Abelian Pattern Matching

The presentation is about finding the abelian patterns in strings. Problem of *Abelian Pattern Matching* differs from *Classical Pattern Matching* in the sense that in case of classical pattern matching we seek for exact occurrence of a pattern substring in the given input string and order of characters in the pattern substring is kept preserved while looking for a match. In case of abelian pattern matching, however, order of characters in the pattern substring doesn't matter. Hence *'abc'* and *'bac'* would be considered matching (abelian) patterns. Hence we are not looking for an exact (ordered) occurrence of a substring, *rather we want to find any permutation of the given combination of characters that form our pattern substring.*

In the presentation I will present **two algorithms**, each based on a different strategy. General idea is that we start with a search window of pattern size and then slide this window from left to right along the input stream to look for the pattern. *The two algorithms differ in the way the pattern is matched inside the window.* In the first algorithm the characters in the search window are read from left to right *(using a prefix based approach)*, whereas in the second algorithm the characters in the search window are read from right to left*(using a suffix based approach)*, thus giving a possibility of skipping some characters from reading (but also with a danger of a need to read same characters many times).

Worst case time complexity of first algorithm is $O(nX)$[1], whereas the worst case time complexity of second algorithm is $O(nmX)$[2] but on average it's expected to perform better than first algorithm. In future we intend to explore the time efficiency of second algorithm on the average and to see that which of the two algorithms perform better in practice.

[1]Here n is the length of the string and X is a variable that depends on the datastructure used for book-keeping

[2]Here m is the size of pattern substring, and n and X are same as defined in 1