НАЦІОНАЛЬНИЙ ТЕХНІЧНИЙ УНІВЕРСИТЕТ УКРАЇНИ

«КИЇВСЬКИЙ ПОЛІТЕХНІЧНИЙ ІНСТИТУТ

ІМЕНІ ІГОРЯ СІКОРСЬКОГО»

Факультет прикладної математики

Кафедра прикладної математики

на тему

«Задача про призначення для магазину-складу»

|  |  |
| --- | --- |
| Виконала: |  |
| студентка групи КМ-73 |  |
| Гнідак М.В. |  |

Київ – 2020

Зміст

1 Вступ3

2 Теоретичні відомості4

3 Опис вибраного методу5

4 Проектування архітектури розроблюваних засобів6

5 Формат вхідних та вихідних даних7

6 Блок-схема алгоритму8

7 Результат виконання програми9

Висновок12

Література13

Додаток А(програмний код)14

1 ВСТУП

У процесі керування виробництвом найчастіше виникають **задачі призначення** виконавців на різні види робіт, наприклад: підбор кадрів і призначення кандидатів на вакантні посади, розподіл джерел капітальних вкладенні між різними проектами науково-технічного розвитку, розподіл екіпажів літаків між авіалініями.

2 ТЕОРЕТИЧНІ ВІДОМОСТІ

Задачі про призначення можна сформулювати в такий спосіб. Необхідно виконати *N*різних робіт. Для їхнього виконання можна залучити *N*робітників. Кожний робітник за певну плату готовий виконати будь-яку роботу. Виконання будь-якої роботи варто доручити одному робітникові. Потрібно так розподілити роботи між робітниками, щоб загальні витрати на виконання всіх робіт були мінімальними.

Метод заснований на редукції вихідної матриці і призначенні за нульовими елементами редукованої матриці. При неможливості призначення за нульовими елементами матриця перетворюється так, щоб допустимих нулів було більше, ніж в редукованій матриці.

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| Робота  Працівник | 1 | 2 | … | j | … | m |
| 1 |  |  | … |  | … |  |
| 2 |  |  | … |  | … |  |
| … | … | … | … | … | … | … |
| i |  |  | … |  | … |  |
| … | … | … | … | … | … | … |
| m |  |  | … |  | … |  |

При рішенні задачі про призначення вихідною інформацією є таблиця задач про призначення с = {сij}, елементами якої служать показники ефективності призначень. Для задачі про призначення, записаної в стандартній формі, кількість рядків цієї таблиці збігається з кількістю стовпців:

Виконаємо перетворення цієї матриці: *aij = cij – uij – vij,*

де *uij* – мінімальний елемент для рядка *i*, а *vij* – мінімальний елемент для стовпця *j.*

сij — показник ефективності призначення i-го робітника на j-й роботі, наприклад витрати виконання i-м робітником j-й роботи;

xij — змінна моделі (хij = 1, якщо i-й робітник використовується на j-й роботі, і xij = 0 у противному випадку).

*aijxij = cij xij – uijxij – vijxij,*

**Крок 1.** Перейти від початкової матриці до редукованої. Для цього від усіх елементів кожного рядка відняти мінімальний елемент цього рядка, а потім у редукованій по рядках матриці виконати таку саму операцію зі стовпцями. Після цього початкову матрицю замінити редукованою матрицею.

**Крок 2.** Виконати призначення. Якщо після редукції в кожному рядку і в кожному стовпці можна вибрати по одному нульовому елементу так, щоб виконувалась система обмежень, то таке призначення буде оптимальним. Якщо цього не можна зробити, то треба перейти на наступний крок.

**Крок 3.** Модифікація редукованої матриці. Закреслити в редукованій матриці всі стовпці і рядки, які містять хоча б один нуль, починаючи з тих рядків і стовпців, які містять максимальне число нулів, і продовжувати закреслення в такому порядку й далі. Із всіх невикреслених елементів знайти найменший, відняти його від кожного невикресленого елемента і додати його до кожного елемента, розташованого на перетині двох ліній, за допомогою яких були викреслені рядки та стовпці. Перейти на крок 2 і якщо не вдасться виконати повне призначення, асоційоване з нулями модифікованої матриці, то знову перейти на крок 3. І так доти, доки не одержимо оптимальне призначення, асоційоване з нулями модифікованої матриці.

Якщо необхідно дослідити задачу на максимум, то всі елементи матриці слід помножити на (-1) і потім скласти з найбільшим числом матриці, таким чином, щоб матриця не вміщувала від'ємних елементів.

3 ОПИС ВИБРАНОГО МЕТОДУ

**Угорський алгоритм**

1. У кожному рядку матриці призначення знаходимо найменший елемент і віднімаємо його від усіх елементів у даному рядку.
2. У кожному стовпці отриманої матриці знаходимо найменший елемент і віднімаємо його від усіх елементів цього стовпчика.
3. Знаходимо рядок з одним нулем (0). Цей нуль виділяємо, його називають відміченим. У стопвці, де знаходиться відмічений нуль, всі інші нулі закреслюємо і далі не розглядаємо. Цей крок продовжуємо, поки це можливо.
4. Знаходимо стовпець з одним нулем, тепер він буде відміченим. У рядку, де знаходиться відмічений нуль, всі інші закреслюємо. Цей крок продовжуємо, поки це можливо.
5. Якщо після кроків 3 та 4 ще є невідмічені нулі, то відмічаємо будь-який з них, а в рядку та стовпці, де він знаходиться, всі інші нулі закреслюємо.
6. Якщо у кожному рядку та у кожному стовпці матриці буде рівно один відмічений нуль, то отримали оптимальне розв'язання. Кожен із відмічених нулів закріплюється поставщиком до споживача.

4 ПРОЕКТУВАННЯ АРХІТЕКТУРИ РОЗРОБЛЮВАНИХ ЗАСОБІВ

Модуль для приведення матриці

Модуль для відмічання та закреслення нулів

Модуль для перестановки нулів

Модуль для знаходження чисел які використовуються для приведнення матриці

Модуль для виведення результатів

Модуль для перевірки закінченості розв’язання

Рис. 1 Архітектура програмних засобів

5 ФОРМАТ ВХІДНИХ ТА ВИХІДНИХ ДАНИХ

**Вхідні Дані**

В якості вхідних даних в программу подається матртиця. В кожному стовпці мають міститись числові, непорожні дані.

**Вихідні дані**

В якості вихідних даних, програма має видати розподіл для кожного складу про те в який магазин потрібно перевести товар з кожного складу, щоб витрати мінімальну суму на перевезення

6 БЛОК-СХЕМА АЛГОРИТМУ



Рис. 2 Блок-Схема алгоритму

7 РЕЗУЛЬТАТ ВИКАНОННЯ ПРОГРАМИ

**Приклад 1**

В цьому прикладі використано матрицю розмірності 4.

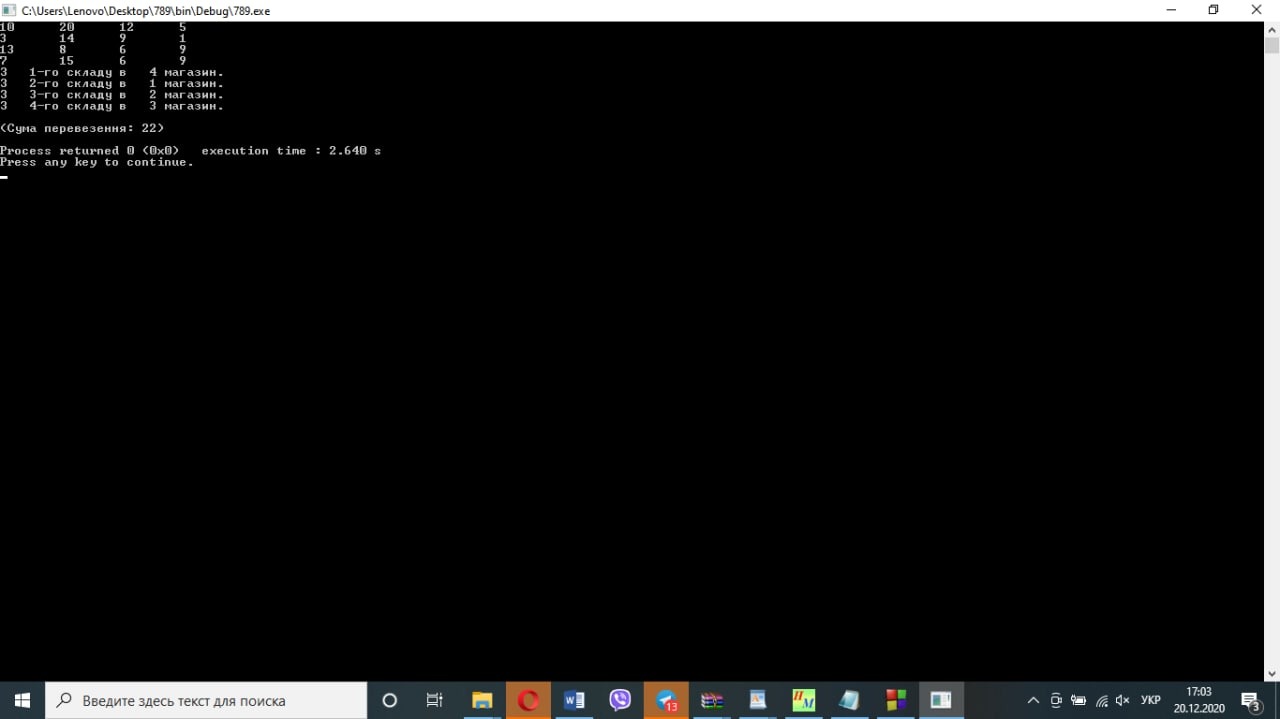


Рис. 3 Результати виконання програми

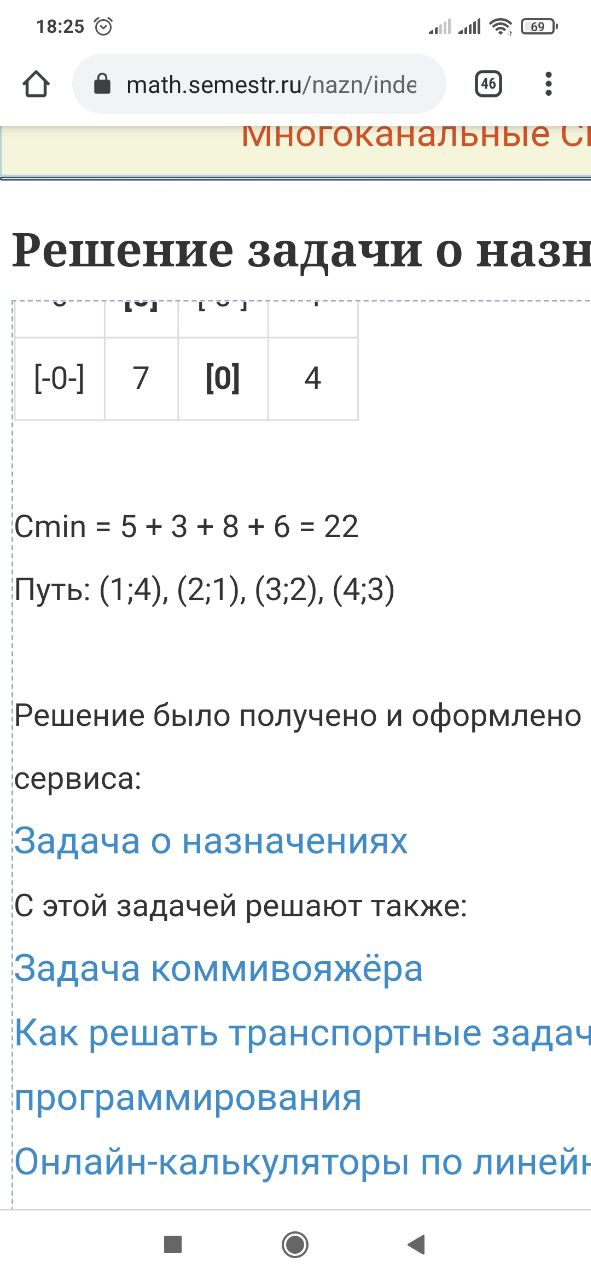


Рис.4 Результат, що були отримані в онлайн калькуляторі

**Приклад 2**

Матриця розмірності 7.

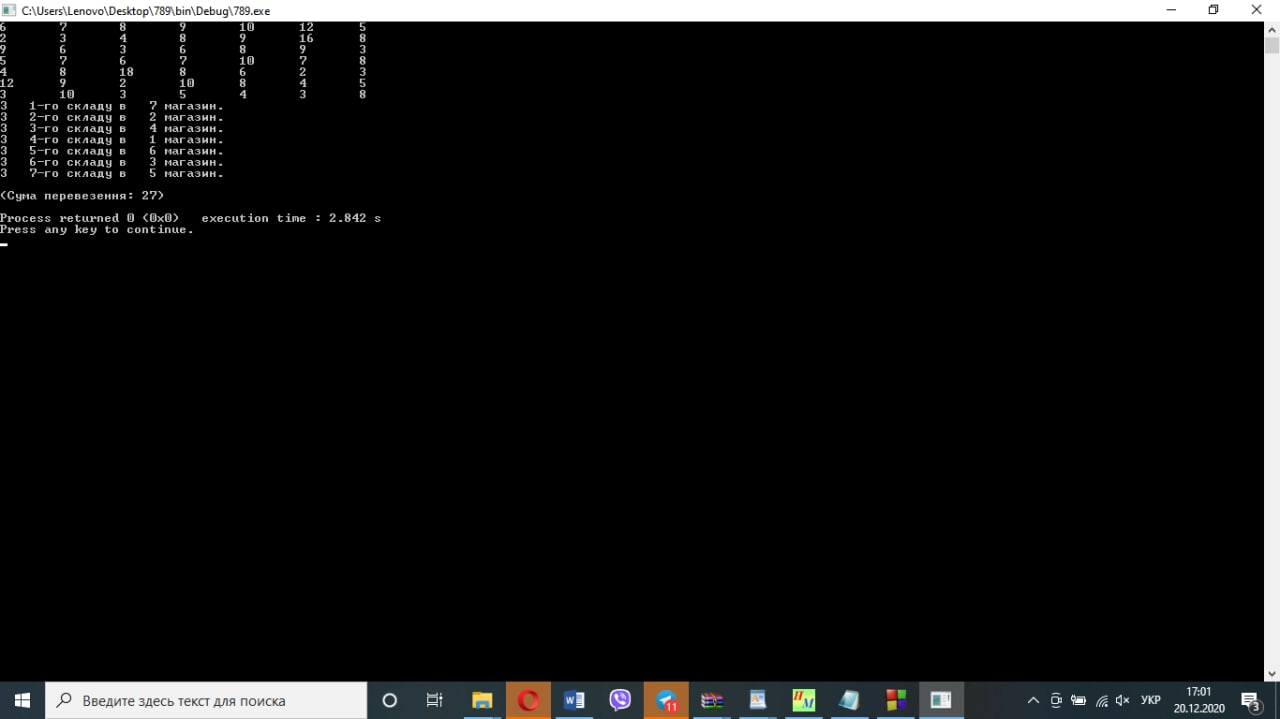
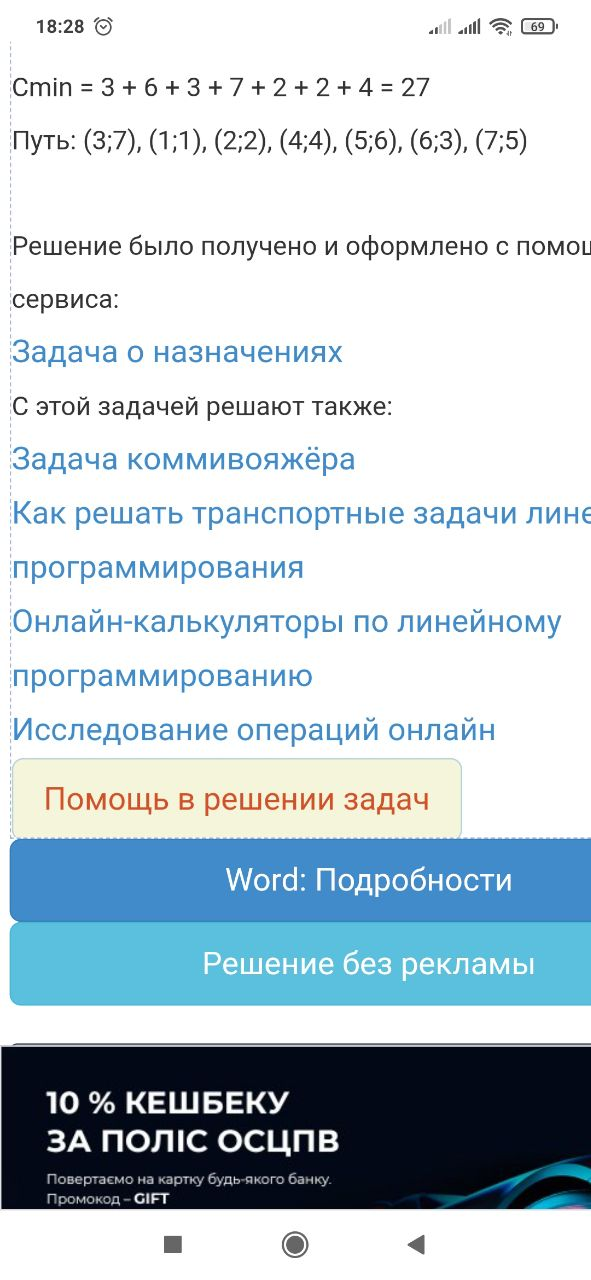


Рис.5 Результат виконання програми



Путь: (1,7), (2,2), (3,4), (4,1), (5,6), (6,3),(7,5)

Рис.6 Результат, що були отримані в онлайн калькуляторі

ВИСНОВОК

В даній роботі використані матриці, які були змодельовані вручну, а отримані результати будуть порівнюватись з тим, що видає популярний онлайн калькулятор, так як провести розрахунок руками можливо, але для матриць великої розмірності є дуже великою можливість зробити помилку, або і зовсім не змогти розв’язати задачі через неуважність, і саме через появу ЕОМ, такі методи стали популярними.

Для першої матриці розмірності 4 ми отримали, що мінімальна сума перевезення складе 22 умовні одиниці, а для матриці розмірності 7 – 27 умовних одиниць і також була отримана схема перевезення при якій дана сума буде досягнена.

ЛІТЕРАТУРА

1. *Костевич Л.С. Математическое программирование: Информ. Технологии оптимальных решений: Учеб. Пособие.- Минск.: Новое знание, 2003.-424 с.*
2. *Ларіонов Ю.І., Левикін В.М., Хажмурадов М.А. Дослідження операцій в інформаційних системах. – Харків.: Компанія СМІТ, 2005.-364 с.*
3. *Лук‘янова В.В. Комп‘ютерний аналіз даних: Посібник. – К.: Видавничий центр „Академія”, 2003. – 344с.*
4. *Терентьєв О.О. Дослідження операцій: навчальний посібник / О.О. Терентьєв, О.В. Доля, О.І. Баліна. – К.: 2020. –116 с.:іл.*

ДОДАТОК А

Програмний код

main.cpp

#include <windows.h>

#include <stdio.h>

#define N 101

#include <iostream>

int u[N][N],

cross0[N][N]; // 1 - отмеченный ноль, -1 - зачеркнутый.

int str0[N],col0[N]; // Помеченные строки и столбцы.

int strround[N],colround[N]; // Пометка кружками (вторая пометка).

int usestr[N];

int vmax=0;

int sum=0;

int n = 7;

bool ToMax = FALSE; // ToMax == TRUE - задача на максимум,

// ToMax == FALSE - задача на минимум.

int v[7][7]={6,7,8,9,10,12,5,

2,3,4,8,9,16,8,

9,6,3,6,8,9,3,

5,7,6,7,10,7,8,

4,8,18,8,6,2,3,

12,9,2,10,8,4,5,

3,10,3,5,4,3,8};

// Функция считывания данных из файла.

// Приведение матрицы

void privod(BOOL ToMax){

int i,j,umin;

for(i=0; i<n; i++)

for(j=0; j<n; j++)

vmax = vmax > v[i][j] ? vmax : v[i][j];

if(ToMax){

for(i=0; i<n; i++)

for(j=0; j<n; j++)

u[i][j] = vmax-v[i][j];

}

else{

for(i=0; i<n; i++)

for(j=0; j<n; j++)

u[i][j] = v[i][j];

}

for(i=0; i<n; i++){ // Приведение по строкам.

umin=u[i][0];

for(j=1; j<n; j++) // Минимум в строке.

umin = umin < u[i][j] ? umin : u[i][j];

for(j=0; j<n; j++)

u[i][j]-=umin;

}

for(j=0; j<n; j++){ // Приведение по столбцам.

umin=u[0][j];

for(i=1; i<n; i++) // Минимум в столбце.

umin = umin < u[i][j] ? umin : u[i][j];

for(i=0; i<n; i++)

u[i][j]-=umin;

}

}

void mark0(){ // Отмечаем и зачеркиваем нули.

int i,j;

for(i=0; i<n; i++)

for(j=0; j<n; j++)

cross0[i][j]=0;

for(i=0; i<n; i++)

str0[i]=col0[i]=0;

for(i=0; i<n; i++)

for(j=0; j<n; j++)

if(u[i][j]==0)

if(str0[i]==0 && col0[j]==0){

cross0[i][j]=1;

str0[i]=col0[j]=1;

}

else

cross0[i][j]=-1;

}

int findcouple(int i){

int i1,j=0;

while(cross0[i][j]!=1) j++;

for(i1=0; i1<n; i1++)

// Если ноль зачеркнут и в этой строке еще не были

if(cross0[i1][j]==-1 && !usestr[i1]){

// Если строка не помечена

if(!str0[i1]){

str0[i1]=1;

cross0[i1][j]=1;

cross0[i][j]=-1;

return 1;

}

else{

usestr[i1]=1;

if(findcouple(i1)){

cross0[i1][j]=1;

cross0[i][j]=-1;

return 1;

}

}

}

return 0;

}

// Нахождение паросочетания.

int upcouple(){

int i,j;

for(i=0; i<n; i++)

usestr[i]=0; // В какой строке побывали.

for(j=0; j<n; j++)

if(!col0[j])

for(i=0; i<n; i++)

if(cross0[i][j]==-1){ // Зачеркнутый ноль в непомеченном столбце.

usestr[i]=1;

if(findcouple(i)){

col0[j]=1;

cross0[i][j]=1;

return 1;

}

else

usestr[i]=0;

}

return 0;

}

// Нахождение максимального паросочетания.

void maxcouple(){

while(upcouple());

}

// Проверка на решенность задачи.

int fin(){

int i;

for(i=0; i<n; i++)

if(!str0[i])

return 0;

return 1;

}

// Нахождение минимальной опоры.

void minsupport(){

int i,j,b;

for(i=0; i<n; i++)

strround[i]=colround[i]=0;

for(i=0; i<n; i++)

strround[i]=1-str0[i];

b=1;

while(b){

b=0;

for(i=0; i<n; i++)

if(strround[i])

for(j=0; j<n; j++)

if(cross0[i][j]==-1)

colround[j]=1;

for(j=0; j<n; j++)

if(colround[j])

for(i=0; i<n; i++)

if(cross0[i][j]==1 && !strround[i]){

b=1;

strround[i]=1;

}

}

}

// Перестановка нулей.

void rotate0(){

int i,j,min=vmax;

for(i=0; i<n; i++)

if(strround[i])

for(j=0; j<n; j++)

if(!colround[j])

if(min>u[i][j])

min=u[i][j];

for(i=0; i<n; i++)

if(!strround[i])

for(j=0; j<n; j++)

u[i][j]+=min;

for(j=0; j<n; j++)

if(!colround[j])

for(i=0; i<n; i++)

u[i][j]-=min;

}

// Выдача результата.

void print(){

int i,j;

for(int i=0; i<n; i++){

printf("З %3i-го складу ", i+1);

for(int j=0; j<n; j++)

if(cross0[i][j]==1){

printf("в %3i магазин.\n",j+1);

sum += v[i][j];

break;

}

}printf("\n(Сума перевезення: %d)\n",sum);}

int main()

{

setlocale(LC\_ALL, "Russian");

for (int i=0; i<n; i++){

for (int j = 0; j<n; j++)

printf("%d\t",v[i][j]);

printf("\n");

}

privod(ToMax);

while(1){

mark0();

maxcouple();

if(fin())

break;

minsupport();

rotate0();

}

print();

MessageBox(0,"Все посчитано.","Результат",64);

}