RESEARCH OF PATTERN MATCHING ALGORITHM BASED ON KMP AND BMHS2

Ever used a text editor and you had to find a word? I'm sure this is a common scenario and it has happened to you many times. Of course, your first instinct isn't to scan the entire article, you make the computer do that for you by using the find function. But have you ever wondered how that works? What makes the find function so fast? How does it find the matches? Here's where "string matching" or "pattern matching" algorithms are used.
What do we mean by string matching? We have two strings, a-the pattern and b - the string in which we want to find the pattern, and we need to find whether a matches some or the complete string b. Essentially, we need to see if the string a is a part of string b.

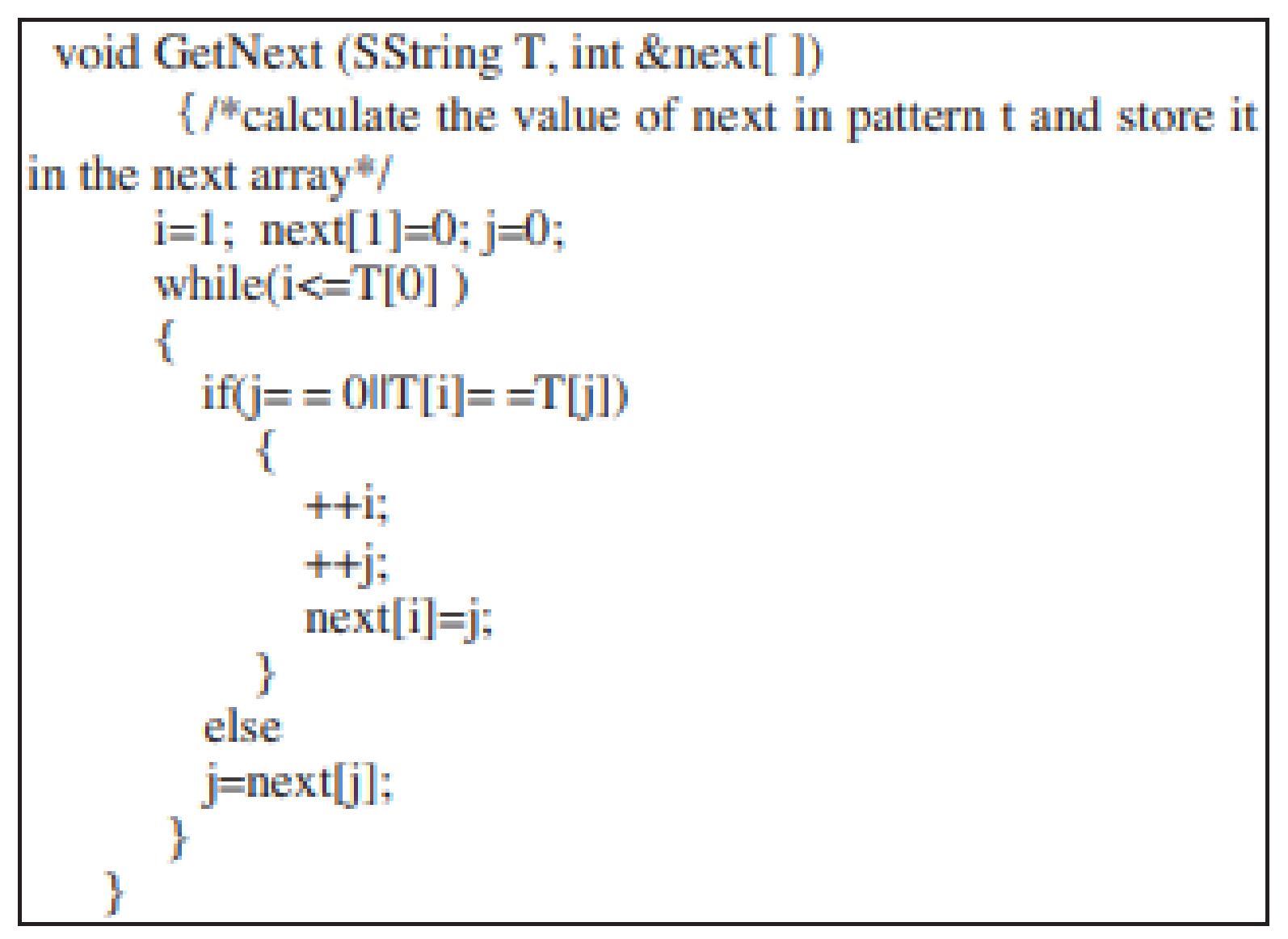
INTRODUCTION

- Pattern matching algorithm is widely used in search engine and text matching and has become a research hotspot. The room for performance improvement of a single pattern matching algorithm is limited.
- The two algorithms have different matching sequences in each match, but they need to move the text string matching window to the right in case of mismatch
- The current research focuses on combining multiple pattern matching algorithms and combining the advantages of each algorithm to further improve the time performance of pattern matching.
- The improvement of performance of pattern matching algorithm mainly lies in reducing the number of character comparisons and increasing the distance of the matching window moving to the right when matching.

IDEA OF THE ALGORITHM

- . This algorithm mainly eliminates the backtracking problem of the main string pointer in the matching process of BF algorithm, and makes use of the partial matching results to slide the pattern string to the right as far as possible and then continue to compare, so as to improve the efficiency of the algorithm to a certain extent. In the worst matching case, the time complexity of KMP algorithm is O(m+n), where the length of the pattern string is m and the length of the main string is n.
- The basic idea: whenever a match fails, the i pointer does not have to backtrack, but to use the "partial match" result which has been obtained to see if it is necessary to adjust the value of I, and then "slide" the pattern to the right several positions before continuing the comparison.

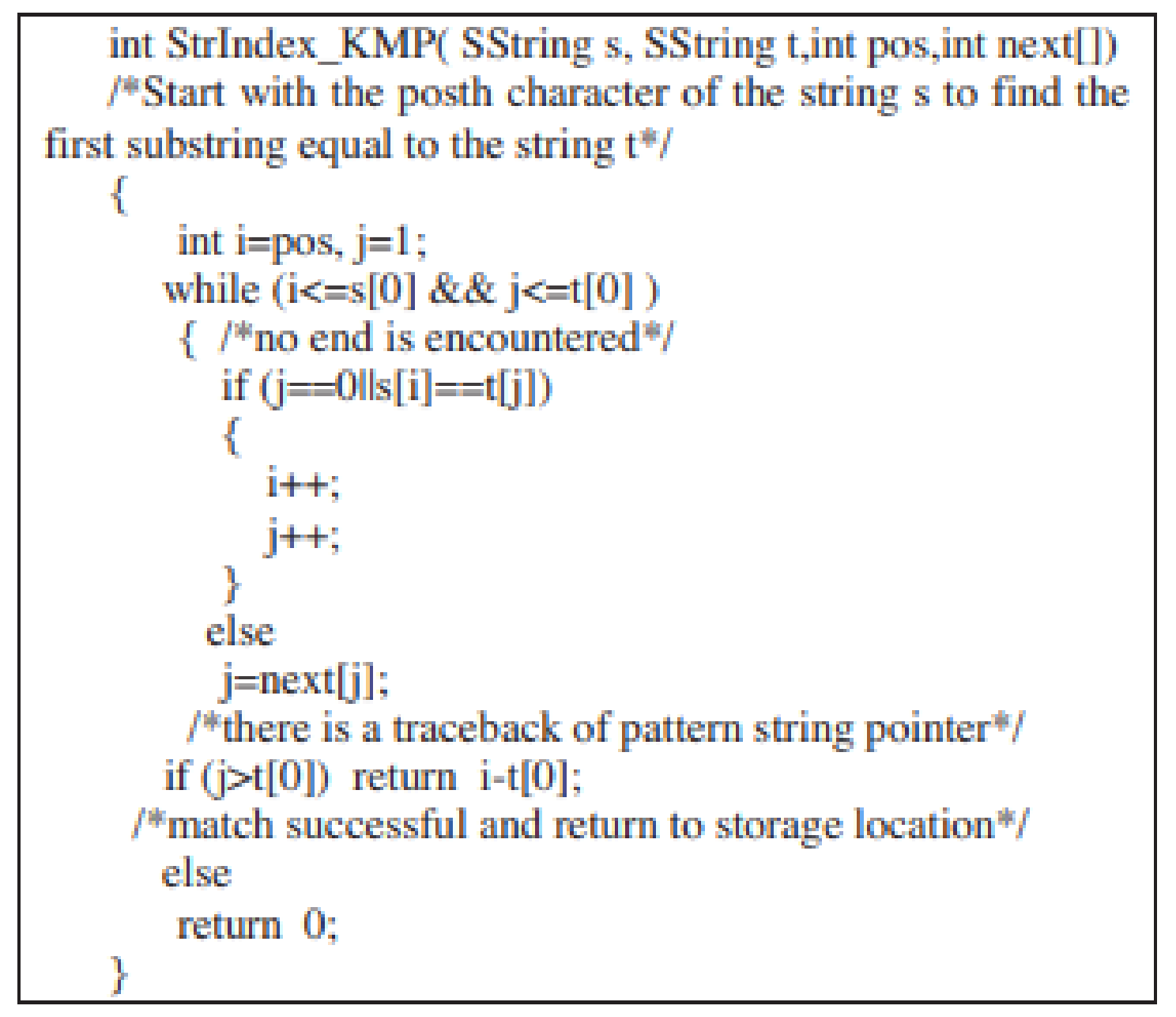
EXPERIMENTS AND EVALUATION



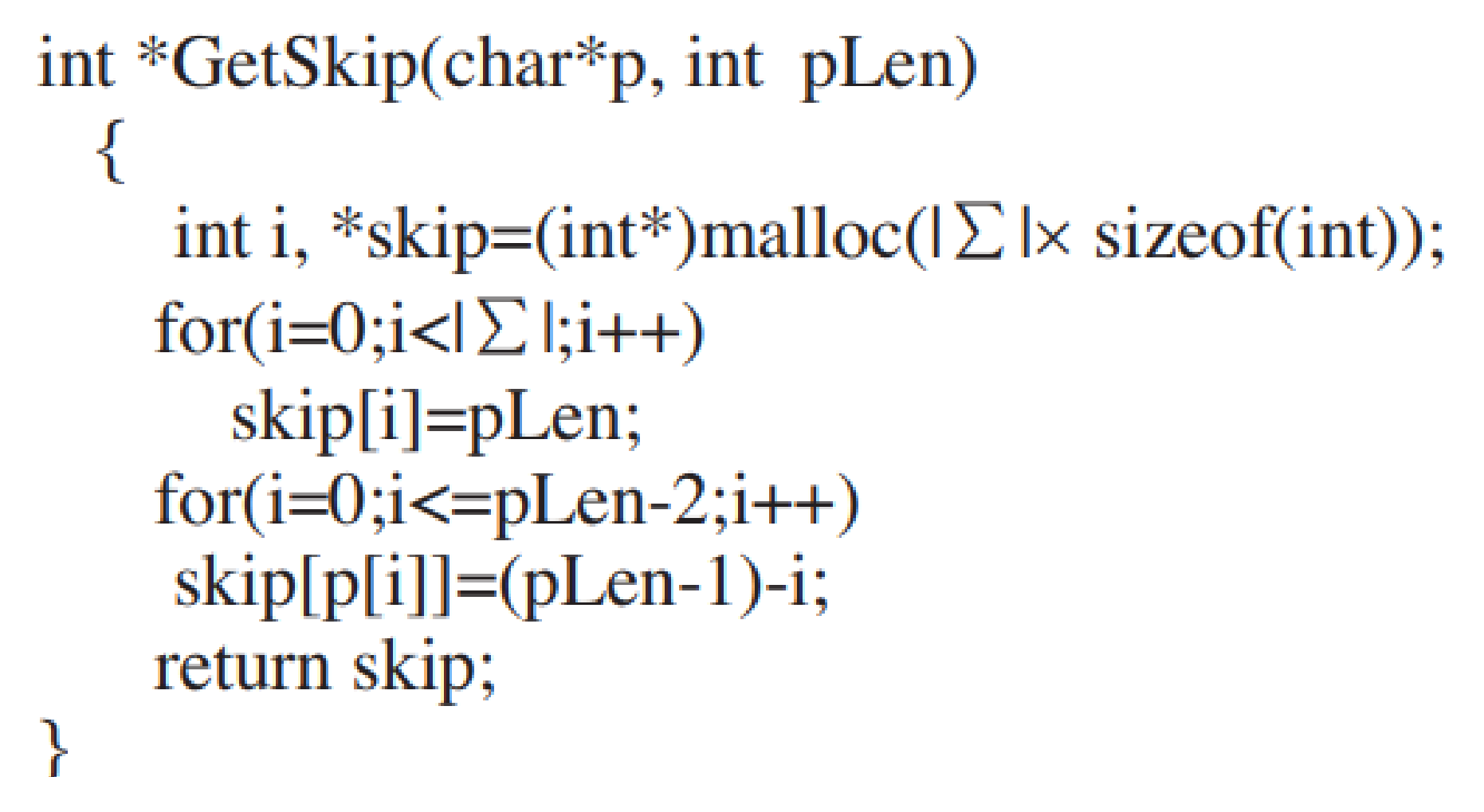
```
  void GetNext (SString T, int &next[ ])
        { /*calculate the value of next in pattern t and store it
in the next array*/
        i=1;  next[1]=0; j=0;
        while(i<=T[0] )
        {
           if(j= = 0||T[i]= =T[j])
              {
                 ++i;
                 ++j;
                 next[i]=j;
              }
           else
           j=next[j];
        }
     }
```

The next array of pattern string is used to pattern matching function of KMP which locates pattern string P in main string T.

The KMP algorithm before improvement looked like this

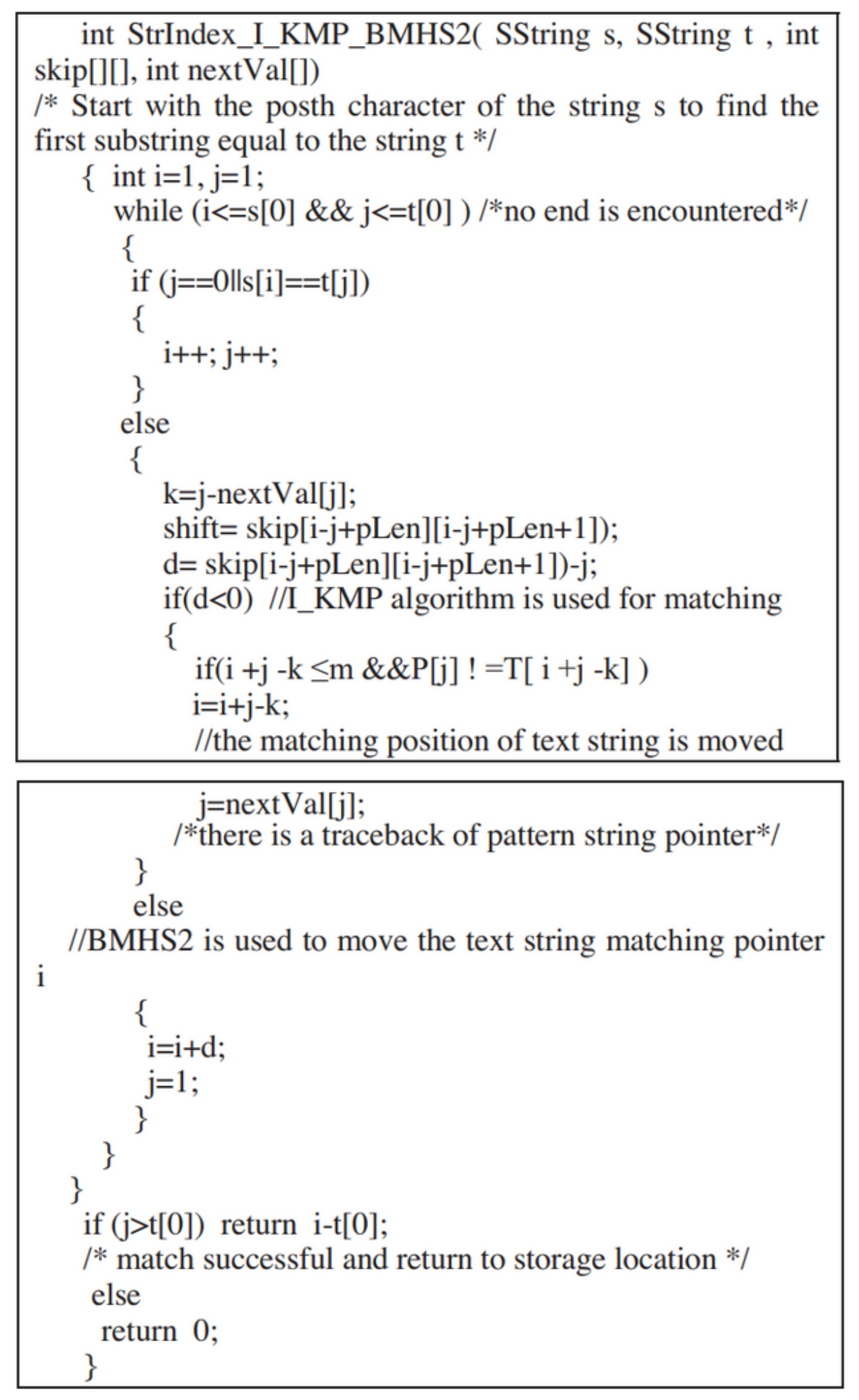


```
    int StrIndex_KMP( SString s, SString t,int pos,int next[])
    /*Start with the posth character of the string s to find the
first substring equal to the string t*/
    {
        int i=pos, j=1;
       while (i<=s[0] && j<=t[0] )
        {  /*no end is encountered*/
           if (j==0||s[i]==t[j])
           {
              i++;
              j++;
           }
          else
           j=next[j];
        /*there is a traceback of pattern string pointer*/
       if (j>t[0])  return  i-t[0];
     /*match successful and return to storage location*/
       else
        return  0;
    }
```



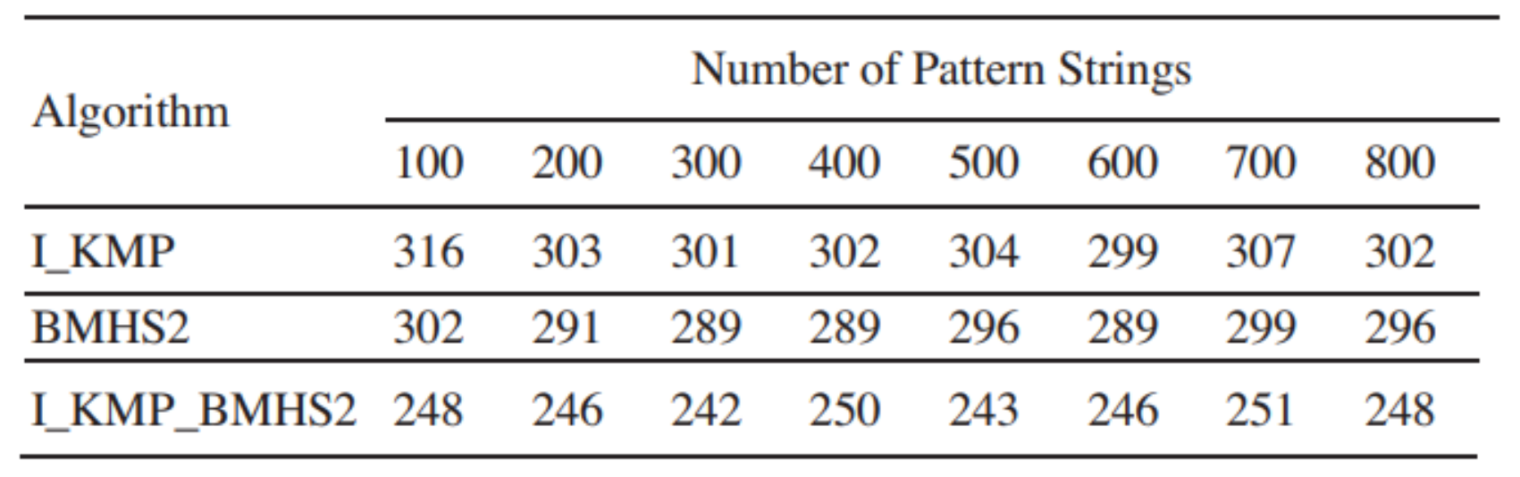
```
int *GetSkip(char*p, int  pLen)
   {
     int i, *skip=(int*)malloc(|Σ|× sizeof(int));
    for(i=0;i<|Σ|;i++)
        skip[i]=pLen;
    for(i=0;i<=pLen-2;i++)
     skip[p[i]]=(pLen-1)-i;
    return skip;
}
```

In the algorithm, |Σ| is the number of character sets in the string. for example, in a string composed of ASCII codes, |Σ| is 256. Algorithm analysis: theoretically, the complexity of BHM algorithm in the worst case is O(mn), but in general, it has better performance than BM. In practice, the time complexity of the algorithm is O(m+n).



```
    int StrIndex_I_KMP_BMHS2( SString s, SString t , int
skip[][], int nextVal[])
/* Start with the posth character of the string s to find the
first substring equal to the string t */
    {  int i=1, j=1;
       while (i<=s[0] && j<=t[0] ) /*no end is encountered*/
        {
         if (j==0||s[i]==t[j])
         {
            i++; j++;
         }
        else
         {
            k=j-nextVal[j];
            shift= skip[i-j+pLen][i-j+pLen+1]);
            d= skip[i-j+pLen][i-j+pLen+1])-j;
            if(d<0)  //I_KMP algorithm is used for matching
            {
               if(i +j -k ≤m &&P[j] ! =T[ i +j -k] )
               i=i+j-k;
               //the matching position of text string is moved
               j=nextVal[j];
             /*there is a traceback of pattern string pointer*/
         }
         else
   //BMHS2 is used to move the text string matching pointer
i
         {
          i=i+d;
          j=1;
         }
      }
   }
    if (j>t[0])  return  i-t[0];
    /* match successful and return to storage location */
     else
      return  0;
    }
```

The improved KMP algorithm is adopted for each match, and every match is matched from left to right in the matching window. In order to improve the distance to the right during each mismatch, the improved KMP, the larger distance move to the right of the improved KMP algorithm and BMHS2 algorithm are adopted, so as to further improve the matching efficiency under the same environment.



Algorithm	Number of Pattern Strings							
	100	200	300	400	500	600	700	800
I_KMP	316	303	301	302	304	299	307	302
BMHS2	302	291	289	289	296	289	299	296
I_KMP_BMHS2	248	246	242	250	243	246	251	248

AUTHORS

Yansen Zhou , Ruixuan Pang

AFFILIATIONS

Department of Information Science and Technology University of International Relations Beijing, China

RESEARCH PAPER INFO

2019 IEEE 5th International Conference on Computer and Communications

REFFERENCES

http://arxiv.org/abs/2201.06718v2

CONCLUSION

- The performance of the improved KMP algorithm combined with BMHS2 algorithm can be improved, mainly lying in the matching window of text string is moved to the right by using the larger jump distance between KMP and BMHS2 in each mismatch.
- Compared with the multi-pattern matching algorithm, An improved algorithm combining KMP with BMHS2 algorithm still has the problem of low efficiency.

APPLICATIONS OF KMP

1. Text editors and word processors: The KMP-BMHS2 algorithm can be used in text editors and word processors to find and replace words and phrases in a document.
2. Information retrieval systems: The algorithm can be used in information retrieval systems to search for keywords in large databases of text.
3. DNA sequencing: The KMP-BMHS2 algorithm can be used in DNA sequencing to search for patterns in genetic sequences.

MADE BY

Aarya Tiwari

ROLL NUMBER

16010421119

BATCH

B2

SOMAIYA
VIDYAVIHAR UNIVERSITY
K J Somaiya College of Engineering