Федеральное государственное бюджетное образовательное учреждение высшего образования
«Сибирский государственный университет телекоммуникаций и информатики»
(СибГУТИ)

Институт информатики и вычислительной техники

09.03.01 "Информатика и вычислительная техника"
профиль "Программное обеспечение средств вычислительной техники и автоматизированных систем"

Кафедра прикладной математики и кибернетики

Курсовая работа по дисциплине
Алгоритмы и вычислительные методы оптимизации

Симплекс-метод

Вариант ____

Выполнил:

студент гр.ИП-_______ ______________/_______________/

ФИО студента

«__» _____________ 2023 г.

Проверил

______________/_______________/

ФИО преподавателя

«__» _____________ 2023 г. Оценка__________________

Новосибирск 2023 г.

Оглавление

Задание на курсовую работу

Написать программу, решающую задачу линейного программирования (далее – ЗЛП), представленной в канонической форме, симплекс-методом, используя в качестве начальной угловой точки опорное решение, найденное методом Жордана-Гаусса, согласно номеру в таблице рейтинга по дисциплине “Алгоритмы и вычислительные методы оптимизации”.

На вход программе подаются данные ЗЛП, заданной в канонической форме (считываются из файла в виде матрицы размера (m+1)x(n+1)):

$$\begin{cases} a_{11}x_1 + a_{12}x_2 + \cdots + a_{1n}x_n = b_1 \\ \dots\dots\dots\dots\dots\dots\dots\dots\dots\dots\dots\dots\dots\dots\dots \\ a_{11}x_1 + a_{12}x_2 + \cdots + a_{1n}x_n = b_n \\ \qquad x_1, x_2, \dots, x_n \geq 0 \end{cases}$$

$$Z = c_1x_1 + c_2x_2 + \cdots + c_nx_n \to max$$

Описание и формулы используемых методов

Для решения ЗЛП, описанной в задании на курсовую работу, необходимо в программе будет реализовать нахождения начального опорного решения с помощью метода Жордана-Гаусса, а также найти оптимальный план с помощью симплекс-метода.

Метод Жордана-Гаусса заключается в нахождении базисных переменных в расширенной матрице системы ограничений $\bar{A}$, используя жордановы исключения.

Во время преобразований расширенной матрицы $\bar{A}$ с помощью жордановых исключений в столбце предполагаемой базисной переменной происходит поиск не нулевого элемента столба – разрешающего элемента. Строка расширенной матрицы с разрешающим элементом делится на разрешающий элемент, столбец – зануляется, кроме разрешающего элемента, а все остальные элементы вычисляются по правилу прямоугольника, где $a_{ij} -$ это элемент матрицы в строке i в столбце j, $k -$ строка разрешающего элемента, $s -$ столбец разрешающего элемента.

$$a_{ij} = a_{ij} - \frac{a_{is} * a_{kj}}{a_{ks}}$$

либо $a_{ij} = a_{ij} - a_{is} * a'_{kj}$, $a'_{kj} = \frac{a_{kj}}{a_{ks}}$ – элемент после деления строки k на a_{ks}

В ходе жордановых исключений могут возникнуть следующие ситуации:

1. Получена строка, состоящая из нулей, кроме последнего коэффициента (правой части уравнения). В этом случае система не имеет решения.
2. Ранг матрицы A равен количеству неизвестных n $(r = n)$. Тогда система имеет единственное решение.
3. Ранг матрицы A меньше количеству неизвестных n $(r < n)$. Тогда система имеет бесконечно много решений.

Так как на переменные наложено условие неотрицательности, то в качестве базисного решения методом Жордана-Гаусса необходимо брать любое опорное решение – базисное решение, у которого все последние коэффициенты (правых частей системы уравнений) больше нуля.

Для нахождения опорного решение будет применён алгоритм перебора r различных переменных из n переменных (сочетания из n по r без повторений) с проверкой на неотрицательность коэффициентов правой части для базисных решений.

В найденном начальном опорном решении необходимо выразить базисные переменные через свободные и подставить полученные уравнения в целевую функции Z. Коэффициенты полученной матрицы $\bar{A}$ и коэффициенты свободных переменных со знаком минус новой целевой функции Z, а также её значение при нулевом значении свободных переменных заносится в симплекс-таблицу.

Нахождение оптимального опорного решения симплекс-методом заключается в переходе от одного опорного решения к другому изменением базисной переменной без "ухудшения" значения целевой функции: увеличения при поиске минимума, уменьшения при поиске максимума.

Средством хранения информации в симплекс-методе является симплекс-таблица, содержание которой было описано выше. Алгоритм этого метода заключается в том, что если в Z-строке имеется отрицательный коэффициент (в случае поиска максимума), то опорное решение является не оптимальным. В качестве новой базисной переменной необходимо брать переменную, соответствующую столбцу с максимальным по модулю отрицательным коэффициентом в Z-строке и заменять ею базисную переменную в строке с минимальным не отрицательным симплексным отношением, используя жорданово исключение с преобразованием Z-строки включительно. Этот процесс необходимо повторять до тех пор, пока все коэффициенты не станут не отрицательными.

В ходе поиска оптимального плана, столбец разрешающего элемента может не иметь симплексного отношения, удовлетворяющего вышесказанным условиям. В таком случае говорят, что "Система не ограничена сверху" и её значение максимально.

Если в Z-строке с оптимальным опорным решением есть нулевые коэффициенты для столбцов свободных переменных, оптимальных решений бесконечно много и различные оптимальные опорные решения, найденные симплекс-методом, создают множество оптимальных решений.

Пусть имеются 2 различных оптимальных опорных решения X^1 и X^2, тогда множество опорных решений X^* будет принадлежать отрезку от решения X^1 до решения X^2 будет равно:

$$X^* = \lambda X^1 + (1-\lambda)X^2 = \lambda(X^1 - X^2) + X^2, 0 \leq \lambda \leq 1$$

Если будет ещё оптимальное опорное решение X^3, то множество можно будет найти, использовав в качестве опорное решения множество для X^1 и X^2 $(X^{**} = \gamma X^* + (1-\gamma)X^3 = \gamma\lambda(X^1 - X^2) + \gamma(X^2 - X^3) + X^3)$

Приводя к общему виду множество, созданное l-ным количеством оптимальных опорных решения будет иметь вид:

$$X^* = m_1 m_2 \dots m_{l-1}(X^1 - X^2) + m_2 \dots m_{l-1}(X^2 - X^3) + \dots + m_{l-1}(X^{l-1} - X^l) + X^l$$

Список используемой литературы и интернет-источников

При выполнении данной курсовой работы был использован только лекционный материал дисциплины “Алгоритмы и вычислительные методы оптимизации” и навыки разработки ПО на языке C++ в интегрированной среде разработки “Dev-Cpp”, приобретённые за время обучения в университете.

Результат работы программы

```
Z=(0/1) + (2/1)x2 -> max

      1/1            2/1            3/1       |       5/1
      2/1            5/1            7/1       |      11/1

Матрица системы с опорным базисом
      1/1            0/1            1/1       |       3/1
      0/1            1/1            1/1       |       1/1


Z=(2/1) + (-2/1)x3;

x1=3/1 + (1/1)x3;
x2=1/1 + (1/1)x3;


 б.п. |         1        |        x1             x2             x3        | С.О.
---------------------------------------------------------------------------
 x1   |        3/1       |        1/1            0/1            1/1       |
 x2   |        1/1       |        0/1            1/1            1/1       |
---------------------------------------------------------------------------
  Z   |        2/1       |        0/1            0/1            2/1       |


Zmax=Z((3/1),(1/1),(0/1))=2/1


--------------------------------
Process exited with return value 0
Press any key to continue . . .
```

Рис.1 Пример выполнения программы с лёгкой ЗЛП

```
Z=(0/1) + (5/1)x1 + (6/1)x2 -> max

      1/1            2/1       |       3/1
      2/1            4/1       |       5/1

      1/1            2/1       |       3/1
      0/1            0/1       |      -1/1

System hasn't solution

--------------------------------
Process exited with return value 0
Press any key to continue . . .
```

Рис.2 Пример выполнения работы программы с ЗЛП, не имеющей решения

```
Z=(0/1) + (5/1)x4 -> max

      1/1            0/1            0/1           -2/1       |       5/1
      5/1            1/1            0/1          -10/1       |      31/1
      3/1            2/1            1/1          -29/1       |      84/1

Матрица системы с опорным базисом
      1/1            0/1            0/1           -2/1       |       5/1
      0/1            1/1            0/1            0/1       |       6/1
      0/1            0/1            1/1          -23/1       |      57/1


Z=(0/1) + (5/1)x4;

x1=5/1 + (-2/1)x4;
x2=6/1 + (0/1)x4;
x3=57/1 + (-23/1)x4;


 б.п. |         1        |        x1             x2             x3             x4        | С.О.
------------------------------------------------------------------------------------------
 x1   |        5/1       |        1/1            0/1            0/1           -2/1       |
 x2   |        6/1       |        0/1            1/1            0/1            0/1       |
 x3   |       57/1       |        0/1            0/1            1/1          -23/1       |
------------------------------------------------------------------------------------------
  Z   |        0/1       |        0/1            0/1            0/1           -5/1       |


System has infinity value;


--------------------------------
Process exited with return value 0
Press any key to continue . . .
```

Рис. 3 Пример работы программы с ЗЛП, не ограниченной сверху

```
Z=(0/1) + (-2/1)x1 + (8/1)x2 -> max

     -1/1            4/1            1/1            0/1            0/1        |       8/1
      2/1           -1/1            0/1            1/1            0/1        |       4/1
      1/1            1/1            0/1            0/1           -1/1        |       1/1

Матрица системы с опорным базисом
      1/1            0/1            1/7            4/7            0/1        |      24/7
      0/1            1/1            2/7            1/7            0/1        |      20/7
      0/1            0/1            3/7            5/7            1/1        |      37/7


Z=(16/1) + (-2/1)x3 + (0/1)x4;

x1=24/7 + (1/7)x3 + (4/7)x4;
x2=20/7 + (2/7)x3 + (1/7)x4;
x5=37/7 + (3/7)x3 + (5/7)x4;


 б.п. |        1         |        x1             x2             x3             x4             x5         | С.О.
-----------------------------------------------------------------------------------------------------------------
 x1   |       24/7       |       1/1            0/1            1/7            4/7            0/1        |
 x2   |       20/7       |       0/1            1/1            2/7            1/7            0/1        |
 x5   |       37/7       |       0/1            0/1            3/7            5/7            1/1        |
-----------------------------------------------------------------------------------------------------------------
  Z   |       16/1       |       0/1            0/1            2/1            0/1            0/1        |


 б.п. |        1         |        x1             x2             x3             x4             x5         | С.О.
-----------------------------------------------------------------------------------------------------------------
 x4   |        6/1       |       7/4            0/1            1/4            1/1            0/1        |
 x2   |        2/1       |      -1/4            1/1            1/4            0/1            0/1        |
 x5   |        1/1       |      -5/4            0/1            1/4            0/1            1/1        |
-----------------------------------------------------------------------------------------------------------------
  Z   |       16/1       |       0/1            0/1            2/1            0/1            0/1        |


Zmax=Z((24/7)*m1+(0/1),(6/7)*m1+(2/1),(0/1)*m1+(0/1),(-6/1)*m1+(6/1),(30/7)*m1+(1/1))=16/1


--------------------------------
Process exited with return value 0
Press any key to continue . . .
```

Рис. 4 Пример работы программы с ЗЛП, имеющей бесконечно много решений

Исходный код программы

```
#include<stdio.h>
#include<stdlib.h>
#include<iostream>
#include<vector>
#include<math.h>
#include<fstream>

using namespace std;

long long gcd(long long a, long long b)
{
	if (a % b == 0) return b;
	if (b % a == 0) return a;
	if (a > b) return gcd(a%b, b);
	return gcd(a, b%a);
}

long long lcm(long long a,long long b)
{
	return (a*b)/gcd(a,b);
}

int Next_SetNum(vector<long long> &setNum, int p, int n)
{
	int k=setNum.size();
	setNum[p]++;
	for(int i=p+1;i<k;i++) setNum[i]=setNum[i-1]+1;
	if(setNum[k-1]==n-1) return p-1;
	return k-1;
}

class simp_frac
{
public:
	long long num;
	long long den;

	simp_frac(): num(0), den(1) {}
	simp_frac(long long _num): num(_num), den(1) {}
	simp_frac(long long _num, long long _den): num(_num), den(_den) {}

	simp_frac operator+(simp_frac other)
	{
		long long LCM=lcm(den,other.den);
		long long Num=num*LCM/den+other.num*LCM/other.den;
		if(Num==0) return simp_frac(0,1);
		else
		{
			long long GCD=gcd(fabs(Num),LCM);
			return simp_frac(Num/GCD,LCM/GCD);
		}
	}
	simp_frac operator-(simp_frac other)
	{
		long long LCM=lcm(den,other.den);
		long long Num=num*LCM/den-other.num*LCM/other.den;
		if(Num==0) return simp_frac(0,1);
		else
		{
			long long GCD=gcd(fabs(Num),LCM);
```

```
                return simp_frac(Num/GCD,LCM/GCD);
            }
        }
        simp_frac operator*(simp_frac other)
        {
            if(num==0 || other.num==0) return simp_frac(0);
            long long GCD_a=gcd(fabs(num),fabs(other.den));
            long long GCD_b=gcd(fabs(other.num),fabs(den));
            return
simp_frac((num/GCD_a)*(other.num/GCD_b),(den/GCD_b)*(other.den/GCD_a));
        }
        simp_frac operator/(simp_frac other)
        {
            if(num==0) return simp_frac(0);
            long long GCD_a=gcd(fabs(num),fabs(other.num));
            long long GCD_b=gcd(fabs(other.den),fabs(den));
            long long _num=num;
            if(other.num<0)
            {
                _num*=-1;
                other.num*=-1;
            }
            return
simp_frac((_num/GCD_a)*(other.den/GCD_b),(den/GCD_b)*(other.num/GCD_a));
        }
        bool operator<(simp_frac other)
        {
            long double d1=num*other.den, d2=other.num*den;
            if(d1<d2) return true;
            else return false;
        }
};

void print_frac(simp_frac x)
{
        printf("%8lli/%-8lli",x.num,x.den);
}

void Print_Matr(vector< vector<simp_frac> > A, vector<simp_frac> B)
{
        for(long long i=0;i<A.size();i++)
        {
            for(long long j=0;j<A[i].size();j++) print_frac(A[i][j]);
            cout<<" | ";
            print_frac(B[i]);
            cout<<endl;
        }
        cout<<endl;
}

void Print_Simp_Table(vector< vector<simp_frac> > A, vector<simp_frac> B,vec-
tor<simp_frac> Z, vector<long long> setNum)
{
        printf(" á.ï. | %8c%-9i | ",' ',1);
        for(long long j=0;j<A[0].size();j++) printf("%8c%-9lli",'x',j+1);
        printf(" | Ñ.Î. ");
        cout<<"\n"<<string(14+18*(A[0].size()+1),'-')<<"\n";
        for(long long i=0;i<setNum.size();i++)
        {
            printf(" x%-4lli| ",setNum[i]+1);
            print_frac(B[i]);
            cout<<" | ";
```

```
		for(long long j=0;j<A[i].size();j++) print_frac(A[i][j]);
		cout<<" | ";
		cout<<endl;
	}
	cout<<string(14+18*(A[0].size()+1),'-')<<"\n";
	printf("  Z   | ");
	print_frac(Z[Z.size()-1]);
	cout<<" | ";
	for(long long j=0;j<Z.size()-1;j++)
	{
		Z[j].num=-Z[j].num;
		print_frac(Z[j]);
	}
	cout<<" | \n\n\n";
}

void Calculate_Matr(vector< vector<simp_frac> > &A, vector<simp_frac> &B,
long long k, long long s)
{
	simp_frac temp=A[k][s];
	simp_frac temp2;
	for(long long j=0;j<A[k].size();j++)
	{
		A[k][j]=A[k][j]/temp;
	}
	B[k]=B[k]/temp;
	for(long long i=0;i<A.size();i++)
	{
		if(i==k) continue;
		temp2=A[i][s];
		for(long long j=0;j<A[k].size();j++)
		{
			A[i][j]=A[i][j]-A[k][j]*temp2;
		}
		B[i]=B[i]-B[k]*temp2;
	}
}

int Check_Matr(vector< vector<simp_frac> > &A, vector<simp_frac> &B)
{
	int er=0;
	bool flag;
	for(long long i=0;i<A.size();i++)
	{
		flag=true;
		for(long long j=0;j<A[0].size();j++)
		{
			if(A[i][j].num!=0)
			{
				flag=false;
				break;
			}
		}
		if(flag)
		{
			if(B[i].num==0)
			{
				er++;
				B.erase(B.begin()+i);
				A[i].clear();
				A.erase(A.begin()+i);
				i--;
```

```
                    }
                    else
                    {
                          return -1;
                    }
             }
      }
      return er;
}

bool Exists_Num(long long x, vector<long long> setNum)
{
      for(long long i=0;i<setNum.size();i++) if(x==setNum[i]) return true;
      return false;
}

void Print_Solution(vector< vector<simp_frac> > A, vector<simp_frac> B, vec-
tor<long long> setNum, long long n)
{
      long long r=setNum.size();
      for(long long i=0;i<r;i++)
      {
             printf("x%lli=%lli/%lli",setNum[i]+1,B[i].num,B[i].den);
             for(long long j=0;j<n;j++)
             {
                    if(!Exists_Num(j,setNum))
                    {
                          printf(" +
(%lli/%lli)x%lli",A[i][j].num,A[i][j].den,j+1);
                    }
             }
             printf(";\n");
      }
      printf("\n");
}

int Jordan_Gauss(vector< vector<simp_frac> > &A, vector<simp_frac> &B)
{
      vector<simp_frac> swap;
      simp_frac temp_b;
      int er;
      long long n=A[0].size();
      long long r=B.size(),k=0,s=0;
      while(k<r)
      {
             for(long long i=k;i<B.size();i++)
             {
                    if(A[i][s].num!=0)
                    {
                          swap=A[i];
                          A[i]=A[k];
                          A[k]=swap;
                          temp_b=B[i];
                          B[i]=B[k];
                          B[k]=temp_b;
                          Calculate_Matr(A,B,k,s);
                          er=Check_Matr(A,B);
                          if(er==-1)
                          {
                                 Print_Matr(A,B);
                                 cout<<"System hasn't solution"<<endl;
                                 return -1;
```

```
                    }
                    else r-=er;
                    k++;
                    break;
                }
            }
            s++;
        }
        return r;
}

int Jordan_Gauss(vector< vector<simp_frac> > &A, vector<simp_frac> &B, vec-
tor<long long> setNum)
{
        bool flag;
        vector<simp_frac> swap;
        simp_frac temp_b;
        int er;
        long long n=A[0].size();
        long long r=setNum.size(),k,s;
        for(k=0;k<r;k++)
        {
            s=setNum[k];
            flag=false;
            for(long long i=k;i<B.size();i++)
            {
                if(A[i][s].num!=0)
                {
                    flag=true;
                    swap=A[i];
                    A[i]=A[k];
                    A[k]=swap;
                    temp_b=B[i];
                    B[i]=B[k];
                    B[k]=temp_b;
                    Calculate_Matr(A,B,k,s);
                    break;
                }
            }
            if(!flag) return -1;
        }
        for(long long i=0;i<B.size();i++) if(B[i]<0) return -1;
        return 0;
}

int Find_Basic_Solution(vector< vector<simp_frac> > &A, vector<simp_frac> &B,
vector<long long> &setNum, long long n)
{
        bool flag=false;
        int r,p;
        r=Jordan_Gauss(A,B);
        if(r==-1) return -1;
        for(int i=0;i<r;i++) setNum.push_back(i);
        p=r-1;
        while(p>=0)
        {
            if(Jordan_Gauss(A,B,setNum)!=-1)
            {
                flag=true;
                break;
            }
            p=Next_SetNum(setNum,p,n);
```

```
	}
	if(Jordan_Gauss(A,B,setNum)!=-1) flag=true;
	if(!flag)
	{
		cout<<"Îïîðíûõ ðåøåíèé íå ñóùåñòâóåò"<<endl;
		return -1;
	}
	cout<<"Ìàòðèöà ñèñòåìû ñ îïîðíûì áàçèñîì"<<endl;
	Print_Matr(A,B);
	return r;
}

void Calculate_Simp_Table(vector< vector<simp_frac> > &A, vector<simp_frac>
&B, vector<simp_frac> &Z,long long k, long long s)
{
	Calculate_Matr(A,B,k,s);
	simp_frac temp=Z[s];
	long long n=Z.size()-1;
	for(long long j=0;j<n;j++)
	{
		Z[j]=Z[j]-A[k][j]*temp;
	}
	Z[n]=Z[n]+B[k]*temp;
}

long long Find_Simp_Div(vector< vector<simp_frac> > A, vector<simp_frac> B,
long long s)
{
	vector<long long> pot;
	for(long long i=0;i<B.size();i++)
	{
		if(A[i][s].num>0)
		{
			if(pot.size()==0) pot.push_back(i);
			else if((B[i]/A[i][s])<(B[pot[0]]/A[pot[0]][s])) pot[0]=i;
		}
	}
	if(pot.size()==0)
	{
		printf("System has infinity value;\n\n");
		return -1;
	}
	return pot[0];
}

vector<simp_frac> Create_Solution(vector<simp_frac> B, vector<long long>
setNum, long long k, long long s, long long n)
{
	setNum[k]=s;
	vector<simp_frac> X=vector<simp_frac>(n);
	for(long long i=0;i<n;i++)
	{
		X[i]=simp_frac(0);
		if(Exists_Num(i,setNum))
		{
			for(long long j=0;j<setNum.size();j++)
			{
				if(setNum[j]==i)
				{
					X[i]=B[j];
					break;
				}
```

```
			}
		}
	}
	return X;
}

vector<simp_frac> Diff_Solutions(vector<simp_frac> X, vector<simp_frac> Y)
{
	vector<simp_frac> Z=vector<simp_frac>(X.size());
	for(long long i=0;i<Z.size();i++) Z[i]=X[i]-Y[i];
	return Z;
}

void Print_Simp_Solution(vector<simp_frac> X[],long long n, simp_frac res)
{
	printf("Zmax=Z(");
	for(long long j=0;j<X[0].size();j++)
	{
		for(long long i=0;i<n;i++)
		{
			printf("(%lli/%lli)",X[i][j].num,X[i][j].den);
			for(long long k=i+1;k<n;k++)
			{
				printf("*m%lli",k);
			}
			if(i!=n-1) cout<<"+";
		}
		if(j!=X[0].size()-1) cout<<",";
	}
	printf(")=%lli/%lli\n\n",res.num,res.den);
}

void Simp_Table(vector< vector<simp_frac> > &A, vector<simp_frac> &B, vec-
tor<simp_frac> &Z, vector<long long> &setNum)
{
	Print_Simp_Table(A,B,Z,setNum);
	vector<long long> pot;
	vector<long long> copy_setNum;
	long long n=A[0].size();
	long long r=B.size(),k=0,s=0;
	vector<simp_frac> *X;
	bool flag_neg;
	int zero_count;
	while(true)
	{
		flag_neg=false;
		zero_count=0;
		pot.clear();
		for(long long j=0;j<n;j++)
		{
			if(Exists_Num(j,setNum)) continue;
			if(-Z[j].num<0)
			{
				if(flag_neg)
				{
					if(Z[pot[0]]<Z[j]) pot[0]=j;
				}
				else
				{
					pot.push_back(j);
					flag_neg=true;
				}
```

```
                }
                else if(Z[j].num==0) zero_count++;
            }
            if(flag_neg)
            {
                s=pot[0];
                k=Find_Simp_Div(A,B,s);
                if(k==-1) return;
                Calculate_Simp_Table(A,B,Z,k,s);
                setNum[k]=s;
                Print_Simp_Table(A,B,Z,setNum);
            }
            else
            {
                X=new vector<simp_frac>[zero_count+1];
                X[0]=Create_Solution(B,setNum,0,setNum[0],n);
                for(long long i=0;i<setNum.size();i++)
copy_setNum.push_back(setNum[i]);
                long long count=1;
                for(long long j=0;j<n;j++)
                {
                    if(!Exists_Num(j,copy_setNum) && Z[j].num==0)
                    {
                        k=Find_Simp_Div(A,B,j);
                        Calculate_Simp_Table(A,B,Z,k,j);
                        setNum[k]=j;
                        Print_Simp_Table(A,B,Z,setNum);
                        X[count]=Create_Solution(B,setNum,k,j,n);
                        count++;
                    }
                }
                if(count!=zero_count+1) printf("Something was wrong!\n");
                for(long long j=0;j<zero_count;j++) X[j]=Diff_Solu-
tions(X[j],X[j+1]);
                Print_Simp_Solution(X,zero_count+1,Z[n]);
                break;
            }
        }
}

int main()
{
        system("chcp 1251>0");
        ifstream cin("InTest.txt");
        long long n,m,temp,r;
        char temp_c;
        vector<long long> setNum;
        vector<simp_frac> temp_v;
        vector< vector<simp_frac> > A;
        vector<simp_frac> B,Z;
        vector<char> Op;
        cin>>n>>m;
        for(long long i=0;i<m;i++)
        {
            temp_v.clear();
            for(long long j=0;j<n;j++)
            {
                cin>>temp;
                temp_v.push_back(simp_frac(temp));
            }
            A.push_back(temp_v);
            cin>>temp_c;
```

```
        Op.push_back(temp_c);
        cin>>temp;
        B.push_back(simp_frac(temp));
}
for(long long j=0;j<n;j++)
{
        cin>>temp;
        Z.push_back(simp_frac(temp));
}
for(long long k=0;k<m;k++)
{
        if(Op[k]!='=')
        {
              for(long long i=0;i<m;i++)
              {
                    if(i==k) continue;
                    A[i].push_back(simp_frac(0));
              }
              if(Op[k]=='<') A[k].push_back(simp_frac(1));
              else A[k].push_back(simp_frac(-1));
              Z.push_back(simp_frac(0));
              n++;
        }
}
Z.push_back(simp_frac(0));
printf("\nZ=(%lli/%lli)",Z[n].num,Z[n].den);
for(long long i=0;i<n;i++)
{
        if(Z[i].num!=0)
        {
              printf(" + (%lli/%lli)x%lli",Z[i].num,Z[i].den,i+1);
        }
}
cout<<" -> max"<<endl<<endl;

Print_Matr(A,B);
r=Find_Basic_Solution(A,B,setNum,n);
if(r==-1) return 0;

for(long long i=0;i<r;i++)
{
        for(long long j=0;j<n;j++)
        {
              if(!Exists_Num(j,setNum))
              {
                    Z[j]=Z[j]-Z[setNum[i]]*A[i][j];
              }
        }
        Z[n]=Z[n]+Z[setNum[i]]*B[i];
        Z[setNum[i]]=simp_frac(0);
}

printf("\nZ=(%lli/%lli)",Z[n].num,Z[n].den);
for(long long i=0;i<n;i++)
{
        if(!Exists_Num(i,setNum))
        {
              printf(" + (%lli/%lli)x%lli",Z[i].num,Z[i].den,i+1);
        }
}
cout<<";"<<endl<<endl;
Print_Solution(A,B,setNum,n);
```

```
    cout<<endl;
    Simp_Table(A,B,Z,setNum);
    return 0;
}
```