IT41: Midterm May 16, 2024

Exercise 1: $2p + 3q$-Decomposition

In this problem we want to know in how many ways we can decompose an integer n as a sum of 2 and 3. For example $8 = 3 + 3 + 2 = 2 + 2 + 2 + 2$. We will say 8 has two decompositions. However $5 = 2 + 3$ has a unique decompostion and $n = 1$ has no decomposition.

1. Find the 3 decompositions of 12.
2. Consider the following recursive algorithm:

```
Data: n an interger
Result: The number of possible decompositions of n as a sum of 3 and 2
if n = 1 then
    Return 0
end
else
    if n = 2  or n = 3 then
        Return 1
    end
    else
        Return Number_of_Decomposition(n − 2)+Number_of_Decomposition(n − 3)
    end
end
```

Algorithm 1: Number_of_Decomposition(n)

   (a) Is the algorithm convergent ?

   (b) Prove its correctness.

Exercise 2: Bubble Sort

In this exercise one studies a sorting algorithm called Bubble Sort whose pseudo-code is:

```
Data: A array of length n
Result: Array A sorted in nondecreasing order
for i ← 1 to n − 1 do
    for j ← 1 to n − i do
        if A[j] > A[j + 1] then
            Exchange A[j] and A[j + 1]
        end
    end
end
```

Algorithm 2: Bubble Sort

1. Consider the following array $T_1 = [2, 5, 3, 4, 1]$. Apply the algorithm to sort T_1 (here $n = 5$) by showing for all values of i, j how T changes (warning: it is a pseudo-code so here $T_1[1] = 2$).
2. What is the complexity of this algorithm ?
3. One would like to prove the correctness of the Bubble Sort algorithm. Consider the following loop invariant after the ith iteration of the first (outer) forloop:

$$\mathcal{P}(i): \text{ The sub-array}[T[n-i], ...T[n]] \text{ is sorted}$$

   (a) Explain why $\mathcal{P}(0)$ is true.

   (b) Show that if $\mathcal{P}(i)$ is true then $\mathcal{P}(i+1)$ is true

   (c) Deduce that Bubble Sort is correct.

4. Consider now the array $T_2 = [1, 2, 3, 5, 4]$ what is the state of T after the iteration of the inner loop for $i = 1$?

5. Consider the following variation of the original pseudo-code:

```
Data: A array of length n
Result: Array A sorted in nondecreasing order
for i ← 1 to n − 1 do
    swapped ← false
    for j ← 1 to n − i do
        if A[j] > A[j + 1] then
            Exchange A[j] and A[j + 1]
            swapped ← true
        end
    end
    if not swapped then
        break
    end
end
```

Algorithm 3: Bubble Sort optimized

what would be the effect of this variation on the array T_2 ?

6. What is the complexity of Bubble sort optimized in the best case scenario ?

Exercise 3: Convolution

In image processing the convolution of matrices is an algebraic operation that allows us to apply filters on an image. Convolution is also used in neural networks (Convolutional Neural Network). In this exercise one considers the convolution of two arrays.

Let $T_1 = [a_0, \dots, a_{n-1}]$ and $T_2 = [b_0, \dots, b_{n-1}]$ be two arrays of size n. First complete T_1 and T_2 with zeros to make them two arrays of size $2n$, $\tilde{T_1} = [a_0, \dots, a_{n-1}, 0, \dots, 0]$ and $\tilde{T_2} = [b_0, \dots, b_{n-1}, 0 \dots, 0]$, then one defines $T_1 \star T_2$ to be the array of size $2n$ defined by:

$$T_1 \star T_2[k] = \sum_{i=0}^{k} \tilde{T_1}[k-i] \times \tilde{T_2}[i] \text{ for } k = 0, \dots 2n-1 \tag{1}$$

The inner product of T_1 and T_2 of size n is the algebraic operation defined by

$$T_1 \cdot T_2 = [a_0 b_0, a_1 b_1, \dots, a_{n-1} b_{n-1}] \tag{2}$$

1. What is the complexity of the (naive) algorithm that computes $T_1 \star T_2$ in terms of the size n following the formula (1) ?
2. What is the complexity of computation of $T_1 \cdot T_2$?
3. Consider $n = 2$ and $T_1 = [1, 1]$ and $T_2 = [0, 2]$.
   (a) Compute $T_1 \star T_2$.
   (b) What is the matrix corresponding to DFT_4 ?
   (c) Compute $Y_1 = DTF_4(\tilde{T_1})$ and $Y_2 = DFT_4(\tilde{T_2})$ where $\tilde{T_i}$ is the array of size 4 obtained from T_i by completing with zeros.
   (d) Compute $Y = Y_1 \cdot Y_2$
   (e) Check that $T_1 \star T_2 = DFT_4^{-1} Y$ (recall from exercise 6 recitation 3: $DFT_4 = \frac{1}{4}\overline{DFT_4}$).
4. Suppose that what we checked for and found in the $n = 2$ case is true in general, i.e. for all arrays T_1 and T_2 of size n we have the following formula:

$$T_1 \star T_2 = DFT_{2n}^{-1}((DFT_{2n} T_1) \cdot (DFT_{2n} T_2)) \tag{3}$$

   (a) Propose an algorithm that computes $T_1 \star T_2$ based on Eq (3).
   (b) What is the complexity of this algorithm ?