DESIGN AND ANALYSIS OF ALGORITHMS
LAB 02: ANALYSIS OF RECURSIVE ALGORITHMS

I. Designing and analyzing recursive algorithms

Iterative	recursive
1. Locate basic operations 2. Identify the worst case 3. Count the number of basic operations in the worst case => T(N) 3.1 Determine Big Theta/Big-O notation of T(N)	1. Locate basic operations 2. Build recurrence relation by Counting the number of basic operations => T(N) 3. Solve the recurrence relation 3.1 Determine Big Theta/Big-O notation of T(N)

4. A sample

- Finding maximum value in an array
- Input: Array A (A[1], A[2], . . . , A[n])
- Output: Largest element in the array.
- Algorithm:
- Findlargest(A[1:n])
    - 1. If n = 1, then return A[1].
    - 2. Else,
        - 3. let m = Findlargest(A[1 : n − 1]).
        - 4. If m > A[n], then return m
        - 5. Else return A[n].

a. Basic operations:
comparison in line 1
assignment in line 3
comparision in line 4

b. Recurrence relation

$$T(n) = \begin{cases} T(n-1) + 3, if\ n > 1 \\ 1\ , \quad if\ n = 1 \end{cases}$$

c. Solving the recurrence relation

T(n) = T(n−1)+3
T(n−1) = T(n−2)+3
T(n−2) = T(n−3)+ 3
…
T(2) = T(1) + 3

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

$$T(n) \in \Theta(n)$$

5. Exercises:

a. Design (pseudocode), implement (Python) and analyze (complexity) a recursive(a) algorithm and an iterative(b) algorithm for computing the *k*-th power of *2*(for $\mathbf{k = 3}$, we have $\mathbf{2^3 = 8}$)
b. Design, implement and analyze a recursive(a) algorithm and an iterative(b) algorithm for finding maximum value in an array
c. Design, implement and analyze a recursive(a) algorithm and an iterative(b) algorithm for finding a value in a sorted array

II. Recurrence Relation

1. Samples

To analyze the complexity of the recurrence relation, we might solve by the iterative method or Master theorem.

a. Sample Solutions with Master theorem

If $T(n) = aT(n/b) + f(n)$, where $f(n) = \Theta(n^k)$, then

$$T(n) = \begin{cases} \Theta(n^k), & \text{if } a < b^k; \\ \Theta(n^k \log n), & \text{if } a = b^k; \\ \Theta(n^{\log_b a}), & \text{if } a > b^k; \end{cases}$$

The theorem also holds if $b > 1$.

Sample 1

C (N) = C (2N/3) + 1 2N/3 = N / (3/2)

$$a = 1, b = \frac{3}{2}, f(N) = 1 \; and \; f(n) \in \Theta(n^k), therefore \; k = 0$$

$$\Rightarrow a = 1 = b^k = \left(\frac{3}{2}\right)^0$$

$$\Rightarrow C(N) \in \Theta(N^k logN) = \Theta(logN)$$

Sample 2:

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

A = 2, b = 2, k = 1, b^k = 2=> a = b^k => $T(n) \in \Theta(nlogn)$

b. A sample solution with iteration

(i). Given the following recurrence relation

$\boldsymbol{C(n) = 2C(n/2) + n - 1}$ with $\boldsymbol{n > 1, C(1) = 0}$

Applying the reverse method, we have:

$$\begin{aligned}
C(n) &= 2C(n/2) + n - 1 \\
&= 2(2C(n/2^2) + n/2 - 1) + n - 1 \\
&= 2^2C(n/2^2) + n - 2 + n - 1 \\
&= 2^2C(n/2^2) + 2n - 2 - 1 \\
&\vdots \\
&= 2^kC(n/2^k) + kn - 2^{k-1} - 2^{k-2} - \ldots - 2 - 1 \\
&= 2^kC(1) + kn - \sum_{j=0}^{k-1} 2^j \\
&= 2^k \times 0 + kn - (2^k - 1) \\
&= kn - 2^k + 1 \\
&= n \log n - n + 1.
\end{aligned}$$

So $C(n) \in \Theta(nlgn)$

(ii). Given another recurrence relation

$C_N = c_{N-1} + N$. With $n \geq 2$, $C_1 = 1$

$C_{N-1} = c_{N-2} + N-1$

$C_{N-2} = c_{N-3} + N-2$

…

$C_2 = C_1 + 2$

$C_N = C_1 + 2 + 3 + \ldots + N = 1 + 2 + 3 + \ldots + N = N(N+1)/2 \in \Theta(N^2)$

2. Exercises

Solve the following recurrence relations with the iterative method

a. $C_N = C_{N/2} + 1000$ with $N \geq 2$, $C_1 = 0$
b. $C_N = 3C_{N/2} + N$ with $n \geq 2$, $C_1 = 0$
c. $C(N) = 2C(N/2) + 1$. With $N \geq 2$, $C(1) = 0$

Solve the following recurrence relations with Master theorem

d. $C_N = C_{N/2} + 1000$ with $N \geq 2$, $C_1 = 0$

e. $C_N = 3C_{N/2} + N$ with $n \geq 2$, $C_1 = 0$
f. C (N) = 2C (N/2) + 1. With N ≥2, C (1) = 0
g. $C_N = 4c_{N/2} + N$. With n ≥2, $C_1 = 1$
h. C (N) = 9C (N/3) + N
i. C (N) = C (2N/3) + 1
j. C (N) = 3C (N/4) + NlgN
k. $T(n) = T(n/3) + 2T(n/3) + n$
