Departement of Computer Science
Markus Püschel, David Steurer
Johannes Lengler, Gleb Novikov, Chris Wendler

28. October 2019

Algorithms & Data Structures — Exercise sheet 6 — HS 19

Exercise Class (Room & TA): ______________________

Submitted by: ______________________

Peer Feedback by: ______________________

Points: ____

Submission: On Monday, 04 November 2019, hand in your solution to your TA *before* the exercise class starts. Exercises that are marked by * are challenge exercises. They do not count towards bonus points.

Exercise 6.1 *Heapsort* (1 point).

Given the array [A, L, G, O, R, I, T, H, M], we want to sort it in ascending alphabetical order using Heapsort.

a) From the lecture and the script you know a method to construct a heap in linear time. Draw the resulting max binary heap if this method is applied to the above array.

b) Sort the above array in ascending alphabetical order with heapsort, beginning with the heap that you obtained in a). Draw the array after each intermediate step in which a key is moved to its final position.

Exercise 6.2 *Sorting algorithms* ***(This exercise is from January 2019 exam)*** .

Below you see four sequences of snapshots, each obtained during the execution of one of the following algorithms: `InsertionSort`, `SelectionSort`, `QuickSort`, `MergeSort`, and `BubbleSort`. For each sequence, write down the corresponding algorithm.

3	8	5	4	1	2	7	6
3	5	4	1	2	7	6	8
3	4	1	2	5	6	7	8

____

3	8	5	4	1	2	7	6
3	5	8	4	1	2	7	6
3	4	5	8	1	2	7	6

____

3	8	5	4	1	2	7	6
3	8	4	5	1	2	6	7
3	4	5	8	1	2	6	7

____

3	8	5	4	1	2	7	6
3	6	5	4	1	2	7	8
3	2	5	4	1	6	7	8

____

Exercise 6.3 *Finding nouns* **(2 points)**.

Assume that you are working at the department of linguistics. You have several files on your computer that contain nouns of a very rare language (words in this language are written using standard English alphabet). Initially, the nouns were separated by spaces, but due to some technical error in your text editor, all the spaces disappeared. For example, if the original file contained

kuzdra bokr gostak doshes

then the corrupted file contains

kuzdrabokrgostakdoshes

Fortunately, you can call an expert in this language and ask him whether a string is a noun of this language or not. Your goal is to recover original sequences of nouns from corrupted files.

a) Provide a *dynamic programming* algorithm that takes as input a content of some file S (which is a string of English letters) and determines whether one can split S by spaces to get a sequence of nouns of the target language. Assume that you call an expert using some function $f(x)$ that returns true if a string x is a noun of a target language and false otherwise. Number of calls of f should be $\mathcal{O}(n^2)$ where n is a length of S (i.e. n is the number of letters in S). Your DP table should be *one-dimentional* and number of entries should be *linear*, i.e. $\mathcal{O}(n)$.

   For example, if all the nouns of the language are $\{$bokr, bokrgos, doshes, drabok, gostak, kuz, kuzdra, takdos$\}$, (that is, function f outputs true only at such strings), then your algorithm should output true on input “kuzdrabokrgostakdoshes” and false on input “kuztakdoshes”.

   Address the following aspects in your solution:

   1) *Definition of the DP table*: What is the meaning of each entry?
   2) *Computation of an entry*: How can an entry be computed from the values of other entries? Specify the base cases, i.e., the entries that do not depend on others.
   3) *Calculation order*: In which order can entries be computed so that values needed for each entry have been determined in previous steps?
   4) *Extracting the solution*: How can the final solution be extracted once the table has been filled?

   Specifically, you can use the following scheme:

   Meaning of a table entry (in words):

   $DP[i]$: ____

Computation of an entry (initialization and recursion):

Order of computation:

Computing the output:

b) State the running time of your algorithm in part a) in Θ-notation in terms of n (where n is the length of S). You can assume that calls of f take unit time.

c) Use DP table from part a) to efficiently find a sequence of nouns in the case when such a sequence exists. More precisely, provide an algorithm that takes as input a string S and a DP table from part a) and either splits S by spaces so that each string surrounded by spaces is a noun in a target language, or outputs false if it's not possible to split S in this way. If there are more than one possibility to split S in such a way, your algorithm can output any of these possibilities.

For example, if the input is

kuzdrabokrgostakdoshes

and the nouns of the language are {bokr, bokrgos, doshes, drabok, gostak, kuz, kuzdra, takdos}, then your algorithm should output

kuzdra bokr gostak doshes

Hint: *Follow your answer back through your DP table (Rückverfolgen).*

d) State the running time of your algorithm in part c) in Θ-notation in terms of n (where n is the length of S). You can assume that calls of f take unit time.