International Journal of Computational Intelligence and Informatics, Vol. 5: No. 4, March 2016

# An Efficient Technique for Sequential Pattern Searching

**S. Deepa**
*Department of Computer science,*
*Vellalar College for Women,*
*Erode.*
*deepasamiappan@gmail.com*

***Abstract*- This The task of Sequential pattern mining aims to extract the sequences from large databases, which in turn can be interpreted as domain knowledge for several purposes. Sequential pattern mining is used in several domains such as studying the customer behaviors, mining several web logs distributed on multiple servers, protein and gene sequence analysis and in computational biology to analyze the amino acid mutation patterns. The searching process in sequence databases plays an important role in many application domains, mainly for information retrieval and data mining. Hence there exist a lot of interests among the researchers towards the development of new concepts related to the sequence search. This research work proposes two new search techniques namely Sequence Search by Partitioning (SSP) and Sequence Search by Indexing (SSI) for performing sequence search efficiently. Performance of the proposed techniques are compared based on the key features such as total execution time, search time and memory utilization.**

Keywords- Sequence, Sequence database, PrefixSpan, SSI, SSP.

## I. INTRODUCTION

Data mining is the process of analyzing data from different perspectives and summarizing it into useful information. It may also be defined as the process of identifying the useful patterns of data from large databases. From these identified patterns, new and important information can be obtained that will lead to the discovery of new meanings which can then be translated into enhancements in many current fields. An important concept of data mining is sequence mining. Sequential pattern mining aims to extract frequently occurred sequences to describe the data or predict future data or mining periodical patterns [1].

Sequential Pattern Mining finds the interesting sequential patterns among the large database. A sequence $\alpha = <A_1 \bullet \bullet \bullet A_n>$ is an ordered list of itemsets. A sequence $\alpha = <A_1 \ldots A_n>$ is called a subsequence of another sequence $\beta = <B_1 \ldots B_m>$ $(n \leq m)$, and $\beta$ a super-sequence of $\alpha$, if there exist integers $1 \leq i_1 < \ldots < i_n \leq m$ such that $X_1\ Y_{i1}, \ldots, X_n\ Y_{in}$. A sequential database is a set of 2-tuples (sid, $\beta$), where sid is a sequence-id and $\beta$ is the sequence. Given a positive integer minimum_sup as the support threshold, a sequence $\lambda$ is considered to be a sequential pattern in sequence database Sdb if sup $(\lambda) \geq$ minimum_sup. The sequential pattern mining problem is used to find the complete set of sequential patterns with respect to a given sequence database and a support threshold minimum support [2].

For performing search operations in data structures, there exist several different algorithms. Different algorithms make different demands on computer memory and running time. In an industrial setting where the rate of commercial software and staff time become significant, the availability and price of off-the-shelf routines that employs the algorithms must be considered. And if no commercial package is available, the programming time required to apply the algorithm must also be considered. It is therefore useful to know things like the running time and memory requirements of algorithms so that an informed decision can be made about which algorithm is best for a particular application.

Searching a list for a particular item is a general task. People deal with different types of data in web applications such as text searching, image searching, audio searching and video searching. Every search engine uses variety of search algorithms for handling different types of data. The search algorithm increases the pattern matching process. Scientific disciplines are confronted with an increasing amount of data and few tools or techniques for extracting meaningful information from it. The questions of what to extract and how to extract it make dealing with large, multi-dimensional datasets become one of the most important and exciting areas of scientific research today. The basic operations carried out on a data structure are search, insertion, and deletion. Search efficiency is considered the most important criterion for selecting data structures because search is frequently carried out.

There are many search algorithms in data structures that are used to search for a particular data item in a large amount of data. But still no specific algorithms have been developed to search for a particular sequence of data items in large volumes of data. For example, in order to perform search of a particular item in a retail database, several algorithms such as linear search, binary search, fibonacci search etc are already available. These algorithms are used for finding whether the item (single), for example, bread is present in a particular transaction or not. But it is difficult to find more number of items, namely bread, butter and jam had been bought together in a particular transaction. In retail dataset, each individual item is stored with a unique product id. Hence the concept of integer sequence search can be used to find the sequence relationships among the items that were bought together. The sequential search algorithms perform search from the first data points to the end of the data sequence $s_1$, $s_2$ : : : $s_n$.

The rest of the paper is organized as follows: Section II describes the related work. Section III describes the problem objective and the proposed techniques are explained in Section IV. Section V deals with the performance Evaluation and the conclusion for the proposed techniques is given in Section VI.

## II. Related Work

First, Searching an item or data from a large data set is a challenging task. Many numbers of searching algorithms are used for performing searching process. Some of the popular searching algorithms are Binary Search, Linear Search, Depth First Search, Binary Search Tree, Particle Swarm Optimization, Genetic algorithm, etc. The simplest method of searching of an element is linear search. It is the simplest searching method which checks for an item one by one linearly [3, 4, 5, and 7]. The Randomized Searching Algorithm selects the positions randomly from the sorted input array and compare elements of those positions with the search key until the match is found or else the array is finished [10, 15]

A number of algorithms have been proposed for performing searching operations for a sequence of strings in a large database. Boyer and Moore [6] proposed the Boyer Moore algorithm that performs the search from right to left in the pattern. The algorithm places the pattern over the leftmost characters in the text and attempts to match it from right to left. If there is no mismatch, then the pattern has been found. Or else the algorithm performs a shift, which is an amount by which the pattern is moved to the right before a new matching attempt is undertaken. Knuth and Pratt [8] proposed the KMP algorithm, each time when a mismatch is found, the false start consists of characters that were already examined. This avoids the repetitive comparisons with the known characters. The pattern is preprocessed to obtain a table that gives the next position in the pattern to be processed after a mismatch.

Sunday [13] proposed a new algorithm known as the Quick-Search Algorithm (QS) that uses the Quick-search bad-character (qsBc) shift table, generated during the preprocessing stage. The shift value for a character in the qsBc table is defined as its corresponding position in the pattern from right to left order. If the character is not present in the pattern, then the shift value is equal to m+1. After an attempt, when the window is positioned on y[j.. j+m-1], the length of the shift is at least equal to one. Therefore, the character y[j+m] is necessarily involved in the next attempt and is used for the bad-character shift of the current attempt. During each attempt of the searching phase, the comparisons between the pattern and the text characters can be performed in any order.

Sheik et al., [12] proposed a new algorithm where the order of comparisons is carried out by comparing the last character of the window and the pattern and after a match, the algorithm further compares the first character of the window and the pattern. The remaining characters are compared from right to left until a complete match or a mismatch occurs. After each attempt, the skip of the window is gained by the Quick-Search bad character shift value for the character that is placed next to the window. Raita [11] designed an algorithm in which the rightmost character of the pattern and the window are compared and on a match, the leftmost character of the pattern and the window are then compared. If the pattern and the window match, it compares the middle character of both the pattern and the window. Then if match occurs, the characters from the second to the penultimate (n-1) position of the pattern and the window are compared. The skip for the window is computed by applying the bmBc shift of the rightmost character in the window.

## III. Problem Objective And Methodology

The sequential pattern mining aims to generate the frequent sequences in the given transaction database based on the user defined minimum support. The sequence generation algorithms are used to generate the sequences and these generated sequences are stored in a sequence database. Many applications require to see whether a given search sequence is found in the sequence database or not. And some applications have the need to count the occurrence of a given search sequence in the sequence database. The main aim of the proposed search techniques are to perform the search operation in the sequence database and as well to count the occurrences of the search sequence.

In this paper, two new techniques are proposed for searching the sequence database. Initially, the sequences are generated from the data set by using several sequence generation algorithms. Important sequence generation

algorithms are Generalized Sequential Patterns (GSP), Sequential Pattern Discovery using Equivalent classes (SPADE), Prefix-Projected Sequential Pattern Growth (PrefixSpan), Sequential PAttern Mining (SPAM), Recursive Prefix Suffix Pattern detection (RPSP), etc. In this research work, three sequence generation algorithms namely GSP, SPADE, PrefixSpan algorithms are used for generating sequences. By measuring the efficiency of these algorithms, the PrefixSpan algorithm performance is better than GSP and SPADE [14]. The sequences generated by the Prefixspan algorithm are stored in a sequence database. In this research work, two new search techniques are proposed to perform the search process and to find the number of occurrences of a particular sequence in a sequence database. The techniques proposed for performing search operations are Sequence Search by Partitioning (SSP) and Sequence Search by Indexing (SSI).

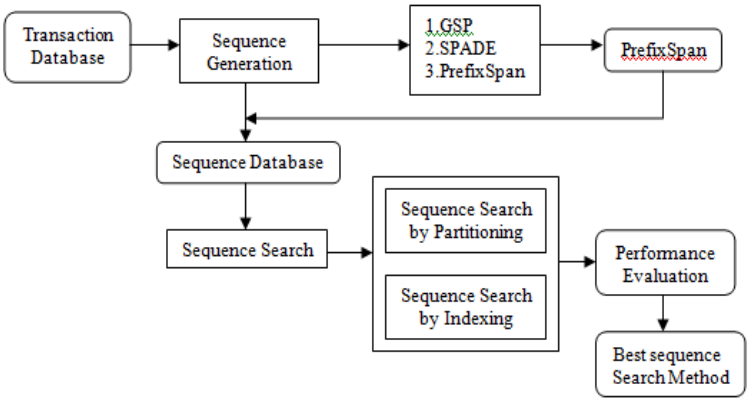


Figure 1. System Architecture for Sequence Search

### *A. Dataset*

The dataset used in this paper is taken from Frequent ItemSet Mining Repository. http://fimi.ua.ac.be/data/retail.dat. Retail dataset is used in this research work. It is a real time dataset collected from a Belgian Retail Supermarket store. The dataset consists of 88,163 transactions and 16,440 different products that are sold in various transactions carried over in a certain period of time. The transactions consist of unique ids that are given for each product in the store. The TABLE I show the sample for the dataset that was taken for the sequence generation. It consists of Transaction Id's and sequence of product Id's.

TABLE I. SAMPLE DATASET

| TRANSACTION | SEQUENCE OF PRODUCTS |
|---|---|
| T1 | 10 21 32 43 54 65 76 |
| T2 | 21 32 43 65 |
| T3 | 21 32 76 89 90 |
| T4 | 21 32 35 67 78 |
| T5 | 23 34 43 67 89 90 |

### *B. The Prefixspan Algorithm*

The major idea of PrefixSpan algorithm [9] is that any frequent subsequences can always be found by growing frequent prefixes. It divides the database into smaller projected databases and solves them recursively. It examines only the prefix subsequences and projects only their corresponding postfix subsequences into projected databases. In each of the projected databases, sequential patterns are grown by exploring only local frequent patterns. Since no candidate sequence needs to be generated, the database need not be scanned multiple times. Since Prefix-projection substantially reduces the size of projected databases, it leads to efficient mining of sequential patterns. The PrefixSpan algorithm is applied to generate the sequential patterns from the retail database.

**Algorithm 1:** Algorithm PrefixSpan

1. PrefixSpan(α, i, S| α)
2. Begin
3. Scan S|α once, find the set of frequent items b such that
   a. b can be assembled to the last element of α to form a sequential pattern; or
   b. <b> can be appended to α to form a sequential pattern.
4. For each frequent item b, appended it to α to form a sequential pattern α' and the output α';
5. For each α', construct α'-projected database S| α' and call PrefixSpan (α', i+1,S| α').
6. End

TABLE II. SAMPLE SEQUENCES PRODUCED FROM RETAIL DATASET BY PREFIXSPAN ALGORITHM

| S.NO | SEQUENCES PRODUCED |
|---|---|
| 1 | 21 |
| 2 | 32 |
| 3 | 43 |
| 4 | 21 32 |
| 5 | 21 43 |
| 6 | 32 43 |
| 7 | 21 32 43 |

The above table shows the sample of sequence Id's for the products that were generated by applying Prefixspan algorithm on the retail dataset.

## IV. PROPOSED SEARCH TECHNIQUES

### *A. Sequence search by partitioning*

This method consists of two steps namely partitioning and searching. In partitioning, the sequence database Sdb is partitioned into different tables as $T_1$, $T_2$ ….$T_n$ based on the length of the sequences. For example, the sequence length is one, two, three, etc. Sequences with one item are stored in table $T_1$ and the sequences with two items are stored in another table $T_2$ and so on. In order to perform search process, the input search sequence Ss is required. Next, the input search sequence length (L) is calculated. Based on the search sequence length (L), the search starts from the table TL and continues up to the table Tn. The search Sequences Ss are retrieved and the count value is calculated.

**The SSP algorithm:**

Input: Sequence database Sdb, Search sequence Ss.

Output: (i) Search successful or unsuccessful (ii) Count.

Method:

Consider the input sequence database, Sdb=<$s_1$,$s_2$…..$s_n$>, $s_j \in I$, where j=1 to n be the set of sequences and I=<$i_1$,$i_2$….$i_m$> where i=1 to m be the set of items.

2. Initialize Count=0 and L=0.
3. Partitioning:
   - 3.1 Partition Sdb into T tables based on the sequence length.
   - 3.2 Consider the search sequence Ss and calculate its length (L).
4. Searching:
   - 4.1 Start search from TL to Tn.
   - 4.2 If (Ss ∈ TL) then display the search sequence Ss;
     Increment the value of count and L;
   - 4.3 If (L>n) Then display the value of count;
     Else Goto step 4.2;
   - 4.4. Else Increment the value of L and Goto step 4.2;

End

*B. Sequence Search by Indexing*

This method has three steps namely indexing, searching and partitioning. In Indexing, each individual item $j_1,j_2....j_m$ in the sequence database Sdb is given an index value and stored in a separate index table J. To perform searching, an input search sequence Ss is given. The input search sequence Ss is split up into individual items as $i_1$, $i_2 \ldots i_n$ and each individual item in the search sequence Ss is taken and checked in the index table J to see whether the item has an index value in the index table J. If at least one item of the search sequence Ss has no index value, then it can be declared that the sequence will not be present in the sequence database Sdb. If the search sequence Ss is present in the index table J, the sequence database Sdb is partitioned into tables $T_1,T_2 \ldots T_n$ based on the sequence length. Then the search proceeds from the table TL and continues until the table Tn. The sequences related to the given search sequence Ss are retrieved and the count value is also calculated.

This method will be effective when the first item i1 in the search sequence Ss itself is not present in the index table J. But if the search sequence Ss is too large and if the last item in the search sequence is not present in the index table J, then it will be time consuming.

**The SSI algorithm:**
Input: Sequence database Sdb, Search sequence Ss
Output: i) Search successful or unsuccessful ii) Count.
Method:
1. Consider the input sequence database, Sdb=<$s_1,s_2 \ldots .s_n$>, $s_i \in I$, where i=1 to n be the set of sequences and I=<$j_1,j_2 \ldots j_m$> where j=1 to m be the set of items
2. Initialize Count=0 and L=0.
3. Indexing:
    3.1 Create an Index table J for each item in Sdb.
    3.2 Consider the input search sequence Ss and split Ss into individual items as $Ss_i$ to $Ss_n$.
    3.3 If ($Ss_i \in J$) Then increment the value of i.
    3.4. If (i<=n) Then Goto step 3.3;
    3.5. Else Goto step 4;
4. Partitioning:
    4.1 Partition Sdb into T tables based on the sequence length.
    4.2Calculate length L of search sequence Ss.
5. Searching:
    5.1 Start search from TL to Tn.
    5.2 If (Ss $\in$ TL) then display sequence;
        Increment the value of count and L
    5.3 If (L>n) Then display count;
Else Goto step 5.2.
    5.4 Else Increment the value of L and Goto step 5.2;
6. Else
    Process terminated;

## V. PERFORMANCE EVALUATION

To test the proposed methods, a series of performance studies were conducted. A performance test is focused on the memory used and in addition, the execution time of the two methods is also analyzed. The evaluation was performed on PC Intel Pentium processor, 2GB RAM, OS Windows 7 Ultimate 32-bit. The subsequent tests compare performance of two different search techniques on retail dataset. The performance of these two search techniques are analyzed under various criteria such as various dataset sizes and various search sequences length. The different sizes of dataset used in this work are 250, 1000 and 2000. The different search sequence lengths are 2, 6 and 10. The results for the search sequence [32, 43] and their occurrence count is shown in Table III.

TABLE III. SAMPLE OUTPUT FOR SEARCH SEQUENCE

| SEARCH SEQUENCE | OUTPUT | COUNT |
|---|---|---|
| 32, 43 | [32, 43] | 2 |

TABLE IV. TOTAL EXECUTION TIME OF SSP AND SSI TECHNIQUES FOR DATASETS OF VARIOUS SIZES

| ALGORITHM | DATASET SIZE | TOTAL EXECUTION TIME(IN MS) |
|---|---|---|
| SSP | 250 | 32.0 |
| | 1000 | 35.2 |
| | 2000 | 35.9 |
| SSI | 250 | 56.8 |
| | 1000 | 56.3 |
| | 2000 | 57.6 |

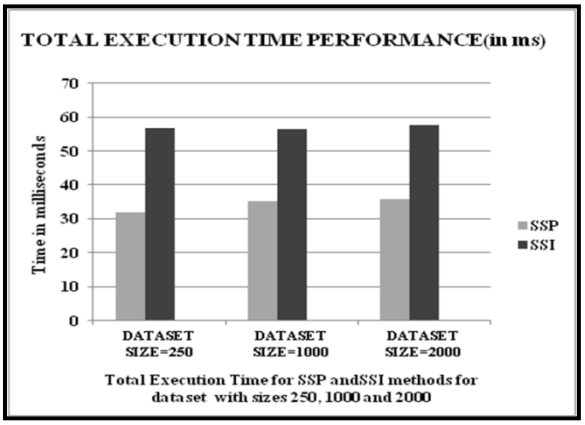


Figure 2. Total Execution Time of SSP and SSI techniques for datasets of various sizes

The above graph shows the total execution time taken for searching the sequences by SSP and SSI techniques. The total execution time includes the time taken for generating the sequences and dividing them in to tables based on their sequence length and finally search operation. From the results, we observed that the SSP technique takes minimum execution time than SSI technique.

Table V shows the search time of SSP and SSI techniques for sequences of various lengths. Search time indicates only the time involved in searching the sequence in the preprocessed tables.

TABLE V. SEARCH TIME OF SSP AND SSI TECHNIQUES FOR DATASETS OF VARIOUS SIZES

| ALGORITHM | DATASET SIZE | SEARCH TIME(IN MS) |
|---|---|---|
| SSP | 250 | 16.0 |
| | 1000 | 17.3 |
| | 2000 | 17.8 |
| SSI | 250 | 14.2 |
| | 1000 | 14.4 |
| | 2000 | 14.8 |

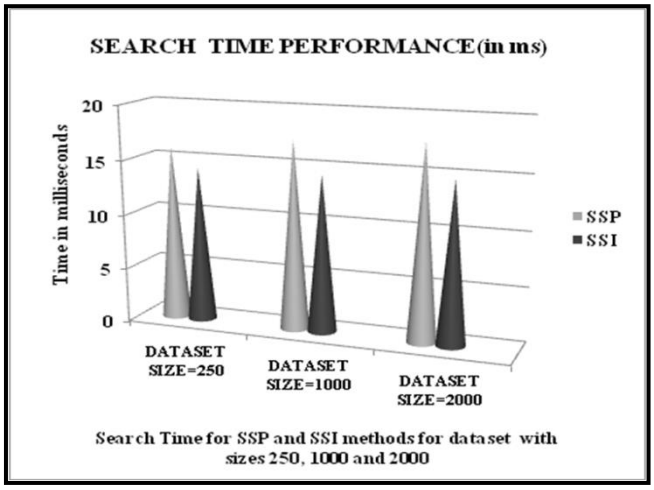


Figure 3. Search Time of SSP and SSI techniques for datasets of various sizes

The above graph shows the search time of the two search techniques. The result shows that the search time of SSI require minimum search time than SSP technique.

Table VI shows the memory space occupied by the SSP and SSI techniques for storing the generated sequences. The sequences were generated for dataset of various sizes and their corresponding memory space occupied is taken.

TABLE VI. TOTAL MEMORY SPACE OF SSP AND SSI TECHNIQUES FOR DATASETS OF VARIOUS SIZES

| ALGORITHM | DATASET SIZE | TOTAL MEMORY(IN KB) |
|---|---|---|
| SSP | 250 | 184.0 |
| | 1000 | 190.0 |
| | 2000 | 190.0 |
| SSI | 250 | 344.0 |
| | 1000 | 345.6 |
| | 2000 | 346.4 |

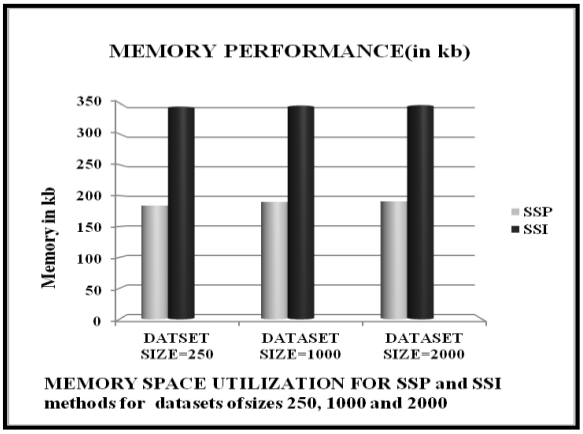


Figure 4. Total Memory Space of SSP and SSI techniques for datasets of various sizes

The above graph shows the total memory space utilized for searching the sequences by SSP and SSI techniques. From the results, we observed that the SSP technique occupies minimum memory space than SSI technique.

The tables VII, VIII and IX shows the results of search process that was taken for search sequences of various lengths.

TABLE VII. TOTAL EXECUTION TIME OF SSP AND SSI METHODS FOR SEARCH SEQUENCES OF VARIOUS LENGTHS

| ALGORITHM | SEQUENCE LENGTH | TOTAL EXECUTION TIME(IN MS) |
|---|---|---|
| SSP | 2 | 32.0 |
| | 6 | 106.8 |
| | 10 | 170.0 |
| SSI | 2 | 58.4 |
| | 6 | 170.4 |
| | 10 | 288.0 |

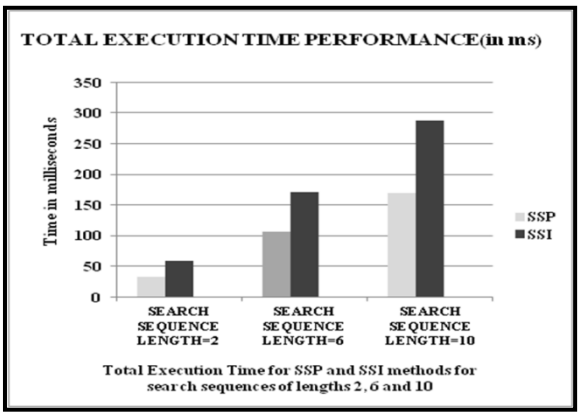


Figure 5. Total Execution Time of SSP and SSI methods for search sequences of various lengths

The above graph shows the total execution time for searching the sequences by SSP and SSI techniques. From the results, we observed that the SSP technique occupies minimum execution time than SSI technique.

TABLE VIII. SEARCH TIME OF SSP AND SSI TECHNIQUES FOR SEARCH SEQUENCES OF VARIOUS LENGTHS

| ALGORITHM | SEQUENCE LENGTH | SEARCH TIME(IN MS) |
|---|---|---|
| SSP | 2 | 16.0 |
| | 6 | 53.4 |
| | 10 | 85.0 |
| SSI | 2 | 14.6 |
| | 6 | 42.6 |
| | 10 | 72.0 |

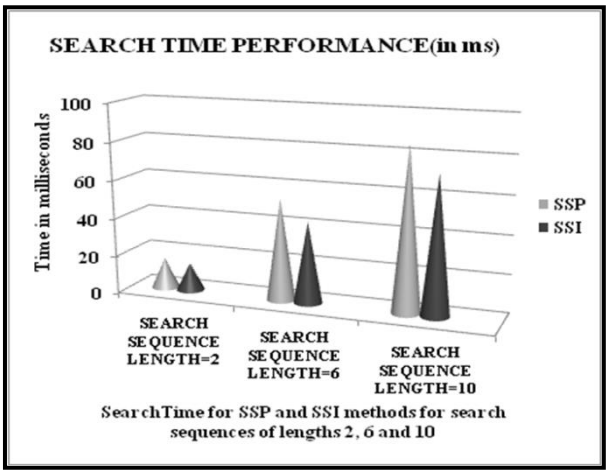


Figure 6. Search Time of SSP and SSI techniques for search sequences of various lengths

The above graph shows the search time for searching the sequences by SSP and SSI techniques. From the results, it is observed that the SSPP technique occupies minimum search time than other two techniques.

TABLE IX. TOTAL MEMORY SPACE OF SSP AND SSI TECHNIQUES FOR SEARCH SEQUENCES OF VARIOUS LENGTHS

| ALGORITHM | SEQUENCE LENGTH | SEARCH TIME(IN MS) |
|---|---|---|
| SSP | 2 | 182.0 |
| | 6 | 567.6 |
| | 10 | 928.0 |
| SSI | 2 | 349.6 |
| | 6 | 1056.0 |
| | 10 | 1776.0 |

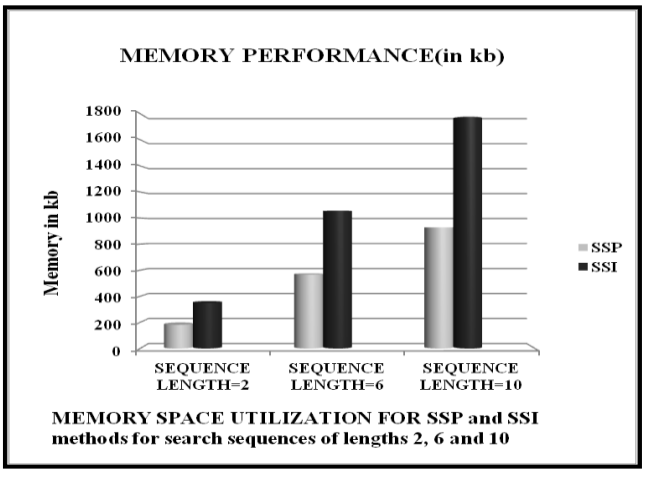


Figure 7. Total memory space of SSP and SSI techniques for search sequences of various lengths

The above graph shows the total memory space utilized for searching the sequences by SSP and SSI techniques. From the results, we observed that the SSP technique occupies minimum memory space than other two techniques.

## VI. CONCLUSION

In recent years, a number of applications may involve the management of the sequential information. The existing sequential search algorithms are inadequate to handle the sequence search operations. Hence there arises a need for the development of better algorithms for performing the search operations in sequence databases efficiently. A number of algorithms have been proposed for searching a single integer value or for searching a sequence of strings. This research work introduced the concept for searching a sequence of integers and two new search techniques namely SSP, SSI are proposed for performing search process for a sequence of integers on retail dataset. By analyzing the experimental results, it is clear that the SSP technique needs minimum execution time and SSI require minimum search time for searching the sequence. In terms of memory utilization, SSP occupies less amount of memory when compared with SSI technique.

## REFERENCES


[1] Agrawal R and Srikant R, "Fast Algorithms for Mining Association Rules", 20th Int. Conf. Very Large Data Bases, VLDB, Morgan Kaufmann, pp. 487-499, 1994.

[2] Agrawal R and Srikant R, "Mining Sequential Patterns",11 th Int. Conf. on Data Engineering, IEEE Computer Society Press, Taiwan, pp. 3-14, 1995.

[3] Beck,"On the linear search Problem".,Israel J. Mathematics, 1964.

[4] R. Bellman, "An optimal search problem", SIAM Rev, 1963.

[5] Booch G, "Object Oriented Analysis and Design", second Edition, Addison-Wesley, 1975.

[6] Boyer and Moore, "The Boyer-Moore Algorithm", 1977.

[7] Collins W. J, "Data Structures", first Edition Addison-Wesley publishing company, page 397, U.S.A, 1992.

[8] Donald Knuth and Vaughan Pratt, "Knuth-Morris-Pratt Algorithm", 1977.

[9] Jian Pei., Jiawei Han., Behzad Mortazavi-Asl., Jianyong Wang., Helen Pinto., Qiming Chen.,Umeshwar Dayal,, and Mei-Chun Hsu, "Mining Sequential patterns by Pattern-Growth:The PrefixSpan Approach", IEEE Transactions on Knowledge and Data Engineering, vol. 16, no. 10, 2004.

[10] Pranesh Das and Prof. Pabitra Mohan Khilar, "A Randomized Searching Algorithm and its Performance analysis with Binary Search and Linear Search Algorithms", The International Journal of Computer Science & Applications (TIJCSA),Volume 1, No. 11, ISSN – 2278-1080, 2013.

[11] Raita, "Tuning the Boyer-Moore-Horspool string-searching algorithm. Software - Practice Experience", 22(10), 879-884, 1992.

[12] S.S. Sheik, Sumit K.Aggarwal, Anindya Poddar, N.Balakrishnan and K.Sekar, "A FAST Pattern Matching Algorithm", J.Chem. Inf. Comput. Sci., 44, 1251-1256, 2004.

[13] Sunday D.M. A very fast substring search algorithm", Commun. ACM, 33(8), 132-142, 1990.

[14] Vijayarani and Deepa, "An efficient algorithm for sequence generation in Data mining", International Journal of Cybernetics and Informatics, 2013. (Accepted Paper).

[15] Zelda B. Zabinsky and Robert L.Smith, "An adaptive Random Search algorithm with linear complexity in dimension", Technical Report 90-15, 1990.