Algorithms and Complexity
2017
Extra Mästarprov 1: Algorithms

This test is given to students who failed to get E on the ordinary Mästarprov 1. It consists of two problems. If both problems are solved correctly (basically) the test gives grade E. Your solutions should be handed in latest May 19th 16.00. No collaboration is allowed.

1. At the service center

Let us assume that we have a service center where students can come with there questions. Let us assume that n students arrive and that each student i has a task which it takes t_i minutes to handle. There is just one line so the students have to wait for their turn. For each student we define the *service time* as the sum of the time the student has to wait plus the time for the student's task. We define the *total service time* (TST) as $\sum_i s_i$. This is the sum of the individual times the students spend in the service center.

For instance, if we have 3 students and serve them in order $3, 1, 2$, we get $s_1 = t_3 + t_1$, $s_2 = t_3 + t_1 + t_2$, $s_3 = t_3$. Then we have $TST = 2t_1 + t_2 + 3t_3$.

It can be shown that if the students are served in random order, the mean value of TST is $\frac{n+1}{2} \sum_i t_i$. But if we want to minimize TST we can do better than this. Describe an algorithm in pseudo-code that find the optimal order to serve the students in, given that we want to minimize TST. Show that your algorithm is correct and analyze the complexity.

2. Readable subsets

We have a string $s = s_1, s_2, ..., s_n$ of normal letters. Some substrings of consecutive letters of s might form readable words. We want to find a maximal (maximum number of words) set of disjoint readable words in the string. The words don't have to form a readable sentence.

As an example, the string WHATZZHOWQRUHELP of contains WHAT HOW HELP as an obviously maximal set.

How do we find such a set of maximal size? Here is an idea using dynamic programming: If the string has length n we can define $M[k]$ as the size of an optimal set when we have the substring of the first k letters in the string. How do we find $M[k+1]$? Maybe we can test if there is a readable word ending with letter $k+1$. If so, then... Or else ...

Use this idea and implement a dynamic programming algorithm (preferably in pseudo-code) that solves to problem, that is, finds $M[n]$ in polynomial time in n. We assume that we can use a dictionary function $read[w]$ which decides if w is a readable word in time $O(1)$. (Read returns TRUE or FALSE.) Analyze the complexity and explain why your algorithm is correct.