

Project in ME001 – Sampling system Group 1

By 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	Steps to Run the Program	10
5	Program Test	15
6	Summary	17
	Appendices	18
A	Programs for Algorithms	18
B	Programs for GUI	27

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 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.

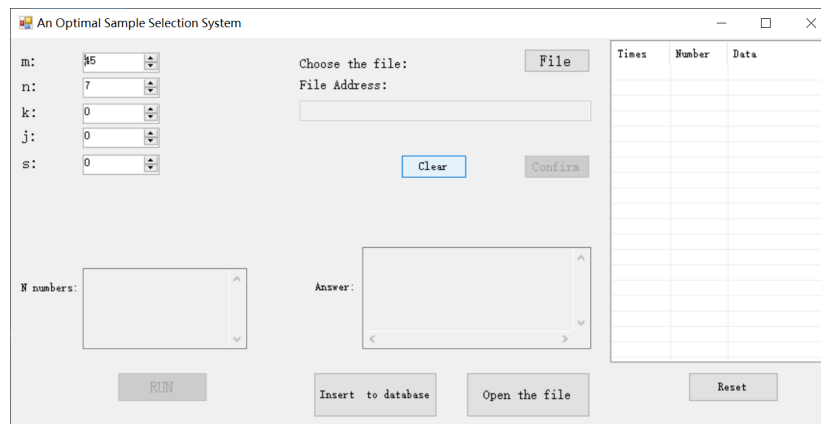


Figure. 4.0.1. Initial GUI

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

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

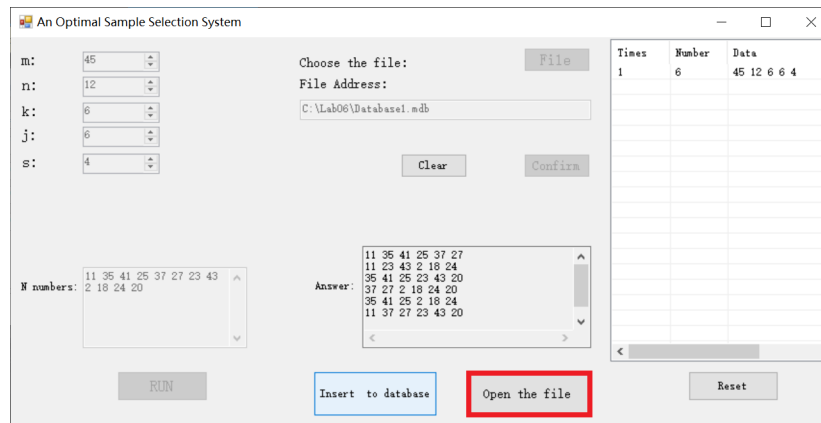


Figure. 4.0.7. Step6

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

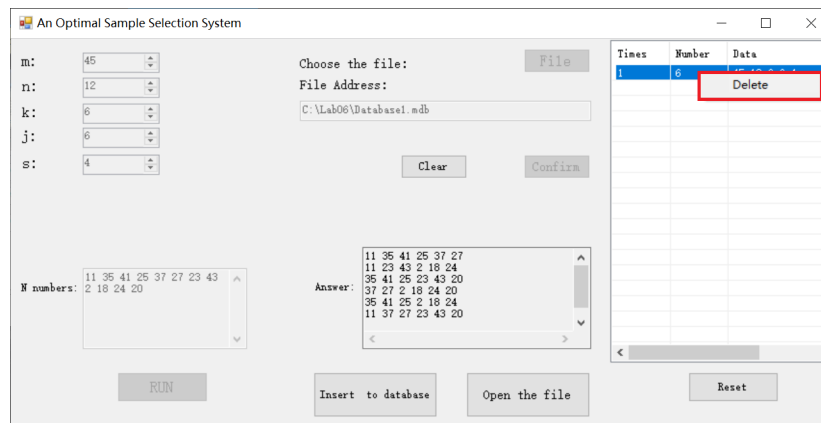


Figure. 4.0.8. Step7

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

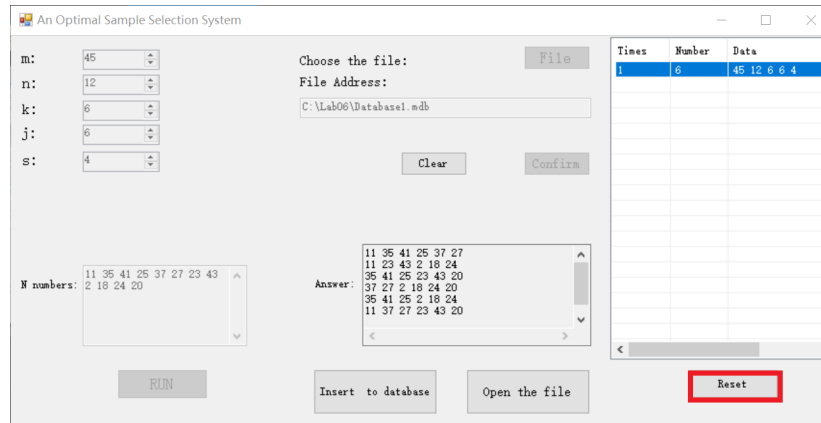


Figure. 4.0.9. Step8

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.

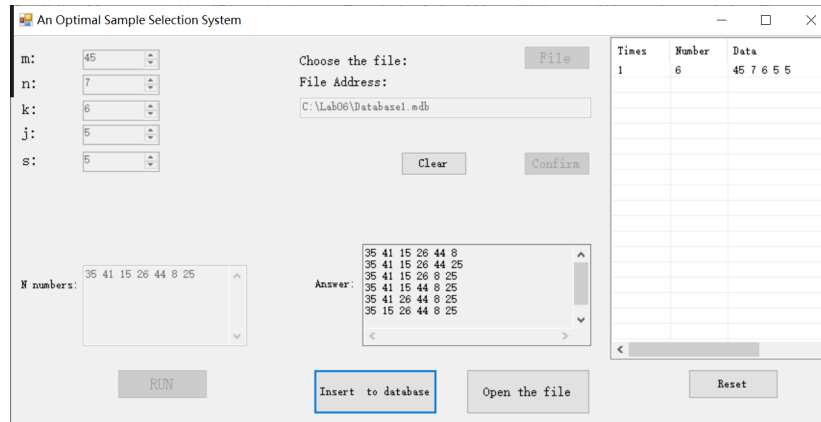


Figure. 5.0.1. E.g.1: Input the data: $m = 45, n = 7, k = 6, j = 5, s = 5$.

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

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

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 = new List<int>();
23         private List<int> setNumberForJ = new List<int>();
24         private Dictionary<int, int> dictionaryForAllSets
25             = new Dictionary<int, int>();
26         private Dictionary<int, int> dictionaryForAllSets2
27             = new Dictionary<int, int>();
28         private bool[] visit = new bool[10000000];
29         private List<int> totalList = new List<int>();
30         private Queue<int> queueForSet = new Queue<int>();
31         private bool[] judgeNumber = new bool[46];
32         private int[,] C = new int[26, 26];
```

```

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

```

```

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

```

```

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

```

```

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

```

```

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

```

```

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

```



```

261         }
262         if (cnt == k)
263         {
264             Combination.Add(i);
265             //myBV.Add(new BitVector32(i));
266         }
267     }
268     return Combination;
269 }
270 private int TotalNumberForJ(int n, int j)
271 {
272     for (int i = 0; i <= n; i++)
273     {
274         for (int m = 0; m <= j; m++)
275         {
276             C[i, m] = 0;
277         }
278     }
279     for (int i = 0; i <= n; i++)
280     {
281         C[i, 0] = 1;
282         if (i == n && j == 0)
283             return C[n, j];
284     }
285     for (int i = 1; i <= n; i++)
286     {
287         for (int m = 1; m <= i; m++)
288         {
289             C[i, m] = C[i - 1, m - 1] + C[i - 1, m];
290             if (i == n && m == j)
291                 return C[n, m];
292         }
293     }
294     return C[n, j];
295 }
296
297 private void Dfs(int start, int setNum,
298                 int currentNumber, int totalNum, ref int result)

```

```

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

```

```

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

```

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) // Confirm button
62     {
63         button1.Enabled=false;
64         ChooseTotalList();
65         //DisableNumericUpDown();
66         textBox2.Enabled = false;
67         button1.Enabled = false;
68         button3.Enabled = false;
69         button2.Enabled = true;
70     }
71
72     private async void button2_Click
73         (object sender, EventArgs e) // Run button
74     {
75         button2.Enabled = false;
76         Algorithm algorithm =
77             new Algorithm(
78                 (int)numericUpDown2.Value,
79                 (int)numericUpDown3.Value,
80                 (int)numericUpDown4.Value,
81                 (int)numericUpDown5.Value,
82                 totalList, judgeNumber);
83         if (numericUpDown4.Value
84             == numericUpDown5.Value)
85         {
86             vs= await Task.Run
87                 (()=>algorithm.ExecuteAlgorithm1());
88         }
89         else
90         {
91             vs = await Task.Run
92                 (()=>algorithm.ExecuteAlgorithm2());
93         }

```

```

94         //InsertToMdb(openFileDialog1.FileName);
95         //UpdateToMdb(openFileDialog1.FileName);
96         textBox3.Text = GetSeries2();
97         //textBox3.Enabled = false;
98
99
100     }
101
102     private void button3_Click
103         (object sender, EventArgs e) // File button
104     {
105
106         openFileDialog1.InitialDirectory = "c:\\ ";
107         openFileDialog1.Filter
108             = "Database files (*.mdb)|*.mdb";
109         openFileDialog1.FilterIndex = 0;
110         openFileDialog1.RestoreDirectory = true;
111
112
113         if (openFileDialog1.ShowDialog()
114             == DialogResult.OK)
115         {
116             DisableNumericUpDown();
117             textBox1.Enabled = false;
118             button1.Enabled = true;
119             textBox2.Text = openFileDialog1.FileName;
120             CreateTableInToMdb
121                 (openFileDialog1.FileName);
122         }
123
124     }
125
126     private void button4_Click
127         (object sender, EventArgs e) //Clear function
128     {
129         InitializeFunctionForClear();
130     }
131

```

```

132     private void button5_Click
133         (object sender, EventArgs e)
134         //Open the file button
135     {
136         Process proc = new Process();
137         proc.EnableRaisingEvents = false;
138         proc.StartInfo.FileName = openFileDialog1.FileName;
139         proc.Start();
140     }
141
142     private void button6_Click
143         (object sender, EventArgs e)
144         //Insert to database button
145     {
146         InsertToMdb(openFileDialog1.FileName);
147         string [] arr = new string [3];
148         arr[0] = cnt.ToString();
149         arr[1] = vs.Count.ToString();
150         arr[2] = numericUpDown1.Value.ToString()
151             + " "
152             + numericUpDown2.Value.ToString() +
153             " " + numericUpDown3.Value.ToString()
154             + " "
155             + numericUpDown4.Value.ToString()
156             + " "
157             + numericUpDown5.Value.ToString();
158         ListViewItem itm=new ListViewItem(arr);
159         listView1.Items.Add(itm);
160         cnt++;
161     }
162
163     private void button7_Click
164         (object sender, EventArgs e)//reset button
165     {
166         listView1.Items.Clear();
167         cnt = 1;
168         DeleteAllRecordFromMdb
169             (openFileDialog1.FileName);

```

```

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

```



```

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

```

```

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

```

```

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

```

```

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

```

```

360         return series2;
361     }
362
363     private string GetSeries2()
364     {
365         string series2 = "";
366         foreach (var num in vs)
367         {
368             for (int i = 0;
369                 i < numericUpDown2.Value; i++)
370             {
371                 if (((1 << i) & num) != 0)
372                 {
373                     series2 +=
374                         totalList[i].ToString();
375                     series2 += " ";
376                 }
377             }
378             series2 += "\r\n";
379         }
380         return series2;
381     }
382
383     private void listView1_SelectedIndexChanged
384         (object sender, EventArgs e)
385     {
386         //var selectedItemText = "+
387         //(listView1.SelectedItem ?? "(none)")
388         //.ToString();
389         //MessageBox.Show"
390         //+"("Selected: " + selectedItemText);
391     }
392
393     private void listView1_MouseDown
394         (object sender, MouseEventArgs e)
395     {
396         if (listView1.SelectedItems.Count >= 1

```

```

398         && e.Button==MouseButtons.Right)
399     {
400         ListViewItem item = listView1.SelectedItems[0];
401
402         //here i check for the Mouse pointer location on click if its conta
403         // in the actual selected item's bounds or not .
404         // cuz i ran into a problem with the ui once because of that ..
405         if (item.Bounds.Contains(e.Location))
406         {
407             ContextMenu cm = new ContextMenu();
408             MenuItem menuItemForDelete
409                 = new MenuItem();
410             menuItemForDelete.Text = "Delete";
411             menuItemForDelete.Click +=
412                 new EventHandler
413                     (menuItemForDelete_Click);
414             cm.MenuItems.Add(menuItemForDelete);
415             listView1.ContextMenu = cm;
416         }
417     }
418 }
419
420 private void menuItemForDelete_Click
421     (object sender, EventArgs e)
422 {
423     var element = listView1.SelectedItems[0];
424     DeleteRecordFromMdb
425         ( openFileDialog1.FileName ,
426           element.SubItems[0].Text );
427     listView1.Items.Remove
428         ( listView1.SelectedItems[0] );
429 }
430 }
431 }
432
433 }

```