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

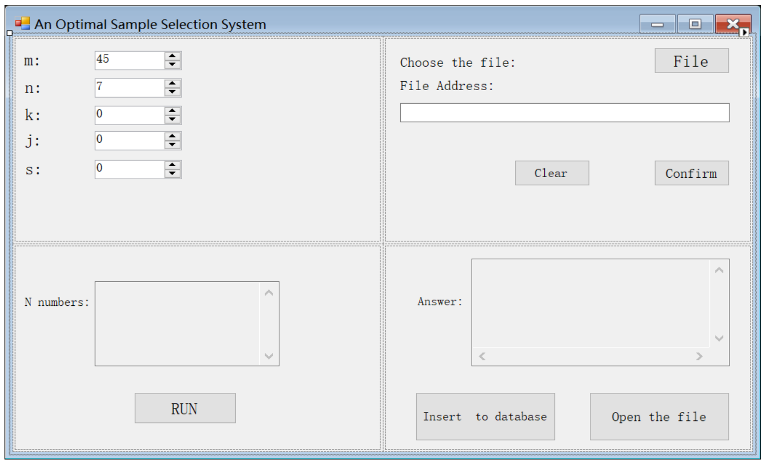
December 2, 2020

Contents

1 Steps to run the program

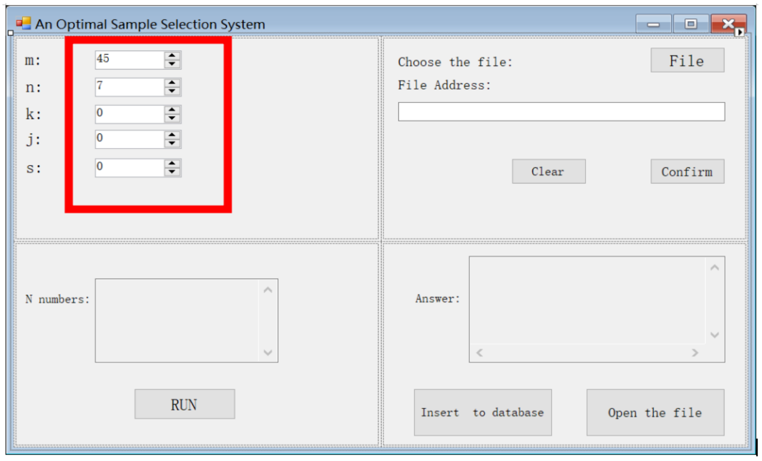
In order to make the operation more smooth, all the program environment and settings are completed and included in the file package. Just required to follow the steps below to run the program.

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

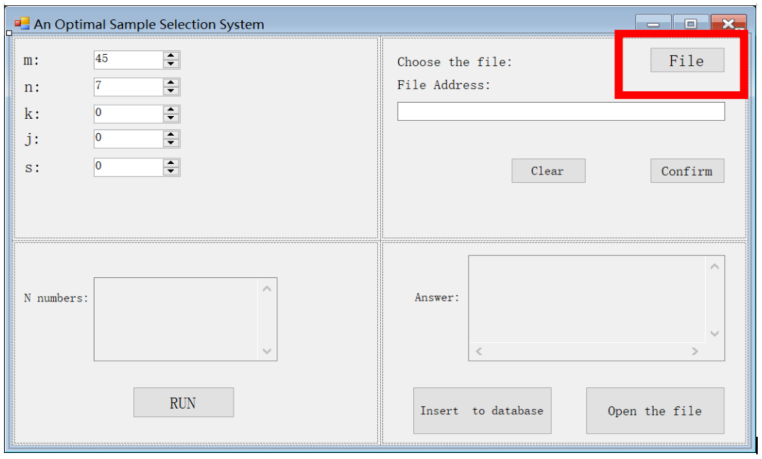


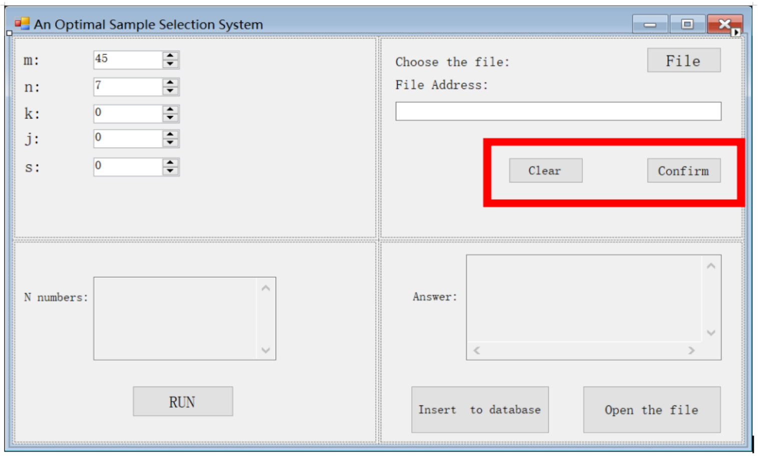
2. In order to record the relevant output data of the program and facilitate display and modification later. It is required to have a .mdb file to store it, which is called —.mdb in project package.

3. Choose the data of each parameter and input on the program surface.

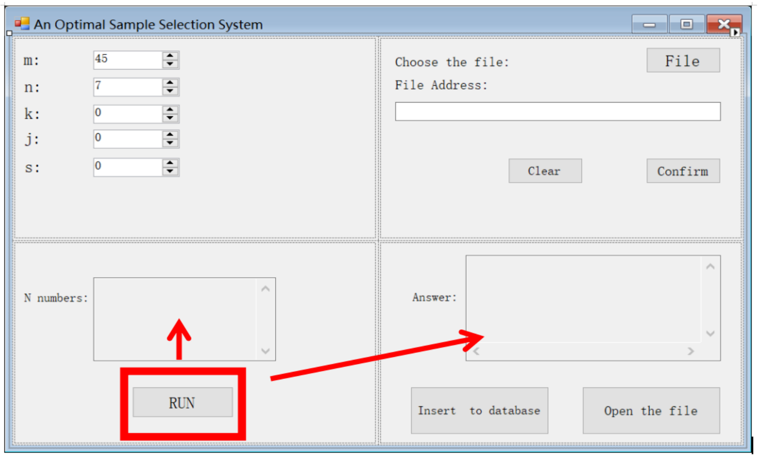


4. Choose the DB file to store and operate the data, click the 'File' and choose the —.mdb in the previous step and 'Confirm' if all get right.(*'clear' is a function that clear all the data you have input, include the parameter in step 3*)



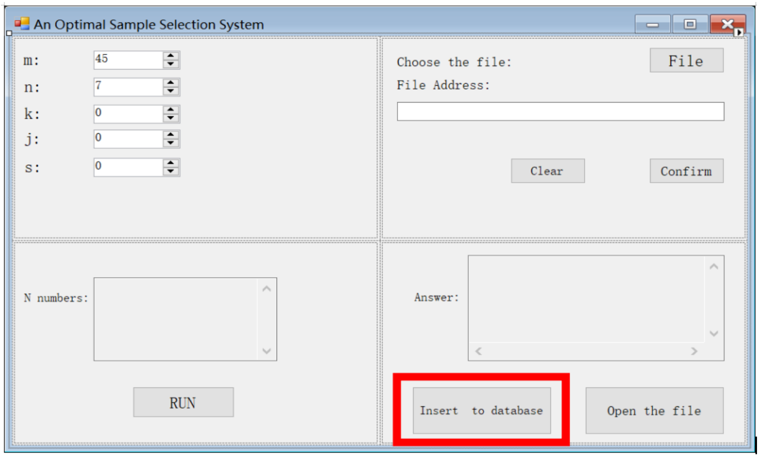


5. Push the 'RUN' button and the N number and final answer of your input will be shown on the surface window, you can check the answer after that.

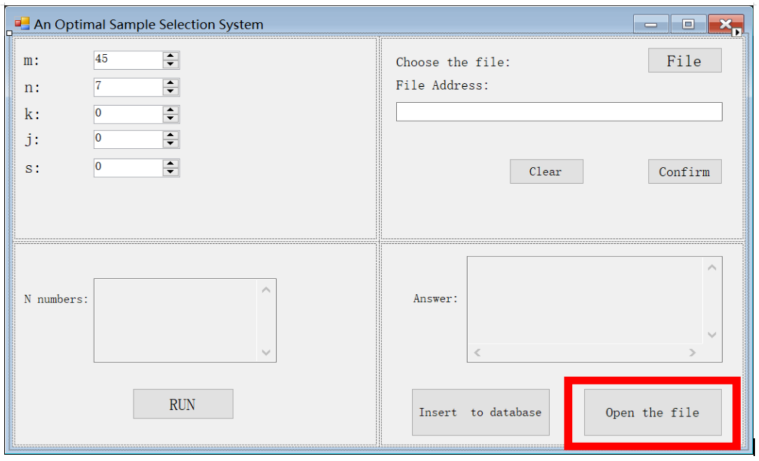


m: 45
n: 7
k: 0
j: 0
s: 0

6. After confirm the data is correct, use 'Insert to database' to download the data on the DB file(—.mdb), and 'Open the file' can open it to display the data you have calculate. It is also easy for you to delete or use any other operation on the data though your DB file.



m: 45
n: 7
k: 0
j: 0
s: 0

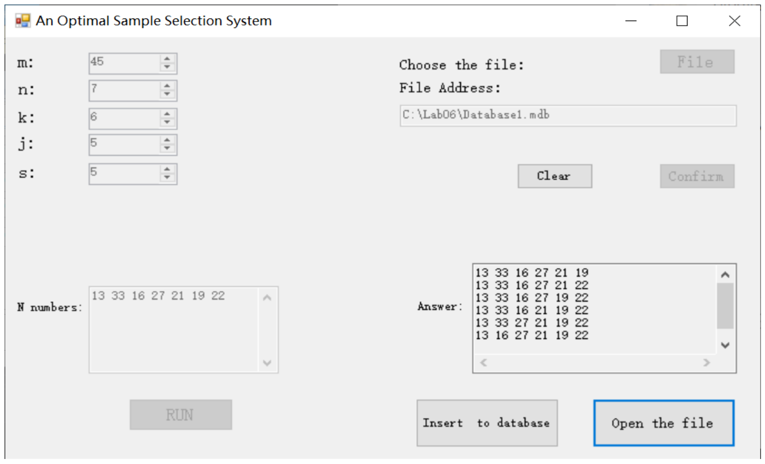


m: 45
n: 7
k: 0
j: 0
s: 0
Choose the file:
File Address:
N numbers:
Answer:

2 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, 4, and 6 as the demo of our program.

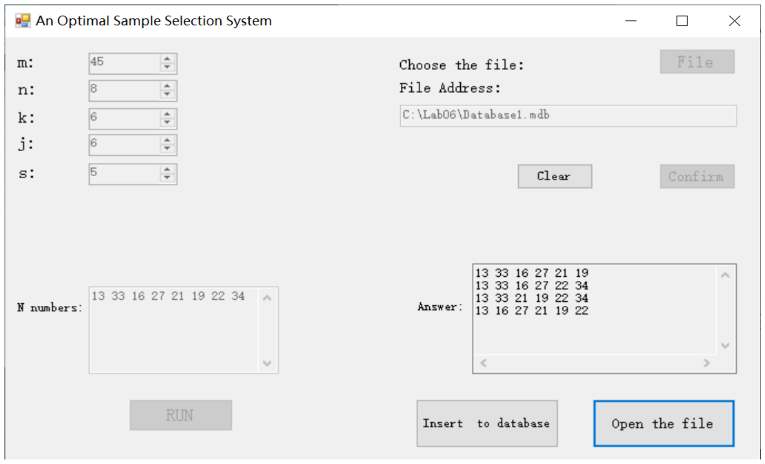
- E.g.1: Input the data: $m = 45, n = 7, k = 6, j = 5, s = 5$.



m: 45
n: 7
k: 6
j: 5
s: 5
Choose the file:
File Address: C:\Lab06\Database1.mdb
N numbers: 13 33 16 27 21 19 22
Answer:

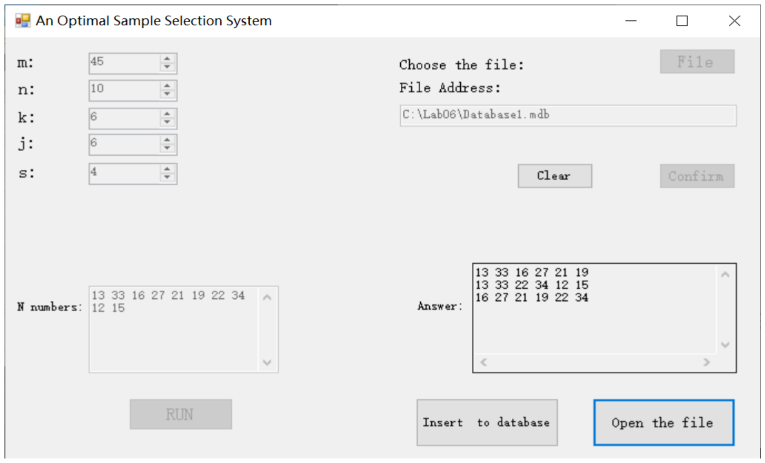
```
13 33 16 27 21 19
13 33 16 27 21 22
13 33 16 27 19 22
13 33 16 21 19 22
13 33 27 21 19 22
13 16 27 21 19 22
```

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



```
m: 45
n: 8
k: 6
j: 6
s: 5
Choose the file:
File Address:
C:\Lab06\Database1.mdb
N numbers: 13 33 16 27 21 19 22 34
Answer: 13 33 16 27 21 19
        13 33 16 27 22 34
        13 33 21 19 22 34
        13 16 27 21 19 22
```

- E.g.6: Input the data: $m = 45, n = 10, k = 6, j = 6, s = 4$.



```
m: 45
n: 10
k: 6
j: 6
s: 4
Choose the file:
File Address:
C:\Lab06\Database1.mdb
N numbers: 13 33 16 27 21 19 22 34
           12 15
Answer: 13 33 16 27 21 19
        13 33 22 34 12 15
        16 27 21 19 22 34
```

3 Basic ideas

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

3.1 $j = s$

1. **Lemma one: 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 orign set to a container, and then we label every element to one(illustrate the picture below). We assume that the orign set is S, $S = \{1, 2, 3, 4, 5\}$:

5	4	3	2	1
1	1	1	1	1

  Then, the subset which has the same element with the orignal set's, I'll label the element to 1, otherwise I'll label it to 0. For example, we suppose that one the subset is S_1,$S_1 = \{1, 2, 4\}$. We can represent it as below:

5	4	3	2	1
0	1	0	1	1

  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 $2^n - 1$(orignal set). Just like the example above set S can be represented by $11111_2 = 31_{10}$ and S_1 can be expressed as $01011_2 = 11_{10}$

- Now, 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 above, $S_1 = \{1, 2, 4\}$:

5	4	3	2	1
0	1	0	1	1

  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 illustrates below:

```
void findSubsetOfk(int n,int k, vector<int> subsetK){
    int count=0;//number of 1s
    for(int i = 1 ; i < (1<<n); i++){
        for(int j = 0; j < n; j++){
            //the binary number representation
            //of subset has an 1 on the jth position
            if(i & (1<<j)!=0){
                count++;
            }
        }
        if(count==k)
            susetK.empalce_back(i);
        count=0;
    }

}
```

But,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 is below:

```
void findSubsetOfk(int n,int k, vector<int> subsetK){
    int count=0;//number of 1s
    for(int i = (1<<k)-1 ; i < (1<<n); i++){
        for(int j = 0; j < n; j++){
            //the binary number representation
            //of subset has an 1 on the jth position
            if(i & (1<<j)!=0){
                count++;
            }
        }
        if(count==k)
            susetK.empalce_back(i);
        count=0;
    }

}
```

- 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. **Lemma two: Calculate the Combination number**
   If we calculate the combination number directly, it is likely to out of bounds of int. So we can use this combination formula below:

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

   to calculate the combination number. The code is below:

```
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];
}
```

3. **Lemma three: Greedy algorithm to calculate the set covergae**
   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 possible from S such that their union covers $\mathcal{U}$. We can solve this problem easily by greedy algorithm. The algorithm is below:

```
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
```

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.

3.2 $j \neq s$

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_2. Therefore, we can expand it as below:

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

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

4 Essential codes and functions analysis

5 Pros and cons of the program

1. We adopt multithreading 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 stucked.

```
private async void button2_Click(object sender, EventArgs e)// Rur
{
    button2.Enabled = false;
    Algorithm algorithm = new Algorithm((int)numericUpDown2.Value,
        (int)numericUpDown4.Value, (int)numericUpDown5.Value, tota
    if (numericUpDown4.Value == numericUpDown5.Value)
    {
        vs= await Task.Run(()=>algorithm.ExecuteAlgorithm1());
    }
    else
    {
        vs = await Task.Run(()=>algorithm.ExecuteAlgorithm2());
    }
    //InsertToMdb(openFileDialog1.FileName);
    //UpdateToMdb(openFileDialog1.FileName);
    textBox3.Text = GetSeries2();
    //textBox3.Enabled = false;


}
```

6 Summary