Abstract

We study the design of efficient algorithms for combinatorial pattern matching. More concretely, we study algorithms for tree matching, string matching, and string matching in compressed texts.

Tree Matching Survey We begin with a survey on tree matching problems for labeled trees based on deleting, inserting, and relabeling nodes. We review the known results for the tree edit distance problem, the tree alignment distance problem, and the tree inclusion problem. The survey covers both ordered and unordered trees. For each of the problems we present one or more of the central algorithms for each of the problems in detail.

Tree Inclusion Given rooted, ordered, and labeled trees P and T the *tree inclusion problem* is to determine if P can be obtained from T by deleting nodes in T. We show that the tree inclusion problem can be solved in $O(n_T)$ space with the following running times:

$$\min \begin{cases} O(l_P n_T), \\ O(n_P l_T \log\log n_T + n_T), \\ O(\frac{n_P n_T}{\log n_T} + n_T \log n_T). \end{cases}$$

Here n_S and l_S denotes the number of nodes and leaves in tree $S \in \{P, T\}$, respectively, and we assume that $n_P \leq n_T$. Our results matches or improves the previous time complexities while using only $O(n_T)$ space. All previous algorithms required $\Omega(n_P n_T)$ space in worst-case.

Tree Path Subsequence Given rooted and labeled trees P and T the *tree path subsequence problem* is to report which paths in P are subsequences of which paths in T. Here a path begins at the root and ends at a leaf. We show that the tree path subsequence problem can be solved in $O(n_T)$ space with the following running times:

$$\min \begin{cases} O(l_P n_T + n_P), \\ O(n_P l_T + n_T), \\ O(\frac{n_P n_T}{\log n_T} + n_T + n_P \log n_P). \end{cases}$$

As our results for the tree inclusion problem this matches or improves the previous time complexities while using only $O(n_T)$ space. All previous algorithms required $\Omega(n_P n_T)$ space in worst-case.

Regular Expression Matching Using the Four Russian Technique Given a regular expression R and a string Q the *regular expression matching problem* is to determine if Q matches any of the strings specified by R. We give an algorithm for regular expression matching using $O(nm/\log n + n + m \log m)$ and $O(n)$ space, where m and n are the lengths of R and Q, respectively. This matches the running time of the fastest known algorithm for the problem while improving the space from $O(nm/\log n)$ to $O(n)$. Our algorithm is based on the Four Russian Technique. We extend our ideas to improve the results for the *approximate regular expression matching problem*, the *string edit distance problem*, and the *subsequence indexing problem*.

Regular Expression Matching Using Word-Level Parallelism We revisit the regular expression matching problem and develop new algorithms based on word-level parallel techniques. On a RAM with a standard instruction set and word length $w \geq \log n$, we show that the problem can be solved in $O(m)$ space with the following running times:

$$\begin{cases} O(n\frac{m \log w}{w} + m \log w) & \text{if } m > w \\ O(n \log m + m \log m) & \text{if } \sqrt{w} < m \leq w \\ O(\min(n + m^2, n \log m + m \log m)) & \text{if } m \leq \sqrt{w}. \end{cases}$$

This improves the best known time bound among algorithms using $O(m)$ space. Whenever $w \geq \log^2 n$ it improves all known time bounds regardless of how much space is used.

Approximate String Matching and Regular Expression Matching on Compressed Texts Given strings P and Q and an *error threshold* k, the *approximate string matching problem* is to find all ending positions of substrings in Q whose *unit-cost string edit distance* to P is at most k. The unit-cost string edit distance is the minimum number of insertions, deletions, and substitutions needed to convert one string to the other. We study the approximate string matching problem when Q is given in compressed form using Ziv-Lempel compression schemes (more precisely, the `ZL78` or `ZLW` schemes). We present a time-space trade-off for the problem. In particular, we show that the problem can be solved in $O(nmk + occ)$ time and $O(n/mk + m + occ)$ space, where n is the length of the compressed version of Q, m is the length of P, and occ is the number of matches of P in Q. This matches the best known bound while improving the space by a factor $\Theta(m^2k^2)$. We extend our techniques to improve the results for regular expression matching on Ziv-Lempel compressed strings.