University of Zurich

Department of Informatics
Database Technology Group
Prof. Dr. Anton Dignös

Informatics II
Exercise 11

May 10, 2021

Goals:

- Solve dynamic programming problems on 2-D array.
- Practice writing programs with dynamic programming algorithms.
- Compare programs in dynamic programming and recursive approaches.

Dynamic programming

Task 1 Longest Palindromic Subsequence (LPS) Problem A palindromic sequence is a sequence that reads the same backwards as forwards, e.g., "racecar" or "lagerregal". Given a string `A[]` with n characters, a subsequence is a sequence that can be derived from `A` by deleting some or no elements without changing the order of the remaining elements. Given a string `A[]`, LPS finds the length of the longest subsequence of `A` that is a palindrome. The idea for a dynamic programming solution is to compute an auxiliary 2-D matrix `L` with the dimension n × n. The value of element `L[i, j]` is the length of the longest LPS from the ith character to the jth character, both included. Note that $j \geq i$, and that the 0th character is the first character of `A`.

1. Determine the length of the LPS of the sequence "bbcab" using `L`.
2. Give a recursive solution to find out the length of the LPS of a given sequence `A`
3. Is a dynamic programming solution more efficient than a recursive solution? Explain.
4. Provide a recursive problem formulation for determining `L[i, j]` that is the length of the longest LPS from from $i-$th character to $j-$th character, both included.
5. Implement the dynamic programming solution using `L` in C.
6. Analyse the time complexity of the program in (5).

Task 2 Knapsack Problem Given n items, each with a weight `w[i]` and a value `v[i]`. Suppose we want to put some items in a knapsack of capacity W, i.e. the sum of weights of all the selected items must be less than or equal to W. Assume that each item can be used only once. Our goal is to maximise the value in the knapsack.

The idea is to create a two dimensional matrix `K` of $n+1$ rows and $W+1$ columns. As for `K[i][j]`, j means that the allocated weight in the knapsack is j, and i means that we can put first i items into the knapsack; `K[i][j]` means the maximum value that we can get by putting first i items with the allocated weight j. We can't put items in the knapsack if their weights exceeds the allocated weight.

1. Suppose we have a knapsack which can hold $W = 5$ and there are $n = 3$ items to choose from. The values of these items are `v={10,20,30}` and the weights of these items are `w={1,2,3}`. Determine the maximal value in the knapsack.

2. Write a C program that determines which items should be put into the knapsack and the corresponding value of selected items. Your program should work for general cases as well (not only the first sub-task).

More exercises: dynamic programming tasks in the exercises and exams in the past.