**Search Program Project**

**Requirements:**

1. Write Command Line interface for search sequence of integers in given matrix of integers rows.
2. Input matrix data should be private to the class.
3. Implement search functions for below criteria.
   1. Find all rows that have a specific sequence of numbers
   2. Find all rows that contain all the required numbers
   3. Find the row that has the closest match to a specific number sequence

And display the row numbers in console.

1. Allow to give multiple search sequence in input text file for processing.
2. Speed up search time

**Project Plan:**

**Sprint 1:  Design the search program - 2 days**

01. Class Design for the search program.

02. Setup Project and Git repository.

03. Implement code to process the input matrix data, sequence data and CLI.

04. Implement search program classes from design.

**Sprint 2: Create I/O for search program - 2 days**

05. Implement ordered search function.

06. Implement un-ordered search function.

07. Implement best match search function.

**Sprint 3: Implementation of search functions - 2 days**

08. Optimize the ordered search functions

**Sprint 4: Creating Test case and Reports - 2 days**

09. Create and implement the test cases for search program

10. Creating report on optimizing the search functionality and design of the project.

**Design and Optimization of Search Program**

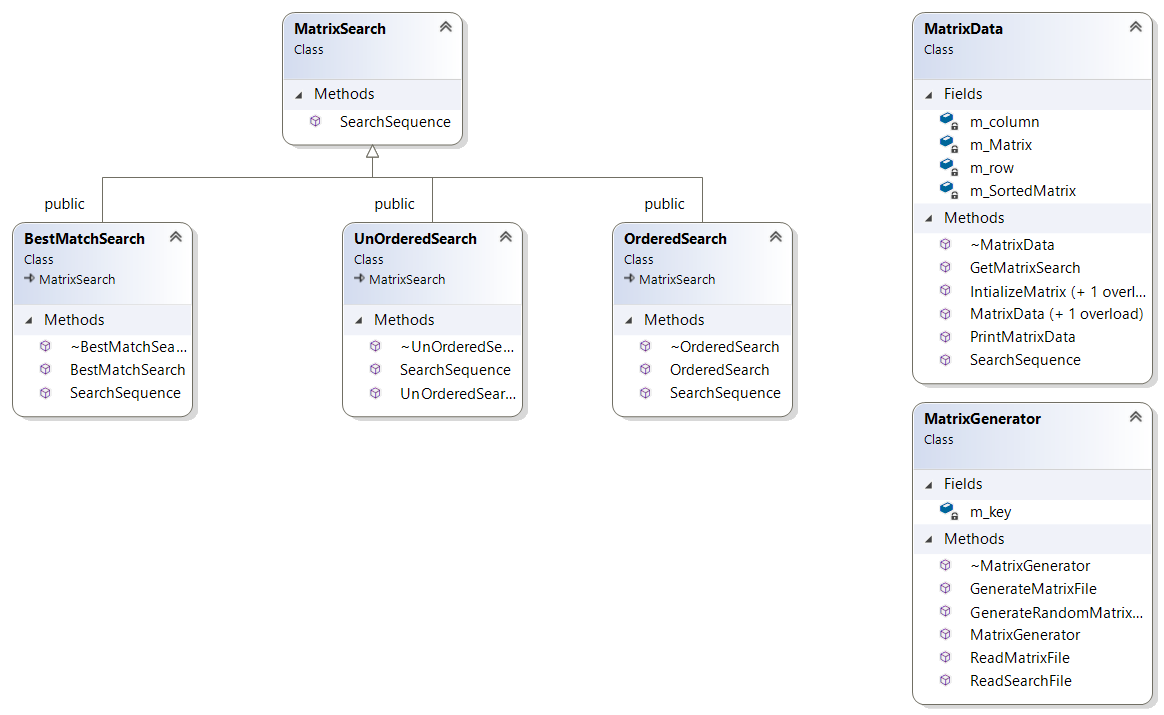
**Search Program:**

Search Program is used to find the matched rows for given sequence from matrix data provided in input (matrix.dat) file. sequence in matrix data can be found using different search functions defined according to user requirements. To implement search program created the “**MatrixData**” class for holding matrix values and its rows and columns.

To accommodate different search functionality this program implements strategy design pattern. In this design, different classes are created to allow implementation of different search functions such as “**OrderedSearch**”, “**UnorderedSearch**” and “**BestMatchSearch**” classes which are inherited from “**MatrixSearch**” base class. And these classes will implement “**SearchSequence**()” Method as defined for each search functions. This design can be extended to add more search functions according to user requirements.

Class “**MatrixData**” is implemented with below methods.

1. **GetMatrixSearch()** – To get the proper matrix search class depending on the search type string.
2. **IntializeMatrix()** – Method to generate the matrix from given input strings.
3. **SearchSequnence()** – Method to initiate the search sequence methods on matrix using input sequence string.



**Matrix Generator program:**

Matrix Generator program will take number of rows and columns as input arguments from user and generates the encrypted matrix data file(matrix.dat) at program location.

Created MatrixGenerator class which implements below methods.

1. **GenerateRandomMatrixFile()** – Generate the random matrix data is generated using random() function of standard library. This matrix data will have encrypted and written to the matrix.dat file, so that the data will not be readable as per project requirement.
2. **ReadMatrixFile()** - This method will be called from Search Program to read matrix data file and generate the **MatrixDataType** data. Which is used to in initializing the **MatrixData** object using **IntializeMatrix()** method.
3. **ReadSearchFile()** – This method is used to read the search sequence from text file is used in Search Program and then call search sequence functions on the matrix data.

These methods are part of MatrixGenerator.cpp file shared as part of Search program compilation.

**Ordered Search function:**

Ordered Search function is implemented in **OrderedSearch** class which is derived from **MatrixSearch** class and it overrides **SrearchSequence()** method as shown in class design diagram. This class method implements the “searchSequence” type of search functions where expectation is to find the row with matching input sequence. In this function at least one match of sequence is enough for considering the matrix row. And function output will be vector of matching rows.

Before Optimization, for each row of the matrix to match the sequence used the linear search (i.e. naïve search). This logic is implemented in method called.

int LinearSearchSequenceCount(const std::vector<int> &row, const std::vector<int> &sequence, bool bCount)

This method will take the row and sequence to search as input, and to find the sequence we will take each value of sequence and linearly search through until full sequence matches. The number of comparison in the worst case is O(m\*(n-m+1)). Where m is number values in sequence and n is number of values in row .

To optimize the sequence matching implemented KMP (Knuth Morris Pratt) algorithm. For KMP algorithm, first we need to pre-process the input sequence numbers and get the prefix array and this array will be used in matching algorithm to avoid duplicate matching of the row values with sequence i.e. number of values to be skipped after mismatch.

std::vector<int> GetSequenePrefixVector(const std::vector<int> &vecSequence)

In KMP algorithm, start by comparing the values of sequence with rows values. Keep matching the values of rows and sequence and if mismatch happens then with help of prefix array we will avoid matching already compared sequence which are duplicated one and consider next value to match. This algorithm improves the number of comparison from O(m\*(n-m+1)) to O(m+n).

int KMPSearchSequenceCount(const std::vector<int> &row, const std::vector<int> &sequence, const std::vector<int> &vecSeqPrefix, bool bCount )

To reduce the number of comparison further with above algorithm, first find the index of the first value of the sequence using std::find method from standard library. Then start search the sequence from that index. This would speed up the process by avoiding the through matching of first value from sequence through each element of the row.

**Best Match Search function:**

Best match Search function is implemented in “**BestMatchSearch**” class which is derived from **MatrixSearch** class and it overrides **SrearchSequence()** method as shown class design diagram. This class method implements the “searchBestMatch” type of search functions where we need to find the best matching row by considering the number of matches with input sequence. Unlike ordered search function, in this method will consider all the sequence matches in the matrix row.

This method will call the same method as in ordered search function and then get total number of sequence matches for each row and compare with all the rows match counts and consider row which is the maximum number of matches. The number of comparison are like ordered search function i.e. O(m+n).

**Unordered Search function:**

Unordered Search function is implemented in “**UnOrderedSearch**” class which is derived from **MatrixSearch** class and it overrides **SrearchSequence()** method as shown class design diagram. This class method implements the “searchUnordered” type of search functions to find the rows with all values of sequence without considering the order. But if the value repeats in sequence, row must contain at least that many number of values.

In unordered search function, take each number from sequence and search linearly in the row with below method. And then modifies that number by negating it so that in the next search the index should not be considered. This is to satisfy the second requirement to allow to find atleast that many sequence number. Here the number of comparisons in the worst case will be O(m\*n).

To improve above logic, sorted the rows while initializing the matrix and stored in **MatrixData** class object. And these sorted matrix data used only for unordered search type function. In search sequence method, sort the sequence and match with sorted rows like in ordered search function. Here we need to allow the mismatch of the sequence with row. In this method at least one match is enough to consider that row. Here the number of comparisons in the worst case for each row is O(m+n). below method is implemented with above logic.

int UnorderedSearchSortedSequence(const std::vector<int> &row, const std::vector<int> &sequence);

**Optimization Report:**

For calculating the optimization of search functions, generated random matrix data with 10000x10000 size and created matrix.dat file and taken search sequence of 20 numbers from last but one row of the matrix to make the number comparison to worst case. I have captured the elapsed time for each type search function with generated matrix data and input sequence. Below are the details of the elapsed time before and after optimization search functions as explained above.

|  |  |  |
| --- | --- | --- |
| Search type | Before Optimization (sec) | After Optimization(sec) |
| searchSequence | 1.35938 | 0.765625 |
| searchUnordered | 1.26562 | 0.78125 |
| searchBestMatch | 1.34375 | 0.765625 |