컴퓨터 알고리즘(HW#2)

학부 : 정보컴퓨터공학부
학번 : 201624548
이름 : 이재윤

1. T(625) ?

$T(n) = 7 \cdot T(\frac{n}{5}) + 10n$

$T(1) = 1$.

Python :

```
def T(n) :
    if n==1 :
        return 1.
    return 7·T(n//5) +10n.
print (T(625)).
```

result : 46801

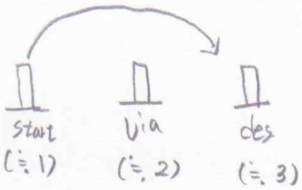
$T(625) : 6250 + 7 \cdot T(125) = 46801$.

$T(125) : 1250 + 7 \cdot T(25) = 5793$

$T(25) : 250 + 7 \cdot \underbrace{T(5)}_{57} = 649$

$T(5) : 50 + 7 \cdot \underbrace{T(1)}_{1} = 57$.

17. 하노이 타워 알고리즘.(재귀)



start, via, des 를 편의상 1, 2, 3으로 두자.

Python :

```
def hanoi(n,start, via, des). :
    count = 0
    if n==1 :
        return 1.              # 원판이 단 하나 남았을 때 start에서 des로 이동.
    else :
        count += hanoi(n-1, start, des, via) # n-1 개 만큼을 via로 이동        S(n-1)
        count += 1.                          # n번째의 가장 큰 원판을 des로 이동   1.
        count += hanoi(n-1, via, start, des). # via에 있던 n-1개 만큼의 원판을 des로 이동   S(n-1)
    return count.
n=int(input())
print(hanoi(n,1,2,3)).
```

a) 이 알고리즘에 대해. $S(n) = 2^n - 1$ 임을 보여라.

1. $S(1) = 1$. $\Rightarrow 2^n - 1$ 에 대해 성립.
2. $S(n) = k$ 이고, 이가 $2^n - 1$ 이라고 하면.

$S(n+1) = 2k + 1$ 이고 (else 구문 안의 알고리즘에 의해)

이는 $2(2^n - 1) + 1 = 2^{n+1} - 1$ 이므로

$S(n) = 2^n - 1$ 에 대해 성립한다.

b) 위 알고리즘이 junk move 가 없는 최소한의 이동 경로 이므로. $2^n - 1$ 이 최소 이동 횟수이다.

3. 트로미노 알고리즘 설계.

0	1
2	3

편의상 사각형을 4등분 했을 때 왼쪽 위 : 0, 오른쪽 위 : 1, 왼쪽 아래 : 2 오른쪽 아래 : 3으로 지칭.

Python.

```
import numpy

count = 1.

# n은 2의 제곱수. empty는 트로미노가 빈 좌표. (L 자 모양이라 default : 1로 둠)
def tromino(n, array, empty =1):
    global count
    # n이 2일 때 타일링
    if n== 2:
        if empty != 0:
            array[0][0] = count
        if empty != 1:
            array[0][1] = count
        if empty != 2:
            array[1][0] = count
        if empty != 3:
            array[1][1] = count
        count += 1.
    else:
        tromino (n//2, array[n//4 : (n//4)*3 , n//4 : (n//4)*3], empty) # 중앙의 empty 부분이 빈 사각형 모양
        if empty != 0:
            tromino(n//2, array[0:n//2 , 0:n//2], 3)
        if empty != 1:
            tromino (n//2, array[:n//2 , n//2:],2)
        if empty != 2:
            tromino(n//2, array[n//2: , 0:n//2], 1)
        if empty != 3:
            tromino(n//2 , array[n//2: , n//2 : ], 0)
m = int(input())
array = numpy.array([[0]*m for _ in range(m)])

tromino (n, array)

for arr in array:
    for i in arr:
        print("%3d" % i ,end = ' ')
    print()
```

4b. Subarray의 max 값을 구하는 알고리즘을 설계.

1. for 문 2중으로 구성 — $O(n^2)$ → 생략.

2. 분할 정복 : $O(n \log n)$

Python:

```
import sys.

def   sum_subarray(arr ,start_index, end_index):
    func_type = 1
    if  start_index > end_index :
            func_type = -1
    result = -sys.maxsize
    s = 0

    for i in range(start_index, end_index, func_type):
        s = s + arr[i]
        if (s > result):
            result = s
    return result


def  max_mid_subarray (arr, low, high)
    mid = (low+high) // 2
    left_sum = sum_subarray (arr, mid, low-1)  #왼쪽
    right_sum = sum_subarray (arr, mid+1, high+1)  #오른쪽  mid 중복 제거
    return  left_sum + right_sum

def  max_subarray (arr, low, high):
    if (low == high) : return arr[high]

    mid = (low + high) // 2

    max_left_subarray = max_subarray (arr, low, mid)
    max_right_subarray = max_subarray (arr, mid+1, high)
    max_middle_subarray = max_mid_subarray (arr, low, high)
    return  max (max_left_subarray, max_right_subarray, max_middle_subarray)
```