Algorithms and Complexity
2014
Mästarprov1: Algorithms

Mästarprov 1 should be solved individually in written form and presented orally. No collaboration is allowed.

Written solutions should be handed in latest on **Friday, February 28th 17.00,** to Mladen (personally or his mailbox). Be sure to save a copy of your solutions. Mästarprov 1 is a mandatory and rated part of the course. The test consists of four tasks. The test is roughly graded as follows: Two task correctly solved give an E. Three tasks correctly solved give a C and all tasks correctly solved give an A. You can read more about the grading criteria and the final grade on the course web page. The report should be written in English.

In all problems you should give an analysis of the time complexity of your algorithm and you should be able to argue for its correctness.

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.

3. A tale of two strings

We have two list $(x_1, x_2, ..., x_n)$ and $y_1, y_2, ..., y_m)$ of real numbers. The list are sorted in ascending order. We can assume that no pair of numbers from either lists are equal. So there are $n+m$ numbers. We want to find the median, that is the number $a = x_i$ or y_j such that exactly $\lceil \frac{n+m}{2} - 1 \rceil$ numbers from the lists are smaller than a. We can solve this problem in time $O(max(n, m))$.(We can merge the lists and then find the median.) But there is a better way of doing it. Develop an algorithm that solves this problem in time $O(\log(max(n, m)))$. Show that your algorithm is correct and carefully compute the complexity.

4. Bad railroad

In this problem we have a railroad which we can represent as a real line with $n+1$ points $x_0, x_1, x_2, ..., x_n$ corrsponding to stations s_0 = start position, $s_1, s_2, ..., s_n$ = end postion. We have a passenger train going from s_0 to s_n. There are m passengers boarding the train at s_0 and they want to go to different stations along the line. Normally, the train would stop at all n stations, s_0 excluded. But on a particular day, for some reasons the train can just stop at $k < n$ stations between s_0 and s_n. The staff on the train must decide at which stations the stops should be, guided by the demands of the passengers. They collect information on the form $g_1, g_2, ..., g_m$ where g_i is the index of the station passenger i wants to go to. The staff should find a set $h_0, h_1, h_2, ..., h_k$ of indices of the stations the train will stop at and $0 = h_0 < h_1 < h_2 < ... < h_k < n$. We can see that passengers with $g_j = n$ don't cause any problems. The staff now employs the policy that if for any passenger, say j, the train doesn't stop at g_j, the passenger is asked to get of the train at the largest h_i such that $h_i < g_j$. (This includes the possibility that the passenger "get off" the train at h_0, i.e., doesn't border the train at all.)

Of course, this will make some passengers very angry. How angry? The mathematically inclined driver of the train makes the following estimate. The anger $anger(j)$ of passenger j will be 0 if the train stops at station number g_j. Otherwise, if the passenger is forced to get of at h_i we have the estimate $anger(j) = A + B\sqrt{x_{g_j} - x_{h_i}}$, where A and B are some positive constants.

The staff wants to find $h_0, h_1, h_2, ..., h_k$ such that $\sum_j anger(j)$ is minimal. Find an algorithm that solves this problem. The algorithm must be polynomial in n, m, k.