1. Introduction
Web usage mining (WUM), WUM is the process of extracting useful information from server logs. Web Usage Mining is the application of data mining techniques to discover interesting usage patterns from Web data in order to understand and better serve the needs of Web-based applications [1]. Web Usage Mining can be described as the discovery and analysis of user access patterns, through the mining of log files and associated data from a particular Web site. With using weblogs we can find out frequent patterns to improve the website and it can be useful for the web recommendation. Web log files contain the information about user like Date, Time, Site Name, IP Address, URI, User-Agent, Status, Access time etc.

Sequential pattern mining (SPM) is an important data mining task of discovering time-related behaviors in sequence databases. Sequential Pattern mining is a topic of data mining concerned with finding statistically relevant patterns between data examples where the values are delivered in a time series data [2]. The concept of sequence Data Mining was first introduced by Rakesh Agrawal and Ramakrishna Srikant in the year 1995 [3]. SPM technology has been applied in many domains, like web-log analysis, the analyses of customer purchase behavior, process analysis of scientific experiments, medical record analysis etc. Sequential pattern mining discovers frequent subsequences as patterns in a sequence database. A sequence database stores a number of records, where all records are sequences of ordered events, with or without concrete notions of time. An example sequence database is retail customer transactions or purchase sequences in a grocery store showing, for each customer, the collection of store items they purchased every week for one month.

With using SPM methods for web log mining we can propose a good recommendation for web. It can be more beneficial to find the sequence of users’ behavior in web usage mining. SPM algorithms are broadly categorized into three approaches: Apriori based Pattern Growth and Early pruning [9]. These techniques are for finding sequence pattern mining using different algorithms. Basically it provides the sequence patterns not the frequent pattern, web recommendation and also from application point of view it is helpful. Apriori algorithm generates very large amount of candidate sequence and perform too many database scan. Pattern growth based algorithm can solve above problem like Prefix span. It works on projected database but it fails if we consider the time to generate to projected database. Early pruning is a way to reduce search space and processing time for mining. In comparison of these algorithms, the hybrid algorithm of pattern growth and early pruning is more efficient for Sequence pattern mining.

2. Categorization of SPM algorithms.
SPM algorithms are mainly categorized in three terms namely apriori based, pattern growth and early pruning [9].

Key features of apriori based methods
Breadth first search: Apriori-based algorithms are described as breath-first (level-wise) search algorithms because they construct all k-sequences together in each kth iteration of the algorithm as they traverse the search space.
Generate and test: Algorithms that depend on this feature only display an inefficient pruning method and generate an explosive number of candidate sequences, consuming a lot of memory in the early stages of mining.

Multiple database scan:
Disadvantage
- It is a very undesirable characteristic of most apriori-based algorithms.
- Requires a lot of processing time and I/O cost.

Key features of pattern growth based methods
Sampling /Compression: Compression is used in the data structure that holds the candidate sequences, usually a tree.
Sampling: The problem with sampling is that the support threshold must be kept small, which causes a combinatorial explosion in the number of candidate patterns.
Candidate Sequence Pruning: Pattern-growth algorithms can prune candidate sequences early while maintaining a smaller search space and maintain a more directed and narrower search procedure.
Prefix span [6] - Uses direct antimonotic app of apriori property to prune candidate sequence along with projected database.
PLWAP [10] - It also has a position-coded feature that enables it to identify locations of nodes relevant to each other as a look-ahead capability and to prune candidate sequences early in the mining process.
Search Space Partitioning: It allows partitioning of the generated search space of large candidate sequences for efficient memory management.

Tree Projection: Here algorithms implement a physical tree data structure representation of the search space, which is traversed breadth-first or depth-first in search of...
frequent sequences.

**Depth first Traversal**: It has been stressed a lot and made very clear in several works that depth-first search of the search space makes a big difference in performance, and also helps in the early pruning of candidate sequences as well as mining of closed sequences.

**Suffix/Prefix growth**: This greatly reduces the amount of memory required to store all the different candidate sequences that share the same prefix/suffix.

**Memory only**: This feature targets algorithms that do not spawn an explosive number of candidate sequences, which enables them to have minimum I/O cost.

Key features of early pruning based methods

**Support counting avoidance**: A sequence database can be removed from memory and no longer be used once the algorithm finds a way to store candidate sequences along with support counts in a tree structure, or any other representation for that matter. Vertical projection of db: The amount of computation incurred by bitwise (usually AND) operations used to count the support for each candidate sequence.

**Position Coded**: It enables an algorithm to look-ahead to avoid generating infrequent candidate sequences.

<table>
<thead>
<tr>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>Uses Candidate generation and test approach</td>
<td>Uses Regular Expressions (REs) as a flexible Constraint</td>
<td>Uses vertical format sequential pattern mining method</td>
<td>Uses divide-and-conquer approach</td>
<td>Uses Projected database concept</td>
</tr>
<tr>
<td>Requires Multiple database scan.</td>
<td>Requires Multiple scans</td>
<td>Requires only three database scans</td>
<td>Reduces the cost of scanning multiple projected databases</td>
<td>Requires single database scan</td>
</tr>
<tr>
<td>Generates long sequential pattern, large number of candidates</td>
<td>Generates long sequential pattern that satisfies user-specified RE constraints</td>
<td>Generates large number of patterns, many of them are trivial or useless</td>
<td>Projects a large sequence database recursively into a set of small projected sequence databases</td>
<td>Generates long sequential pattern and less number of projected databases</td>
</tr>
<tr>
<td>Generates some candidates which doesn’t have any existence in sequence database</td>
<td>Generates fewer candidates which have the potential to be frequent for higher values of minimum support.</td>
<td>Using equivalence classes on frequent sequences, the original problem Decomposes into smaller sub-problems</td>
<td>Recursively project a sequence database into a set of smaller databases based on the current set of frequent patterns</td>
<td>Never generates any prefix which is not present in sequence database</td>
</tr>
<tr>
<td>Not good for those applications where low support thresholds are used</td>
<td>Performs well, even if it contains a large number of cycles of moderate length</td>
<td>Good for fast mining of sequential patterns in large databases</td>
<td>Good for large set of sequential patterns</td>
<td>Good for those applications where low support thresholds are used</td>
</tr>
<tr>
<td>Performance is poor than PrefixSpan algorithm</td>
<td>Performance is poor than PrefixSpan algorithm</td>
<td>Performance is better than GSP and poor than PrefixSpan</td>
<td>Performance is better than GSP and poor than PrefixSpan</td>
<td>Performance is better than GSP algorithm</td>
</tr>
</tbody>
</table>

Table 1: A Comparative Study of Apriori and pattern growth based algorithms.[5]

It is true that in comparison of apriori based algo and pattern growth based algo prefix span is more efficient algorithm compare to others. But another algorithm that is PLWAP[10], a hybrid algorithm of Pattern growth and early pruning is more efficient for Sequence pattern mining. The table shows the comparison of prefixspan and PLWAP algorithm which is hybrid algorithm of pattern growth and early pruning. Comparative performance analysis of algorithms from each of the categories. Two data sets were used, a medium size data set described as CST35S5N0D200K and a large-size data set described as C15T8S8N120D800K. These were run at different minimum support values: low minimum supports of between 0.1% and 0.9% and regular minimum supports of 1% to 10%. PrefixSpan needs memory space to hold the sequence database plus a set of header tables and pseudoprojection tables[8], PLWAP enjoys the fastest execution times, as it clearly separates itself from WAP-mine and Prefix Span, especially at low minimum support values when more frequent patterns are found and with large data sets[9].
4. Conclusion
With the help of analysis of different algorithms of SPM and theoretical study, we can say that PLWAP, a hybrid algorithm outperforms pattern growth algorithms like PrefixSpan[6] and Apriori based algorithm like GSP[3]. It is clear that PLWAP Algorithm is more efficient with respect to running time, space utilization and scalability than other algorithms.