Exercises No 1

Exercise 1

Write pseudocode for an algorithm for finding real roots of equation $ax^2 + bx + c = 0$ for arbitrary real coefficient a, b, c. (You may assume the availability of the square root function $sqrt(x)$.)

Exercise 2

Describe the standard algorithm for finding the binary representation of a positive decimal integer.

Exercise 3

Consider the algorithm for sorting problem (Algorithm 1) that sorts an array by counting, for each of its elements, the number of smaller elements and then uses this information to put the element in its appropriate position in the sorted array.

Exercise 4

For each of the following functions, indicate the class $\Theta(g(n))$ the function belongs to. (use the simplest $g(n)$ possible in your answers). prove your assertions.

- $(n^2 + 1)^10$
- $\sqrt{10n^2 + 7n + 3}$
- $2n \lg (n + 2)^2 + (n + 2)^2 \lg \frac{n}{2}$

Algorithm 1 *ComparisonCountingSort(A[0 . . . n-1])*

```
1: {Sorts an array by comparison counting}
2: Input: Array A[0 . . . n − 1] of values.
3: Output: Array S[0 . . . n − 1] of values.
4: for i=0 to n-1 do
5:    Count[i] = 0
6: end for
7: for i=0 to n-2 do
8:    for j=i+1 to n-1 do
9:       if A[i]¡A[j] then
10:         Count[j] = Count[j] + 1
11:      else
12:         Count[i] = Count[i] + 1
13:      end if
14:   end for
15: end for
16: for i=0 to n-1 do
17:   S[Count[i]] = A[i]
18: end for
19: return S
```

- $2^{n+1} + 3^{n-1}$
- $\lfloor \log_2 n \rfloor$

Exercise 5

List the following functions according to their order of growth from the lowest to the highest:

- $(n-2)!$
- $5 \lg (n+100)^10$
- 2^{2^n}
- $0.001n^4 + 3n^3 + 1$
- $\ln^2 n$

- $\sqrt[3]{n}$
- 3^n

Exercise 6

Compute the following sums:

- $1 + 3 + 5 + 7 + \ldots + 999$
- $2 + 4 + 8 + 16 + \ldots + 1024$
- $\sum_{i=3}^{n+1} 1$
- $\sum_{i=3}^{n+1} i$
- $\sum_{i=0}^{n-1} i(i+1)$
- $\sum_{j=1}^{n} 3^{j+1}$
- $\sum_{i=1}^{n} \sum_{j=1}^{n} ij$
- $\sum_{i=1}^{n} \frac{1}{i(i+1)}$

Exercise 7

Find the order of growth of the following sums. Use the $\Theta(g(n))$ with the simplest function $g(n)$ possible.

- $\sum_{i=0}^{n-1} (i^2 + 1)^2$
- $\sum_{i=2}^{n-1} \lg i^2$
- $\sum_{i=1}^{n} (i+1)2^{i-1}$
- $\sum_{i=0}^{n-1} \sum_{j=0}^{n-1} (i+j)$

Exercise 8

Consider the following algorithm (Algorithm 2)

1. What does this algorithm compute ?

Algorithm 2 *Mystery(n)*

Input: A non-negative integer n.
$S = 0$
for *i=1 to n* **do**
    $S = S + i \times i$
end for
return S

2. What is the basic operation ?

3. How many times is the basic operation executed ?

4. What is the efficiency class of this algorithm ?

5. Suggest an improvement, or a better algorithm altogether,a nd indicate its efficiency class.

Exercise 9

Consider the following algorithm (Algorithm 3)

Algorithm 3 *Secret(A[0 . . . n-1])*

Input: Array $A[0 \ldots n-1]$ of n real numbers.
$minval = A[0]$
$maxval = A[0]$
for *i=0 to n-1* **do**
    if $A[i] < minval$ **then**
        $minval = A[i]$
    end if
    if $A[i] > maxval$ **then**
        $maxval = A[i]$
    end if
end for
return $maxval - minval$

Answer questions (1)-(5) of exercise 8 about this algorithm.

Exercise 10

Consider the following algorithm (Algorithm 4)

Algorithm 4 *Enigma(A[0 ... n-1, 0 ... n-1])*

```
1: Input: Array A[0...n − 1] of n real numbers.
2: for i=0 to n-2 do
3:    for j=i+1 to n-1 do
4:       if A[i, j] ≠ A[j, i] then
5:          return false
6:       end if
7:    end for
8: end for
9: return true
```

Answer questions (1)-(5) of exercise 8 about this algorithm.

Exercise 11

Solve the following recurrence relations.

1. $x(n) = x(n-1) + 5$ for $n > 1$, $x(1) = 0$
2. $x(n) = 3x(n-1)$ for $n > 1$, $x(1) = 4$
3. $x(n) = x(n-1) + n$ for $n > 0$, $x(0) = 0$
4. $x(n) = x(n/2) + n$ for $n > 1$, $x(1) = 1$ (solve for $n = 2^k$)
5. $x(n) = x(n/3) + 1$ for $n > 1$, $x(1) = 1$ (solve for $n = 3^k$)

Exercise 12

Consider the following recursive algorithm (Algorithm 5) for computing the sum of the first n cubes: $S(n) = 1^3 + 2^3 + \ldots + n^3$

1. Set up and solve a recurrence relation for the number of times the algorithm's basic operation is executed.
2. How does this algorithm compare with the straightforward nonrecursive algorithm for computing this sum ?

Algorithm 5 *S(n)*

1: **Input**: A positive integer n.
2: **Output**: The sum of the first n cubes
3: **if** $n = 1$ **then**
4: **return** 1
5: **return false**
6: **else**
7: $S(n-1) + n \times n \times n$
8: **end if**