Eidgenössische
Technische Hochschule
Zürich

Ecole polytechnique fédérale de Zurich
Politecnico federale di Zurigo
Federal Institute of Technology at Zurich

Institut für Theoretische Informatik
Peter Widmayer
Thomas Tschager
Antonis Thomas

13th April 2016

Datenstrukturen & Algorithmen Exercise Sheet 7 FS 16

Exercise 7.1 *Frequencies.*

Please note: In the original version of this exercise, the keys K and L were accidentally swapped.

a) We want to encode a long text in binary numbers. The text consists of five characters and we are given for each character the frequency of appearance. Construct an optimal coding tree and encode the word `AGGLGSK`.

Key	A	G	K	L	S
Frequency	9	8	8	7	7

b) We now want to construct an optimal search tree. We assume an alphabetical order on the keys. Besides the access frequencies of the keys from a), we are given the access frequencies for every interval between two neighboring keys. For example, in 18% of all cases we get a query for some key larger than A and smaller than G. Compute an optimal search tree and provide the weighted path length of your tree.

Interval	(_,A)	(A,G)	(G,K)	(K,L)	(L,S)	(S,_)
Frequency	0	18	7	0	19	17

Exercise 7.2 *Enumerating Palindromes.*

This exercise is concerned with dynamic programming. *Please consider the remarks about the description of a dynamic program on exercise sheet 6.*

A *palindrome* is a word whose meaning may be interpreted the same way in either forward or reverse direction, e.g. the word `RACECAR`. Formally, a palindrome is a sequence $\langle a_1, \ldots, a_n \rangle$ where either $n = 1$, or $a_1 = a_n$ and $\langle a_2, \ldots, a_{n-1} \rangle$ is a palindrome (for $n = 2$ we only require $a_1 = a_2$). Let $A[1..n]$ be an array storing a string of length n. A subarray $A[i..j]$, $1 \leq i \leq j \leq n$, is called *palindrome in* A if $\langle A[i], \ldots, A[j] \rangle$ is a palindrome.

Example: The array [`L`, `A`, `R`, `A`] contains the palindromes `A`, `R`, `L` and `ARA` (the palindrome `A` occurs twice). The array [`A`, `N`, `N`, `A`] contains the palindromes `A`, `N`, `NN` and `ANNA` (the palindromes `A` and `N` occur twice).

a) Let A be an array containing a string of length n. Describe a dynamic programming algorithm that outputs all pairs (i, j) where $\langle A[i], \ldots, A[j] \rangle$ is a palindrome. Provide also the running time of your solution.

Example: For the input [`L`, `A`, `R`, `A`], the output consists of the pairs $(1, 1)$, $(2, 2)$, $(3, 3)$, $(4, 4)$, $(2, 4)$. For the input [`A`, `N`, `N`, `A`], the output consists of the pairs $(1, 1)$, $(2, 2)$, $(3, 3)$, $(4, 4)$, $(2, 3)$, $(1, 4)$. In the output, no special order of the pairs is required.

Hint: Notice that the exercise can easily be solved without dynamic programming by trivial enumeration of all palindromes in time $\mathcal{O}(n^3)$. We search for a more efficient algorithm.

b) Suppose that the algorithm of a) computed the DP table already. Describe in detail, how a longest palindrome in A can be extracted from the DP table. Provide also the required running time.

Exercise 7.3 *Longest common subsequence problem (**Programming Exercise**).*

In this exercise we are going to implement a dynamic programming algorithm for solving the *longest common subsequence problem*. The problem is defined as follows. We are given two strings of text $A = a_1 \cdots a_n$ and $B = b_1 \cdots b_m$, where $a_1, \ldots, a_n, b_1, \ldots, b_m$ are characters from an alphabet Σ, and we look for the length of a longest string that is a subsequence of both A and B. For example, if $A =$ "AGCAT" and $B =$ "GAC", the longest common subsequences would be one of the following: "AC","GC","GA", with length 2.

Input The first line contains only the number t of test instances. After that, we have exactly two lines per test instance. The first line contains the sequence A, and the second line contains the sequence B. Note that the alphabet used is $\Sigma = \{A, B, \ldots, Z\}$.

Output For every test instance we output only one line. This line contains the length of the longest common subsequence.

Example

Input:

```
2
AGCAT
GAC
ROCK
ROLL
```

Output:

```
2
2
```

Directions This exercise is to be solved using *dynamic programming*. We provide you with a template where you have to fill in the functions `computeTable` and `longestCommonSubseqLen`.

In function `main` you will also see how to read strings with Java scanner.

There is only one testset for 100 point in this exercise.

Hand-in: Wednesday, 20th April 2016 in your exercise group.