AROUND OPTIMALITY OF DYNAMIC PROGRAMMING ALGORITHMS

PROJECT DESCRIPTION FOR THE GENERAL PUBLIC

Dynamic programming is one of the basic and very popular techniques for constructing algorithms. It works by splitting a problem to smaller subproblems and solving them, one by one, from the smallest to the largest. It differs from a similar *divide-and-conquer* technique in that the subproblems need not to be disjoint. Dynamic programming has plentiful of applications, e.g. in biological sequence alignment, resource allocation, or graph problems.

There is a growing list of recent *lower bounds* implying that for a number of problems with natural dynamic programming algorithms these algorithms are optimal, meaning that any faster algorithms are unlikely to exist.

Even though this kind of results might seem pessimistic at a first glance, they play an important role in computer science, both in theory and in practice. In practice, they let us avoid wasting time on optimizing algorithms that cannot be further optimized. In theory, they give us a good insight into the structure of computational problems.

Unfortunately, currently known lower bounds mentioned above significantly differ from each other, even if they target very similar problems. This leads us to the general question of optimality of dynamic programming algorithms. The ultimate goal would be to identify structural properties of dynamic programs which lead to tight lower bounds.

For this project we selected a few computational problems which are elusive for existing techniques. We intend to study their complexity, i.e. time required to solve them with a computer program. The problems are:

- the **Subset Sum** problem, a fundamental question in combinatorial optimization and a building block of many scheduling and resource allocation algorithms;
- the **Optimal Binary Search Tree** problem, asking to construct a dictionary data structure allowing fast searches;
- the **Longest Common (Increasing) Subsequence** problem, a simple method of sequence comparison used to search for similarities in biological sequences or plagiarism among students homework;
- the k-**Mismatch Pattern Matching** problem, occurring when a specified pattern needs to be located in a large body of text.

For each of these problems the basic question is essentially the same: is the current state-of-the-art algorithm optimal? This means that we will try to either provide a faster algorithm or rigorously prove that it is unlikely to exist.