

Project in ME001 – Sampling system

Group 1

CHEN YUXUAN 1809853J-I011-0011 D1

WANG YUAN 1809853G-I011-0030 D1

HE PEILIN 1809853U-I011-0078 D1

December 2, 2020

Contents

1	Restatement of the Problem	3
2	Basic Ideas	3
2.1	$j = s$	3
2.1.1	Algorithm to Find Subsets	3
2.1.2	Calculate the Combination Number	5
2.1.3	Greedy Algorithm to Calculate the Set Covered	5
2.2	$j \neq s$	6
3	Essential Codes and Functions Analysis	7
3.1	Realization of Modifying DB files	7
3.2	Multi-Threading	9
4	User Guide	10
5	Program Test	15
6	Summary	17
	Appendices	18
A	Programs for Algorithms	18
B	Programs for GUI	28

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 ($45 \leq m \leq 54$), any n ($7 \leq n \leq 25$) 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 ($4 \leq k \leq 7$) 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 ($3 \leq s \leq 7$) samples have been selected from the j (where $s \leq j \leq k$) 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

5	4	3	2	1
1	1	1	1	1

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 $01011_2 = 11_{10}$

5	4	3	2	1
0	1	0	1	1

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.

```

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

```

1 void findSubsetOfkOptim(int n, int k, vector<int> subsetK){
2     int count=0; //number of 1s
3     for(int i = (1<<k)-1 ; i < (1<<n); i++){
4         for(int j = 0; j < n; j++){
```

```

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

```

1 int calculateCombinationNumber(int n, int m){
2     for (int i=0; i<=n; i++)
3         C[i][0]=1;
4     for (int i=1; i<=n; i++)
5         for (int j=1; j<=i; j++)
6             C[i][j]=C[i-1][j-1]+C[i-1][j];
7     return C[n][m];
8 }

```

2.1.3 Greedy Algorithm to Calculate the Set Covered

We denote that the input is a set \mathcal{U} of n elements, and a collection $S = \{S_1, S_2, \dots, S_m\}$ 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 $\mathcal{U} = \{1, 2, \dots, C_n^j\}$, 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 01011₂. 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).

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

1 public void InsertToMdb(string fileNameWithPath)
2 {
3     var con = new OleDbConnection(
4         "Provider = Microsoft.Jet.OLEDB.4.0; Data Source = "
5         + fileNameWithPath);
6     var cmd = new OleDbCommand();
```

```

7      cmd.Connection = con;
8      cmd.CommandText = "insert into my_table ([m],[n],[k],[j]," +
9                          "[s],[n numbers],[minium number of sets], [answer]) " +
10                         "values (@m, @n, @k,@j,@s,@series1, @number, @answer);";
11      cmd.Parameters.AddWithValue( "@m", numericUpDown1.Value );
12      cmd.Parameters.AddWithValue( "@n", numericUpDown2.Value );
13      cmd.Parameters.AddWithValue( "@k", numericUpDown3.Value );
14      cmd.Parameters.AddWithValue( "@j", numericUpDown4.Value );
15      cmd.Parameters.AddWithValue( "@s", numericUpDown5.Value );
16      cmd.Parameters.AddWithValue( "@series1", series1Fordb() );
17      cmd.Parameters.AddWithValue( "@number", vs.Count() );
18      cmd.Parameters.AddWithValue( "@answer", series2Fordb() );
19      con.Open();
20      cmd.ExecuteNonQuery();
21      con.Close();
22  }

```

```

1  private void DeleteRecordFromMdb
2      (string fileNameWithPath, string num)
3  {
4      int number = Int32.Parse(num);
5      var con = new OleDbConnection
6          ("Provider = Microsoft.Jet.OLEDB.4.0; Data Source = "
7           + fileNameWithPath);
8      var cmd = new OleDbCommand();
9      con.Open();
10     cmd.Connection = con;
11     cmd.CommandText = "DELETE FROM [my_table] " +
12                       "WHERE [order]= " + number + " ";
13     cmd.ExecuteNonQuery();
14     con.Close();
15 }
16
17 private void DeleteAllRecordFromMdb
18     (string fileNameWithPath)
19 {
20     var con = new OleDbConnection
21         ("Provider = Microsoft.Jet.OLEDB.4.0; Data Source = "

```



```

22         + fileNameWithPath);
23     var cmd = new OleDbCommand();
24     con.Open();
25     cmd.Connection = con;
26     cmd.CommandText = "DELETE FROM [my_table] ";
27     cmd.ExecuteNonQuery();
28     con.Close();
29 }

```

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 [button2_Click](#).

```

1 private async void button2_Click(object sender, EventArgs e)
2 // Run button
3 {
4     button2.Enabled = false;
5     Algorithm algorithm = new Algorithm(
6         (int)numericUpDown2.Value,
7         (int)numericUpDown3.Value,
8         (int)numericUpDown4.Value,
9         (int)numericUpDown5.Value,
10        totalList, judgeNumber);
11     if (numericUpDown4.Value == numericUpDown5.Value)
12     {
13         vs = await Task.Run(() => algorithm.ExecuteAlgorithm1());
14     }
15     else
16     {
17         vs = await Task.Run(() => algorithm.ExecuteAlgorithm2());
18     }
19     //InsertToMdb(openFileDialog1.FileName);
20     //UpdateToMdb(openFileDialog1.FileName);
21     textBox3.Text = GetSeries2();
22     //textBox3.Enabled = false;
23

```

24

25

}

4 User Guide

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.

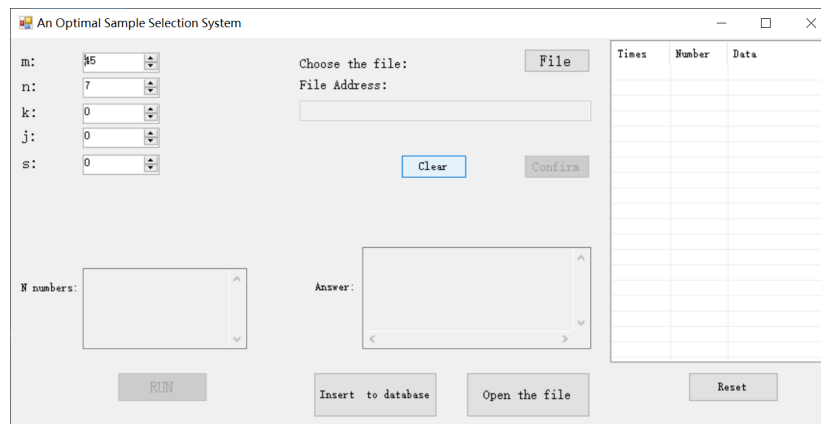


Figure. 4.0.1. Initial GUI

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

An Optimal Sample Selection System

m: 45
n: 12
k: 6
j: 6
s: 4

Choose the file: File
File Address: C:\Lab06\Database1.mdb
Clear Confirm

N numbers: 11 36 41 25 37 27 23 43
2 18 24 20

Answer:

Times Number Data

RUN Insert to database Open the file Reset

Figure. 4.0.5. Step4

An Optimal Sample Selection System

m: 45
n: 12
k: 6
j: 6
s: 4

Choose the file: File
File Address: C:\Lab06\Database1.mdb
Clear Confirm

N numbers: 11 36 41 25 37 27 23 43
2 18 24 20

Answer: 11 36 41 25 37 27
11 23 43 2 18 24
36 41 25 23 43 20
37 27 2 18 24 20
36 41 25 2 18 24
11 37 27 23 43 20

Times Number Data

RUN Insert to database Open the file Reset

Figure. 4.0.6. Step5

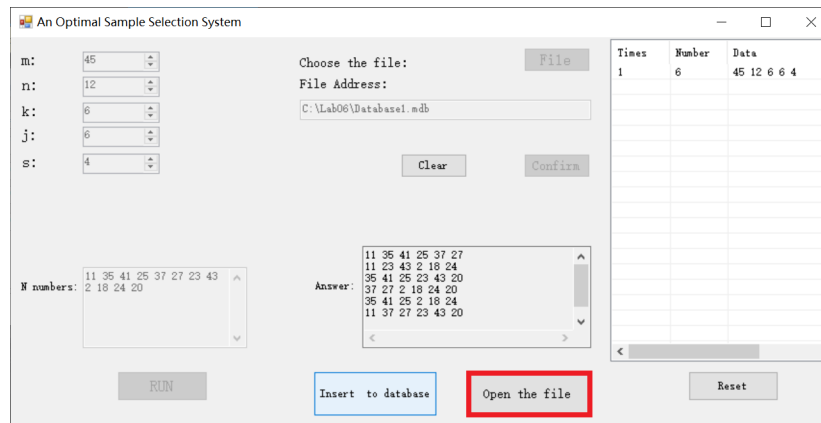


Figure. 4.0.7. Step6

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

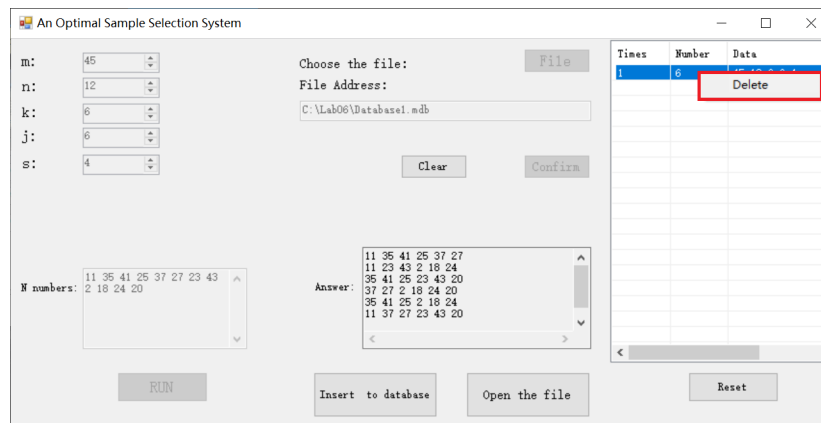


Figure. 4.0.8. Step7

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

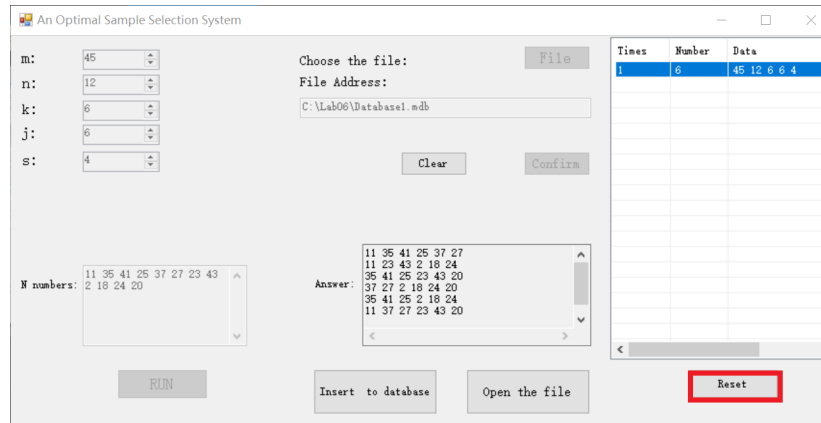


Figure. 4.0.9. Step8

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.

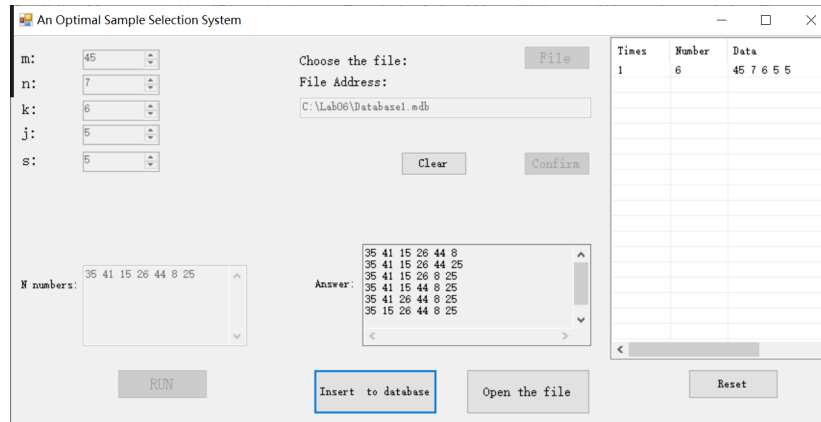


Figure. 5.0.1. E.g.1: Input the data: $m = 45, n = 7, k = 6, j = 5, s = 5$.

An Optimal Sample Selection System

m: 45
n: 8
k: 6
j: 6
s: 5

Choose the file: File
File Address: C:\Lab06\Database1.mdb
Clear Confirm

N numbers: 37 1 25 22 14 17 4 38
Answer: 37 1 25 22 14 17
37 1 25 22 4 38
37 1 14 17 4 38
37 25 22 14 17 4

RUN Insert to database Open the file Reset

Times	Number	Data
1	6	45 7 6 5 5
2	4	45 8 6 6 5

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

An Optimal Sample Selection System

m: 45
n: 10
k: 6
j: 6
s: 4

Choose the file: File
File Address: C:\Lab06\Database1.mdb
Clear Confirm

N numbers: 44 35 8 28 7 32 12 20 39 21
Answer: 44 35 8 28 7 32
44 35 12 20 39 21
8 28 7 32 12 20

RUN Insert to database Open the file Reset

Times	Number	Data
1	6	45 7 6 5 5
2	4	45 8 6 6 5
3	3	45 10 6 6 4

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

```
1 using System;
2 using System.Collections.Generic;
3 using System.Collections.Specialized;
4 using System.ComponentModel;
5 using System.ComponentModel.Design;
6 using System.Diagnostics;
7 using System.Drawing.Drawing2D;
8 using System.Drawing.Text;
9 using System.Linq;
10 using System.Security.Cryptography;
11 using System.Text;
12 using System.Threading.Tasks;
13
14 namespace Information_System_Project
15 {
16     public class Algorithm
17     {
18         private int n;
19         private int k;
20         private int j;
21         private int s;
22         private List<int> setNumberForK
23             = new List<int>();
24         private List<int> setNumberForJ
25             = new List<int>();
26         private Dictionary<int, int> dictionaryForAllSets
27             = new Dictionary<int, int>();
28         private Dictionary<int, int>
29             dictionaryForAllSets2
30             = new Dictionary<int, int>();
31         private bool[] visit = new bool[10000000];
32         private List<int> totalList = new List<int>();
```

```

33     private Queue<int> queueForSet
34         = new Queue<int>();
35     private bool[] judgeNumber = new bool[46];
36     private int[,] C = new int[26,26];
37     public Algorithm(int n, int k, int j,
38         int s, List<int> totalList,
39         bool[] judgeNumber)
40     //class of the Algorithm
41     {
42         this.n = n;
43         this.k = k;
44         this.j = j;
45         this.s = s;
46         this.totalList = totalList;
47         this.judgeNumber = judgeNumber;
48     }
49     public Queue<int>
50         ExecuteAlgorithm1()//j==s
51     {
52         GreedyAlgorithm();//j==s algorithm
53         return queueForSet;
54     }
55     public Queue<int>
56         ExecuteAlgorithm2()//j!=s
57     {
58         GreedyAlgorithm2();//j!=s algorithm
59         return queueForSet;
60     }
61     private void GreedyAlgorithm()
62     //the main body of algorithm
63     {
64         setNumberForK
65             = CombinationForAllNum(n, k);
66         //find all the k sets
67         int max;
68         int now = 0;
69         int allNum
70             = TotalNumberForJ(n, j);

```

```

71         // find the number of j sets
72         int node;
73         int index;
74         List<int> vis = new List<int>();
75         List<int> result = new List<int>();
76         while (allNum > 0)
77         //Greedy Cover algorithm
78         {
79             //Debug.WriteLine(allNum);
80             max = 0;
81             node = 0;
82             index = 0;
83             foreach (var element in setNumberForK)
84             {
85                 int numOfUnfound = 0;
86                 for (int j1 = (1 << s) - 1;
87                     j1 <= element; j1++)
88                     //find j sets in k sets
89                 {
90                     if ((j1 & element) != j1)
91                     //j sets don't the subsets of k sets
92                     {
93                         continue;
94                     }
95                     int cnt = 0;
96                     for (int k1 = 0; k1 < n; k1++)
97                     {
98                         if (cnt > s)
99                             break;
100                         if ((j1 & (1 << k1)) != 0)
101                         {
102                             cnt++;
103                         }
104                     } //find how many 1s the set has
105                     if (cnt == s
106                         && !dictionaryForAllSets.
107                             ContainsKey(j1))
108                         //find the subset hasn't found

```

```

109         {
110             numOfUnfound++;
111             vis.Add(j1);
112         }
113     }
114     if (max < numOfUnfound)
115 //find the maximum number of j sets that the k sets cover
116     {
117         node = index;
118         max = numOfUnfound;
119         result.Clear();
120         foreach (var eachNum in vis)
121         {
122             result.Add(eachNum);
123         }
124     }
125     vis.Clear();
126     index++;
127 }
128 queueForSet.Enqueue(setNumberForK[node]);
129 setNumberForK.RemoveAt(node);
130 foreach (var eachNum in result)
131 {
132     dictionaryForAllSets[eachNum]
133         = ++now;
134 }
135 allNum -= max;
136
137 }
138
139 }
140 private void GreedyAlgorithm2()
141 {
142     setNumberForK=CombinationForAllNum(n, k);//
143     int max;
144     int now = 0;
145     int allNum = TotalNumberForJ(n, j);
146     int node;

```

```

147         int index;
148         List<int> vis = new List<int>();
149         List<int> result = new List<int>();
150         while (allNum > 0)
151         {
152             max = 0;
153             node = 0;
154             index = 0;
155             //Debug.WriteLine(allNum + " " + dictionaryForAllSets2.Count());
156             foreach (var element in setNumberForK)
157             {
158                 int numOfUnfound = 0;
159                 //int origin = dictionaryForAllSets2.Count();
160                 //var value = dictionaryForAllSets2.Count();
161                 for (int j1 = (1 << s) - 1; j
162                     1 <= element; j1++)
163                 {
164                     if ((j1 & element) != j1)
165                     {
166                         continue;
167                     }
168                     int cnt = 0;
169                     var answer = 0;
170                     for (int k1 = 0; k1 < n; k1++)
171                     {
172                         if (cnt > s)
173                             break;
174                         if ((j1 & (1 << k1)) != 0)
175                         {
176                             cnt++;
177                         }
178                     }
179                     if (cnt == s &&
180                         !dictionaryForAllSets.
181                             ContainsKey(j1))
182                     {
183                         int num = j - s;
184                         numOfSetContainj1

```

```

185         (0, j1, num, ref answer);
186         numOfUnfound += answer;
187         vis.Add(j1);
188     }
189 }
190 foreach (var eachNum in setNumberForJ)
191 {
192     dictionaryForAllSets2.
193         Remove(eachNum);
194 }
195 setNumberForJ.Clear();
196 if (max < numOfUnfound)
197 {
198     max = numOfUnfound;
199     node = index;
200     result.Clear();
201     foreach (var eachNum in vis)
202     {
203         result.Add(eachNum);
204     }
205 }
206 index++;
207 vis.Clear();
208 }
209 //Debug.WriteLine(max);
210 SetDictionary2(result);
211 queueForSet.Enqueue(setNumberForK[node]);
212 foreach (var eachNum in result)
213 {
214     //Debug.WriteLine(eachNum + " ");
215     dictionaryForAllSets[eachNum]
216         = ++now;
217 }
218 setNumberForK.RemoveAt(node);
219 allNum -= max;
220 }
221 Debug.WriteLine(allNum);
222 }

```

```

223     private void numOfSetContainj1
224         (int node, int j1,
225         int num, ref int answer)
226     {
227         if (num == 0 &&
228             !dictionaryForAllSets2.ContainsKey(j1))
229         {
230             answer += 1;
231             setNumberForJ.Add(j1);
232             dictionaryForAllSets2[j1] =
233                 dictionaryForAllSets2.Count() + 1;
234             return;
235         }
236         else if (num == 0
237             && dictionaryForAllSets2.ContainsKey(j1))
238             return;
239         for(int i1 = node; i1 < n; i1++)
240         {
241             if(((1<<i1) & j1) == 0)
242             {
243                 numOfSetContainj1
244                     (i1+1, j1 |
245                     (1 << i1), num - 1, ref answer);
246             }
247         }
248     }
249
250     private void SetDictionary2(List<int> result)
251     {
252         foreach(var eachNum in result)
253         {
254             FindEachElement(0, eachNum, j-s);
255         }
256     }
257
258     private void FindEachElement
259         (int node, int element, int num)
260     {

```



```

261         if (num == 0 &&
262             !dictionaryForAllSets2.
263             ContainsKey(element))
264         {
265             dictionaryForAllSets2[element]
266             = dictionaryForAllSets2.Count() + 1;
267             return;
268         }
269         else if (num == 0 &&
270             dictionaryForAllSets2.
271             ContainsKey(element))
272             return;
273         for (int i1 = node; i1 < n; i1++)
274         {
275             if (((1 << i1) & element) == 0)
276             {
277                 FindEachElement
278                     (i1, element |
279                     (1 << i1), num - 1);
280             }
281         }
282     }
283
284     private List<int> CombinationForAllNum
285         (int n, int k)
286     {
287         List<int> Combination = new List<int>();
288         for (int i = (1<<(k))-1; i < (1 << n); i++)
289         {
290             var cnt = 0;
291             for (int j1 = 0; j1 < n; j1++)
292             {
293                 if ((i & (1 << j1)) != 0)
294                 {
295                     cnt++;
296                 }
297             }
298             if (cnt == k)

```

```

299         {
300             Combination.Add(i);
301             //myBV.Add(new BitVector32(i));
302         }
303     }
304     return Combination;
305 }
306 private int TotalNumberForJ
307     (int n,int j)
308     //calculate the combination number
309     {
310         for(int i = 0; i <= n; i++)
311         {
312             for(int m = 0; m <= j; m++)
313             {
314                 C[i , m] = 0;
315             }
316         }
317         for(int i = 0; i <= n; i++)
318         {
319             C[i , 0] = 1;
320             if (i == n && j == 0)
321                 return C[n, j];
322         }
323         for (int i = 1; i <= n; i++)
324         {
325             for (int m = 1; m <= i; m++)
326             {
327                 C[i , m]
328                     = C[i - 1, m - 1]
329                     + C[i-1, m];
330                 if (i == n && m == j)
331                     return C[n, m];
332             }
333         }
334         return C[n, j];
335     }
336

```

```

337     private void Dfs
338         (int start ,int setNum ,
339          int currentNumber ,
340          int totalNum ,
341          ref int result )
342     {
343
344         int now = currentNumber ;
345         List<int> vis = new List<int> ();
346         if (setNum >= result )
347             return ;
348         if (currentNumber == totalNum )
349             {
350
351                 //if (setNum < result)
352                 result = setNum ;
353                 return ;
354             }
355         for (int i = start ;
356             i < setNumberForK.Count ; i++)
357             {
358                 if (!visit [ i ])
359                     {
360                         for (int j1 = (1 << s) - 1 ;
361                             j1 <= setNumberForK [ i ] ; j1++)
362                             {
363                                 int cnt = 0 ;
364                                 for (int k1 = 0 ; k1 < n ; k1++)
365                                     {
366                                         if ((j1 & (1 << k1)) != 0)
367                                             {
368                                                 cnt++ ;
369                                             }
370                                     }
371                                 if ((j1 & setNumberForK [ i ]) == j1
372                                    && cnt == s &&
373                                    !dictionaryForAllSets
374                                        .ContainsKey (j1)) //hh

```

```

375         {
376             now++;
377             dictionaryForAllSets[j1]
378                 = now;
379             vis.Add(j1);
380
381         }
382     }
383     //Debug.WriteLine(" ");
384     Dfs(i + 1, setNum + 1,
385         now, totalNum, ref result);
386     visit[i] = false;
387     now = currentNumber;
388
389 }
390 foreach (var eachNum in vis)
391 {
392     dictionaryForAllSets.Remove(eachNum);
393 }
394 vis.Clear();
395 }
396 return ;
397 }
398 }
399 }

```

B Programs for GUI

source/Form1.cs

```

1 using System;
2 using System.Collections.Generic;
3 using System.ComponentModel;
4 using System.Data;
5 using System.Data.OleDb;
6 using System.Diagnostics;
7 using System.Drawing;
8 using System.IO;
9 using System.Linq;

```

```

10 using System.Text;
11 using System.Threading;
12 using System.Threading.Tasks;
13 using System.Windows.Forms;
14
15 namespace Information_System_Project
16 {
17     public partial class Form1 : Form
18     {
19         int cnt = 1;
20         Queue<int> vs;
21         List<int> totalList = new List<int>();
22         bool[] judgeNumber = new bool[55];
23         OpenFileDialog openFileDialog1
24             = new OpenFileDialog();
25
26         public Form1()
27         {
28             InitializeComponent();
29             listView1.View = View.Details;
30             listView1.GridLines = true;
31             listView1.FullRowSelect = true;
32             listView1.Columns.Add("Times", 60);
33             listView1.Columns.Add("Number", 60);
34             listView1.Columns.Add("Data", 244);
35         }
36
37
38         private void numericUpDown3_ValueChanged
39             (object sender, EventArgs e)
40         {
41             numericUpDown3.Maximum
42                 = numericUpDown2.Value;
43         }
44
45
46         private void numericUpDown4_ValueChanged
47             (object sender, EventArgs e)

```

```

48     {
49         numericUpDown4.Maximum
50         = numericUpDown3.Value;
51     }
52
53     private void numericUpDown5_ValueChanged
54         (object sender, EventArgs e)
55     {
56         numericUpDown5.Maximum
57         = numericUpDown4.Value;
58     }
59
60     private void button1_Click
61         (object sender, EventArgs e)
62     //Confirm button
63     {
64         button1.Enabled=false;
65         ChooseTotalList();
66         //DisableNumericUpDown();
67         textBox2.Enabled = false;
68         button1.Enabled = false;
69         button3.Enabled = false;
70         button2.Enabled = true;
71     }
72
73     private async void button2_Click
74         (object sender, EventArgs e)\
75     // Run button
76     {
77         button2.Enabled = false;
78         Algorithm algorithm = new Algorithm
79             ((int)numericUpDown2.Value,
80             (int)numericUpDown3.Value,
81             (int)numericUpDown4.Value,
82             (int)numericUpDown5.Value,
83             totalList, judgeNumber);
84         if (numericUpDown4.Value
85             == numericUpDown5.Value)

```

```

86         {
87             vs= await Task.Run
88                 (()=>algorithm.ExecuteAlgorithm1());
89         }
90         else
91         {
92             vs = await Task.Run
93                 (()=>algorithm.ExecuteAlgorithm2());
94         }
95         //InsertToMdb(openFileDialog1.FileName);
96         //UpdateToMdb(openFileDialog1.FileName);
97         textBox3.Text = GetSeries2();
98         //textBox3.Enabled = false;
99
100
101     }
102
103     private void button3_Click
104         (object sender, EventArgs e)
105         // File button
106     {
107
108         openFileDialog1.InitialDirectory = "c:\\ ";
109         openFileDialog1.Filter
110             = "Database files (*.mdb)|*.mdb";
111         openFileDialog1.FilterIndex = 0;
112         openFileDialog1.RestoreDirectory = true;
113
114
115         if (openFileDialog1.ShowDialog()
116             == DialogResult.OK)
117         {
118             DisableNumericUpDown();
119             textBox1.Enabled = false;
120             button1.Enabled = true;
121             textBox2.Text = openFileDialog1.FileName;
122             CreateTableInToMdb
123                 (openFileDialog1.FileName);

```

```

124         }
125
126     }
127
128     private void button4_Click
129         (object sender, EventArgs e)
130     //Clear function
131     {
132         InitializeFunctionForClear();
133     }
134
135     private void button5_Click
136         (object sender, EventArgs e)
137     //Open the file button
138     {
139         Process proc = new Process();
140         proc.EnableRaisingEvents = false;
141         proc.StartInfo.FileName
142             = openFileDialog1.FileName;
143         proc.Start();
144     }
145
146     private void button6_Click
147         (object sender, EventArgs e)
148     //Insert to database button
149     {
150         InsertToMdb(openFileDialog1.FileName);
151         string[] arr = new string[3];
152         arr[0] = cnt.ToString();
153         arr[1] = vs.Count.ToString();
154         arr[2] =
155             numericUpDown1.Value.ToString() +
156             " " + numericUpDown2.Value.ToString() +
157             " " + numericUpDown3.Value.ToString() +
158             " " + numericUpDown4.Value.ToString() +
159             " " + numericUpDown5.Value.ToString();
160         ListViewItem itm=new ListViewItem(arr);
161         listView1.Items.Add(itm);

```



```

162         cnt++;
163     }
164
165     private void button7_Click
166         (object sender, EventArgs e) //reset button
167     {
168         listView1.Items.Clear();
169         cnt = 1;
170         DeleteAllRecordFromMdb
171             (openFileDialog1.FileName);
172     }
173
174     private void ChooseTotalList()
175     {
176         InitializeJudgeNumber();
177         totalList.Clear();
178         var rand = new Random();
179         StringBuilder str = new StringBuilder();
180         for (int i = 1;
181             i <= numericUpDown2.Value; i++)
182         {
183             int randNumber =
184                 rand.Next
185                     (1,
186                     (int)numericUpDown1.Value + 1);
187             if (!judgeNumber[randNumber])
188             {
189                 judgeNumber[randNumber] = true;
190                 totalList.Add(randNumber);
191                 str.Append
192                     (totalList[i - 1].ToString());
193                 str.Append(" ");
194             }
195             else
196             {
197                 i--;
198             }
199         }

```

```

200         textBox1.Text = str.ToString();
201     }
202     private void InitializeJudgeNumber()
203     // set all the number in the totallist to 0
204     {
205         for (int i = 0; i < judgeNumber.Length; i++)
206             judgeNumber[i] = false;
207     }
208
209     private void InitializeFunctionForClear()
210     {
211         EnableNumericUpDown();
212         button1.Enabled = true;
213         textBox1.Enabled = true;
214         textBox1.Clear();
215         //textBox2.Enabled = true;
216         //textBox2.Clear();
217         textBox3.Clear();
218         button3.Enabled = true;
219     }
220
221     private void DisableNumericUpDown()
222     {
223         numericUpDown1.Enabled = false;
224         numericUpDown2.Enabled = false;
225         numericUpDown3.Enabled = false;
226         numericUpDown4.Enabled = false;
227         numericUpDown5.Enabled = false;
228     }
229
230     private void EnableNumericUpDown()
231     {
232         numericUpDown1.Enabled = true;
233         numericUpDown2.Enabled = true;
234         numericUpDown3.Enabled = true;
235         numericUpDown4.Enabled = true;
236         numericUpDown5.Enabled = true;
237     }

```

```

238
239     private void CreateTableInToMdb
240         (string fileNameWithPath)
241         //create the table in the database
242     {
243         try
244         {
245             OleDbConnection myConnection =
246                 new OleDbConnection
247                 ( "Provider=Microsoft.Jet.OLEDB.4.0; "
248                   + "Data Source=" + fileNameWithPath );
249             myConnection.Open();
250             OleDbCommand myCommand
251                 = new OleDbCommand();
252             myCommand.Connection
253                 = myConnection;
254             myCommand.CommandText =
255                 "CREATE TABLE my_table" +
256                 "([order] NUMBER, "+
257                 "[m] NUMBER, " +
258                 "[n] NUMBER, " +
259                 "[k] NUMBER, " +
260                 "[j] Number, " +
261                 "[s] NUMBER, " +
262                 "[n numbers] TEXT, " +
263                 "[minium number of sets] NUMBER, " +
264                 "[answer] TEXT)";
265             myCommand.ExecuteNonQuery();
266             myCommand.Connection.Close();
267         }
268         catch { }
269     }
270
271     private void InsertToMdb
272         (string fileNameWithPath)
273         //insert infomation in the database
274     {
275         var con = new OleDbConnection

```

```

276         ( "Provider = Microsoft.Jet.OLEDB.4.0;" +
277         " Data Source = "
278         + fileNameWithPath );
279     var cmd = new OleDbCommand();
280     cmd.Connection = con;
281     cmd.CommandText = "insert into my_table " +
282     "([order],[m],[n],[k],[j],[s],[n numbers]," +
283     "[minium number of sets],[answer]) " +
284     "values (@order, @m, @n, @k,@j,@s,@series1," +
285     " @number, @answer);";
286     cmd.Parameters.AddWithValue( "@order", cnt );
287     cmd.Parameters.AddWithValue
288     ( "@m", numericUpDown1.Value );
289     cmd.Parameters.AddWithValue
290     ( "@n", numericUpDown2.Value );
291     cmd.Parameters.AddWithValue
292     ( "@k", numericUpDown3.Value );
293     cmd.Parameters.AddWithValue
294     ( "@j", numericUpDown4.Value );
295     cmd.Parameters.AddWithValue
296     ( "@s", numericUpDown5.Value );
297     cmd.Parameters.AddWithValue
298     ( "@series1", series1Fordb() );
299     cmd.Parameters.AddWithValue
300     ( "@number", vs.Count() );
301     cmd.Parameters.AddWithValue
302     ( "@answer", series2Fordb() );
303     con.Open();
304     cmd.ExecuteNonQuery();
305     con.Close();
306 }
307
308 private void DeleteRecordFromMdb
309     (string fileNameWithPath,string num)
310     //delete records in the database
311 {
312     int number = Int32.Parse(num);
313     var con = new OleDbConnection

```

```

314         ( "Provider = Microsoft.Jet.OLEDB.4.0;" +
315         " Data Source = "
316         + fileNameWithPath );
317     var cmd = new OleDbCommand();
318     con.Open();
319     cmd.Connection = con;
320     cmd.CommandText = "DELETE FROM [my_table] " +
321         "WHERE [order]=" + number + " ";
322     cmd.ExecuteNonQuery();
323     con.Close();
324 }
325
326 private void DeleteAllRecordFromMdb
327 (string fileNameWithPath)
328 //delete all the records in the database
329 {
330     var con = new OleDbConnection
331         ( "Provider = Microsoft.Jet.OLEDB.4.0;" +
332         " Data Source = "
333         + fileNameWithPath );
334     var cmd = new OleDbCommand();
335     con.Open();
336     cmd.Connection = con;
337     cmd.CommandText = "DELETE FROM [my_table] ";
338     cmd.ExecuteNonQuery();
339     con.Close();
340 }
341
342 private string series1Fordb()
343 // print the numbers choose from m
344 {
345     string series1 = "";
346     foreach (var num in totalList)
347     {
348         series1 += num.ToString();
349         series1 += " ";
350     }
351     return series1;

```

```

352     }
353
354     private string series2Fordb()
355     //print the result into database
356     {
357         string series2 = "";
358         int index = 0;
359         foreach (var num in vs)
360         {
361             if (index != 0)
362                 series2 += "; ";
363             for (int i = 0;
364                 i < numericUpDown2.Value; i++)
365             {
366                 if (((1 << i) & num) != 0)
367                 {
368                     series2 +=
369                         totalList[i].ToString();
370                     series2 += " ";
371                 }
372             }
373
374             index++;
375         }
376         return series2;
377     }
378
379     private string GetSeries2()
380     // print the result in the textbox
381     {
382         string series2 = "";
383         foreach (var num in vs)
384         {
385             for (int i = 0;
386                 i < numericUpDown2.Value; i++)
387             {
388                 if (((1 << i) & num) != 0)
389                 {

```

```

390         series2 +=
391             totalList[i].ToString();
392         series2 += " ";
393     }
394 }
395     series2 += "\r\n";
396 }
397     return series2;
398 }
399
400     private void listView1_SelectedIndexChanged
401         (object sender, EventArgs e)
402     {
403         //var selectedItemText = (listView1.SelectedItem ?? "(none)").ToString();
404         //MessageBox.Show("Selected: " + selectedItemText);
405     }
406
407     private void listView1_MouseDown
408         (object sender, MouseEventArgs e)
409         //click the right button of the mouse
410     {
411         if (listView1.SelectedItems.Count >= 1
412             && e.Button==MouseButtons.Right)
413         {
414             ListViewItem item =
415                 listView1.SelectedItems[0];
416
417             //here i check for the Mouse pointer location on click if its contained
418             // in the actual selected item's bounds or not .
419             // cuz i ran into a problem with the ui once because of that ..
420             if (item.Bounds.Contains(e.Location))
421             {
422                 ContextMenu cm = new ContextMenu();
423                 MenuItem menuItemForDelete
424                     = new MenuItem();
425                 menuItemForDelete.Text = "Delete";
426                 menuItemForDelete.Click +=

```

```

428         new EventHandler
429             (menuItemForDelete_Click);
430         cm.MenuItems.Add(menuItemForDelete);
431         listView1.ContextMenu = cm;
432     }
433 }
434 }
435
436 private void menuItemForDelete_Click
437     (object sender, EventArgs e) //set the display button
438 {
439     var element = listView1.SelectedItems[0];
440     DeleteRecordFromMdb
441         (openFileDialog1.FileName,
442         element.SubItems[0].Text);
443     listView1.Items.Remove
444         (listView1.SelectedItems[0]);
445 }
446 }
447 }
448
449 }

```