Weekplan:
Approximate String Matching

Hjalte Wedel Vildhøj

References and Reading

[1] *Efficient string matching with k mismatches*, G. M. Landau and U. Vishkin, Theoretical Computer Science, Volume 43, 1986.

[2] *String Matching and Other Products*, M. J. Fischer and M. S. Paterson, 1974.

[3] *Generalized string matching*, K. Abrahamson, SIAM Journal on Computing, Volume 16 Issue 6, 1987.

[4] *Faster Algorithms for String Matching with k Mismatches*, Amihood Amir, Moshe Lewenstein and Ely Porat, JACM 2000.

We recommend reading [1] and [4] in detail.

Exercises

1 A faster algorithm for k-mismatches We saw an $O(n\sqrt{k}\log m)$-time algorithm for the k-mismatch problem. Improve the time complexity to $O(n\sqrt{k\log m})$. **Hint:** Consider the definition of frequent symbols.

2 Patterns with wildcards A wildcard $*$ is a special symbol that matches any other symbol from the alphabet Σ. Show how to solve the k-mismatch problem in $O(n|\Sigma|\log m)$ time when some of the symbols in the pattern P and the text T are wildcards.

3 Periodic patterns A string x is periodic with period p if $x = yy\cdots y$ for some string y of length p. Suppose the pattern P has period p. Show how to solve the k-mismatch problem in $O(np)$ time.

4 Exact matching with convolutions Give a convolution-based algorithm that finds all exact occurrences of P in T in $O(n\log m)$ time. **Hint:** Consider the sum $\sum_{j=0}^{m-1}(T[i+j]-P[j])^2$

5 Approximate text indexing with one mismatch Design a data structure for a string T of length n that supports the following approximate pattern query for a string P:

> SEARCH(P): Return all positions i in T where $T[i, i+|P|-1]$ and P mismatches in at most one position.

Your data structure should use $O(n\log n)$ space and preprocessing time, and queries should be answered in $O(|P|^2 \operatorname{polylog} n + occ)$ time where occ denotes the number of occurrences of P in T with at most one mismatch. If necessary you may assume that T contains no exact matches of P. **Hint:** Use suffix trees and 2D-range reporting. Extra challenge: Improve the query complexity to $O(|P| \operatorname{polylog} n + occ)$.

6 Nearly dual strings A string is x is *dual* if $x = yy$ for some other string y. We say that x is *k-nearly dual* if x can be made dual by changing at most k symbols in x. Here changing a symbol means replacing it with another symbol (i.e., deleting or adding symbols are not allowed). Let T be a string of length n.

1. [w] Show how to find all k-nearly dual strings in T in $O(n^2k)$ time.
2. Show an $O(n)$-time algorithm that given a position i in T outputs all dual strings yy where the first copy of y contains position i. **Hint:** Consider all possible lengths of y separately.
3. Show how to find all dual strings in T in $O(n \log n + occ)$ time, where occ denotes the number of dual strings in T. **Hint:** Use your solution from (2) to make a divide-and-conquer algorithm.
4. Generalize your algorithm from (3) to find all k-nearly dual strings in T in $O(kn \log n + occ)$ time, where occ denotes the number of k-nearly dual strings in T.