moves to solve the problem with n disks.

Submission.
Please submit your assignment (one **PDF** file) to moodle on or before 5:00pm, February 22, 2020. Make sure all contents are readable.

Please feel free to post your questions on **Moodle forum**, or contact us (dhding2@cs.hku.hk, dpshan@cs.hku.hk or gwyuan@cs.hku.hk)if you encounter any difficulty in this assignment. We are very happy to help.

TA Dan, Damon and Gavin.