IDMA 2026
Problem set 1

Abdulkareem Al-Rifai, kbt185

16-02-26

Contents

1 Different representations of integers

1.a Write the binary number $(110)_2$ in decimal notation

Binary notation refers to the number system of base 2, where as the decimal notation refers to base 10. I will start by expanding $(110)_2 = 1 * 2^2 + 1 * 2^1 + 0 * 2^0 = 6$, now I will try to write the number 6 in decimal notation. To do that I will use the equation

$$m = q_0 \cdot b + r_0 ...$$

$$...m = q_n \cdot b + r_n$$

where : m : is the number we are trying to rewrite, b : the base, we continue running the recursion process until $q_n = 0$

$$6 = 0 \cdot 10 + 6$$

since $q_0 = 0$ we stop and now we have

$$(110)_2 = (6)_{10}$$

1.b Write the decimal number 110 in binary notation

Following the steps of the previous question, we have

$$(110)_{10} = 1 \cdot 10^2 + 1 \cdot 10^1 + 0 \cdot 10^0 = 110$$

to convert the number 110 to base 2 we use the equation :

$$110 = 55 \cdot 2 + 0$$

$$55 = 27 \cdot 2 + 1$$

$$27 = 13 \cdot 2 + 1$$

$$13 = 6 \cdot 2 + 1$$

$$6 = 3 * 2 + 0$$

$$3 = 1 \cdot 2 + 1$$

$$1 = 0 \cdot 2 + 1$$

Then we have that

$$(110)_{10} = (1101110)_2$$

1.c Write the octal number $(110)_8$ in decimal notation

$$(110)_8 = 1 \cdot 8^2 + 1 \cdot 8^1 + 0 \cdot 8^0 = 64 + 8 + 0 = 72$$

$$72 = 7 \cdot 10 + 2$$

$$7 = 0 \cdot 10 + 7$$

hence

$$(110)_8 = (72)_{10}$$

2 Find the greates common divisor using the algorithm we learned, show the steps then express d as linear combination of m and n

2.a m=38, n=14

2.b m=117, n=69

3 Consider the following algorithm

```
j := 1
while (j<=n) {
        A[j] := 0
        for i := j downto 1 {
                A[j] := A[j]+i*i
        }
        j := j+1
        }
```

3.a Explain what the algorithm does, and what are the number stored and computed in A

Let start with the easier part, what are the values stored in array A, from j to n. The values stored in A are the sum of squares of j downto 1, meaning that for A[2], A[2] is equal to $2^2 + 1^2 = 5$.

j is an index variable showing our position in the array. This algorithm has 2 loops, the outer while-loop runs n times, going through every element in the array. The inner for-loop, that also run n times, is used to update the values of the array from j to n. Initially A[j:n] = 0, a condition/ command specified at the begining of the outer-loop. The inner for-loop takes A[j] and iterates throught it i times, until i hits 1. At each iteration in the for loop, the value of A[j] is updated, then the updated value is used for the next iteration, once i hits 1, the for-loop terminates, and j is updated to j+1. To reduce abstraction, suppose that n=3 and execute the algorithm at j=3 manually. j=3

$$A[3] = 0 \rightarrow j \leq n(true, continue) \tag{1}$$

we enter for-loop

$$A[3] = 0 + 3 * 3 = 9 \rightarrow A[3'] = A[3] + 2 * 2 = 13 \rightarrow A[3''] = A[3'] + 1 * 1 = 13 + 1 = 14 \tag{2}$$

$$i = 1\textit{(for-oop terminates)} \rightarrow j = 3 + 1 = 4 \rightarrow 4 \leq 3\textit{(j larger than, outer-loop terminates)} \tag{3}$$

3.b Provide an asymptotic analysis of the running time as function of the array size n

We can analyse the running time of the algorithm by looking at the inner and outer loop. Assuming that every operation takes a constant time c, we can avoid detailed analysis and look at the recursive operations. The outer-loop will run n-times, and for each j in the while loop run, the for loop runs j times. This means that if j = 3, the for loop runs 3 times, j=n, the for-loop with run n-times.The complexity of this such a relationship is n*n = $\mathbf{n^2}$. Asymptotic analysis refers to the behavior of the algorithm as n grows to infinity.

$$O(n^2) \tag{4}$$

3.c Improve the code to run faster while retaining the same functionality, analyse the time complexity

This algorithm should run faster considering that it only has one loop that runs n times, rather than 2-loops running n times each.

```
A[1:n]
for i = 2 up to n :
        if i <= n :
                A[i] = i*i +A[i-1]
        i = i+1
```

4

4.a

4.b

4.c