Chapter 4: Algorithms
Correction of Exercises

I. Introduction:

Exercise 1:

Write an algorithm that returns the smallest value in the sequence s1, …, sn.

```
Input : s, n
Output : small, the smallest value of the sequence s
min(s, n) {
        small = s1
        for i = 2 to n
            if (si > small) // smaller value found
                small = si
        return small
 }
```

Exercise 2:

Write an algorithm that returns the sum of the sequence numbers s1, …, sn.

```
Input : s, n
Output : sum
seq_sum(s, n) {
        sum = 0
        for i = 1 to n
            sum = sum + si
                small = si
        return sum
 }
```

Exercise 3:

Write an algorithm that receives as input of the matrix of a relation R and tests whether R is reflexive.

Input : A (an n × n matrix of the relation R), n

Output : true, if R is reflexive; false, if R is not reflexive

```
is_reflexive(A, n) {
        for i = 1 to n
            if (A_ii == 0)
                return false
                small = si
        return true
}
```

II. Example of Algorithms:

Exercise 1: Trace the algorithm of the Text Search (slide 13) for the input

t = “balalaika” and p = “bala”

```
Input: p (indexed from 1 to m), m, t (indexed from 1 to n), n
Output: I
text_search(p, m, t, n) {
    for i = 1 to n – m + 1 {
        j = 1
        // i is the index in t of the first character of the substring
        // to compare with p, and j is the index p
        // the while loop compares ti … ti+m-1 and p1 … pm
        while (ti+m-1 == pj)
        j = j + 1
        if (j > m)
            return I
        }
    }
    return 0
}
```

First i and j are set to 1. The while loop then compares $t_1 \ldots t_4$= “bala” with p = “bala”. Since the comparison succeeds, the algorithm returns i = 1 to indicate that p was found in t starting at index 1 in t.

Exercise 2: Trace the algorithm of the Insertion Sort (slide 18) for the input

34 20 144 55

Slide 18 : the insertion sort

```
Input: s, n
Output: s (sorted)
insertion_sort(s, n) {
    for i = 2 to n {
        val = si // save si so it can be inserted into the correct place
        j = i - 1
        // if val < sj, move sj right to make room for si
        while (j ≥ 1 ∧ val < sj) {
        sj+1 = sj
      j = j – 1
        }
        sj+1 = val // insert val
    }
}
```

First, 20 is inserted in

34

Since 20 < 34, 34 must move one position to the right

	34

Now 20 is inserted

20	34

Since 144 > 34, it is immediately inserted to 34's right

20	34	144

Since 55 < 144, 144 must move one position to the right

20	34		144

Since55 > 34, 55 is now inserted

20	34	55	144

The sequence is now sorted.

<u>Exercise 3:</u> Trace the algorithm of the Suffle (slide 21) for the input

34 57 72 101 135

Assume that the values of *rand* are

rand(1, 5) = 5, *rand(2, 5) = 4*

rand(3, 5) = 3, *rand(4, 5) = 5*

Input : A, n

Output : A (shuffled)

shuffle(A n) {

for i = 1 to n - 1

swap(Ai, Arand(i, n)

}

We first swap Ai and Aj, where i = 1 and j = rand(1, 5) = 5.

After the swap we have

135	57	72	101	34
i				j

We next swap Ai and Aj, where i = 2 and j = rand(2, 5) = 4.

After the swap we have

135	101	72	57	34
	i		j	

We next swap Ai and Aj, where where i = 3 and j = rand(3, 5) = 3.

The sequence is unchanged.

We next swap Ai and Aj, where where i = 4 and j = rand(4, 5) = 5.

After the swap we have

135	101	72	34	57

i j

Exercise 4: Write the algorithm that returns the index of the last occurrence of the value *key* in the sequence s1, …, sn. If *key* is not in the sequence, the algorithm returns the value 0. For example, if the sequence

12 11 12 23

And *key* is 12, the algorithm returns to 3.

Input : s (the sequence s1, …, sn), n, and key

Output : i (the index of the last occurrence of key in s), or 0 if key is not in s

```
Reverse_linear_search(s, n, key) {
    i = n
    while ( i ≥ 1) {
        if (si == key)
            return i
            i = i – 1
    }
    return 0
}
```

Exercise 5:

The selection sort algorithm sorts the sequence s1, …, sn in nondecreasing order by first finding the smallest item, say si, and placing it first by swapping s1 and si. It then finds the smallest item in s2, …., sn, again say si, and places it second by swapping s2 and si. It continues until the sequence is sorted.

Write selection sort in pseudocode.

```
Input : s (the sequence s1, …, sn) and n
Output : s (sorted in nondecreasing order)
        Selection_sort(s, n) {
                For i = n to n – 1 {
                  // find smallest in si, …, sn
                  small_index = i
                  for j = i + 1 to n
                      if (si < s small_index)
                          s small_index = j
                   swap(si, s small_index)
                }
         }
```

III. Analyse of Algorithms:

Exercise 1 :

Find a theta notation for each expression

a) $6n + 1 = \Theta(n)$
b) $3n^2 + 2n \lg n = \Theta(n^2)$
c) $2 + 4 + 6 + \ldots + 2n = \Theta(n^2)$
d) $2 + 4 + 8 + 16 + \ldots + 2^n = \Theta(n^{n+1})$

Exercise 2 :

Find a theta notation for the number of times the statement x = x +1 is executed

a) or i = 1 to 2n

x = x + 1

$\Theta(n)$

b) for i = 1 to 2n

for j = 1 to n

x = x + 1

$\Theta(n^2)$

c) for i = 1 to n

for j = 1 to n

for k = 1 to n

x = x + 1

$\Theta(n^3)$

Exercise 3 :

_Let consider that

$$1 + 2 + \ldots + n = An^2 + Bn + C$$

For all n, and for some constant A, B and C.

a) Assuming that this is true, plug in n = 1, 2, 3 to obtain three equations in the three unknowns A, B and C.

When n = 1, we obtain

$$1 = A + B + C$$

When n = 2, we obtain

$$3 = 4\,A + 2\,B + C$$

When n = 3, we obtain

$$6 = 9\,A + 3\,B + C$$

b) Solve for A, B and C with the three equations obtained in the previous question.

Solving this system for A, B, C. We obtain

$$\mathbf{A = B = ½, \qquad C = 0}$$

We obtain this formula $1 + 2 + \ldots + n = \frac{n2}{2} + \frac{n}{2} + 0 = \frac{n\,(n+1)}{2}$

c) Prove using the mathematical induction that the statement is true.

We must show that for all n, if equation n is true : Sn = n(n+1)/2 then, equation n+1 is also true. Sn+1= (n+1)(n+2)/2

a) ***Basis Step:*** S(1) : 1 =1(2)/2 = 1 is true

b) ***Inductive Step:*** If S(n) = n(n+1)/2 is true.

$$\begin{aligned} S(n+1) &= n(n+1)/2 + (n+1) \\ &= \{n(n+1) + 2(n+1)\}/2 \\ &= (\,n^2 + n) + 2n + 2)\,/2 \\ &= (\,n^2 + 3n + 2\,)/2 \\ &= \{(n+1)\,(n+2)\}/2 \end{aligned}$$

So, S(n) is true for every positive integer n.

Exercise 4 :

Let consider the formula

$$\frac{b^{n+1} - a^{n+1}}{b-a} = \sum_{i=0}^{n} a^i b^{n-1} \qquad 0 \leq a \leq b$$

Prove that

$$\frac{b^{n+1} - a^{n+1}}{b-a} < (n+1)b^n \qquad 0 \leq a \leq b$$

Replacing a by b in the sum yields

$$\frac{b^{n+1} - a^{n+1}}{b-a} = \sum_{i=0}^{n} a^i b^{n-1} < \sum_{i=0}^{n} b^i b^{n-1} = \sum_{i=0}^{n} b^n = (n+1)b^n$$

IV. Recursive Algorithms:

Exercise 1:

Trace the algorithm of the computing n Factorial (slide 48) for n = 4.

Slide 47 : the computing n Factorial

This recursive algorithm computes n!

Input : n, an integer greater than or equal to 0

Output : n!

```
1. factorial(n) {
2.      if (n ==0)
3.          return 1
4.      return n * factorial(n – 1)
5. }
```

At line 2, since $4 \neq 0$, we proceed to line 4. The algorithm is invoked with input 3.

Exercise 2:

Let consider the formula

$$s_1 = 2, \qquad s_n = s_{n-1} + 2n \qquad \text{for all } n \geq 2$$

a) Write the recursive algorithm that computes: $s_n = 2 + 4 + 6 + \ldots + 2n$.

Input : n

Output : 2 + 4 + 6 + … + 2n.

```
1. sum(s, n) {
2.      if (n ==1)
3.          return 2
4.      return sum(n – 1) + 2
5. }
```

b) Proof using the mathematical induction that the recursive algorithm that computes s_n is correct.

a) **<u>Basic step :</u>** (n = 1) if n is equal to 1, we correctly return 2.

b) **<u>Inductive step:</u>** Assume that the algorithm correctly computes the sum when the input is n – 1. Now suppose that the input to this algorithm is n > 1. At line 2, since n ≠ 1, we proceed to line 4, where we invoke this algorithm with input n – 1. By the inductive assumption, the value returned, sum(n – 1), is equal to

$$2 + 4 + 6 + \ldots + 2\,(n - 1).$$

At line 4, we then return

$$\text{Sum}\,(n - 1) + 2n = 2 + 4 + 6 + \ldots + 2\,(n - 1) + 2n$$

Which is the correct value.

<u>Exercise 3:</u>

Write a recursive algorithm to find the maximum of a finite sequence of numbers. Give a proof using mathematical induction that your algorithm is correct.

Input : s (the sequence s1, …, sn) and the length n of the sequence

Output : the maximum value in the sequence

```
find_max(s, n) {
        if ( n == 1)
                return si
        x = find_max(s, n – 1)
        if (x > sn)
                return x
        else
                return sn
}
```

We prove that the algorithm is correct using the induction on n.

a) **<u>Basic step:</u>** (n= 1) if n = , the only item in the sequence is s1 and the algorithm correctly returns it.

b) **<u>Inductive step:</u>** Assume that the algorithm computes the maximum for input of size n – 1, and suppose that the algorithm receives input of size n. By assumption, the recursive call

x = find_max(s, n-1)

correctly computes x as the maximum value in the sequence s1, …, sn-1. If x is greater than sn, the maximum value in the sequence s1, …, sn is x, the value returned by the algorithm. If x is not greater than sn, the maximum value in the sequence s1, …, sn is sn, again, the value returned by the algorithm.

In either case, the algorithm correctly computes the maximum value in the sequence. The inductive step is complete, and we have proved that the algorithm is correct.

<u>Exercise 4:</u>

Use the mathematical induction to show that

$$f_n^2 = f_{n-1} f_{n+1} + (-1)^{n+1} \qquad \text{for all } n \geq 2$$

a) Basic step: (n = 2)

$$f_2^2 = 1 = 1 \times 2 - 1 = f_1 \ f_3 + (-1)^3$$

b) Inductive step: Assume that the statement is true.

$$\begin{aligned} f_n f_{n+2} + (-1)^{n+2} &= f_n \ (f_{n+1} + f_n) + (-1)^{n+2} \\ &= f_n \ f_{n+1} + f_n^2 + (-1)^{n+2} \\ &= f_n \ f_{n+1} + (f_{n-1} f_{n+1} + (-1)^{n+1}) + (-1)^{n+2} \\ &= f_{n+1} \ (f_n + f_{n-1}) = f_{n+1}^2 \end{aligned}$$

We can conclude that the statement is true.

Exercise 5:

Use the mathematical induction to show that

$$\sum_{k=1}^{n} f_k^2 = f_n f_{n+1}$$

for all n ≥ 1

a) Basic step: (n = 1)

$$f_1^2 = 1^2 = 1 = 1 \times 1 = f_1 \times f_2$$

b) inductive step:

$$\sum_{k=1}^{n+1} f_k^2 = \sum_{k=1}^{n} f_k^2 + f_{n+1}^2 = f_n f_{n+1} + f_{n+1}^2 = f_{n+1} (f_n + f_{n+1}) = f_{n+1} f_{n+2}$$

We can conclude that the statement is true.

Exercise 6:

Let assume the formula for differentiating products:

$$\frac{d(fg)}{dx} = f\frac{dg}{dx} + g\frac{df}{dx}$$

for all n ≥ 1

Use mathematical induction to prove that

$$\frac{dx^n}{dx} = nx^{n-1}$$

for n = 1, 2, ….

a) **Basic step: (n = 1)**

$$\frac{dx}{dx} = 1 = 1\, x^{1-1}$$

The statement is true.

b) **Inductive step:** Assume that the statement is true.

$$\frac{dx^{n+1}}{dx} = \frac{d\,(x \times x^n)}{dx} = x\,\frac{dx^n}{dx} + x^n\,\frac{dx}{dx} = x\, n\, x^{n-1} + x^n \times 1 = (n+1)\, x^n$$

We can conclude that the statement is true.