American University of Beirut
Department of Computer Science
CMPS 211 - Fall 20-21
Practice Exercises Set 7
Recursive Algorithms

Exercise 1

For each of the problems perform the following tasks:

1. write the recursive definition of the function
2. Describe the algorithm in pseudocode
3. prove the correctness of the algorithm using induction.

Algorithm 1

Devise a recursive algorithm to find the floor of a non-negative real number x.

Algorithm 2

Given two non-negative integers x and y, devise a recursive algorithm for multiplying them. The time complexity of the algorithm should be $\in O(log(n))$.
HINT: $xy = 2(x \cdot (y/2))$ *when* y *is even,* $xy = 2(x \cdot \lfloor \frac{y}{2} \rfloor + x)$ *when* y *is odd*

Algorithm 3

Given a real number a and a non-negative integer n, devise a recursive algorithm to find a^{2^n}. The time complexity of the algorithm should be $\in O(n)$.
HINT: $a^{2^{n+1}} = (a^{2^n})^2$

Exercise 2

For the following algorithms, state the recurrence relation and use it to get the time complexity of the algorithm in Big-O notation.

Algorithm 1

int linear_search(**int list**: $a_1, a_2, ..., a_n$, **int** key, **int** index):
    if index < 0
        return -1;
    else if key = a[index]
        return index
    else
        return linear_search($a_1, a_2, ..., a_n$, key, index -1)

This is initially called on index = n.

Algorithm 2

int ternary_search(**int list**: $a_1, a_2, ..., a_n$, **int** key, **int** left, **int** right):
    If left $>$ right
        return 0

    $third_1 := \lfloor \frac{left+right}{3} \rfloor$
    $third_2 := \lfloor 2 \times \frac{left+right}{3} \rfloor$

    if key = a[$third_1$]
        return $third_1$

    else if key = a[$third_2$]
        return $third_2$

    else if key $<$ a[$third_1$]
        right := $third_1 - 1$

    else if key $>$ a[$third_2$]
        left := $third_2 + 1$

    else
        left := $third_1 + 1$
        right := $third_2 - 1$

    return ternary_search($a_1, a_2, ..., a_n$, key, left, right);

This is usually initially called as on left = 1, right = n.

Algorithm 3

int power(**int:** x, **int:** n)
    If $n = 0$
        return 1
    Else If $n = 1$
        return x
    Else If $(n\%2) = 0$
        return $power(x, n/2) \times power(x, n/2)$
    Else
        return $power(x, n/2) \times power(x, n/2) \times x$