Design and analysis of algorithms
MID-TERM REVIEW

I. Sample test with solutions

1. Given an algorithm

```
def f(A):
    # input A is an array of n number
    n = len(A)
    for i in range(0, n-1):
        max = A[i]
        imax = i
        for j in range(i+1, n):
            if A[j] > max:
                max = A[j]
                imax = j
        A[i], A[imax] = A[imax], A[i]
```

b. State the purpose of the algorithm
c. Locate the most frequently executed operations
d. Calculate the complexity of the algorithm as a function of n

e. Identify Θ-notation of the complexity of the algorithm

Solution

a) sorting arrays in descending order
b) comparison in line" if A[j] > max"
c) T(n) = 1 + 2 +…+ n-1 = $\frac{n(n-1)}{2}$
d) $T(n) \in \Theta(n^2)$

2. Solve the recurrence relations

2.a) $T(n) = \begin{cases} 3T\left(\frac{n}{3}\right) + 5^3 \; if \; n > 1 \\ 1 \; if \; n = 1 \end{cases}$

a = 3, b = 3, k = 0

Since $a = 3 > b^k = 1$, then according to the master theorem, we have $T(n) \in \Theta\left(n^{\log_3 3}\right) = \Theta(n)$

2.b) $T(n) = \begin{cases} 2T(n-1) + 1 \; if \; n > 1 \\ 1 \; if \; n = 1 \end{cases}$

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

$$2T(n-1) = 2^2T(n-2) + 2$$

…

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

$$\therefore T(n) = 1 + 2 + \cdots + 2^{n-2} + 2^{n-1} = \frac{1 \times (1 - 2^n)}{1 - 2} = 2^n - 1 \in \Theta(2^n)$$

3. Design and analysis an algorithm for multiplying two matrices

```
def mul(A,B, m,n,k)

#Input: matrix A with size mxn, matrix B with size nxk

#Output: C = A x B with size m x k

C = empty-matrix(m,k)

for i from 0 to m-1 do

   for j from 0 to k -1 do

      for v from 0 to n -1 do

      C[I,j] = C[I,j] + A[I,v]*B[v,j]

return C
```

Analysis:

basic operation: Assignment in “C[I,j] = C[I,j] + A[I,v]*B[v,j]”

Complexity: $T(m, n, k) = m \times n \times k \in \Theta(m \times n \times k)$

II. Exercises

1. Analyze the complexity of the following algorithms to obtain their $\mathbf{\Theta}$-notation

a)

INPUT: , k integer, positive and $n = 3^k$

OUTPUT: count

```
1. count ← 0; i = n
2. while (i ≥ 1)
      1. for j ← 1 to n do
            1. count ← count + 1
            2. print(j)
      2. end for
      3. i ← i/3
3. end while
```

4. return count

b)

ALGORITHMS Secret_2($A[0..n-1]$)
1. **for** $i \leftarrow 0$ **to** $n-2$ **of the**
2. **for** $j \leftarrow i+1$ **to** $n-1$ **of the**
3. **if** $A[i] == A[j] * A[j]$ **return false**
4. **return true**

c) Algorithm:
int ***i*, *even***;
i **:=** ***1***;
even **:=** ***0***;
while(*i* < *k*) {
even **:=** ***even*** **+** ***2***;
i **:=** ***i*** **+** ***1***;
}
return ***even*** **.**

d) Algorithm: Even(positive integer ***k*****)**
Input: ***k*** , a positive integer
Output: ***k***-th even natural number (the first even being ***0***)

Algorithm:
if *k* = *1*, then return ***0***;
else return **Even(*k*-*1*) + *2* .**

e)Algorithm: Power_of_2(natural number **k)**
Input: ***k*** , a natural number
Output: ***k***-th power of ***2***

Algorithm:
if *k* = *0*, then return ***1***;
else return **2*Power_of_2(*k* - *1*) .**

f) Algorithm: Power_of_2(natural number **k)**
Input: ***k*** , a natural number
Output: ***k***-th power of ***2***

Algorithm:
int ***i*, *power***;

```
i := 0;
power := 1;
while( i < k ) {
        power := power * 2;
        i :=  i + 1;
}
return power .
```

2. Solve the following recurrence relations to obtain their **Θ**-notation
   a) $T(n) = 3T(n-1) + 2$ and $T(1) = 0$
   b) $T(n) = T(n-1) + n^2$ and $T(1) = 0$
   c) $C(n) = 9C(n/3) + n$
   d) $C(n) = C(2n/3) + 1$
   e) $C(n) = 3C(n/4) + nlgn$ (O-notation accepted)
   f) $T(n) = 2T(n/2) + nlgn$ (O-notation accepted)
   g) $T(n) = 2T(n/4) + 1$
   h) $T(n) = 2T(n/4) + \sqrt{n}$
   i) $T(n) = 2T(n/4) + n$
   j) $T(n) = 2T(n/4) + n^2$

3. Design an algorithm in the form of pseudocode for solving a problem and analyze the complexity of the algorithm:
   a) calculate the sum of a sequence

$A = \frac{1}{1+1} + \frac{1}{1+2^2} + \frac{1}{1+3^3} + \cdots + \frac{1}{1+n^n}$ (non-recursive algorithm)

$B = \frac{1}{1+m} + \frac{1}{1+m^2} + \frac{1}{1+m^3} + \cdots + \frac{1}{1+m^n}$(non-recursive algorithm)

$C = 1^2 + 2^2 + 3^2 + \cdots + n^2$(non-recursive and recursive algorithms)

   b) operations with matrices: multiplication, addition, subtraction, multiplication with a number (non-recursive algorithm)
   c) selection sort (less important for midterm-test: quicksort, mergesort)
   d) searching binary, sequencial (non-recursive and recursive algorithms)
   e) find min, max of a sequence of numbers (non-recursive and recursive algorithms)
   f) median, average value of a sequence of numbers (non-recursive algorithm)

---THE END---