Algorithms and their applications

Assignment #1

2020314303 Jinduk Park

1. Definition of theta is described below.

- Def of Θ

$f(n)$ is $\Theta(g(n))$ if

$c_2 g(n) \le f(n) \le c_1 g(n)$ for every $n \ge n_0$

Ans. (a)

(a) $(n^3+1)^6$ is $\Theta(n^{18})$

(proof)

$1 \cdot (n^3)^6 \le (n^3+1)^6 \le 2 \cdot (n^3)^6$

for $n \ge n_0$

$\therefore (n^3+1)^6$ is $\Theta(n^{18})$.

Ans. (b)

(b) $\sqrt{10n^2+7n+3}$ is $\Theta(n)$

(proof) $\sqrt{10}\, n \le \sqrt{10n^2+7n+3} \le \sqrt{11}\, n$

for $n \ge n_0$

$\therefore \sqrt{10n^2+7n+3}$ is $\Theta(n)$

Ans.(c)

(c) $2n\log(n+2)^2 + (n+2)^2\log n$ is $\Theta(n^2\log n)$

(proof)

$$n^2\log n \leq 2n\log(n+2)^2 + (n+2)^2\log n \leq 2n^2\log n$$

$$\qquad\qquad\qquad\qquad \shortparallel$$

$$4n\log(n+2) + (n+2)^2\log n$$

$\therefore$ $2n\log(n+2)^2 + (n+2)^2\log n$ is $\Theta(n^2\log n)$

Ans.(d)

(d) $1+\frac{1}{2}+\frac{1}{3}+\cdots+\frac{1}{n}$ is $\Theta(\ln(n))$

(proof)

$$\ln(n) = \int_1^n \frac{dx}{x} \leq 1+\frac{1}{2}+\cdots+\frac{1}{n} = \sum_{k=1}^{n}\frac{1}{k} \leq 1+\int_2^n \frac{dx}{x}$$

$$\qquad\qquad\qquad\qquad \Rightarrow 1+\ln(n)$$

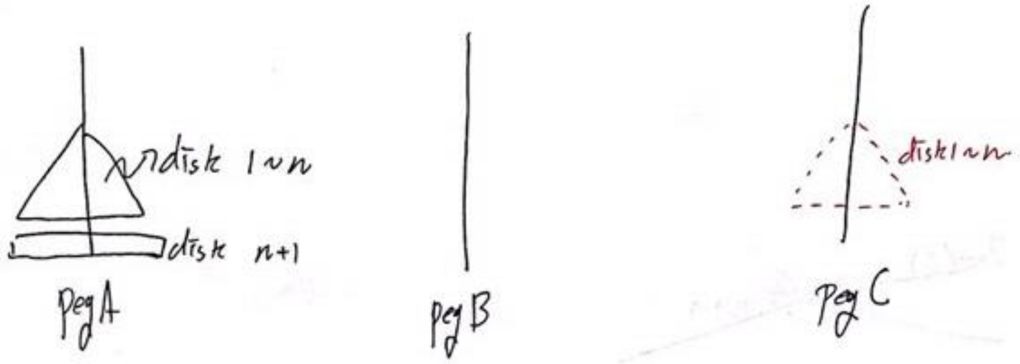
since $\Theta(1) < \Theta(\ln(n))$

$1+\frac{1}{2}+\cdots\frac{1}{n}$ is $\Theta(\ln(n))$

2. Restricted tower of Hanoi

Ans.(a)

2-(a)



let $H(n)$ the total number of moving from A to C via B.

And suppose there are n+1 disks to move.

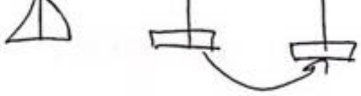
1. It will take $H(n)$ to move n disk from A to C.

+ $H(n)$ step

2. than move $disk_{n+1}$ from A to B

+1 step

3. move backward disk 1~n from C to A & move $disk_{n+1}$ to C



+ $(H(n)+1)$ step

4. than move disk 1~n from A to C again

+ $H(n)$ step

Sum (1~4) : $H(n+1) = 3H(n)+2$

Ans.(b)

```
# 2-(b). Hanoi problem.

# Calculate summation number of moving disks
# n is natural number here.
def Hanoi_sum(n):

    if n == 1:     # Return end value for the recursive func which condition is that the disk is only one.
        return 2

    else:
        return 3*Hanoi_sum(n-1)+2    # Return recursive function that justified in 2-(a)
```

(example output)

```
In [12]: Hanoi_sum(5)
Out[12]: 242
```

Ans.(c)

2-(c) $H_{(n+1)} = 3H_{(n)} + 2$

$$H_{(n+1)} + 1 = 3\left(H_{(n)} + 1\right) \Rightarrow \frac{H_{(n+1)}+1}{H_{(n)}+1} = 3$$

$$H_{(n)} + 1 = 3^n$$

$$H_{(n)} = 3^n - 1 \quad \left(H_{(1)} = 2\right)$$

3. Selection Algorithm

```
# 3. Selection Algorithm

# Calculate entire number of case - choosing 'k' in 'n' candidates.
# n is natural nuber // k is integer which is >= 0
def Combination(n, k):

    if k == 0 or n == k:      #Return end(k=0) / exceptional(n=k) value for the reculsive function.
        return 1
    else:
        return Combination(n-1, k-1) + Combination(n-1, k)
"""
# Description.
 Pulling k out of n equals the sum of the cases in which, after selecting particular one,
 the number of cases in which k is pulled out of it (n-1), and the number in which k-1 is selected including
it.

"""
)
```

(example output)

```
In [13]: Combination(5,2)
Out[13]: 10
```

4. Newton's method

```
# 4. Newton's Method

# Input ans as your starting rough guessing number
# tol is abbreviation of tollerance, the parameter that you think that the gap is good enough
def MySqrt(num, ans, tol):
    if abs(ans*ans - num) <= tol:
        return ans       # if the gap btw guessing num and real one is close enough, than return the newly guessing number.
    else:
        return MySqrt(num,(ans*ans + num)/(2*ans), tol) # Newton method that defined as average btw (ans & num/ans)
```

(example output)

```
In [16]: MySqrt(2,1.5,0.0001)
Out[16]: 1.4142156862745099
```

5. Infix to prefix using *manual process*

5 - (a)

<infix>

A*B+C*D

→ ((A*B) + (C*D))

<prefix>

AB* CD* +

5 - (b)

<infix>

(A-2*(B+C)-D*E)*F

→ (((A-2*(B+C))-(D*E))*F)

<prefix>

A BC+2 * - DE* - F*

6. Prefix to Infix using *manual process*

6-(a) <postfix>

AB*C-D+

→ (A*B) - C+D

<Infix>

A*B-C+D

6-(b) <postfix>

ABC+*D-

→ (A*(B+C)-D)

<Infix>

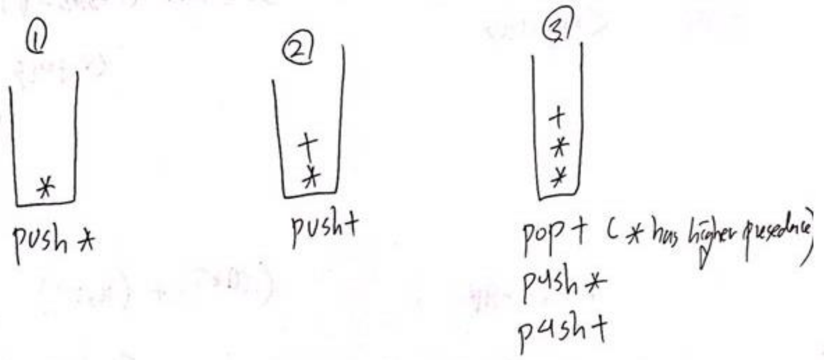
A*(B+C) - D

7. Infix to postfix using *stack*

7-(a) A*B + C*D

sol) A ① * B ② + C ③ * D

Scan direction →



∴ postfix : AB*CD*+

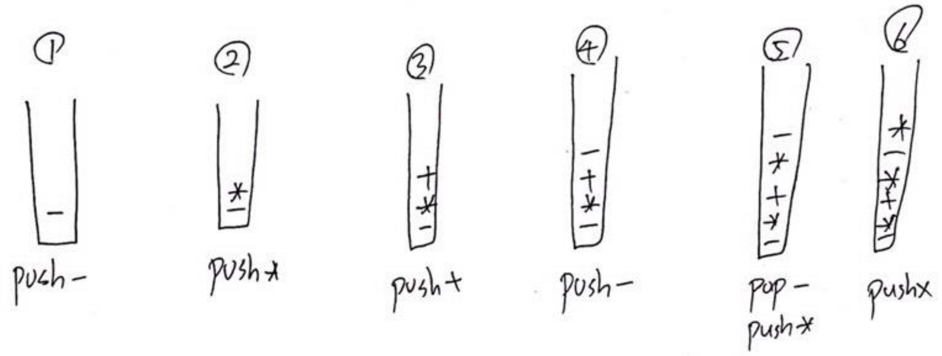
7-(b)

① ② ③ ④ ⑤ ⑥

(A - 2*(B+C) - D*E)*F

$\xrightarrow{\hspace{6em}}$

scan operator



postfix : A B C + 2 * - D E * - F *