The Messenger Problem

Hrishikesh V
Department of Computer Science
PES University
Bangalore
PES1201700276

***Abstract*—This problem requires that the user calculate the minimum cost of sending a text message of length l, given the set of conditions. The cost of sending a character, which is not a substring of the set of characters already sent is c1 and the cost of sending a set of characters which is a substring of the characters already sent is c2.**

***Index Terms*—Dynamic Programming, Substring, String Matching, Longest prefix, cost minimization**

I. Introduction

String processing problems usually have a worst case time complexity of O(mn). There are many methods that can be used to improve the average time complexity of string matching, thereby making them faster. In this paper, I shall explore a method to do so.

II. Approach

The problem requires the user to calculate the lowest cost of constructing a message of pre-specified length. The cost of adding a character (c1) and a substring (c2) are given beforehand. We shall first start the construction of the string one character at a time. The first character takes a cost of c1 because it is obviously not a substring.

For the subsequent characters, the string is divided into two halves. The first half represents the string that has been constructed and the second half contains the characters whose cost is to be determined.

III. Dynamic Programming with Memoize table

Calculating the cost of the secondhalf of the string is a redundant task. Repeated calculation of this cost is minimized with the help of a memoize table. For a string of length n, Table[i] contains the cost of the string starting at index i.

IV. Recurrence Relation

If the cost of the string starting at index i is not available in the memoize table, it is explicitly calculated.

The base case of this recursion is when secondhalf is 0, i.e when no character has been added to the message. In this case, only one character (the first character) is added to the message, costing c1 units.

For every second half i (0¡= i ¡ n), the cost of constructing the message is the minimum of the cost of appending a single character and the cost of appending the longest prefix starting at index i which is the substring of the text between indices 0 and i-1.

There are two ways to calculate the longest prefix that is a substring of the set of characters whose cost is already calculated. The first method involves constructing a suffix array for the first half of the string and checking if the secondhalf is a substring.

The second method uses Knuth-Morris-Pratt algorithm to calculate the longest prefix.

The recurrence relation is

$$cost[sh] = min(c1{+}cost[sh{+}1], c2{+}cost[sh{+}prefix]) \quad (1)$$

V. Conclusion

A combination of Dynamic Programming and an efficient string matching algorithm can be used to reduce the time complexity of calculating the cost of constructing a text message given c1 and c2.