# $\begin{array}{c} \textbf{Project in ME001-Sampling system} \\ \textbf{Group 1} \end{array}$

By Chen YuXuan 1809853 J-I<br/>011-0011 D1  $\,$ 

Wang Yuan 1809853G-I011-0030 D1

He PeiLin 1809853U-I011-0078 D1

December 2, 2020

## Contents

1	Restatement of the Problem			
2	$\begin{array}{llllllllllllllllllllllllllllllllllll$	3 3 5 5 6		
3	Essential Codes and Functions Analysis 3.1 Realization of Modifying DB files	<b>7</b> 7 9		
4	Steps to Run the Program			
5	Program Test			
6	Summary			
Aı	ppendices	18		
A	Programs for Algorithms	18		
В	Programs for GUI			

#### 1 Restatement of the Problem

In this project, we are expected to extract a subset of samples of big data. Assume there are m samples ( $45 \le m \le 54$ ), any n ( $7 \le n \le 25$ ) samples out of these m samples are selected. There are  $C_n^m$  groups of n samples. From one of these groups of n samples, we randomly selected k ( $4 \le k \le 7$ ) samples to form some groups. So there will be  $C_n^k$  groups of k samples selected. There are at least **ONE** group of k samples, in which s ( $3 \le s \le 7$ ) samples have been selected from the j (where  $s \le j \le k$ ) samples. Among these groups of k samples, we would like to optimize them by selecting ONLY some of them.

#### 2 Basic Ideas

We can divide the problem into two parts, j = s and  $j \neq s$ .

**2.1** 
$$j = s$$

#### 2.1.1 Algorithm to Find Subsets

Now, we have a set whose number of the element is n. Then we want to find out all the subsets whose number of the element is k.

#### Algorithm:

• First, we put the origin set to a container, and then we label every element to one(illustrate the picture below). We assume that the origin set is S,  $S = \{1, 2, 3, 4, 5\}$  in Table.2.1.1. Then, the subset which has the same

Table. 2.1.1

element with the original set's is labeled the element to 1, otherwise labeling it to 0. For example, we suppose that one the subset is  $S_1, S_1 = \{1, 2, 4\}$ . We can represent it as Table.2.1.2. Now we can change the number below the array to a binary number, which means that each subset can be represented by a unique number from 0(empty set) to  $2^n - 1$ (original set). Just like the example above set S can be represented by  $11111_2 = 31_{10}$  and  $S_1$  can be expressed as  $01011_2 = 11_{10}$ 

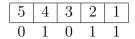


Table. 2.1.2

• Subsequently, we know how to find subsets of the original set, but I want to know how to find the subset with the specific number of elements. Therefore, we only need to know the subset whose binary number representation contains k 1s. As the example in Table.2.1.2,  $S_1 = \{1, 2, 4\}$ : So, the  $S_1$  contains three elements, because it has three 1s.

In this way, we can easily find out the subset whose number of elements is k from 0 to  $2^n - 1$ , the code block findSubsetOfk illustrates the situation.

```
void findSubsetOfk(int n, int k, vector<int> subsetK){
2
       int count=0; //number \ of \ 1s
       for (int i = 1; i < (1 << n); i++){
3
            for (int j = 0; j < n; j++){
4
                 //the binary number representation
5
                //of subset has an 1 on the jth position
6
                 if (i & (1 << j)!=0) {
7
                     count++;
8
                 }
9
10
            if (count==k)
11
                 susetK.empalce back(i);
12
            count = 0;
13
14
       }
15
16
```

However, we can easily find that the binary number representation of the subset whose number of elements is k is no less than  $2^k - 1$ . Therefore, we the code above, we can have an optimization on the i. The optimized code findSubsetOfkOptim is

```
//the binary number representation
5
                 //of subset has an 1 on the jth position
6
                 if (i & (1 << j)! = 0) {
7
8
                      count++;
                 }
9
10
             if (count==k)
11
                 susetK.empalce_back(i);
12
            count = 0;
13
        }
14
15
16
```

• Currently, we can use the same way what we say above to find out the subset of the set whose number of element is k and its number of elements is s.

#### 2.1.2 Calculate the Combination Number

If we calculate the combination number directly, it is likely to out of bounds of int. So we can use combination formula:

$$C_n^m = C_{n-1}^{m-1} + C_{n-1}^m$$

to calculate the combination number. And the specific implementation code can be seen in calculateCombination.

```
int calculateCombinationNumber(int n, int m){
    for(int i=0;i<=n;i++)
        C[i][0]=1;

for(int i=1;i<=n;i++)
        for(int j=1;j<=i;j++)
        C[i][j]=C[i-1][j-1]+C[i-1][j];

return C[n][m];

}</pre>
```

#### 2.1.3 Greedy Algorithm to Calculate the Set Coveraged

We denote that the input is a set  $\mathcal{U}$  of n elements, and a collection  $S = \{S_1, S_2, ..., S_m\}$  of m subsets of  $\mathcal{U}$  such that  $\cup_i S_i = \mathcal{U}$ . Our goal is to take as few subsets as pos-

sible from S such that their union covers  $\mathcal{U}$ . We can solve this problem easily by greedy algorithm. The algorithm is below in Table.2.1.3:

Greedy Cover $(S,\mathcal{U})$ 

- 1. repeat
- 2. pick the set that covers the maximum number of uncover element
- 3. mark elements in the chosen set as covered
- 4. remove the set from S to the result set
- 5. done

Table. 2.1.3. Greedy Cover

Based on the three lemmas above, we can easily transform the problem to that the set  $\mathcal{U} = \{1, 2, \cdots, C_n^j\}$ , which means that we map each different subset whose the number of the elements is j to a unique code from 1 to  $C_n^j$ . Each subset of S, represents the each k set's subsets whose number of elements is j. Ultimately, we can solve the problem easily.

#### **2.2** $j \neq s$

The way to solve the problem is just like the way we mentioned above. However, after finishing finding the subset of the k set whose element number is s, we should know how many sets whose the number of elements is j include it. Therefore, we use **DFS(depth first search)** to find out them. Assuming that n = 5, s = 3, j = 4, and the subset whose number of elements is equal to 3 is labeled as 01011<sub>2</sub>. Therefore, we can expand it as below in Table.2.2.1.

5	4	3	2	1
0	1	0	1	1
0	1	1	1	1
1	1	0	1	1

Table. 2.2.1

Then, we should mark the last two rows of the set above in the  $\mathcal{U}$  as covered.

### 3 Essential Codes and Functions Analysis

#### 3.1 Realization of Modifying DB files

As the request said, we need output the group of k samples and corresponding result in DB files. Consequently, the OOP program language C# can provide abundant libraries to help realize combine with modifying DB files.

Depending on **C**# powerful library and interface, we can apply our algorithm source code on GUI platform, and realize the operation of creating new files(Code.1), exporting result into corresponding files(Code.2) as well as deleting the specular data(Code.3).

```
public void CreateTableInToMdb(string fileNameWithPath)
2
3
       try
       {
4
            OleDbConnection myConnection = new OleDbConnection
5
                ("Provider=Microsoft.Jet.OLEDB.4.0; Data Source="
6
                    + fileNameWithPath);
7
           myConnection. Open();
8
           OleDbCommand myCommand = new OleDbCommand();
9
10
           myCommand. Connection = myConnection;
           myCommand.CommandText =
11
            "CREATE TABLE my_table([m] NUMBER," +
12
                "[n] NUMBER, [k] NUMBER, [j] Number, "+
13
                    "[s] NUMBER, [n numbers] TEXT," +
14
                         "[minium number of sets] NUMBER, "+
15
                             "[answer] TEXT)";
16
           myCommand. ExecuteNonQuery();
17
           myCommand. Connection. Close();
18
19
       catch { }
20
21
   public void InsertToMdb(string fileNameWithPath)
1
2
       var con = new OleDbConnection(
3
            "Provider = Microsoft. Jet. OLEDB. 4.0; Data Source = "
4
               + fileNameWithPath);
5
       var cmd = new OleDbCommand();
6
```

```
7
       cmd.Connection = con;
       cmd. CommandText = "insert into my_table ([m], [n], [k], [j], " +
8
            "[s], [n numbers], [minium number of sets], [answer])" +
9
                "values (@m, @n, @k, @j, @s, @series1, @number, @answer);";
10
       cmd. Parameters. AddWithValue ("@m", numericUpDown1. Value);
11
       cmd. Parameters. AddWithValue("@n", numericUpDown2. Value);
12
       cmd. Parameters. AddWithValue ("@k", numericUpDown3. Value);
13
       \verb|cmd.Parameters.AddWithValue("@j", numericUpDown4.Value)|;|
14
       cmd. Parameters. AddWithValue("@s", numericUpDown5. Value);
15
       cmd. Parameters. AddWithValue("@series1", series1Fordb());
16
       cmd. Parameters. AddWithValue("@number", vs.Count());
17
18
       cmd. Parameters. AddWithValue("@answer", series2Fordb());
       con. Open();
19
       cmd. ExecuteNonQuery();
20
       con. Close ();
21
22
   private void DeleteRecordFromMdb
1
2
       (string fileNameWithPath, string num)
3
       int number = Int32. Parse (num);
4
       var con = new OleDbConnection
5
            ("Provider = Microsoft.Jet.OLEDB.4.0; Data Source = "
6
                + fileNameWithPath);
7
       var cmd = new OleDbCommand();
8
       con. Open();
9
       cmd.Connection = con;
10
       cmd.CommandText = "DELETE FROM [my table] " +
11
            "WHERE [order]=" + number + "";
12
       cmd. ExecuteNonQuery();
13
       con. Close();
14
15
16
   private void DeleteAllRecordFromMdb
17
18
       (string fileNameWithPath)
19
       var con = new OleDbConnection
20
            ("Provider = Microsoft.Jet.OLEDB.4.0; Data Source = "
21
```

```
+ fileNameWithPath);
var cmd = new OleDbCommand();
con.Open();
cmd.Connection = con;
cmd.CommandText = "DELETE FROM [my_table]";
cmd.ExecuteNonQuery();
con.Close();
}
```

#### 3.2 Multi-Threading

We adopt multi-threading programming way. We split the program into two parts, which are the GUI part and the calculation part. In this way, even if the program haven't figured out, the window of the program won't be stick. The specific implemented function is bound in <a href="mailto:button2\_Click">button2\_Click</a>.

```
private async void button2_Click(object sender, EventArgs e)
1
2
   // Run buttom
3
        button2. Enabled = false;
4
        Algorithm algorithm = new Algorithm (
5
            (int) numeric Up Down 2. Value,
6
                 (int) numeric Up Down 3. Value,
7
                     (int) numeric Up Down 4. Value,
8
                          (int) numeric Up Down 5. Value,
9
                              totalList, judgeNumber);
10
           (numericUpDown4. Value = numericUpDown5. Value)
11
12
       {
            vs= await Task.Run(()=>algorithm.ExecuteAlgorithm1());
13
14
        else
15
16
            vs = await Task.Run(()=>algorithm.ExecuteAlgorithm2());
17
18
       //InsertToMdb(openFileDialog1.FileName);
19
        //UpdateToMdb(openFileDialog1.FileName);
20
       textBox3. Text = GetSeries2();
21
       //\text{textBox3}. Enabled = false;
22
23
```

## 4 Steps to Run the Program

The detailed information of programming are listed in Table.4.0.1.

Attribute	Content
Operation System	Windows SDK edition: 10.0
Integrated Development Environment	Visual Studio 2019(v142)
Solution Settings	Release in x86 Platform
Optimization	O2 optimize
Programming Language	C#
Framework	. Net framework 4.7.2
Source Code hosting Platform	Github

Table. 4.0.1. Settings and Attribute

In order to make the operation more smooth, all the program environment and settings are completed and included in the file package. Just required to follow the steps below to run the program.

1. Open the package and find the *Information System Project.exe* file. Double-click the file to enter the program interface as Figure.4.0.1 exactly.

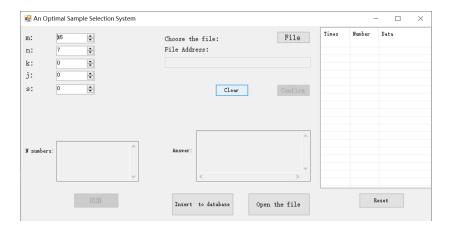
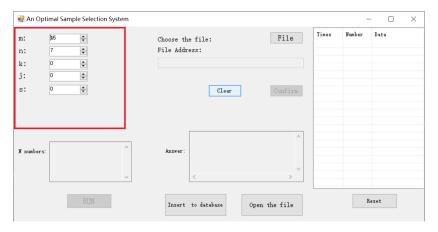


Figure. 4.0.1. Initial GUI

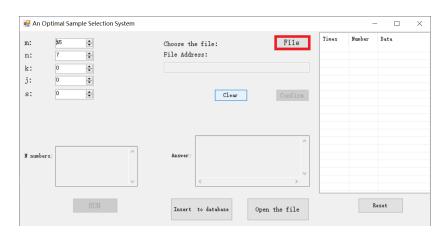
In order to record the relevant output data of the program and facilitate display and modification later. It is required to create a .mdb file to store it, which is called DataBase.mdb in project package for example.

2. Choose the data of each parameter and input on the program surface as Figure.4.0.2.

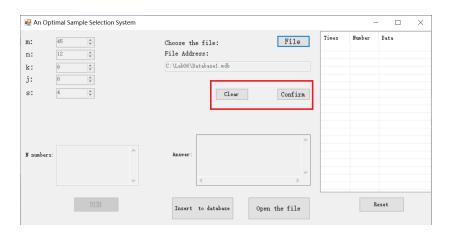


**Figure. 4.0.2.** Step1

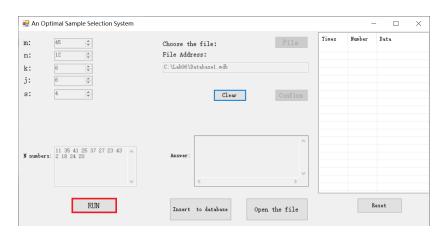
- 3. Choose the DB file to store and operate the data, click the button **File** and choose the .mdb as Figure.4.0.3 In the previous step and **Confirm** if all get right.(**Clear** is a function that clear all the data you have input, including the parameter followed Figure.4.0.4)
- 4. Push the **RUN** button and the **N** number and final answer of your input will be shown on the surface window as Figure.4.0.5, you can check the answer after that.
- 5. After confirming the data is correct, use the button **Insert to database** (Figure.4.0.6) to download the data on the DB file(.mdb), and the button **Open the file**(Figure.4.0.7) can open it to display the data you have calculated. It is also easy for you to delete or use any other operation on the data though your DB file.
- 6. After adding the data into your DB file, you can get the record of the message including order, calculated result and data information, which you can see the tips of first row. Moreover, you can choose the data which



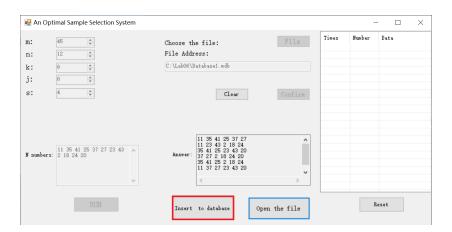
**Figure. 4.0.3.** Step2



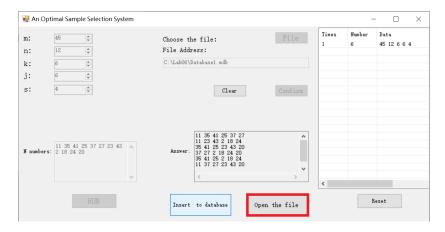
**Figure. 4.0.4.** Step3



**Figure. 4.0.5.** Step4

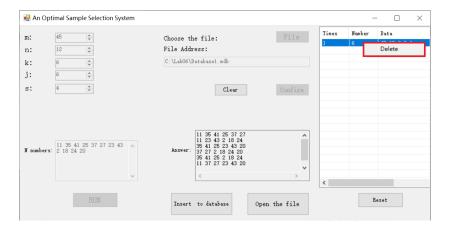


**Figure. 4.0.6.** Step5



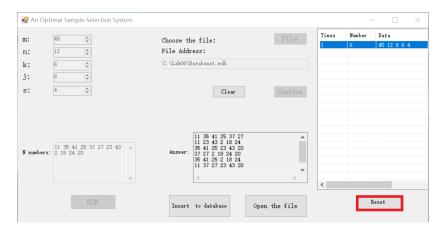
**Figure. 4.0.7.** Step6

you plan to delete by clicking the right button of mouse, then you can see the menu item **Delete** (Figure.4.0.8).



**Figure. 4.0.8.** Step7

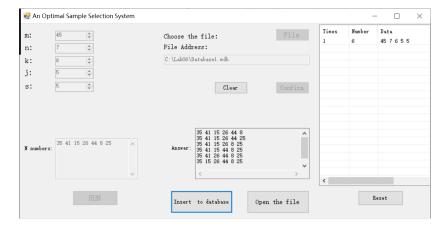
7. Besides, you can clear all the data from your DB file by using button **Reset** (Figure.4.0.9).



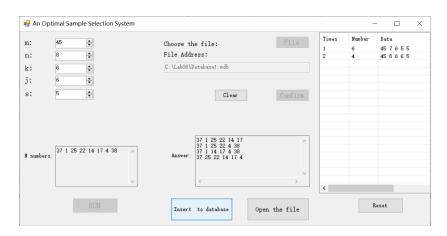
**Figure. 4.0.9.** Step8

## 5 Program Test

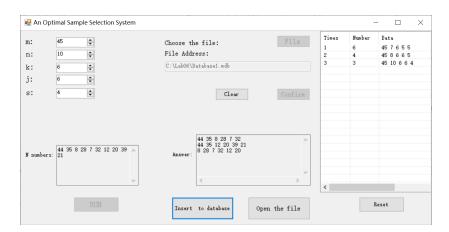
If the program window can be displayed normally, you can enter the value for verification. The conditions of 1, 2, 3 and 4, 5 and 6, 7 in the project requirement file are similar, so we choose 1(Figure.5.0.1), 4(Figure.5.0.2), and 6(Figure.5.0.3) as the demo of our program.



**Figure. 5.0.1.** E.g.1: Input the data: m = 45, n = 7, k = 6, j = 5, s = 5.



**Figure. 5.0.2.** E.g.4: Input the data: m = 45, n = 8, k = 6, j = 6, s = 5.



**Figure. 5.0.3.** E.g.6: Input the data: m = 45, n = 10, k = 6, j = 6, s = 4.

## 6 Summary

This project is based on the theoretical direction of **ME001** subject and combines some knowledge of data structure and mathematic, including optimal algorithms and combinatorics. But there are no correct understanding of some part of difficult and profound mathematic problems like fuzzy set. Authors point out about converse decimal digits to the binary make the big data abstraction in order to descend the time complexity. By using program, authors realize the process from theory to practice reflecting the theoretical view of unity of knowledge and practice. When writing large-scale projects, people often need cooperation and collaborative development, authors use **GitHub** for collaborative development and submit own patch code to collaborator's repository. In this article, we will utilize ideas to achieve team cooperation on **GitHub**. The last but not least, the goal of the future study and work is to work harder to learn this knowledge, in order to enrich, improve our level.

# Appendices

## A Programs for Algorithms

#### source/Algorithm.cs

```
using System;
  using System. Collections. Generic;
   using System. Collections. Specialized;
  using System. Component Model;
  using System. Component Model. Design;
  using System. Diagnostics;
  using System. Drawing. Drawing2D;
   using System. Drawing. Text;
   using System. Linq;
   using System. Security. Cryptography;
   using System. Text;
11
   using System. Threading. Tasks;
12
13
   namespace Information_System_Project
14
15
       public class Algorithm
16
17
18
            private int n;
            private int k;
19
            private int j;
20
21
            private int s;
            private List<int> setNumberForK = new List<int>();
22
            private List<int> setNumberForJ = new List<int>();
23
            private Dictionary<int , int> dictionaryForAllSets
24
                = new Dictionary \langle int, int \rangle();
25
            private Dictionary<int , int> dictionaryForAllSets2
26
27
                = new Dictionary \langle int, int \rangle();
            private bool [] visit = new bool [10000000];
28
            private List<int> totalList = new List<int>();
29
            private Queue<int> queueForSet = new Queue<int>();
30
            private bool [] judgeNumber = new bool [46];
31
            private int [,] C = new int [26,26];
32
```

```
public Algorithm (int n, int k, int j, int s,
33
                List <int > totalList, bool[] judgeNumber)
34
            {
35
36
                 this.n = n;
                 this.k = k;
37
                 this.j = j;
38
                 this.s = s;
39
                 this.totalList = totalList;
40
                 this.judgeNumber = judgeNumber;
41
            }
42
            public Queue<int> ExecuteAlgorithm1()
43
44
45
                Greedy Algorithm ();
                return queueForSet;
46
47
            public Queue<int> ExecuteAlgorithm2()
48
49
                 GreedyAlgorithm2();
50
                 return queueForSet;
51
52
            private void GreedyAlgorithm()
53
54
                setNumberForK=CombinationForAllNum(n, k);
55
                int max;
56
                int now = 0;
57
                int allNum = TotalNumberForJ(n, j);
58
                int node;
59
                int index;
60
                List < int > vis = new List < int > ();
61
                List < int > result = new List < int > ();
62
                 while (allNum > 0)
63
                {
64
65
                     //Debug.WriteLine(allNum);
66
                     \max = 0;
                     node = 0;
67
                     index = 0;
68
                     foreach (var element
69
                          in setNumberForK)
70
```

```
71
                           int numOfUnfound = 0;
72
                           for (int j1 = (1 \ll s) - 1;
73
                                j1 \ll element; j1++)
74
                           {
75
                                int cnt = 0;
76
                                for (int k1 = 0; k1 < n; k1++)
77
78
                                    if ((j1 \& (1 << k1)) != 0)
79
                                    {
80
                                         cnt++;
81
82
83
                                if ((j1 \& element) = j1
84
                                    && cnt == s &&
85
                                         ! dictionary For All Sets.
86
                                              ContainsKey(j1))
87
                                {
88
                                    numOfUnfound++;
89
                                    vis.Add(j1);
90
91
                           }
92
                              (max < numOfUnfound)
                           i f
93
94
95
                                node = index;
                                \max = \text{numOfUnfound};
96
                                result. Clear();
97
                                foreach (var each Num in vis)
98
99
                                    result.Add(eachNum);
100
101
102
                           vis. Clear();
103
                           index++;
104
105
                      queueForSet.Enqueue(setNumberForK[node]);
106
                      setNumberForK . RemoveAt(node);
107
                      foreach (var each Num in result)
108
```

```
109
                           dictionaryForAllSets[eachNum] = ++now;
110
111
                      allNum -= max;
112
113
                  }
114
115
116
             private void GreedyAlgorithm2()
117
118
                  setNumberForK=CombinationForAllNum(n, k);
119
120
                  int max;
121
                  int now = 0;
                  int allNum = TotalNumberForJ(n, j);
122
                  int node;
123
                  int index;
124
                  List < int > vis = new List < int > ();
125
                  List < int > result = new List < int > ();
126
                  while (allNum > 0)
127
128
                  {
                      \max = 0;
129
                      node = 0;
130
                      index = 0;
131
                      foreach (var element in setNumberForK)
132
                      {
133
                           int numOfUnfound = 0;
134
                           for (int j1 = (1 \ll s) - 1;
135
                               j1 \ll element; j1++)
136
                           {
137
                                int cnt = 0;
138
                                var answer = 0;
139
                                for (int k1 = 0; k1 < n; k1++)
140
141
                                    if ((j1 \& (1 << k1)) != 0)
142
143
                                         cnt++;
144
145
146
```

```
if ((j1 \& element) = j1
147
                                    && cnt == s &&
148
                                         ! dictionaryForAllSets.
149
                                             ContainsKey(j1))
150
                               {
151
152
                                    int num = j - s;
                                    numOfSetContainj1
153
                                         (0,j1, num, ref answer);
154
                                    numOfUnfound += answer;
155
                                    vis .Add(j1);
156
157
158
                           foreach (var eachNum in setNumberForJ)
159
160
                               dictionaryForAllSets2.Remove(eachNum);
161
162
                           setNumberForJ. Clear ();
163
                           if (max < numOfUnfound)
164
                           {
165
166
                               \max = \text{numOfUnfound};
                               node = index;
167
                               result. Clear();
168
                               foreach (var each Num in vis)
169
170
                                    result.Add(eachNum);
171
172
173
174
                           index++;
                           vis. Clear();
175
176
                      //Debug.WriteLine(max);
177
                      SetDictionary2 (result);
178
179
                      queueForSet.Enqueue(setNumberForK[node]);
                      foreach (var each Num in result)
180
181
                           //Debug.Write(eachNum + "");
182
                           dictionaryForAllSets[eachNum] = ++now;
183
184
```

```
setNumberForK.RemoveAt(node);
185
                      allNum = max;
186
                  }
187
188
             private void numOfSetContainj1(int node, int j1,
189
                  int num, ref int answer)
190
             {
191
                  if (num = 0 \&\&
192
                      ! dictionaryForAllSets2.
193
                           ContainsKey(j1))
194
195
                  {
196
                      answer += 1;
                      setNumberForJ.Add(j1);
197
                      dictionaryForAllSets2[j1] =
198
                           dictionaryForAllSets2.Count() + 1;
199
200
                      return;
                  }
201
                  else if (num = 0 \&\&
202
                      dictionaryForAllSets2.
203
204
                           ContainsKey(j1))
205
                      return;
                  for (int i1 = node; i1 < n; i1++)
206
                  {
207
                      if(((1 << i1) \& j1) == 0)
208
                      {
209
                           numOfSetContainj1(i1+1, j1 | (1 << i1),
210
                               num -1, ref answer);
211
                      }
212
                 }
213
             }
214
215
             private void SetDictionary2(List<int> result)
216
217
                  foreach (var each Num in result)
218
219
                      FindEachElement (0, eachNum, j-s);
220
221
222
```

```
223
             private void FindEachElement(int node, int element,
224
                  int num)
225
226
227
                  if (num = 0 \&\&
                      ! dictionaryForAllSets2.
228
                           ContainsKey (element))
229
230
                      dictionaryForAllSets2 [element]
231
                           = dictionaryForAllSets2.Count() + 1;
232
233
                      return;
234
                  }
                  else if (num = 0 \&\&
235
                      dictionaryForAllSets2.
236
                           ContainsKey (element))
237
238
                      return;
                  for (int i1 = node; i1 < n; i1++)
239
240
                      if (((1 << i1) \& element) == 0)
241
242
                      {
                           FindEachElement(i1, element | (1 << i1),
243
                               num -1);
244
245
                 }
246
             }
247
248
             private List<int> CombinationForAllNum(int n, int k)
249
250
                  List < int > Combination = new List < int > ();
251
                  for (int i = (1 << (k)) - 1; i < (1 << n); i++)
252
253
                      var cnt = 0;
254
255
                      for (int j1 = 0; j1 < n; j1++)
256
                           if ((i & (1 << j1)) != 0)</pre>
257
258
                                cnt++;
259
260
```

```
261
                      if (cnt == k)
262
                      {
263
                           Combination . Add(i);
264
                           //myBV. Add(new Bit Vector32(i));
265
                      }
266
267
                  return Combination;
268
269
             private int TotalNumberForJ(int n, int j)
270
271
272
                  for (int i = 0; i <= n; i++)
273
                      for (int m = 0; m \le j; m++)
274
275
                          C[i, m] = 0;
276
277
278
                  for (int i = 0; i \le n; i++)
279
280
                      C[i, 0] = 1;
281
                      if (i == n && j == 0)
282
                           return C[n, j];
283
284
                  for (int i = 1; i <= n; i++)
285
286
                      for (int m = 1; m \le i; m++)
287
288
                           C[i, m] = C[i - 1, m - 1] + C[i-1, m];
289
                           if (i == n && m == j)
290
                               return C[n, m];
291
                      }
292
293
                  return C[n, j];
294
             }
295
296
             private void Dfs(int start, int setNum,
297
                  int currentNumber, int totalNum, ref int result)
298
```

```
299
300
                  int now = currentNumber;
301
                  List < int > vis = new List < int > ();
302
                  if (setNum >= result)
303
                       return;
304
                  if (currentNumber == totalNum)
305
306
307
                       //if (setNum < result)
308
                       result = setNum;
309
310
                       return;
311
                  for (int i = start; i < setNumberForK.Count; i++)</pre>
312
313
                       if (! visit[i])
314
                       {
315
                            for (int j1 = (1 \ll s) - 1;
316
                                 j1 \le \operatorname{setNumberForK}[i]; j1++)
317
                            {
318
                                 int cnt = 0;
319
                                 for (int k1 = 0; k1 < n; k1++)
320
                                 {
321
                                      if ((j1 & (1 << k1)) != 0)
322
323
                                      {
324
                                          cnt++;
325
326
                                     ((j1 \& setNumberForK[i])==j1
327
                                     && cnt == s &&
328
                                          ! dictionary For All Sets.
329
                                               ContainsKey(j1))
330
331
                                 {
332
                                     now++;
                                      dictionaryForAllSets[j1] = now;
333
                                      vis.Add(j1);
334
335
336
```

```
//Debug.WriteLine("");
337
                           Dfs(i + 1, setNum + 1, now,
338
                                totalNum, ref result);
339
                           visit[i] = false;
340
341
                           now = currentNumber;
342
343
                      foreach (var eachNum in vis)
344
345
                           dictionaryForAllSets.Remove(eachNum);
346
347
348
                      vis.Clear();
349
350
                  return ;
             }
351
        }
352
353
```

## B Programs for GUI

#### source/Form1.cs

```
using System;
  using System. Collections. Generic;
  using System. Component Model;
  using System. Data;
  using System. Data. OleDb;
  using System. Diagnostics;
  using System. Drawing;
  using System. IO;
  using System. Linq;
  using System. Text;
10
   using System. Threading;
   using System. Threading. Tasks;
   using System. Windows. Forms;
13
14
  namespace Information_System_Project
15
16
       public partial class Form1 : Form
17
```

```
18
            int cnt = 1;
19
            Queue<int> vs;
20
            List < int > totalList = new List < int > ();
21
            bool [] judgeNumber = new bool [55];
22
            OpenFileDialog openFileDialog1
23
                = new OpenFileDialog();
24
25
26
            public Form1()
27
                 InitializeComponent();
28
29
                 list View 1. View = View. Details;
                 listView1.GridLines = true;
30
                 listView1.FullRowSelect = true;
31
                 list View 1. Columns. Add("Times", 60);
32
                 list View 1. Columns. Add ("Number", 60);
33
                 list View 1. Columns. Add ("Data", 244);
34
            }
35
36
37
            private void numericUpDown3_ValueChanged
38
                 (object sender, EventArgs e)
39
40
                numeric Up Down 3. Maximum
41
                     = numericUpDown2. Value;
42
43
            }
44
45
            private void numericUpDown4_ValueChanged
46
                 (object sender, EventArgs e)
47
48
                numericUpDown4. Maximum
49
50
                     = numericUpDown3. Value;
            }
51
52
            private void numericUpDown5_ValueChanged
53
                 (object sender, EventArgs e)
54
55
```

```
numericUpDown5. Maximum
56
                     = numericUpDown4. Value;
57
            }
58
59
            private void button1 Click
60
                 (object sender, EventArgs e)//Confim button
61
            {
62
                 button1. Enabled=false;
63
                 ChooseTotalList();
64
                 //DisableNumericUpDown();
65
                 textBox2.Enabled = false;
66
67
                 button1. Enabled = false;
                 button3. Enabled = false;
68
                 button2. Enabled = true;
69
            }
70
71
72
            private async void button2_Click
                 (object sender, EventArgs e) // Run buttom
73
            {
74
                 button2.Enabled = false;
75
                 Algorithm algorithm =
76
                     new Algorithm (
77
                          (int) numeric Up Down 2. Value,
78
                          (int) numeric Up Down 3. Value,
79
                          (int) numeric Up Down 4. Value,
80
81
                          (int) numericUpDown5. Value,
                          totalList , judgeNumber);
82
                 if (numericUpDown4. Value
83
                     == numericUpDown5. Value)
84
85
                 {
                     vs= await Task.Run
86
                          (()=>algorithm. ExecuteAlgorithm1());
87
88
                 }
                 else
89
90
                     vs = await Task.Run
91
                          (()=>algorithm. ExecuteAlgorithm2());
92
93
```

```
//InsertToMdb (openFileDialog1.FileName);
94
                 //UpdateToMdb(openFileDialog1.FileName);
95
                 textBox3. Text = GetSeries2();
96
                 //textBox3.Enabled = false;
97
98
99
            }
100
101
            private void button3 Click
102
                 (object sender, EventArgs e) // File button
103
            {
104
105
                 openFileDialog1.InitialDirectory = "c:\\";
106
                 openFileDialog1. Filter
107
                     = "Database files (*.mdb)|*.mdb";
108
                 openFileDialog1.FilterIndex = 0;
109
                 openFileDialog1. RestoreDirectory = true;
110
111
112
                 if (openFileDialog1.ShowDialog()
113
                     = DialogResult.OK)
114
115
                 {
                     DisableNumericUpDown();
116
                     textBox1.Enabled = false;
117
                     button1. Enabled = true;
118
                     textBox2. Text = openFileDialog1. FileName;
119
                     CreateTableInToMdb
120
                          (openFileDialog1.FileName);
121
                 }
122
123
            }
124
125
126
            private void button4_Click
                 (object sender, EventArgs e)//Clear function
127
            {
128
                 InitializeFunctionForClear();
129
            }
130
131
```

```
132
             private void button5_Click
                 (object sender, EventArgs e)
133
                 //Open the file button
134
             {
135
                 Process proc = new Process();
136
                 proc. EnableRaisingEvents = false;
137
                 proc. StartInfo. FileName = openFileDialog1. FileName;
138
                 proc. Start();
139
             }
140
141
             private void button6 Click
142
143
                 (object sender, EventArgs e)
                 //Insert to database button
144
             {
145
                 InsertToMdb (openFileDialog1.FileName);
146
                 string[] arr = new string[3];
147
                 arr[0] = cnt. ToString();
148
                 arr[1] = vs.Count.ToString();
149
                 arr [2] = numericUpDown1. Value. ToString()
150
151
                     + numericUpDown2. Value. ToString() +
152
                      " " + numericUpDown3. Value. ToString()
153
154
                     + numericUpDown4. Value. ToString()
155
156
157
                     + numericUpDown5. Value. ToString();
                 ListViewItem itm=new ListViewItem(arr);
158
159
                 list View 1. Items. Add(itm);
                 cnt++;
160
             }
161
162
             private void button7 Click
163
                 (object sender, EventArgs e)//reset button
164
165
                 list View 1. Items. Clear ();
166
                 cnt = 1;
167
                 DeleteAllRecordFromMdb
168
                      (openFileDialog1.FileName);
169
```

```
170
171
             private void ChooseTotalList()
172
173
174
                 InitializeJudgeNumber ();
                 totalList.Clear();
175
                 var rand = new Random();
176
                 StringBuilder str = new StringBuilder();
177
                 for (int i = 1;
178
                       i <= numericUpDown2. Value; i++)
179
                 {
180
181
                      int randNumber = rand.Next
                           (1, (int)numericUpDown1.Value + 1);
182
                      i f
                         (!judgeNumber [randNumber])
183
                      {
184
                          judgeNumber [randNumber] = true;
185
                          totalList.Add(randNumber);
186
                          str. Append
187
                               (totalList[i - 1].ToString());
188
                          str.Append("");
189
                      }
190
                      else
191
192
                          i --;
193
194
195
                 textBox1.Text = str.ToString();
196
197
             private void InitializeJudgeNumber()
198
199
                 for (int i = 0; i < judgeNumber.Length; <math>i++)
200
                      judgeNumber[i] = false;
201
202
203
             private void InitializeFunctionForClear()
204
205
                 EnableNumericUpDown();
206
                 button1. Enabled = true;
207
```

```
208
                 textBox1.Enabled = true;
                 textBox1. Clear();
209
                 //textBox2.Enabled = true;
210
                 //textBox2. Clear();
211
212
                 textBox3. Clear();
                 button3. Enabled = true;
213
             }
214
215
216
             private void DisableNumericUpDown()
217
                 numericUpDown1. Enabled = false;
218
219
                 numericUpDown2. Enabled = false;
                 numericUpDown3. Enabled = false;
220
                 numericUpDown4. Enabled = false;
221
                 numericUpDown5. Enabled = false;
222
             }
223
224
             private void EnableNumericUpDown()
225
226
             {
                 numericUpDown1. Enabled = true;
227
                 numericUpDown2. Enabled = true;
228
                 numericUpDown3. Enabled = true;
229
                 numericUpDown4. Enabled = true;
230
                 numericUpDown5. Enabled = true;
231
232
             }
233
             private void CreateTableInToMdb
234
                 (string fileNameWithPath)
235
             {
236
237
                 try
                 {
238
                      OleDbConnection myConnection =
239
240
                          new OleDbConnection
                               ("Provider=Microsoft. Jet. OLEDB. 4.0; Data Source:
241
                               fileNameWithPath);
242
                      myConnection. Open();
243
                      OleDbCommand myCommand
244
                          = new OleDbCommand();
245
```

```
246
                     myCommand. Connection = myConnection;
                     myCommand.CommandText
247
                         = "CREATE TABLE my_table" +
248
                          "([order] NUMBER, "+
249
                          "[m] NUMBER, " +
250
                          "[n] NUMBER,
251
                          "[k] NUMBER,
252
                          "[j] Number, " +
253
                          "[s] NUMBER,
254
                          "[n numbers] TEXT, "+
255
                          "[minium number of sets] NUMBER,
256
                          "[answer] TEXT)";
257
                     myCommand.ExecuteNonQuery();
258
                     myCommand. Connection. Close();
259
260
                 catch { }
261
            }
262
263
            private void InsertToMdb(string fileNameWithPath)
264
265
                 var con = new OleDbConnection
266
                     ("Provider = Microsoft.Jet.OLEDB.4.0; Data Source = "
267
                     + fileNameWithPath);
268
                 var cmd = new OleDbCommand();
269
                 cmd. Connection = con;
270
                 cmd.CommandText = "insert into my_table"+
271
                     "([order],[m],[n],[k],[j],[s],"+
272
                     "[n numbers], [minium number of sets], [answer])"
273
                     +"values (@order, @m, @n, "+
274
                     "@k, @j, @s, @series1, @number, @answer);";
275
                 cmd. Parameters. AddWithValue("@order", cnt);
276
                 cmd. Parameters. AddWithValue
277
278
                     ("Om", numericUpDown1. Value);
                 cmd. Parameters. AddWithValue
279
                     ("On", numericUpDown2. Value);
280
                 cmd. Parameters. AddWithValue
281
                     ("@k", numericUpDown3. Value);
282
                 cmd. Parameters. AddWithValue
283
```

```
("@j", numericUpDown4. Value);
284
                 cmd. Parameters. AddWithValue
285
                      ("@s", numericUpDown5. Value);
286
                 cmd. Parameters. AddWithValue
287
                     ("@series1", series1Fordb());
288
                 cmd. Parameters. AddWithValue
289
                      ("@number", vs.Count());
290
                 cmd. Parameters. AddWithValue
291
                      ("@answer", series2Fordb());
292
                 con. Open();
293
                 cmd. ExecuteNonQuery();
294
295
                 con. Close ();
             }
296
297
             private void DeleteRecordFromMdb
298
                 (string fileNameWithPath, string num)
299
             {
300
                 int number = Int32.Parse(num);
301
                 var con = new OleDbConnection
302
                      ("Provider = Microsoft. Jet.OLEDB. 4.0;"+
303
                      "Data Source = " + fileNameWithPath);
304
                 var cmd = new OleDbCommand();
305
                 con. Open();
306
                 cmd.Connection = con;
307
                 cmd.CommandText = "DELETE FROM [my_table] " +
308
                      "WHERE [order]=" + number + "";
309
                 cmd.ExecuteNonQuery();
310
                 con. Close();
311
             }
312
313
             private void DeleteAllRecordFromMdb
314
                 (string fileNameWithPath)
315
316
                  var con = new OleDbConnection
317
                      ("Provider = Microsoft.Jet.OLEDB.4.0; "+
318
                      "Data Source = " + fileNameWithPath);
319
                 var cmd = new OleDbCommand();
320
                 con. Open();
321
```

```
322
                 cmd.Connection = con;
                 cmd.CommandText = "DELETE FROM [my_table] ";
323
                 cmd. ExecuteNonQuery();
324
325
                  con. Close ();
             }
326
327
             private string series1Fordb()
328
329
330
                  string series1 = "";
                  foreach (var num in totalList)
331
332
333
                      series1 += num. ToString();
                      series1 += "";
334
335
                  return series1;
336
             }
337
338
             private string series2Fordb()
339
340
                  string series 2 = "";
341
                  int index = 0;
342
                  foreach (var num in vs)
343
                  {
344
                      if (index != 0)
345
                           series2 += "; ";
346
                      for (int i = 0;
347
                           i < numericUpDown2. Value; i++)
348
                      {
349
                           if (((1 << i) \& num) != 0)
350
351
                               series2 +=
352
                                    totalList[i]. ToString();
353
                               series2 += "";
354
355
                      }
356
357
                      index++;
358
359
```

```
360
                  return series2;
             }
361
362
             private string GetSeries2()
363
364
                  string series 2 = "";
365
                  foreach (var num in vs)
366
367
368
                       for (int i = 0;
                           i < numericUpDown2. Value; i++)
369
                       {
370
                           if (((1 << i) \& num)!=0)
371
372
                           {
                                series2 +=
373
                                     totalList[i]. ToString();
374
                                series2 += "";
375
                           }
376
                       }
377
                       series2 += "\langle r \rangle n";
378
379
                  return series2;
380
             }
381
382
             private void listView1 SelectedIndexChanged
383
                  (object sender, EventArgs e)
384
             {
385
                  //var\ selectedItemText = "+
386
                  // "(listView1. SelectedItem ?? "(none)")
387
                  //. ToString();
388
                  //MessageBox.Show"
389
                  //+ "("Selected: " + selectedItem Text);
390
391
392
             }
393
             private void listView1 MouseDown
394
                  (object sender, MouseEventArgs e)
395
             {
396
                  if (listView1.SelectedItems.Count >= 1
397
```

```
&& e.Button=MouseButtons.Right)
398
                 {
399
                      ListViewItem item = listView1. SelectedItems [0];
400
401
402
        //here i check for the Mouse pointer location on click if its conta
        // in the actual selected item's bounds or not .
403
        // cuz i ran into a problem with the ui once because of that ...
404
                      if (item. Bounds. Contains (e. Location))
405
406
                      {
                          ContextMenu \ cm = new \ ContextMenu();
407
                          MenuItem menuItemForDelete
408
409
                              = new MenuItem();
                          menuItemForDelete. Text = "Delete";
410
                          menuItemForDelete.Click +=
411
                                new EventHandler
412
                                   (menuItemForDelete_Click);
413
                          cm. MenuItems. Add (menuItemForDelete);
414
                          listView1.ContextMenu = cm;
415
                      }
416
                 }
417
            }
418
419
             private void menuItemForDelete Click
420
                 (object sender, EventArgs e)
421
422
             {
423
                 var element = listView1. SelectedItems [0];
                 DeleteRecordFromMdb
424
                      (openFileDialog1.FileName,
425
                          element. SubItems [0]. Text);
426
                 list View 1. Items. Remove
427
                      (listView1.SelectedItems[0]);
428
429
430
             }
431
432
```

433