Лабораторная работа 1. **Вспомогательные функции**

**Цель работы:** приобретение навыков составления и отладки программ с использованием пользовательских функций для замера продолжительности процесса вычисления.

**Листинг:**

**lab1.срр**

#include <iostream>

#include "auxil.h"

#include <ctime>

#include <locale>

#define CYCLE 1000000 // количество циклов

int main()

{

double av1 = 0, av2 = 0;

clock\_t t1 = 0, t2 = 0;

setlocale(LC\_ALL, "rus");

auxil::start(); // старт генерации

t1 = clock(); // фиксация времени

for (int i = 0; i < CYCLE; i++)

{

av1 += (double)auxil::iget(-100, 100); // сумма случайных чисел

av2 += auxil::dget(-100, 100); // сумма случайных чисел

}

t2 = clock(); // фиксация времени

std::cout << std::endl << "количество циклов: " << CYCLE;

std::cout << std::endl << "среднее значение (int): " << av1 / CYCLE;

std::cout << std::endl << "среднее значение (double): " << av2 / CYCLE;

std::cout << std::endl << "продолжительность (у.е): " << (t2 - t1);

std::cout << std::endl << " (сек): "

<< ((double)(t2 - t1)) / ((double)CLOCKS\_PER\_SEC);

std::cout << std::endl;

system("pause");

return 0;

}

**auxil.h**

#pragma once

namespace auxil

{

void start(); // старт генератора сл. чисел

double dget(double rmin, double rmax); // получить случайное число

int iget(int rmin, int rmax); // получить случайное число

};

**auxil.cpp**

#include <cstdlib>

#include <ctime>

namespace auxil

{

void start() // старт генератора сл. чисел

{

srand((unsigned)time(NULL));

};

double dget(double rmin, double rmax) // получить случайное число

{

return ((double)rand() / (double)RAND\_MAX) \* (rmax - rmin) + rmin;

};

int iget(int rmin, int rmax) // получить случайное число

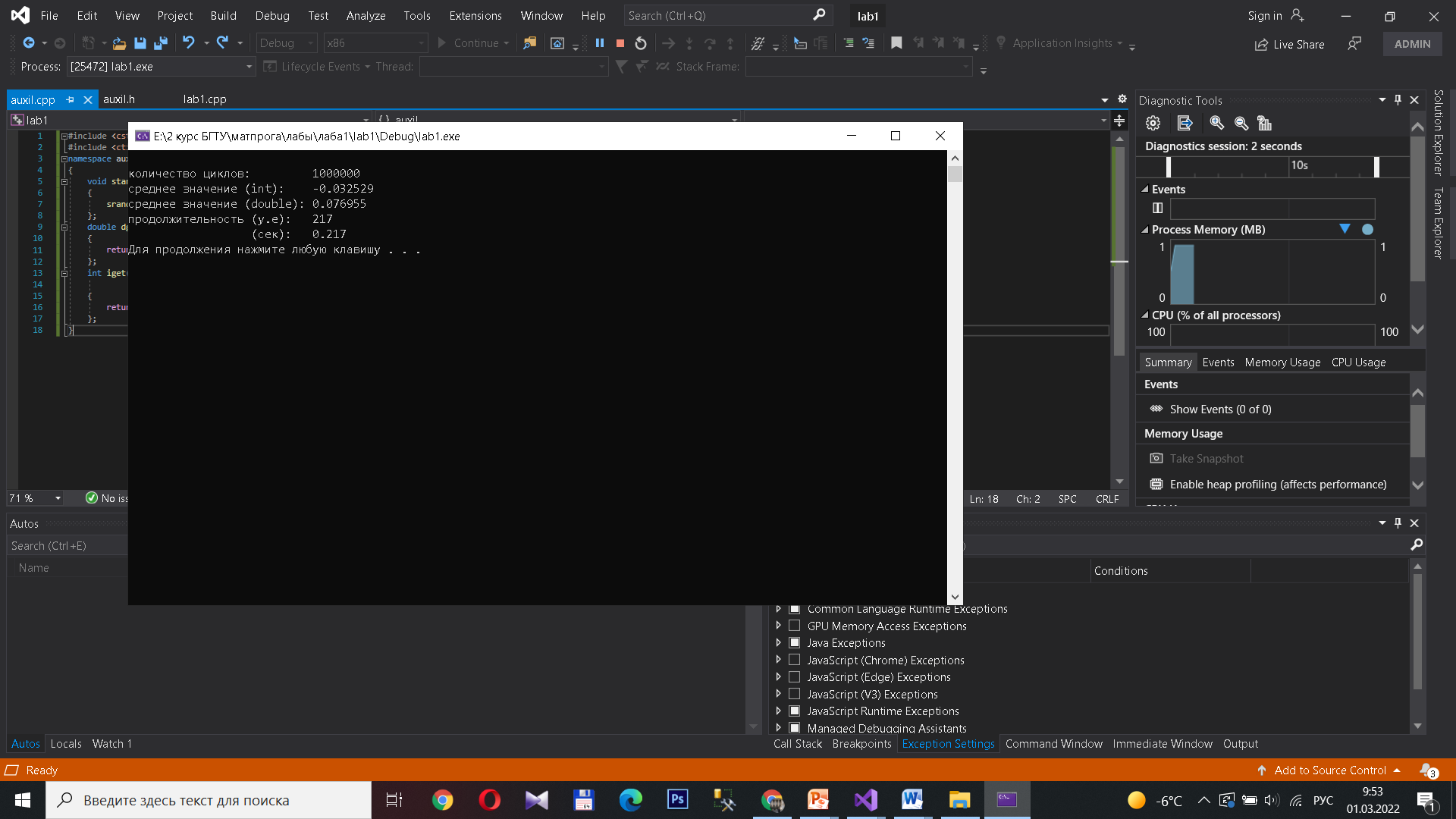
{

return (int)dget((double)rmin, (double)rmax);

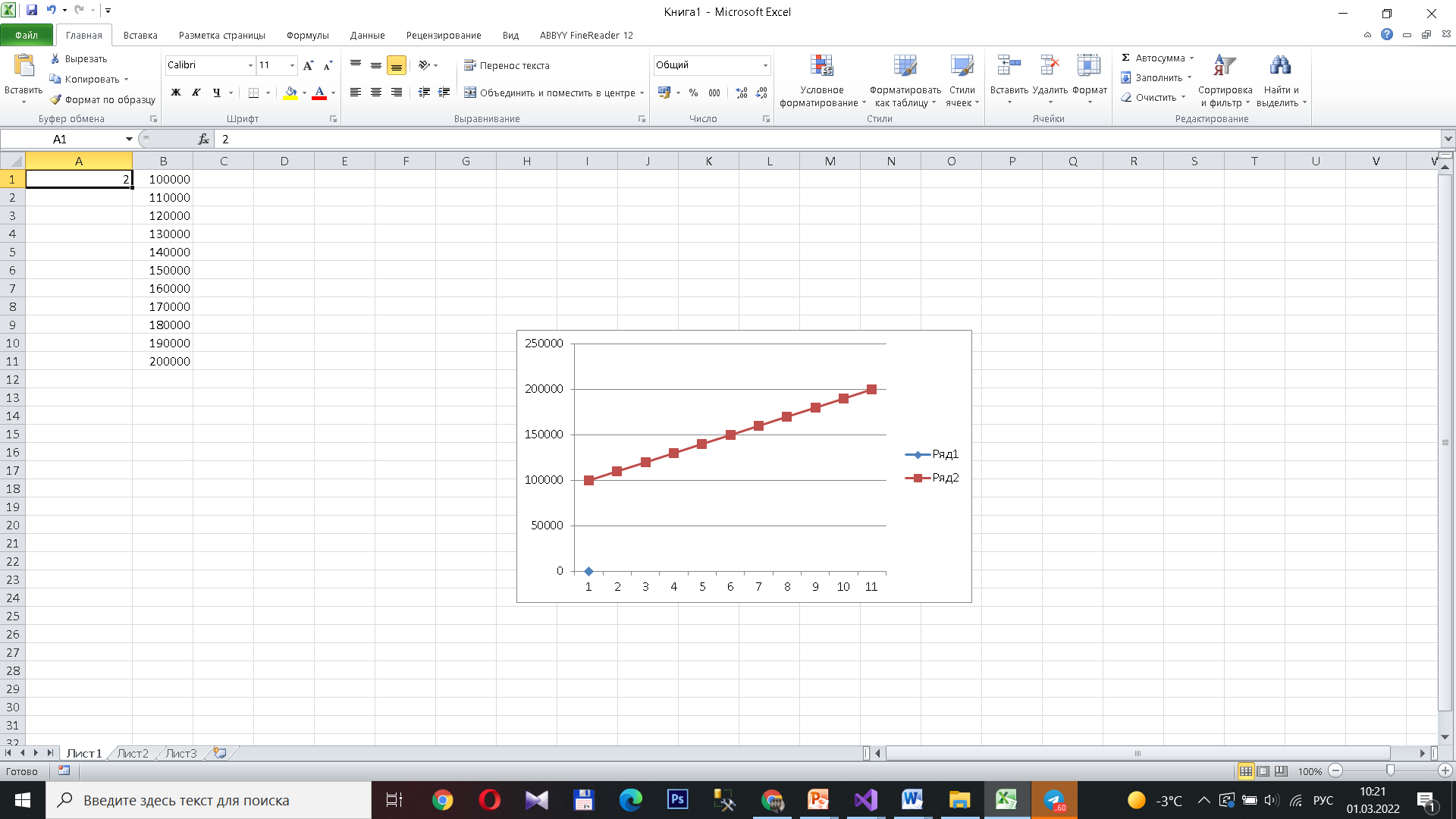
};

}

**Результат:**



**График:**



Начальное значение: 100000

Шаг: 10000

Время: 2 у.е.

**Вывод:** в ходе работы были приобретены навыки составления и отладки программ с использованием пользовательских функций для замера продолжительности процесса вычисления.

Лабораторная работа 2. **Комбинаторные алгоритмы решения оптимизационных задач. Вариант 10**

**Цель работы:** приобрести навыки разработки генераторов подмножеств, перестановок, сочетаний и размещений на С++; научиться применять разработанные генераторы для решения задач о рюкзаке (упрощенную, коммивояжера, об оптимальной загрузке судна и об оптимальной загрузке судна с центровкой.

**Задание 1**: решить в соответствии с вариантом 10 задачу упрощенную о рюкзаке (веса предметов и их стоимость сгенерировать случайным образом: вместимость рюкзака 300 кг, веса предметов 10 – 300 кг, стоимость предметов 5 – 55 у.е.; количество предметов – 18 шт.);

**Листинг:**

**main.срр**

#include "Combi1.h"

#include <tchar.h>

#include "Knapsack.h"

#define NN 18 //количество предметов – 18 шт

int main(int argc, \_TCHAR\* argv[])

{

setlocale(LC\_ALL, "rus");

int V = 300; // вместимость рюкзака 300 кг

auxil::start();

int v[NN], c[NN];

for (int i = 0; i < NN; i++) //заполнение рандомными числами

{

v[i] = auxil::iget(10, 300); //веса предметов 10 – 300 кг

c[i] = auxil::iget(5, 55); //стоимость предметов 5 – 55 у.е

}

short m[NN]; // количество предметов каждого типа {0,1}

int maxcc = knapsack\_s(

V, // [in] вместимость рюкзака

NN, // [in] количество типов предметов

v, // [in] размер предмета каждого типа

c, // [in] стоимость предмета каждого типа

m // [out] количество предметов каждого типа

);

std::cout << std::endl << "-------- Задача о рюкзаке ---------";

std::cout << std::endl << "- количество предметов : " << NN;

std::cout << std::endl << "- вместимость рюкзака : " << V;

std::cout << std::endl << "- размеры предметов : ";

for (int i = 0; i < NN; i++) std::cout << v[i] << " ";

std::cout << std::endl << "- стоимости предмета : ";

for (int i = 0; i < NN; i++) std::cout << c[i] << " ";

std::cout << std::endl << "- стоимости предметов : ";

for (int i = 0; i < NN; i++) std::cout << v[i] \* c[i] << " ";

std::cout << std::endl << "- оптимальная стоимость рюкзака: " << maxcc;

std::cout << std::endl << "- вес рюкзака: ";

int s = 0; for (int i = 0; i < NN; i++) s += m[i] \* v[i];

std::cout << s;

std::cout << std::endl << "- выбраны предметы: ";

for (int i = 0; i < NN; i++) std::cout << " " << m[i];

std::cout << std::endl << std::endl;

system("pause");

return 0;

}

**combi1.h**

// Combi1.h

// Шаблон структуры генератора множества всех подмножеств

#pragma once

namespace combi1

{

struct subset // генератор множества всех подмножеств

{

short n, // количество элементов исходного множества < 64

sn, // количество элементов текущего подмножества

\* sset; // массив индексов текущего подмножества

unsigned \_\_int64 mask; // битовая маска

subset(short n = 1); // конструктор(кол-во эл-ов исх. мн-ва)

short getfirst(); // сформ. массив индексов по битовой маске

short getnext(); // ++маска и сформировать массив индексов

short ntx(short i); // получить i-й элемент массива индексов

unsigned \_\_int64 count(); // вычислить общее кол-во подмножеств

void reset(); // сбросить генератор, начать сначала

};

};

**combi1.cpp**

// Combi1.cpp

// Реализация методов структуры subset

#include "Combi1.h"

#include "Auxil.h"

namespace combi1

{

subset::subset(short n) {

this->n = n;

this->sset = new short[n];

this->reset();

};

void subset::reset() {

this->sn = 0;

this->mask = 0;

};

short subset::getfirst() {

\_\_int64 buf = this->mask;

this->sn = 0;

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

if (buf & 0x1) this->sset[this->sn++] = i;

buf >>= 1;

}

return this->sn;

};

short subset::getnext() {

int rc = -1;

this->sn = 0;

if (++this->mask < this->count()) rc = getfirst();

return rc;

};

short subset::ntx(short i) {

return this->sset[i];

};

unsigned \_\_int64 subset::count() {

return (unsigned \_\_int64)(1 << this->n);

};

};

**auxil.h**

#pragma once

#include <cstdlib>

namespace auxil

{

void start(); // старт генератора сл. чисел

double dget(double rmin, double rmax); // получить случайное число

int iget(int rmin, int rmax); // получить случайное число

};

**auxil.cpp**

#include "Auxil.h"

#include <ctime>

namespace auxil

{

void start() // старт генератора сл. чисел

{

srand((unsigned)time(NULL));

};

double dget(double rmin, double rmax) // получить случайное число

{

return ((double)rand() / (double)RAND\_MAX) \* (rmax - rmin) + rmin;

};

int iget(int rmin, int rmax) // получить случайное число

{

return (int)dget((double)rmin, (double)rmax);

};

}

**knapsack.h**

// Knapsack.h

// Прототип функции knapsack\_s

#pragma once

#include "Combi1.h"

int knapsack\_s(

int V, // [in] вместимость рюкзака

short n, // [in] количество типов предметов

const int v[], // [in] размер предмета каждого типа

const int c[], // [in] стоимость предмета каждого типа

short m[] // [out] количество предметов каждого типа

);

**knapsack.cpp**

// Knapsack.cpp

#include "Knapsack.h"

#define NINF 0x80000000 // самое малое int-число

int calcv(combi1::subset s, const int v[]) // объем в рюкзаке

{

int rc = 0;

for (int i = 0; i < s.sn; i++) rc += v[s.ntx(i)];

return rc;

};

int calcc(combi1::subset s, const int v[], const int c[]) //стоимость в рюкзаке

{

int rc = 0;

for (int i = 0; i < s.sn; i++) rc += (v[s.ntx(i)] \* c[s.ntx(i)]);

return rc;

};

void setm(combi1::subset s, short m[]) //отметить выбранные предметы

{

for (int i = 0; i < s.n; i++) m[i] = 0;

for (int i = 0; i < s.sn; i++) m[s.ntx(i)] = 1;

};

int knapsack\_s(

int V, // [in] вместимость рюкзака

short n, // [in] количество типов предметов

const int v[], // [in] размер предмета каждого типа

const int c[], // [in] стоимость предмета каждого типа

short m[] // [out] количество предметов каждого типа {0,1}

)

{

combi1::subset s(n);

int maxc = NINF, cc = 0;

short ns = s.getfirst();

while (ns >= 0)

{

if (calcv(s, v) <= V)

if ((cc = calcc(s, v, c)) > maxc)

{

maxc = cc;

setm(s, m);

}

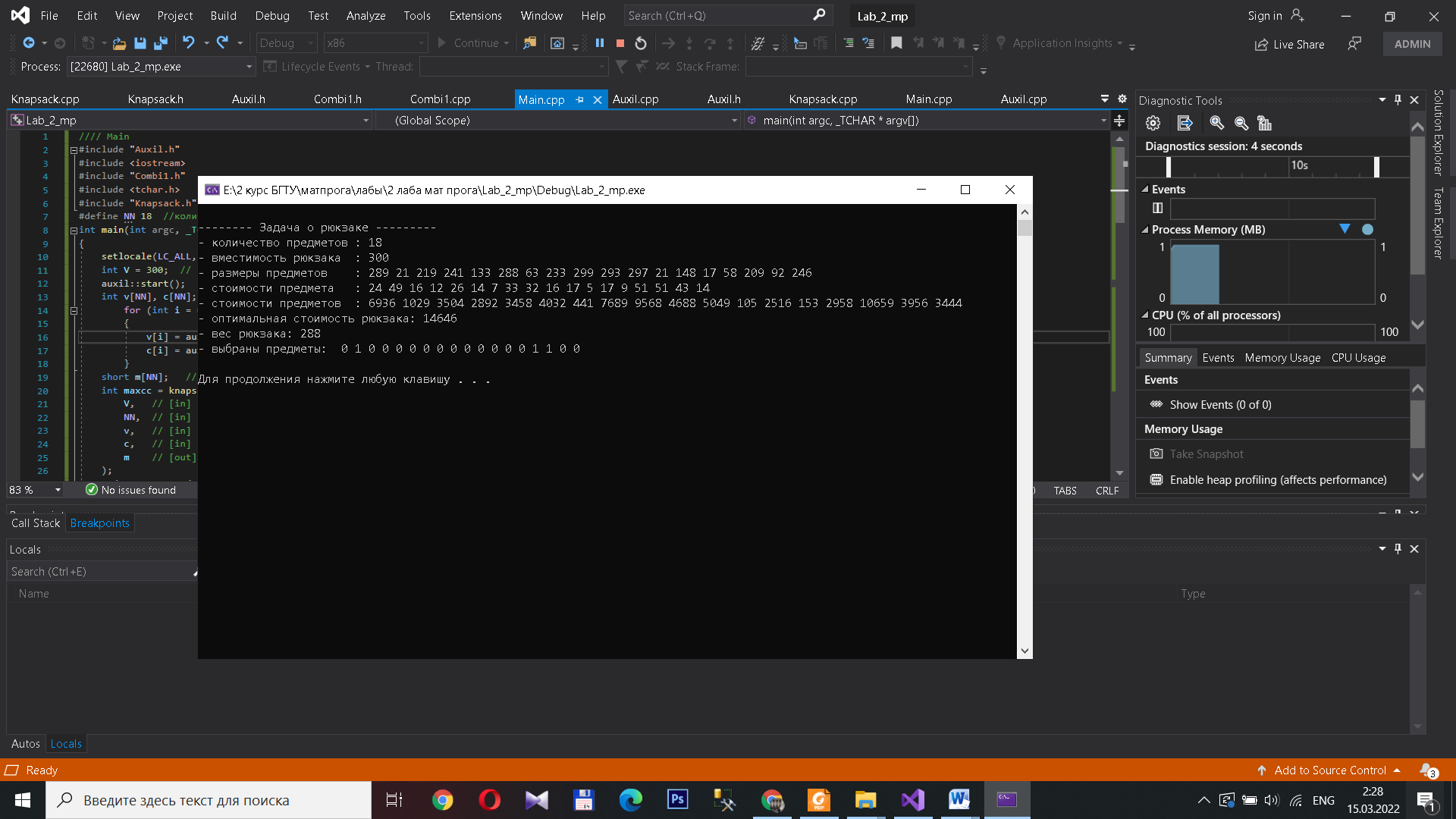
ns = s.getnext();

};

return maxc;

}

**Результат:**



**Задание 2**: исследовать зависимость времени вычисления необходимое для решения задачи в соответствии с вариантом 10 от размерности задачи о рюкзаке (количество предметов 12 – 20 шт.);

**Листинг:**

**main.срр**

#include <iostream>

#include "Combi1.h"

#include "Knapsack.h"

#include <time.h>

#include <tchar.h>

#include <iomanip>

#define NN 20

int main(int argc, \_TCHAR\* argv[])

{

setlocale(LC\_ALL, "rus");

int V = 300, // вместимость рюкзака

v[] = { 25, 56, 67, 40, 20, 27, 37, 33, 33, 44, 53, 12,

60, 75, 12, 55, 54, 42, 43, 14, 30, 37, 31, 12 },

c[] = { 15, 26, 27, 43, 16, 26, 42, 22, 34, 12, 33, 30,

12, 45, 60, 41, 33, 11, 14, 12, 25, 41, 30, 40 };

short m[NN];

int maxcc = 0;

clock\_t t1, t2;

std::cout << std::endl << "-------- Задача о рюкзаке ---------";

std::cout << std::endl << "- вместимость рюкзака : " << V;

std::cout << std::endl << "-- количество ------ продолжительность -- ";

std::cout << std::endl << " предметов вычисления ";

for (int i = 12; i <= NN; i++)

{

t1 = clock();

maxcc = knapsack\_s(V, i, v, c, m);

t2 = clock();

std::cout << std::endl << " " << std::setw(2) << i

<< " " << std::setw(5) << (t2 - t1);

}

std::cout << std::endl << std::endl;

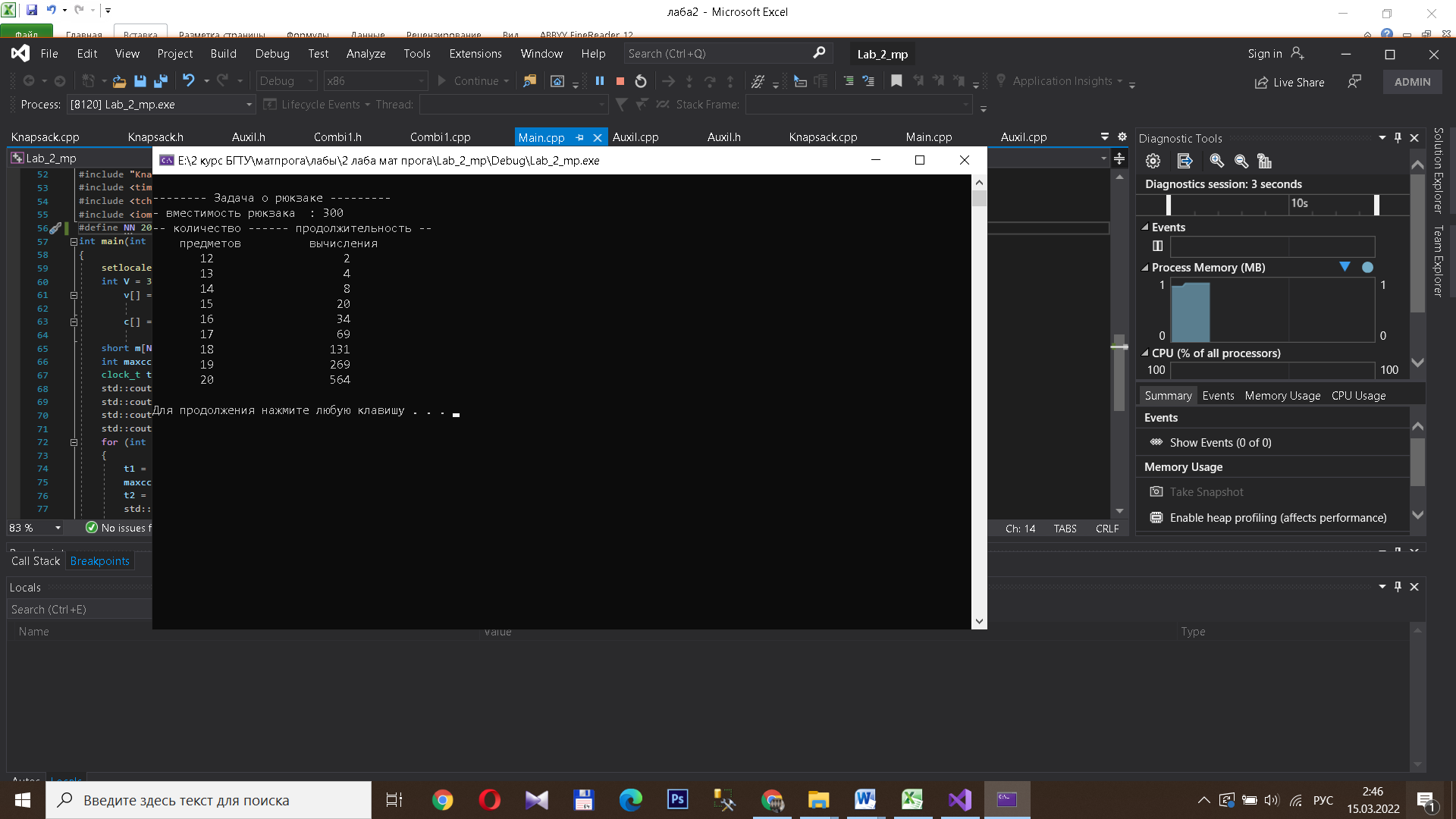
system("pause");

return 0;

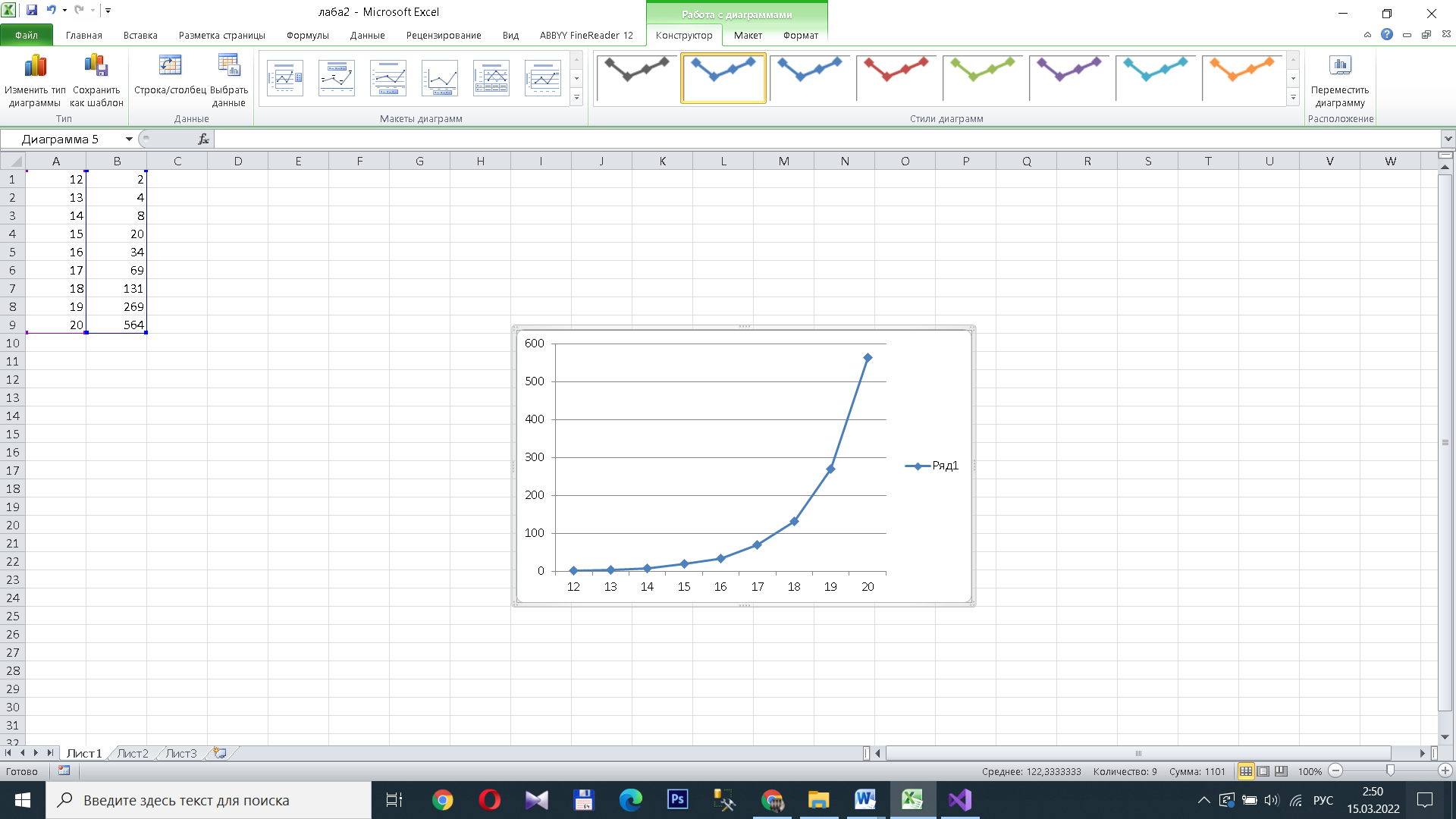
}

Листинг файлов combi1.h, combi1.cpp, auxil.h, auxil.cpp, knapsack.h, knapsack.cpp в задании 1.

**Результат:**



**График:**



**Вывод:** в ходе работы были приобретены навыки разработки генераторов подмножеств, перестановок, сочетаний и размещений на С++; а также применены разработанные генераторы для решения задач о рюкзаке.

Учреждение образования

«БЕЛОРУССКИЙ ГОСУДАРСТВЕННЫЙ ТЕХНОЛОГИЧЕСКИЙ УНИВЕРСИТЕТ»

Решение задачи коммивояжера методом ветвей и границ

10 вариант

Студент: Косс В.А.

ФИТ 2 курс 9 группа

Минск 2022

**ЦЕЛЬ РАБОТЫ:** освоить общие принципы решения задач методом ветвей и границ, решить задачу о коммивояжере данным методом, сравнить полученное решение задачи с комбинаторным методом перестановок.

**УСЛОВИЕ**

**Задание 1.** Сформулировать условие задачи коммивояжера с параметром. Для этого:

* принять элементы матрицы расстояний равными:

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **Город** | **1** | **2** | **3** | **4** | **5** |
| **1** |  | 2 \* n | 21 + n |  | n |
| **2** | n |  | 15 + n | 68 - n | 84 - n |
| **3** | 2 + n | 3 \* n |  | 86 | 49 + n |
| **4** | 17 + n | 58 - n | 4 \* n |  | 3 \* n |
| **5** | 93 - n | 66 + n | 52 | 13 + n |  |

где *n* – номер варианта, n=10;

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
|  | **0** | **1** | **2** | **3** | **4** |
| **0** | INF | 20 | 31 | INF | 10 |
| **1** | 10 | INF | 25 | 58 | 74 |
| **2** | 12 | 30 | INF | 86 | 59 |
| **3** | 27 | 48 | 40 | INF | 30 |
| **4** | 83 | 76 | 52 | 23 | INF |

**Задание 2.** Решить сформулированную задачу методом ветвей и границ.

**Задание 3.** Проверить полученное решение при помощи генератора перестановок (см. лаб. 2, задание 5.1.) и включить копию экрана с решением в отчет.

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
|  | **0** | **1** | **2** | **3** | 5 |
| **0** | INF | 20 | 31 | INF | 10 |
| **1** | 10 | INF | 25 | 58 | 74 |
| **2** | 12 | 30 | INF | 86 | 59 |
| **3** | 27 | 48 | 40 | INF | 30 |
| **4** | 83 | 76 | 52 | 23 | INF |

|  |
| --- |
| **αi** |
| 10 |
| 10 |
| 12 |
| 27 |
| 23 |

Столбец, помеченный символом αi, содержит константы приведения для каждой строки (наименьшее значение в строке). В окружности указана сумма этих констант (число 82).

Производим редукцию строк: из каждого элемента в строке вычитаем соответствующее значение найденного минимума.

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
|  | **0** | **1** | **2** | **3** | **5** |
| **0** | INF | 10 | 21 | INF | 0 |
| **1** | 0 | INF | 15 | 48 | 64 |
| **2** | 0 | 18 | INF | 74 | 47 |
| **3** | 0 | 21 | 13 | INF | 3 |
| **4** | 60 | 53 | 29 | 0 | INF |

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| βi | 0 | 10 | 13 | 0 | 0 |

Столбец, помеченный символом βi, содержит константы приведения для каждого столбца, а в окружности указана сумма этих констант.

Далее из каждого элемента в столбце вычитаем соответствующее значение найденного минимума. В конечном итоге получаем полностью приведенную таблицу.

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
|  | **0** | **1** | **2** | **3** | **5** |
| **0** | INF | 0 | 8 | INF | 0 |
| **1** | 0 | INF | 2 | 48 | 64 |
| **2** | 0 | 8 | INF | 74 | 47 |
| **3** | 0 | 11 | 0 | INF | 3 |
| **4** | 60 | 43 | 16 | 0 | INF |

Общая сумма констант приведения равна φ=82+23=105. Эту величину можно принять в качестве нижней границы длины кратчайшего кольцевого маршрута, проходящего через все города. Тогда корневой вершиной будет R=105.

Далее начинаем исследовать, какая дуга сильнее повлияет на нижнюю границу длины дополнительных кольцевых маршрутов. Для этого поочередно заменяем нули на знак бесконечности, высчитывая сумму минимального элемента по строке и минимального элемента по столбцу, в которых и располагается знак бесконечности.

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
|  | **0** | **1** | **2** | **3** | **5** |
| **0** | INF | INF | 8 | INF | 0 |
| **1** | 0 | INF | 2 | 48 | 64 |
| **2** | 0 | 8 | INF | 74 | 47 |
| **3** | 0 | 11 | 0 | INF | 3 |
| **4** | 60 | 43 | 16 | 0 | INF |

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
|  | **0** | **1** | **2** | **3** | **4** |
| **0** | INF | 0 | 8 | INF | INF |
| **1** | 0 | INF | 2 | 48 | 64 |
| **2** | 0 | 8 | INF | 74 | 47 |
| **3** | 0 | 11 | 0 | INF | 3 |
| **4** | 60 | 43 | 16 | 0 | INF |

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
|  | **0** | **1** | **2** | **3** | **4** |
| **0** | INF | 0 | 8 | INF | 0 |
| **1** | INF | INF | 2 | 48 | 64 |
| **2** | 0 | 8 | INF | 74 | 47 |
| **3** | 0 | 11 | 0 | INF | 3 |
| **4** | 60 | 43 | 16 | 0 | INF |

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
|  | **0** | **1** | **2** | **3** | **4** |
| **0** | INF | 0 | 8 | INF | 0 |
| **1** | 0 | INF | 2 | 48 | 64 |
| **2** | INF | 8 | INF | 74 | 47 |
| **3** | 0 | 11 | 0 | INF | 3 |
| **4** | 60 | 43 | 16 | 0 | INF |

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
|  | **0** | **1** | **2** | **3** | **4** |
| **0** | INF | 0 | 8 | INF | 0 |
| **1** | 0 | INF | 2 | 48 | 64 |
| **2** | 0 | 8 | INF | 74 | 47 |
| **3** | INF | 11 | 0 | INF | 3 |
| **4** | 60 | 43 | 16 | 0 | INF |

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
|  | **0** | **1** | **2** | **3** | **4** |
| **0** | INF | 0 | 8 | INF | 0 |
| **1** | 0 | INF | 2 | 48 | 64 |
| **2** | 0 | 8 | INF | 74 | 47 |
| **3** | 0 | 11 | INF | INF | 3 |
| **4** | 60 | 43 | 16 | 0 | INF |

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
|  | **0** | **1** | **2** | **3** | **4** |
| **0** | INF | 0 | 8 | INF | 0 |
| **1** | 0 | INF | 2 | 48 | 64 |
| **2** | 0 | 8 | INF | 74 | 47 |
| **3** | 0 | 11 | 0 | INF | 3 |
| **4** | 60 | 43 | 16 | INF | INF |

Далее из таблицы вычеркиваем тот столбец и ту строку, в котором располагается элемент с наибольшей суммой, а именно (4,3). И при построении новой таблицы элемент (3,4) заменяем на бесконечность, чтобы исключить зацикливание. Параллельно мы нижнюю границу увеличиваем на соответствующее наибольшей сумме число.

R(~~4,3)~~=105+64=169

|  |
| --- |
| αi |
| 0 |
| 0 |
| 0 |
| 0 |

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
|  | **0** | **1** | **2** | **4** |
| **0** | INF | 0 | 8 | 0 |
| **1** | 0 | INF | 2 | 64 |
| **2** | 0 | 8 | INF | 47 |
| **3** | 0 | 11 | 0 | INF |

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| βi | 0 | 0 | 0 | 0 |

Исходя из новой таблицы, видим, что матрица уже приведенная, поэтому нижняя граница с дугой (3,4) будет оставаться неизменной.

R(4,3)=105+0=105

Далее повторяем все те же действия, что и с первой матрицей.

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
|  | **0** | **1** | **2** | **4** |
| **0** | INF | INF | 8 | 0 |
| **1** | 0 | INF | 2 | 64 |
| **2** | 0 | 8 | INF | 47 |
| **3** | 0 | 11 | 0 | INF |

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
|  | **0** | **1** | **2** | **4** |
| **0** | INF | 0 | 8 | INF |
| **1** | 0 | INF | 2 | 64 |
| **2** | 0 | 8 | INF | 47 |
| **3** | 0 | 11 | 0 | INF |

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
|  | **0** | **1** | **2** | **4** |
| **0** | INF | 0 | 8 | INF |
| **1** | INF | INF | 2 | 64 |
| **2** | 0 | 8 | INF | 47 |
| **3** | 0 | 11 | 0 | INF |

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
|  | **0** | **1** | **2** | **4** |
| **0** | INF | 0 | 8 | INF |
| **1** | 0 | INF | 2 | 64 |
| **2** | INF | 8 | INF | 47 |
| **3** | 0 | 11 | 0 | INF |

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
|  | **0** | **1** | **2** | **4** |
| **0** | INF | 0 | 8 | INF |
| **1** | 0 | INF | 2 | 64 |
| **2** | 0 | 8 | INF | 47 |
| **3** | INF | 11 | 0 | INF |

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
|  | **0** | **1** | **2** | **4** |
| **0** | INF | 0 | 8 | INF |
| **1** | 0 | INF | 2 | 64 |
| **2** | 0 | 8 | INF | 47 |
| **3** | 0 | 11 | INF | INF |

Видим, что наибольшей суммой минимальных значений по строке и по столбцу обладает матрица со знаком бесконечности в (0,4). Исключая это ребро, увеличиваем значение нижней границы.

R(4,3)(~~0,4~~)=105+47=152

Строим новую матрицу, где элемент (3,0) заменяем на бесконечность, чтобы исключить зацикливание. После производим процедуру полного её приведения. Получая общую сумму констант, увеличиваем нижнюю границу с дугами (5,4), (1,5) на это число.

|  |  |  |  |
| --- | --- | --- | --- |
|  | **0** | **1** | **2** |
| **1** | 0 | INF | 2 |
| **2** | 0 | 8 | INF |
| **3** | INF | 11 | 0 |

|  |
| --- |
| αi |
| 0 |
| 0 |
| 0 |

|  |  |  |  |
| --- | --- | --- | --- |
| βi | 0 | 8 | 0 |

|  |  |  |  |
| --- | --- | --- | --- |
|  | **0** | **1** | **2** |
| **1** | 0 | INF | 2 |
| **2** | 0 | 0 | INF |
| **3** | INF | 3 | 0 |

R(4,3)(0,4)=105+8=113

После чего, опять высчитываем сумму минимального элемента по строке и минимального элемента по столбцу, в которых и располагается знак бесконечности, заменяющий нули.

|  |  |  |  |
| --- | --- | --- | --- |
|  | **0** | **1** | **2** |
| **1** | INF | INF | 2 |
| **2** | 0 | 0 | INF |
| **3** | INF | 3 | 0 |

|  |  |  |  |
| --- | --- | --- | --- |
|  | **0** | **1** | **2** |
| **1** | 0 | INF | 2 |
| **2** | INF | 0 | INF |
| **3** | INF | 3 | 0 |

|  |  |  |  |
| --- | --- | --- | --- |
|  | **0** | **1** | **2** |
| **1** | 0 | INF | 2 |
| **2** | 0 | INF | INF |
| **3** | INF | 3 | 0 |

|  |  |  |  |
| --- | --- | --- | --- |
|  | **0** | **1** | **2** |
| **1** | 0 | INF | 2 |
| **2** | 0 | 0 | INF |
| **3** | INF | 3 | INF |

Видим, что наибольшей суммой минимальных значений по строке и по столбцу обладает матрица со знаком бесконечности в (3,2). Исключая это ребро, увеличиваем значение нижней границы.

R(4,3)(0,4)(~~3,2~~)=113+5=118

Строим новую матрицу, где элемент (2,0) заменяем на бесконечность, чтобы исключить зацикливание.

|  |  |  |
| --- | --- | --- |
|  | **0** | **1** |
| **1** | 0 | INF |
| **2** | INF | 0 |

Матрица уже приведенная, поэтому значение нижней границы остается неизменным.

R(4,3)(0,4)(3,2)=113+0=113

Таким образом, получаем следующие дуги: (4,3)(0,4)(4,2)(2,3)(3,1). Расставляя их в правильном порядке, получаем окончательное решение: (0,4)(4,3)(3,2)(2,1)(1,0) или 0–>4–>3–>2–>1–>0. Сложив расстояния, соответствующие дугам кольцевого маршрута, получим 121, что совпадает с нижней границей, приписанной последнему узлу графа.

(2,1)

(1,0)

Рис. 1. Граф решения **T** задачи коммивояжера

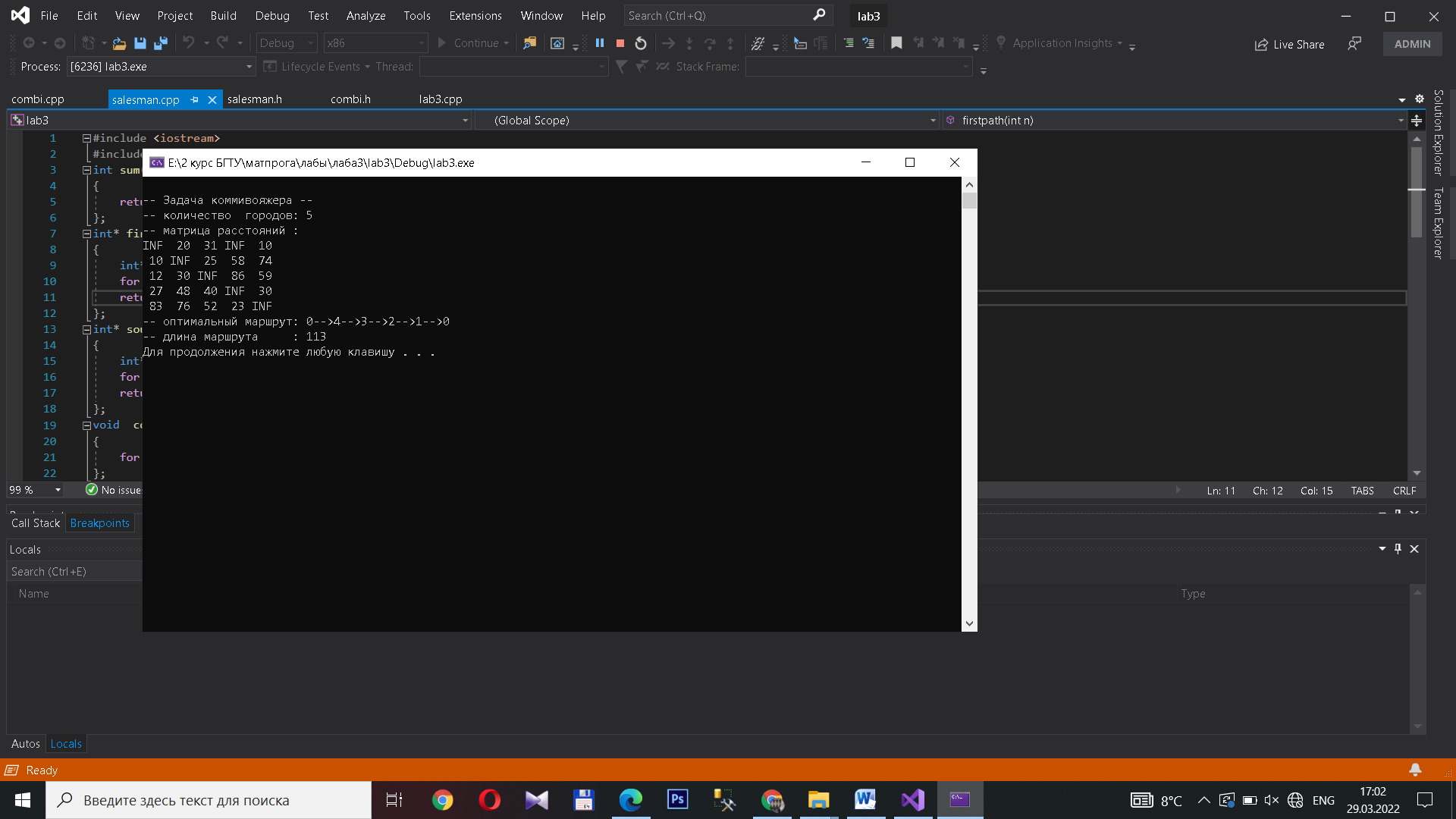


Рис. 2. Вывод значений задачи о коммивояжере с помощью генератора перестановок

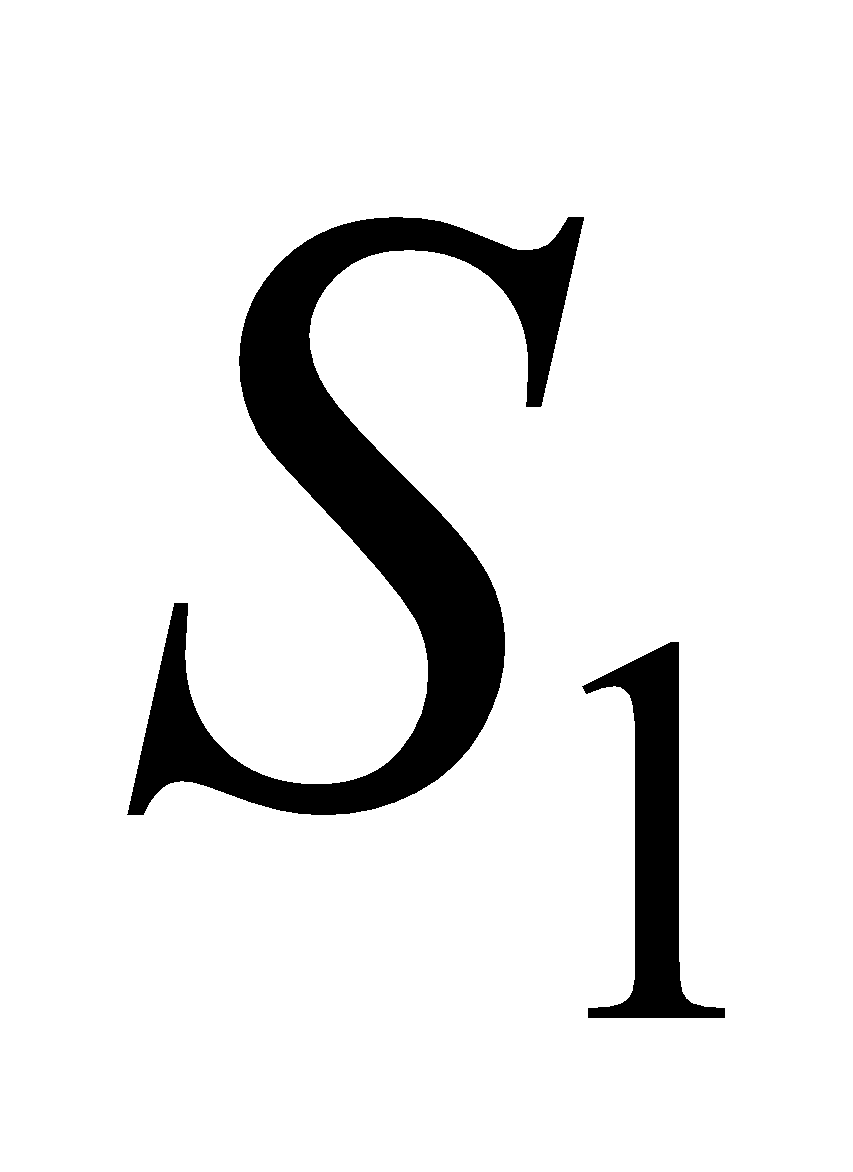
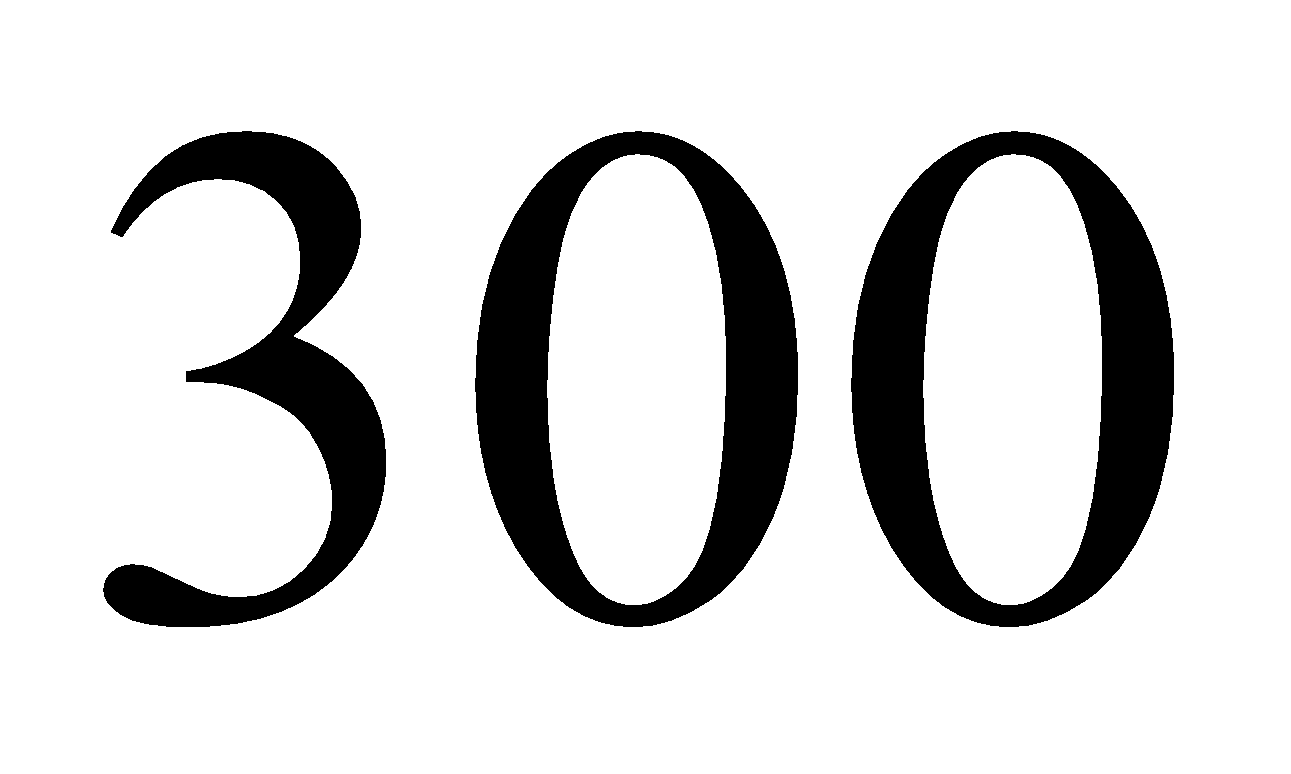
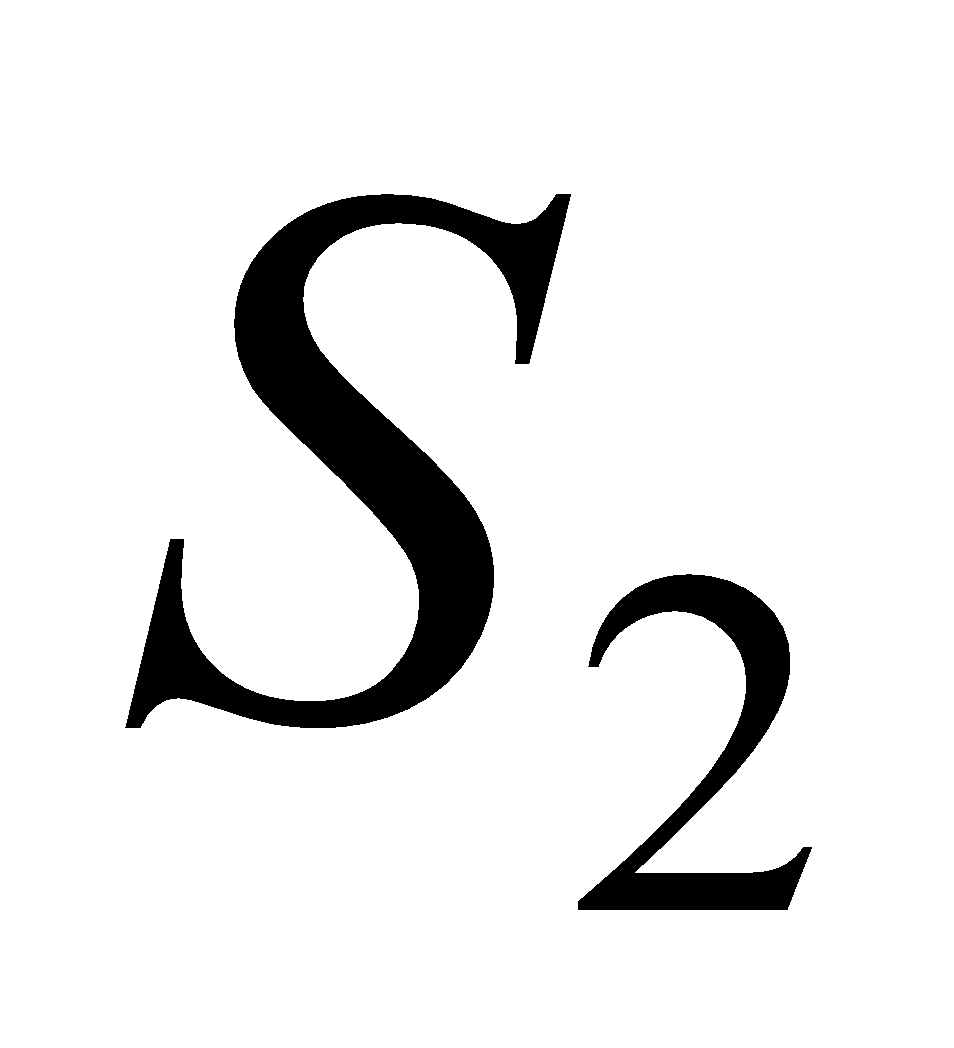
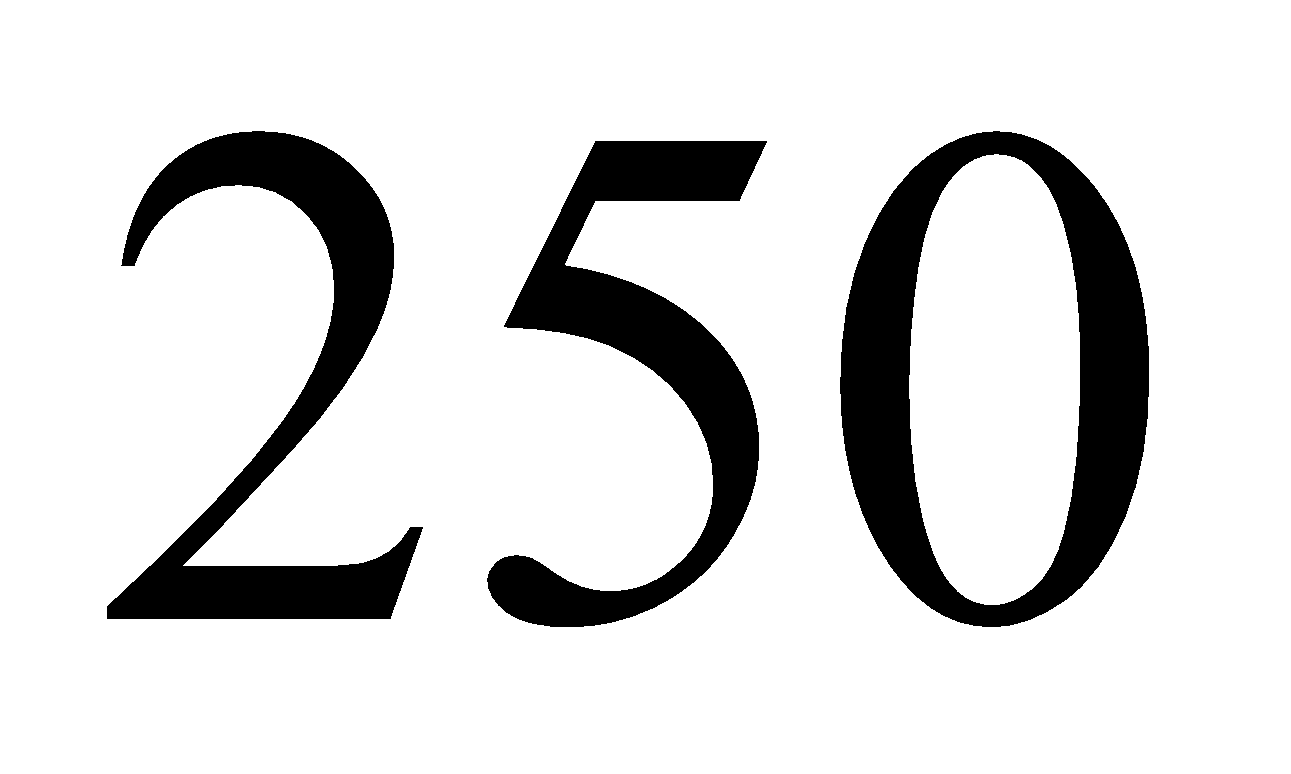
**Вывод:** в ходе лабораторной работы были освоены общие принципы решения задач методом ветвей и границ, решена задача о коммивояжере данным методом.

Лабораторная работа 4. **Динамическое программирование**

**ЦЕЛЬ РАБОТЫ:** освоить общие принципы решения задач методом динамического программирования, сравнить полученные решения задач с рекурсивным методом.

**ЗАДАНИЕ ДЛЯ ВЫПОЛНЕНИЯ**

**Задание 1.**

На языке С++ сгенерировать случайным образом строку букв латинского алфавита  длиной  символов и длиной .

**Листинг:**

int zad1()

{

setlocale(LC\_ALL, "eng");

clock\_t t1 = 0, t2 = 0, t3, t4;

unsigned int lenS1 = 300, lenS2 = 250;

float k[7] = { 0.04F, 0.05F, 0.067F, 0.1F, 0.2F, 0.5F, 1.0F };

char\* S1 = new char[lenS1],

\* S2 = new char[lenS2];

for (unsigned int i = 0; i < (lenS1 > lenS2 ? lenS1 : lenS2); i++)

{

if (i < lenS1)

S1[i] = (int)auxil::iget('a', 'z');

if (i < lenS2)

S2[i] = (int)auxil::iget('a', 'z');

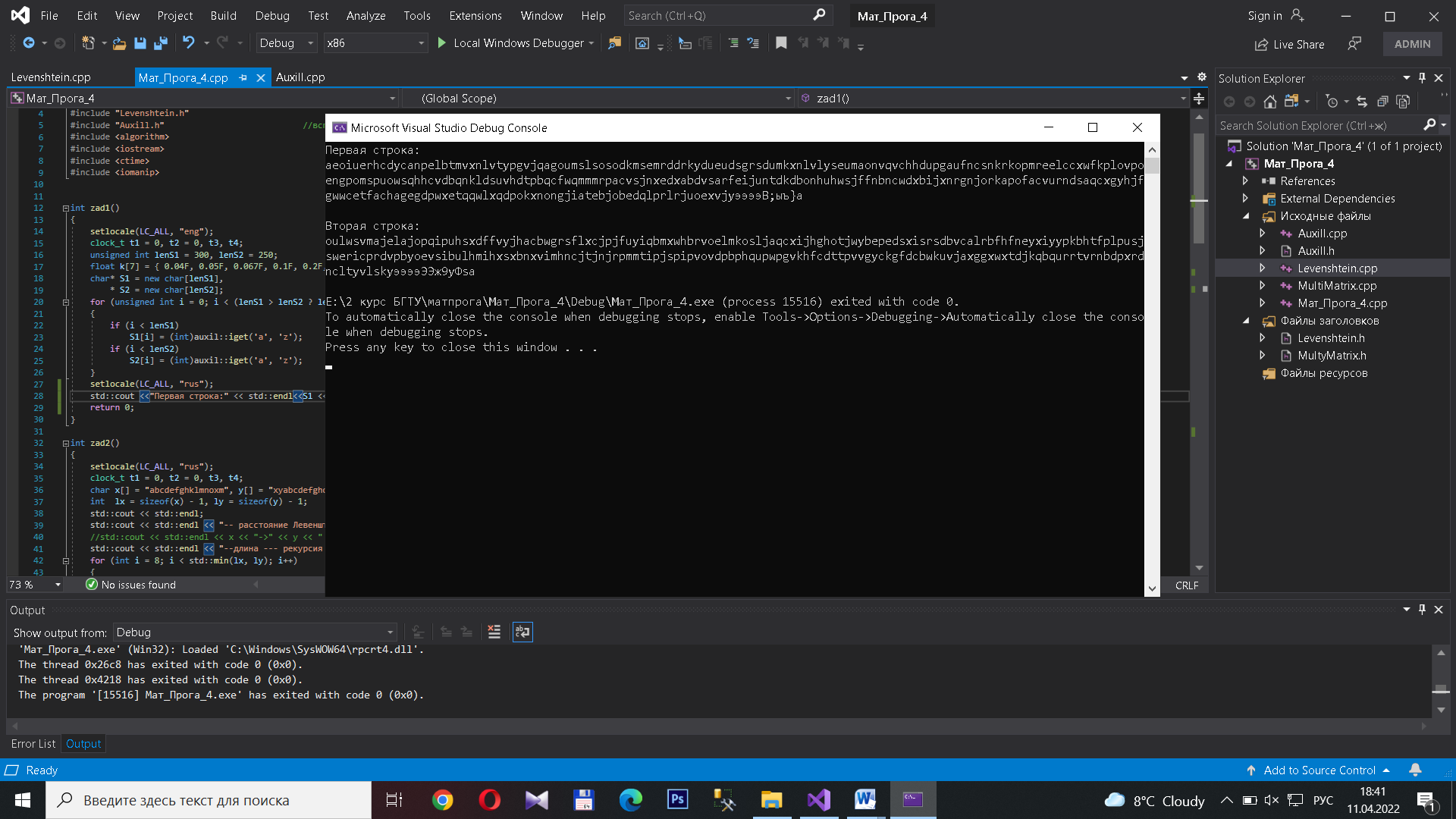
}

setlocale(LC\_ALL, "rus");

std::cout <<"Первая строка:" << std::endl<<S1 << std::endl << std::endl <<"Вторая строка:" << std::endl <<S2 << std::endl;

return 0;

}



**Задание 2.**

Вычислить двумя способами (рекурсивно и с помощью динамического программирования)  – дистанцию Левенштейна для , где - длина строки ,  - строка состоящая из первых  символов строки . (копии экрана и код вставить в отчет).

**Листинг:**

**Levenshtein.h**

// -- дистанции Левенштeйна (динамическое программирование)

int levenshtein(

int lx, // длина слова x

const char x[], // слово длиной lx

int ly, // длина слова y

const char y[] // слово y

);

// -- дистанции Левенштeйна (рекурсия)

int levenshtein\_r(

int lx, // длина строки x

const char x[], // строка длиной lx

int ly, // длина строки y

const char y[] // строка y

);

**Levenshtein.cpp**

#include "stdafx.h"

#include <iomanip>

#include <algorithm>

#include "Levenshtein.h"

#define DD(i,j) d[(i)\*(ly+1)+(j)]

int min3(int x1, int x2, int x3)

{ return std::min(std::min(x1,x2),x3); }

int levenshtein(int lx, const char x[],int ly, const char y[])

{

int\* d = new int[(lx + 1) \* (ly + 1)];

for (int i = 0; i <= lx; i++) DD(i, 0) = i;

for (int j = 0; j <= ly; j++) DD(0, j) = j;

for (int i = 1; i <= lx; i++)

for (int j = 1; j <= ly; j++)

{

DD(i, j) = min3(DD(i - 1, j) + 1, DD(i, j - 1) + 1,

DD(i - 1, j - 1) + (x[i - 1] == y[j - 1] ? 0 : 1));

}

return DD(lx, ly);

}

int levenshtein\_r(int lx, const char x[],

int ly, const char y[] )

{

int rc = 0;

if (lx == 0) rc = ly;

else if (ly == 0) rc = lx;

else if (lx == 1 && ly == 1 && x[0] == y[0]) rc = 0;

else if (lx == 1 && ly == 1 && x[0] != y[0]) rc = 1;

else rc = min3(

levenshtein\_r(lx-1, x, ly, y)+1,

levenshtein\_r(lx, x, ly-1, y)+1,

levenshtein\_r(lx-1, x, ly-1, y)+(x[lx-1] == y[ly-1]?0:1)

);

return rc;

};

**Main.cpp**

#include "stdafx.h"

#include <iostream>

#include <algorithm>

#include <ctime>

#include <iomanip>

#include <locale>

#include "Levenshtein.h"

using namespace std;

#define \_rand(min, max) ( rand() % ((max) - (min) + 1) + (min) )

int main()

{

setlocale(LC\_ALL, "rus");

srand(time(NULL));

char abc[25];

char s1[300];

char s2[250];

for (int i = 97, n = 0; i <= 122; ++i, ++n)

{

abc[n] = (char)i;

}

cout << "S1 = ";

for (int i = 0; i < 301; i++)

{

s1[i] = abc[\_rand(0, 24)];

if (i % 150 == 0)

cout << "\n";

cout << s1[i];

}

cout <<"\nS2 =";

for (int i = 0; i < 251; i++)

{

s2[i] = abc[\_rand(0, 24)];

if (i% 150 == 0)

cout << "\n";

cout << s2[i];

}

cout << "\n";

clock\_t t1 = 0, t2 = 0, t3, t4;

int lx = sizeof(s1) - 1 ;

int ly = sizeof(s2) - 1;

int S1\_size[]{ 12,15,20,30,60,150,300 };

int S2\_size[]{ 10,13,17,25,50,125,250 };

std::cout << "\n\n-- расстояние Левенштейна -----";

std::cout << "\n\n--длина --- рекурсия -- дин.програм. ---\n";

for (int i = 0; i < min(lx, ly); i++)

{

t1 = clock();

levenshtein\_r(S1\_size[i], s1 , S2\_size[i], s2);

t2 = clock();

t3 = clock();

levenshtein(S1\_size[i], s1, S2\_size[i], s2);

t4 = clock();

cout << right << setw(2) << S1\_size[i] << "/" << setw(2) << S2\_size[i]

<< " " << left << setw(10) << (t2 - t1)

<< " " << setw(10) << (t4 - t3) << endl;

}

system("pause");

return 0;

}

**Stdafx.h**

#pragma once

#include "targetver.h"

#include <stdio.h>

#include <tchar.h>

**Targetever.h**

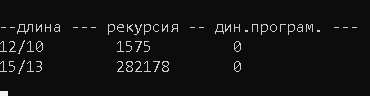
#include <SDKDDKVer.h>

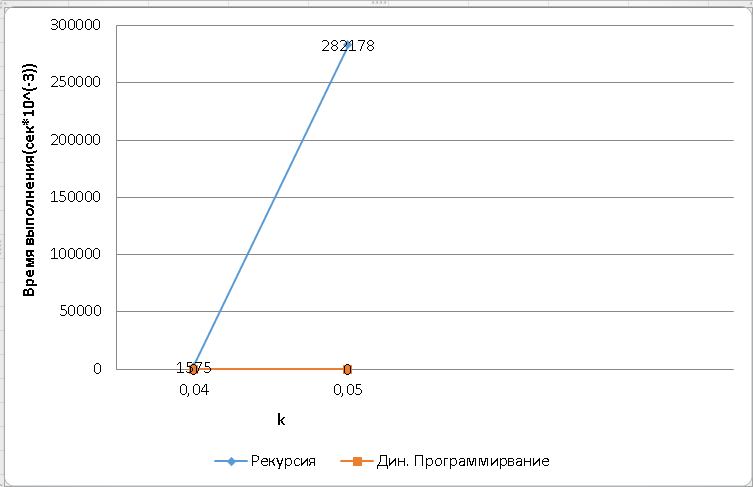
**Stdafx.cpp**

#include "stdafx.h"

**Задание 3.**

Выполнить сравнительный анализ времени, затраченного на вычисление дистанции Левенштейна для двух методов решения. Построить графики зависимости времени вычисления от . (копии экрана и график вставить в отчет).





|  |  |  |  |
| --- | --- | --- | --- |
| 11 | Ель | Дрель | 100\*15, 15\*20, 20\*43, 43\*70, 70\*40, 40\*71 |

**Задание 4.**

Реализовать вручную пример вычисления дистанции Левенштейна при помощи рекурсивного алгоритма (в соответствии с вариантом) (каждый шаг алгоритма по примеру из лекции вставить в отчет).

,

,

,

,

,

,

,

Восходящее вычисление:

**Задание 5. (нет решения, есть excel)**

**Четные варианты**. Выполнить сравнительный анализ времени затраченного на решение задачи об оптимальной расстановке скобок при умножении нескольких матриц для двух методов решения (рекурсивное решение, динамическое программирование). Размерность матриц взять в соответствии с вариантом. Объяснить в отчете принцип расставления скобок по итоговой матрице + код + копии экрана.

|  |
| --- |
| 100\*15, 15\*20, 20\*43, 43\*70, 70\*40, 40\*71 |

**matrix.срр**

#include "stdafx.h"

#include <cmath>

#include <memory.h>

#include <ctime>

#include <iostream>

#include "multyMatrix.h" // умножение матриц

#define N 6

int main()

{

clock\_t t1 = 0;

clock\_t t2 = 0;

clock\_t t3 = 0;

clock\_t t4 = 0;

int Mc[N + 1] = { 100,15,20,43,70,40,71 }, Ms[N][N], r = 0, rd = 0;

memset(Ms, 0, sizeof(int) \* N \* N);

t1 = clock();

r = OptimalM(1, N, N, Mc, OPTIMALM\_PARM(Ms));

t2 = clock();

setlocale(LC\_ALL, "rus");

std::cout << std::endl;

std::cout << std::endl << "-- расстановка скобок (рекурсивное решение) " << std::endl;

std::cout << std::endl << "затраченное время(сек): " << ((double)(t2 - t1)) / ((double)CLOCKS\_PER\_SEC) << std::endl;

std::cout << std::endl << "размерности матриц: ";

for (int i = 1; i <= N; i++) std::cout << "(" << Mc[i - 1] << "," << Mc[i] << ") ";

std::cout << std::endl << "минимальное количество операций умножения: " << r;

std::cout << std::endl << std::endl << "матрица S" << std::endl;

for (int i = 0; i < N; i++)

{

std::cout << std::endl;

for (int j = 0; j < N; j++) std::cout << Ms[i][j] << " ";

}

std::cout << std::endl;

memset(Ms, 0, sizeof(int) \* N \* N);

t3 = clock();

rd = OptimalMD(N, Mc, OPTIMALM\_PARM(Ms));

t4 = clock();

std::cout << std::endl

<< "-- расстановка скобок (динамичеое программирование) " << std::endl;

std::cout << std::endl << "затраченное время(сек): " << ((double)(t4 - t3)) / ((double)CLOCKS\_PER\_SEC) << std::endl;

std::cout << std::endl << "размерности матриц: ";

for (int i = 1; i <= N; i++)

std::cout << "(" << Mc[i - 1] << "," << Mc[i] << ") ";

std::cout << std::endl << "минимальное количество операций умножения: "

<< rd;

std::cout << std::endl << std::endl << "матрица S" << std::endl;

for (int i = 0; i < N; i++)

{

std::cout << std::endl;

for (int j = 0; j < N; j++) std::cout << Ms[i][j] << " ";

}

std::cout << std::endl << std::endl;

system("pause");

return 0;

}

**multyMatrix.срр**

#include "stdafx.h"

#include <memory.h>

#include "multyMatrix.h"

// расстановка скобок (рекурсия)

#define INFINITY 0x7fffffff

#define NINFINITY 0x80000000

int OptimalM(int i, int j, int n, const int c[], int\* s)

{

#define OPTIMALM\_S(x1,x2) (s[(x1-1)\*n+x2-1])

int o = INFINITY;

int bo = INFINITY;

if (i < j)

{

for (int k = i; k < j; k++)

{

bo = OptimalM(i, k, n, c, s) + OptimalM(k + 1, j, n, c, s) + c[i - 1] \* c[k] \* c[j];

if (bo < o)

{

o = bo;

OPTIMALM\_S(i, j) = k;

}

}

}

else o = 0;

return o;

#undef OPTIMALM\_S

};

// расстановка скобок (динамическое программирование)

int OptimalMD(int n, const int c[], int\* s)

{

#define OPTIMALM\_S(x1,x2) (s[(x1-1)\*n+x2-1])

#define OPTIMALM\_M(x1,x2) (M[(x1-1)\*n+x2-1])

int\* M = new int[n \* n], j = 0, q = 0;

for (int i = 1; i <= n; i++)

OPTIMALM\_M(i, i) = 0;

for (int l = 2; l <= n; l++)

{

for (int i = 1; i <= n - l + 1; i++)

{

j = i + l - 1;

OPTIMALM\_M(i, j) = INFINITY;

for (int k = i; k <= j - 1; k++)

{

q = OPTIMALM\_M(i, k) + OPTIMALM\_M(k + 1, j) + c[i - 1] \* c[k] \* c[j];

if (q < OPTIMALM\_M(i, j))

{

OPTIMALM\_M(i, j) = q; OPTIMALM\_S(i, j) = k;

}

}

}

}

return OPTIMALM\_M(1, n);

#undef OPTIMALM\_M

#undef OPTIMALM\_S

};

**multyMatrix.h**

#pragma once

// расстановка скобок при умножении матриц

// функции возвращают минимальное количество операций умножения

#define OPTIMALM\_PARM(x) ((int\*)x) // для представления 2мерного массива

int OptimalM( // рекурсия

int i, // [in] номер первой матрицы

int j, // [in] номер последней матрицы

int n, // [in] количество матриц

const int c[], // [in] массив размерностей

int\* s // [out] результат: позиции скобок

);

int OptimalMD( // динамическое программирование

int n, // [in] количество матриц

const int c[], // [in] массив размерностей

int\* s // [out] результат: позиции скобок

);

**Stdafx.h**

#pragma once

#include "targetver.h"

#include <stdio.h>

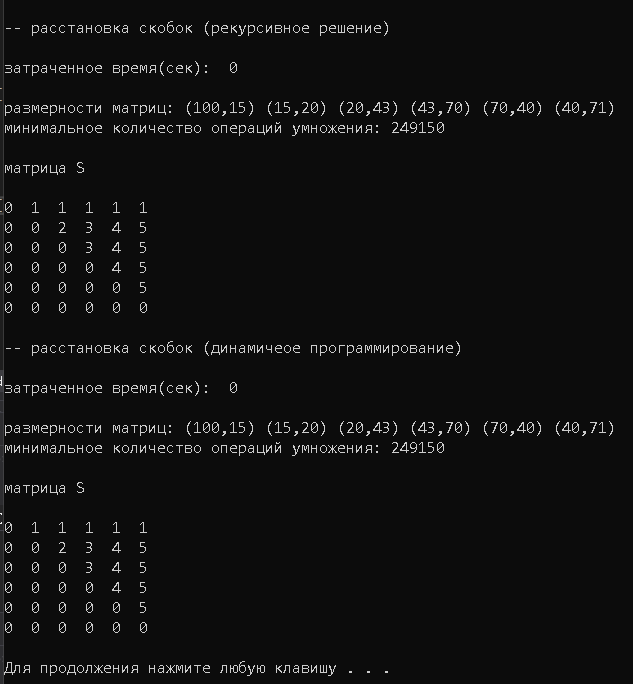
#include <tchar.h>

**Targetever.h**

#include <SDKDDKVer.h>

**Stdafx.cpp**

#include "stdafx.h"



На месте 1;6 стоит 1, значит ставим скобку после A1

A1×(A2×A3×A4×A5×A6)

На месте 2;6 стоит 5, значит ставим скобку после A5

A1×((A2×A3×A4×A5)×A6)

На месте 2;5 стоит 4, значит ставим скобку после A4

A1×(((A2×A3×A4)×A5)×A6)

На месте 2;4 стоит 3, значит ставим скобку после A3

A1×((((A2×A3)×A4)×A5)×A6)=

(15\*20\*43)+(15\*43\*70)+(15\*70\*40)+(15\*40\*71)+(15\*71\*100)=

12900+45165+42000+42600+106500=249165-минимальное кол-в операций умножения

Лабораторная работа 5. **АЛГОРИТМЫ НА ГРАФАХ**

**(алгоритмы поиска в ширину и глубину, топологическая сортировка)**

**ЦЕЛЬ РАБОТЫ:** освоить сущность и программную реализацию: а) способов представления графов; б) алгоритмов поиска в ширину и глубину; в) алгоритма топологической сортировки графов.

**ЗАДАНИЕ ДЛЯ ВЫПОЛНЕНИЯ**

**Задание 1.**

Ориентированный граф G взять в соответствии с вариантом. Осуществить алгоритмы поиска в ширину и глубину, а также алгоритма топологической сортировки аналогично примерам, рассмотренным на лекциях. Оформить отчет, включив в него каждый шаг выполнения алгоритмов.

|  |  |
| --- | --- |
| 10 |  |

Текущее состояние алгоритма хранится в следующих структурах памяти:

Q – очередь вершин,

C – массив окраски вершин,

D – массив расстояний,

P – массив предшествующих вершин.

**Алгоритм поиска в ширину**

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| Q | 0 |  |  |  |  |  |  |
| C | G | W | W | W | W | W | W |
| D | 0 | I | I | I | I | I | I |
| P | N | N | N | N | N | N | N |

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| Q | 1 |  |  |  |  |  |
| C | B | G | W | W | W | W | W |
| D | 0 | 1 | I | I | I | I | I |
| P | N | 0 | N | N | N | N | N |

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| Q | 3 | 4 |  |  |  |
| C | B | B | W | G | G | W | W |
| D | 0 | 1 | I | 2 | 2 | I | I |
| P | N | 0 | N | 1 | 1 | N | N |

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| Q | 4 | 2 | 5 | 6 |  |  |
| C | B | B | G | B | G | G | G |
| D | 0 | 1 | 3 | 2 | 2 | 3 | 3 |
| P | N | 0 | 3 | 1 | 1 | 3 | 3 |

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| Q | 2 | 5 | 6 |  |  |
| C | B | B | G | B | B | G | G |
| D | 0 | 1 | 3 | 2 | 2 | 3 | 3 |
| P | N | 0 | 3 | 1 | 1 | 3 | 3 |

|  |  |  |  |
| --- | --- | --- | --- |
| Q | 5 | 6 |  |
| C | B | B | B | B | B | G | G |
| D | 0 | 1 | 3 | 2 | 2 | 3 | 3 |
| P | N | 0 | 3 | 1 | 1 | 3 | 3 |

|  |  |  |
| --- | --- | --- |
| Q | 6 |  |
| C | B | B | B | B | B | B | G |
| D | 0 | 1 | 3 | 2 | 2 | 3 | 3 |
| P | N | 0 | 3 | 1 | 1 | 3 | 3 |

|  |
| --- |
| Q |
| C | B | B | B | B | B | B | B |
| D | 0 | 1 | 3 | 2 | 2 | 3 | 3 |
| P | N | 0 | 3 | 1 | 1 | 3 | 3 |

Таким образом, BFS-дерево имеет вид: 0134256

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| P | N | 0 | 0 | 2 | 1 | 3 | 3 |

**Алгоритм поиска в глубину**

Текущее состояние алгоритма хранится в следующих структурах памяти:

t – номер шага алгоритма,

C – массив окраски вершин,

D – массив времени обнаружения (шаг окраски в серый цвет) вершины,

P – массив предшествующих вершин,

F – массив хранения времени фиксации (шаг окраски в чёрный цвет) вершины.

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| t | 1 |  |  |  |  |  |  |
| C | G | W | W | W | W | W | W |
| D | 1 | I | I | I | I | I | I |
| P | N | N | N | N | N | N | N |
| F | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| t | 2 |  |  |  |  |  |  |
| C | G | G | W | W | W | W | W |
| D | 1 | 2 | I | I | I | I | I |
| P | N | 0 | N | N | N | N | N |
| F | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| t | 3 |  |  |  |  |  |  |
| C | G | G | W | G | W | W | W |
| D | 1 | 2 | I | 3 | I | I | I |
| P | N | 0 | N | 1 | N | N | N |
| F | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| t | 4 |  |  |  |  |  |  |
| C | G | G | G | G | W | W | W |
| D | 1 | 2 | 4 | 3 | I | I | I |
| P | N | 0 | 3 | 1 | N | N | N |
| F | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| t | 5 |  |  |  |  |  |  |
| C | G | G | B | G | W | W | W |
| D | 1 | 2 | 5 | 3 | I | I | I |
| P | N | 0 | 3 | 1 | N | N | N |
| F | 0 | 0 | 5 | 0 | 0 | 0 | 0 |

(пропускает шаги возвращения через 3 и 1 в 4)

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| t | 6 |  |  |  |  |  |  |
| C | G | G | B | G | G | W | W |
| D(t) | 1 | 2 | 5 | 3 | 6 | I | I |
| P | N | 0 | 3 | 1 | 1 | N | N |
| F | 0 | 0 | 5 | 0 | 0 | 0 | 0 |

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| t | 7 |  |  |  |  |  |  |
| C | G | G | B | G | B | W | W |
| D | 1 | 2 | 5 | 3 | 7 | I | I |
| P | N | 0 | 3 | 1 | 1 | N | N |
| F | 0 | 0 | 5 | 0 | 7 | 0 | 0 |

(пропускает шаги возвращения из 1 и 3 в 5)

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| t | 8 |  |  |  |  |  |  |
| C | G | G | B | G | B | G | W |
| D | 1 | 2 | 5 | 3 | 7 | 8 | I |
| P | N | 0 | 3 | 1 | 1 | 3 | N |
| F | 0 | 0 | 5 | 0 | 7 | 0 | 0 |

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| t | 9 |  |  |  |  |  |  |
| C | G | G | B | G | B | B | W |
| D | 1 | 2 | 5 | 3 | 7 | 9 | I |
| P | N | 0 | 3 | 1 | 1 | 3 | N |
| F | 0 | 0 | 5 | 0 | 7 | 9 | 0 |

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| t | 10 |  |  |  |  |  |  |
| C | G | G | B | G | B | B | G |
| D | 1 | 2 | 5 | 3 | 7 | 9 | 10 |
| P | N | 0 | 3 | 1 | 1 | 3 | 3 |
| F | 0 | 0 | 5 | 0 | 7 | 9 | 0 |

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| t | 11 |  |  |  |  |  |  |
| C | G | G | B | G | B | B | B |
| D | 1 | 2 | 5 | 3 | 7 | 9 | 11 |
| P | N | 0 | 3 | 1 | 1 | 3 | 3 |
| F | 0 | 0 | 5 | 0 | 7 | 9 | 11 |

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| t | 12 |  |  |  |  |  |  |
| C | G | G | B | B | B | B | B |
| D | 1 | 2 | 5 | 12 | 7 | 9 | 11 |
| P | N | 0 | 3 | 1 | 1 | 3 | 3 |
| F | 0 | 0 | 5 | 12 | 7 | 9 | 11 |

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| t | 13 |  |  |  |  |  |  |
| C | G | B | B | B | B | B | B |
| D | 1 | 12 | 5 | 12 | 7 | 9 | 11 |
| P | N | 0 | 3 | 1 | 1 | 3 | 3 |
| F | 0 | 12 | 5 | 12 | 7 | 9 | 11 |

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| t | 14 |  |  |  |  |  |  |
| C | B | B | B | B | B | B | B |
| D | 13 | 12 | 5 | 12 | 7 | 9 | 11 |
| P | N | 0 | 3 | 1 | 1 | 3 | 3 |
| F | 13 | 12 | 5 | 12 | 7 | 9 | 11 |

Таким образом, DFS-дерево имеет вид 2456301

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| P | N | 0 | 3 | 1 | 1 | 3 | 3 |

**Алгоритм топологической сортировки:**

1/

2/

1/

2/

1/

2/

1/

4/

3/

3/

1/

2/

2/

1/

8/

6/7

3/

2/

4/5

1/

6/7

4/5

3/

2/

1/

10/

8/9

6/7

3/

4/5

2/

1/

6/7

4/5

8/9

3/

1/

2/

8/

6/7

4/5

3/

1/

2/

6/7

1/

4/5

3/

2/

6

4/5

3/

3/

4/5

3/

8/9

10/11

6/7

3/12

2/13

4/5

1/14

10/11

10/11

6/7

8/9

3/12

8/9

2/13

1/

4/5

3/12

10/11

6/7

2/

1/

4/5

8/9

4/5

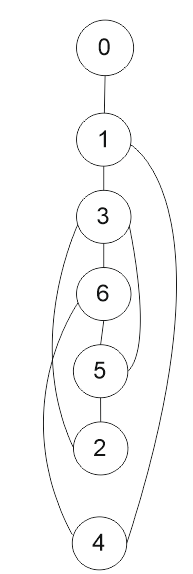
3/

6/7

2/

1/

|  |
| --- |
| 0 |
| 1 |
| 3 |
| 6 |
| 5 |
| 4 |
| 2 |



**Задание 2.** Осуществить программную реализацию алгоритмов на C++. Разработать структуры **AMatrix** и **АList**  для представления ориентированного графа матричным и списковым способом. Разработать функции преобразования из одного способа представления в другой. Разработать функцию **BFS** обхода вершин графа, используя метод поиска в ширину. Продемонстрировать работу функции. Копии экрана вставить в отчет.

**Main.cpp**

#include "stdafx.h"

#include <iostream>

#include "Graph.h"

#include "BFS.h"

#include "DFS.h"

int \_tmain(int argc, \_TCHAR\* argv[])

{

int m[7][7] ={

{0, 0, 1, 0, 0, 0, 0},

{0, 0, 0, 1, 1, 0, 0},

{0, 0, 0, 0, 0, 0, 0},

{0, 0, 1, 0, 0, 1, 1},

{0, 0, 1, 0, 0, 0, 0},

{0, 0, 0, 0, 1, 1, 0},

{0, 0, 0, 0, 0, 0, 0}

};

setlocale(LC\_ALL, "rus");

graph::AMatrix g1(7,(int\*) m);

std::cout<<std::endl;

std::cout<<std::endl<< "-- матрица смежности "<< std::endl;

for (int i = 0; i < g1.nV; i++)

{

std::cout<<std::endl;

for (int j = 0; j < g1.nV; j++)

std::cout<< g1.get(i,j) << " ";

};

std::cout<<std::endl;

graph::AList g2(g1);

std::cout<<std::endl;

std::cout<<std::endl<< "-- списки смежных вершин "<< std::endl;

for (int i = 0; i < g1.nV; i++)

{

std::cout<<std::endl<< i << ": ";

for (int j = 0; j < g2.size(i); j++)

std::cout<<g2.get(i,j)<< " ";

}

std::cout<<std::endl;

BFS b1(g2,0);

std::cout<<std::endl;

std::cout<<std::endl<< "-- поиск в ширину "<< std::endl;

int k1;

while ((k1 = b1.get())!= BFS::NIL)

std::cout<< k1 << " ";

std::cout<<std::endl;

DFS b2(g2);

std::cout<<std::endl;

std::cout<<std::endl<< "-- поиск в глубину "<< std::endl;

for (int i = 0; i < g2.nV; i++)

std::cout<< b2.get(i) << " ";

std::cout<<std::endl;

std::cout << std::endl << "Топологическая сортировка" << std::endl;

for (std::vector <int>::iterator i (b2.topological\_sort.begin ()); i != b2.topological\_sort.end (); ++i)

std::cout << \*i << ' ';

std::cout << std::endl;

system("pause");

return 0;

}

**Graph.cpp**

#include "stdafx.h"

#include "Graph.h"

namespace graph

{

AMatrix::AMatrix(int n, int mr[])

{

this->nV = n;

this->mr = mr;

};

AMatrix::AMatrix(const AList& al)

{

this->nV = al.nV;

this->mr = new int [this->nV \* this->nV];

for(int k = 0; k < this->nV\*this->nV; k++)

mr[k] = 0;

for (int i = 0; i < this->nV; i++)

for (int j = 0; j < al.size(i); j++)

this->set(i,al.get(i,j),1);

};

void AMatrix::set(int i, int j, int r)

{

this->mr[i\*this->nV+j] = r;

};

int AMatrix::get(int i, int j)const

{

return this->mr[i\*this->nV+j];

};

//------------List

void AList::create(int n)

{

this->mr = new std::list<int>[this->nV = n];

};

AList::AList(const AMatrix& am)

{

this->create(am.nV);

for (int i = 0; i < this->nV; i++)

for (int j = 0; j < this->nV; j++)

if (am.get(i,j)!=0)

this->add(i,j);

};

AList::AList(int n, int mr[])

{

this->create(n);

for (int i = 0; i < this->nV; i++)

for (int j = 0; j < this->nV; j++)

if (mr[i\*this->nV+j]!= 0)

this->add(i,j);

};

void AList::add(int i, int j)

{

this->mr[i].push\_back(j);

};

int AList::size(int i) const

{

return (int)this->mr[i].size();

};

int AList::get(int i, int j)const

{

std::list<int>::iterator rc = this->mr[i].begin();

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

rc++;

return (int)\*rc;

};

};

**Graph.h**

#pragma once

#include <list>

#include "Graph.h"

namespace graph

{

struct AList;

struct AMatrix // матрица смежности

{

int nV; // количество вершин

int \*mr; // матрица

AMatrix(int n, int mr[]); // создать матрицу n\*n и

AMatrix(const AList& al); // создать матрицу из спискового

void set(int i, int j, int r); // записать mr[i,j] = r

int get(int i, int j)const; // элемент mr[i,j]

};

struct AList // списки смежности

{

int nV; // количество вершин

std::list<int> \*mr; // массив списков

void create(int n); // создать массив пустых списков

AList(int n, int mr[]); // создать списковое представление

AList(const AMatrix& am); // создать списковое представление

void add(int i, int j); // добавить в i-ый список

int size(int i) const; // размер i-го списка

int get(int i, int j)const; // j-ый элемент i-го списка

};

};

**BFS.cpp**

#include "stdafx.h"

#include "BFS.h"

void BFS::init(const graph::AList& al, int s)

{

this->al = &al;

this->c = new Color[this->al->nV];

this->d = new int[this->al->nV];

this->p = new int[this->al->nV];

for (int i = 0; i < this->al->nV; i++)

{

this->c[i] = WHITE;

this->d[i] = INF;

this->p[i] = NIL;

};

this->c[s] = GRAY;

this->q.push(s);

};

BFS::BFS(const graph::AList& al, int s)

{

this->init(al,s);

};

BFS::BFS(const graph::AMatrix& am, int s)

{

this->init(\*(new graph::AList(am)),s);

};

int BFS::get()

{

int rc = NIL, v = NIL;

if (!this->q.empty())

{

rc = this->q.front();

for (int j = 0; j < this->al->size(rc); j++)

if (this->c[v = this->al->get(rc,j)] == WHITE)

{

this->c[v] = GRAY;

this->d[v] = this->d[rc]+1;

this->p[v] = rc;

this->q.push(v);

};

this->q.pop();

this->c[rc] = BLACK;

};

return rc;

}

**BFS.h**

#pragma once

#include "Graph.h"

#include <queue>

struct BFS // breadth-first search поиск в ширину (связный граф)

{

const static int INF = 0x7fffffff;

const static int NIL = -1;

enum Color {WHITE, GRAY, BLACK}; //

const graph::AList \*al; // исходный граф

Color \*c; // цвет вершины

int \*d; // расстояние до вершины

int \*p; // предшествующая вершина

std::queue<int> q; // очередь

BFS(const graph::AList& al, int s);

BFS(const graph::AMatrix& am, int s);

void init(const graph::AList& al, int s);

int get(); // получить следующую вершину

};

**DFS.cpp**

#include "stdafx.h"

#include "DFS.h"

#define NINF 0x80000000

#define INF 0x7fffffff

void DFS::init(const graph::AList& al)

{

this->al = &al;

this->c = new Color[this->al->nV];

this->d = new int[this->al->nV];

this->f = new int[this->al->nV];

this->p = new int[this->al->nV];

this->t = 0;

for (int i = 0; i < this->al->nV; i++)

{

this->c[i] = WHITE;

this->d[i] = this->f[i] = 0;

this->p[i] = NIL;

};

for (int i = 0; i < this->al->nV; i++)

if (this->c[i] == WHITE)

{

this->visit(i);

this->topological\_sort.push\_back (i);

}

};

DFS::DFS(const graph::AList& al)

{

this->init(al);

};

DFS::DFS(const graph::AMatrix& am)

{

this->init(\*(new graph::AList(am)));

};

void DFS::visit(int u)

{

int v = NIL;

this->c[u] = GRAY;

this->d[u]= ++(this->t);

for (int j = 0; j < this->al->size(u); j++)

if (this->c[v = this->al->get(u,j)] == WHITE)

{

this->p[v] = u;

this->visit(v);

this->topological\_sort.push\_back (v);

}

this->c[u] = BLACK;

this->f[u]= ++(this->t);

};

int DFS::get(int i)

{

int j = 0, min1 = INF , min2 = NINF, ntx = NIL;

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

{

for (int k = 0; k < this->al->nV; k++)

if (this->f[k] < min1 && this->f[k] > min2)

{

min1 = this->f[k];

ntx = k;

};

min2 = min1; min1 = INF;

};

return ntx;

};

**DFS.h**

#pragma once

#include "Graph.h"

#include <vector>

struct DFS // depth-first search поиск в глубину

{

const static int NIL = -1;

enum Color { WHITE, GRAY, BLACK }; //

const graph::AList \*al; // исходный граф

Color \*c; // цвет вершины

int \*d; // время обнаружения

int \*f; // время завершения обработки

int \*p; // предшествующая вершина

int t; // текущее время

DFS(const graph::AList& al);

DFS(const graph::AMatrix& am);

std::vector <int> topological\_sort; //результат топологической сортировки

void visit(int v);

void init(const graph::AList& al);

int get(int i); // получить вершину

};



Лабораторная работа 6. Транспортная задача. Вариант 10

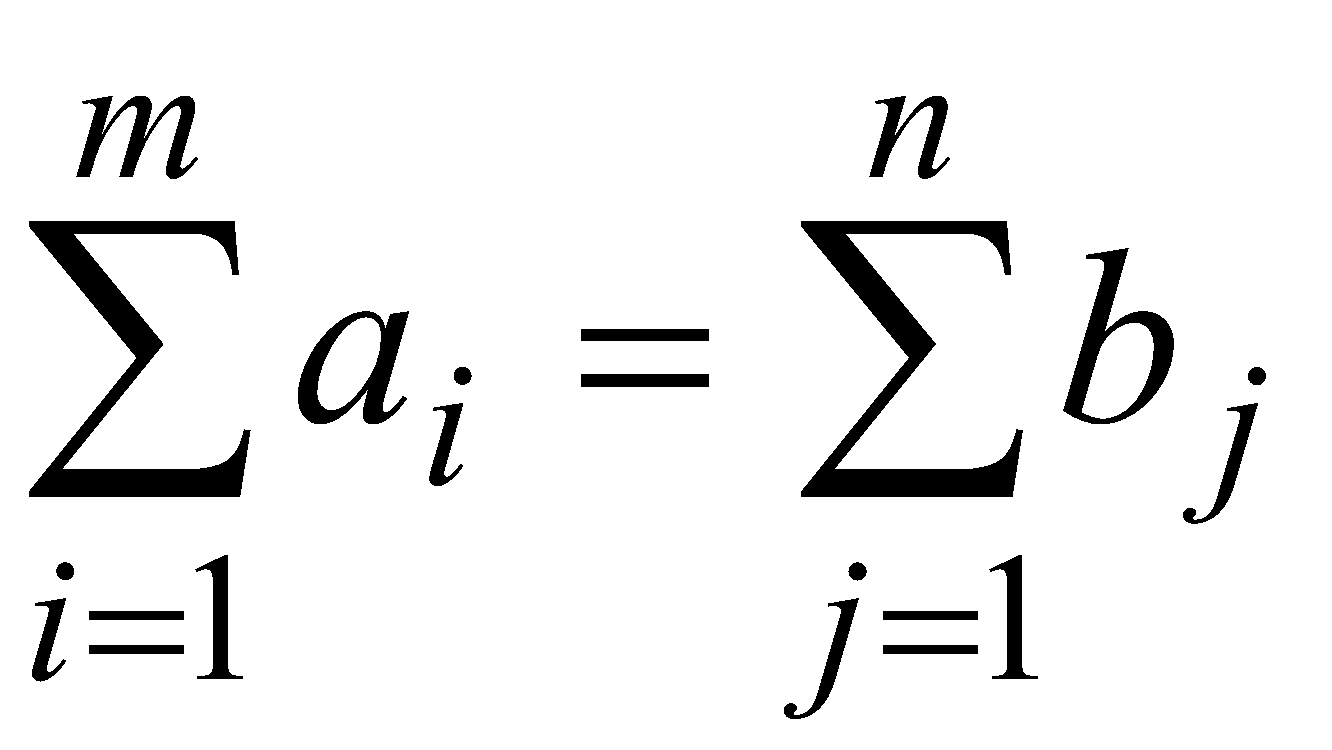
**ЦЕЛЬ РАБОТЫ:** Приобретение навыков решения открытой транспортной задачи

**ЗАДАНИЕ ДЛЯ ВЫПОЛНЕНИЯ**

**Задание.** Решить транспортную задачу. Имеется 5 поставщиков продукции и 6 потребителей. Величина запасов, потребностей и стоимость затрат на перевозку продукции взять в соответствии с вариантом (*N*). Оформить отчет.

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| ПОТРЕБИТЕЛИ  ПОСТАВЩИКИ | 1 | 2 | 3 | 4 | 5 | 6 | ЗАПАСЫ |
| 1 | **22** | **12** | **16** | **13** | **21** | **11** | **178** |
| 2 | **20** | **10** | **18** | **15** | **17** | **23** | **123** |
| 3 | **11** | **15** | **21** | **18** | **12** | **21** | **160** |
| 4 | **14** | **20** | **20** | **13** | **23** | **12** | **169** |
| 5 | **13** | **21** | **19** | **10** | **20** | **14** | **110** |
| ПОТРЕБНОСТИ | **153** | **117** | **141** | **203** | **105** | **173** |  |

Для решения транспортной задачи необходимо, чтобы выполнялось условие:

,

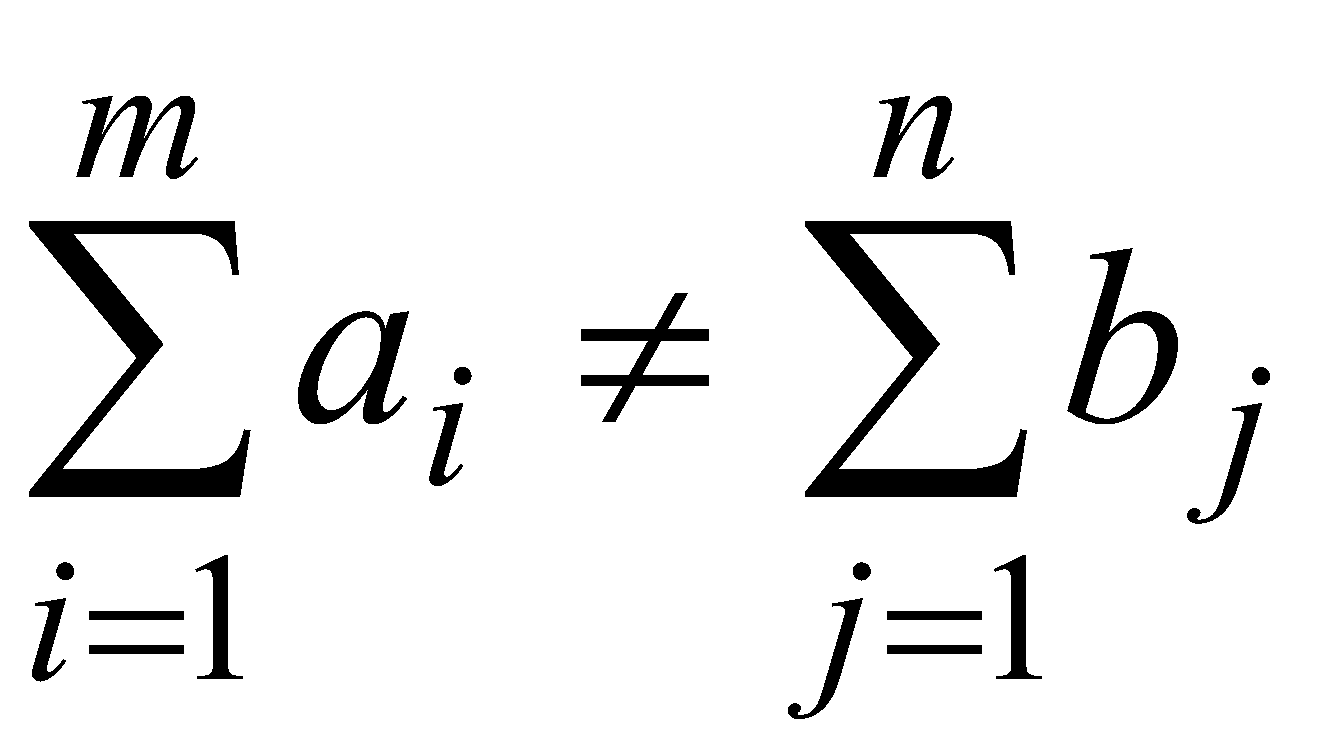
где задача называется закрытой.

Проверим это условие:

∑a=178+123+160+169+110=740

∑b=153+117+141+203+105+173=892

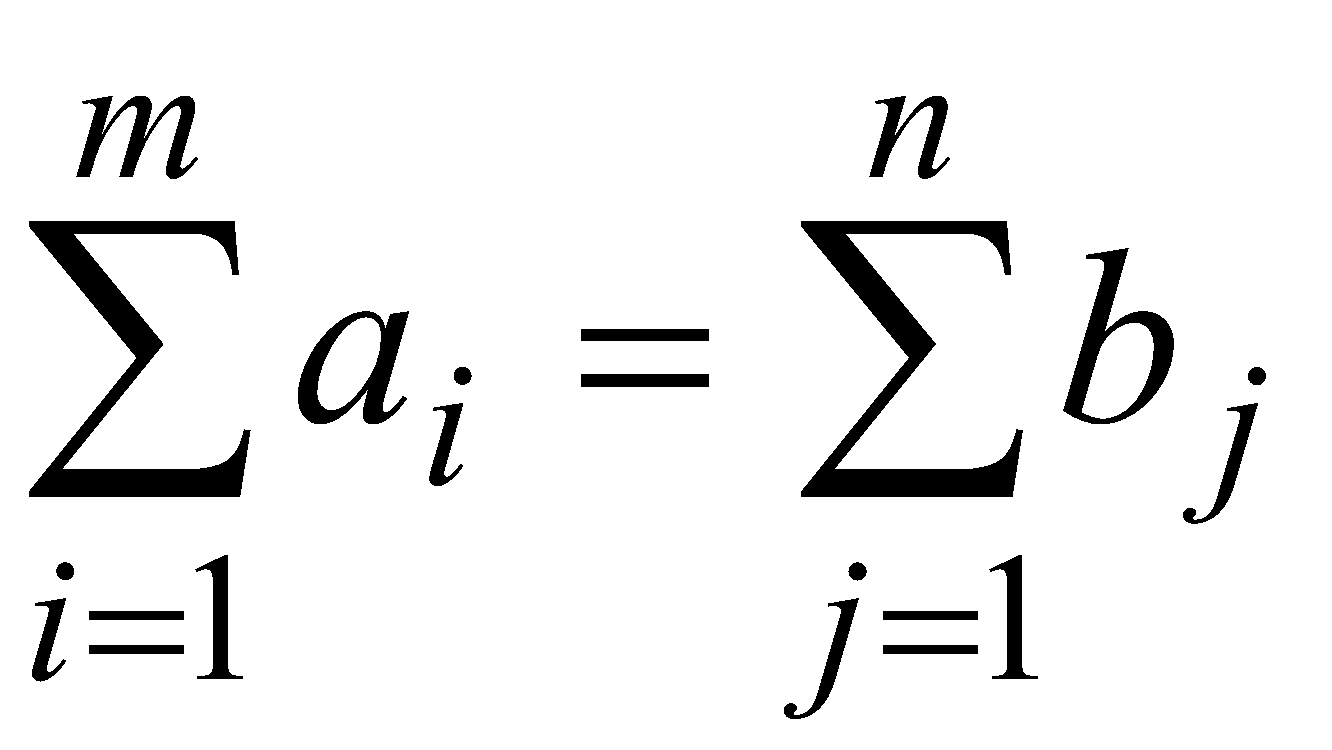
Имеем, что:



Значит, задача называется открытой (с нарушенным балансом). Решение открытой задачи сводится к решению закрытой. С этой целью, имея ∑a < ∑b, добавляем фиктивного поставщика с запасом b-a, а соответствующие тарифы перевозок полагаем равными нулю.

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| ПОТРЕБИТЕЛИ  ПОСТАВЩИКИ | 1 | 2 | 3 | 4 | 5 | 6 | ЗАПАСЫ |
| 1 | **22** | **12** | **16** | **13** | **21** | **11** | **178** |
| 2 | **20** | **10** | **18** | **15** | **17** | **23** | **123** |
| 3 | **11** | **15** | **21** | **18** | **12** | **21** | **160** |
| 4 | **14** | **20** | **20** | **13** | **23** | **12** | **169** |
| 5 | **13** | **21** | **19** | **10** | **20** | **14** | **110** |
| 6 | **0** | **0** | **0** | **0** | **0** | **0** | **152** |
| ПОТРЕБНОСТИ | **153** | **117** | **141** | **203** | **105** | **173** | **892** |

Теперь выполняется условие:



* **Метод наименьшей стоимости.**

1. Выбор ячейки с наименьшим значением c22=10

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| ПОТРЕБИТЕЛИ  ПОСТАВЩИКИ | 1 | 2 | 3 | 4 | 5 | 6 | ЗАПАСЫ |
| 1 | **22** | **12** | **16** | **13** | **21** | **11** | **178** |
| 2 | **20** | **10** | **18** | **15** | **17** | **23** | **123** |
| 3 | **11** | **15** | **21** | **18** | **12** | **21** | **160** |
| 4 | **14** | **20** | **20** | **13** | **23** | **12** | **169** |
| 5 | **13** | **21** | **19** | **10** | **20** | **14** | **110** |
| 6 | **0** | **0** | **0** | **0** | **0** | **0** | **152** |
| ПОТРЕБНОСТИ | **153** | **117** | **141** | **203** | **105** | **173** | **892** |

1. x22 = min (123, 117) = 117

Выбор ячейки с наименьшим значением c54=10

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| ПОТРЕБИТЕЛИ  ПОСТАВЩИКИ | 1 | 2 | 3 | 4 | 5 | 6 | ЗАПАСЫ |
| 1 | **22** | **12** | **16** | **13** | **21** | **11** | **178** |
| 2 | **20** | **10|117** | **18** | **15** | **17** | **23** | **6** |
| 3 | **11** | **15** | **21** | **18** | **12** | **21** | **160** |
| 4 | **14** | **20** | **20** | **13** | **23** | **12** | **169** |
| 5 | **13** | **21** | **19** | **10** | **20** | **14** | **110** |
| 6 | **0** | **0** | **0** | **0** | **0** | **0** | **152** |
| ПОТРЕБНОСТИ | **153** | **0** | **141** | **203** | **105** | **173** | **892** |

1. x54 = min (110, 203) = 110

Выбор ячейки с наименьшим значением c16=11

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| ПОТРЕБИТЕЛИ  ПОСТАВЩИКИ | 1 | 2 | 3 | 4 | 5 | 6 | ЗАПАСЫ |
| 1 | **22** | **12** | **16** | **13** | **21** | **11** | **178** |
| 2 | **20** | **10|118** | **18** | **15** | **17** | **23** | **6** |
| 3 | **11** | **15** | **21** | **18** | **12** | **21** | **160** |
| 4 | **14** | **20** | **20** | **13** | **23** | **12** | **169** |
| 5 | **13** | **21** | **19** | **10|110** | **20** | **14** | **0** |
| 6 | **0** | **0** | **0** | **0** | **0** | **0** | **152** |
| ПОТРЕБНОСТИ | **153** | **0** | **141** | **93** | **105** | **173** | **892** |

1. x16 = min (178, 173) = 173

Выбор ячейки с наименьшим значением c31=11

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| ПОТРЕБИТЕЛИ  ПОСТАВЩИКИ | 1 | 2 | 3 | 4 | 5 | 6 | ЗАПАСЫ |
| 1 | **22** | **12** | **16** | **13** | **21** | **11|173** | **5** |
| 2 | **20** | **10|118** | **18** | **15** | **17** | **23** | **6** |
| 3 | **11** | **15** | **21** | **18** | **12** | **21** | **160** |
| 4 | **14** | **20** | **20** | **13** | **23** | **12** | **169** |
| 5 | **13** | **21** | **19** | **10|110** | **20** | **14** | **0** |
| 6 | **0** | **0** | **0** | **0** | **0** | **0** | **152** |
| ПОТРЕБНОСТИ | **153** | **0** | **141** | **93** | **105** | **0** | **892** |

5) x31 = min (160, 153) = 153

Выбор ячейки с наименьшим значением c35=12

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| ПОТРЕБИТЕЛИ  ПОСТАВЩИКИ | 1 | 2 | 3 | 4 | 5 | 6 | ЗАПАСЫ |
| 1 | **22** | **12** | **16** | **13** | **21** | **11|173** | **5** |
| 2 | **20** | **10|118** | **18** | **15** | **17** | **23** | **6** |
| 3 | **11|153** | **15** | **21** | **18** | **12** | **21** | **7** |
| 4 | **14** | **20** | **20** | **13** | **23** | **12** | **169** |
| 5 | **13** | **21** | **19** | **10|110** | **20** | **14** | **0** |
| 6 | **0** | **0** | **0** | **0** | **0** | **0** | **152** |
| ПОТРЕБНОСТИ | **0** | **0** | **141** | **93** | **105** | **0** | **892** |

6) x35 = min (7, 105) = 7

Выбор ячейки с наименьшим значением c14=13

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| ПОТРЕБИТЕЛИ  ПОСТАВЩИКИ | 1 | 2 | 3 | 4 | 5 | 6 | ЗАПАСЫ |
| 1 | **22** | **12** | **16** | **13** | **21** | **11|173** | **5** |
| 2 | **20** | **10|118** | **18** | **15** | **17** | **23** | **6** |
| 3 | **11|153** | **15** | **21** | **18** | **12|7** | **21** | **0** |
| 4 | **14** | **20** | **20** | **13** | **23** | **12** | **169** |
| 5 | **13** | **21** | **19** | **10|110** | **20** | **14** | **0** |
| 6 | **0** | **0** | **0** | **0** | **0** | **0** | **152** |
| ПОТРЕБНОСТИ | **0** | **0** | **141** | **93** | **98** | **0** | **892** |

7) x14 = min (5, 93) = 5

Выбор ячейки с наименьшим значением c44=13

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| ПОТРЕБИТЕЛИ  ПОСТАВЩИКИ | 1 | 2 | 3 | 4 | 5 | 6 | ЗАПАСЫ |
| 1 | **22** | **12** | **16** | **13|5** | **21** | **11|173** | **0** |
| 2 | **20** | **10|118** | **18** | **15** | **17** | **23** | **6** |
| 3 | **11|153** | **15** | **21** | **18** | **12|7** | **21** | **0** |
| 4 | **14** | **20** | **20** | **13** | **23** | **12** | **169** |
| 5 | **13** | **21** | **19** | **10|110** | **20** | **14** | **0** |
| 6 | **0** | **0** | **0** | **0** | **0** | **0** | **152** |
| ПОТРЕБНОСТИ | **0** | **0** | **141** | **88** | **98** | **0** | **892** |

8) x44 = min (169, 88) = 88

Выбор ячейки с наименьшим значением c25=17

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| ПОТРЕБИТЕЛИ  ПОСТАВЩИКИ | 1 | 2 | 3 | 4 | 5 | 6 | ЗАПАСЫ |
| 1 | **22** | **12** | **16** | **13|5** | **21** | **11|173** | **0** |
| 2 | **20** | **10|118** | **18** | **15** | **17** | **23** | **6** |
| 3 | **11|153** | **15** | **21** | **18** | **12|7** | **21** | **0** |
| 4 | **14** | **20** | **20** | **13|88** | **23** | **12** | **81** |
| 5 | **13** | **21** | **19** | **10|110** | **20** | **14** | **0** |
| 6 | **0** | **0** | **0** | **0** | **0** | **0** | **152** |
| ПОТРЕБНОСТИ | **0** | **0** | **141** | **0** | **98** | **0** | **892** |

9) x25 = min (6, 98) = 6

Выбор ячейки с наименьшим значением c43=20

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| ПОТРЕБИТЕЛИ  ПОСТАВЩИКИ | 1 | 2 | 3 | 4 | 5 | 6 | ЗАПАСЫ |
| 1 | **22** | **12** | **16** | **13|5** | **21** | **11|173** | **0** |
| 2 | **20** | **10|118** | **18** | **15** | **17|6** | **23** | **0** |
| 3 | **11|153** | **15** | **21** | **18** | **12|7** | **21** | **0** |
| 4 | **14** | **20** | **20** | **13|88** | **23** | **12** | **81** |
| 5 | **13** | **21** | **19** | **10|110** | **20** | **14** | **0** |
| 6 | **0** | **0** | **0** | **0** | **0** | **0** | **152** |
| ПОТРЕБНОСТИ | **0** | **0** | **141** | **0** | **92** | **0** | **892** |

10) x43 = min (81, 141) = 81

Выбор ячейки с наименьшим значением c63=0

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| ПОТРЕБИТЕЛИ  ПОСТАВЩИКИ | 1 | 2 | 3 | 4 | 5 | 6 | ЗАПАСЫ |
| 1 | **22** | **12** | **16** | **13|5** | **21** | **11|173** | **0** |
| 2 | **20** | **10|118** | **18** | **15** | **17|6** | **23** | **0** |
| 3 | **11|153** | **15** | **21** | **18** | **12|7** | **21** | **0** |
| 4 | **14** | **20** | **20|81** | **13|88** | **23** | **12** | **0** |
| 5 | **13** | **21** | **19** | **10|110** | **20** | **14** | **0** |
| 6 | **0** | **0** | **0** | **0** | **0** | **0** | **152** |
| ПОТРЕБНОСТИ | **0** | **0** | **60** | **0** | **92** | **0** | **892** |

11) x63 = min (152, 60) = 60

Выбор ячейки с наименьшим значением c65=0

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| ПОТРЕБИТЕЛИ  ПОСТАВЩИКИ | 1 | 2 | 3 | 4 | 5 | 6 | ЗАПАСЫ |
| 1 | **22** | **12** | **16** | **13|5** | **21** | **11|173** | **0** |
| 2 | **20** | **10|118** | **18** | **15** | **17|6** | **23** | **0** |
| 3 | **11|153** | **15** | **21** | **18** | **12|7** | **21** | **0** |
| 4 | **14** | **20** | **20|81** | **13|88** | **23** | **12** | **0** |
| 5 | **13** | **21** | **19** | **10|110** | **20** | **14** | **0** |
| 6 | **0** | **0** | **0|60** | **0** | **0** | **0** | **92** |
| ПОТРЕБНОСТИ | **0** | **0** | **0** | **0** | **92** | **0** | **892** |

12) x65 = min (92, 92) = 92

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| ПОТРЕБИТЕЛИ  ПОСТАВЩИКИ | 1 | 2 | 3 | 4 | 5 | 6 | ЗАПАСЫ |
| 1 | **22** | **12** | **16** | **13|5** | **21** | **11|173** | **0** |
| 2 | **20** | **10|118** | **18** | **15** | **17|6** | **23** | **0** |
| 3 | **11|153** | **15** | **21** | **18** | **12|7** | **21** | **0** |
| 4 | **14** | **20** | **20|81** | **13|88** | **23** | **12** | **0** |
| 5 | **13** | **21** | **19** | **10|110** | **20** | **14** | **0** |
| 6 | **0** | **0** | **0|60** | **0** | **0|92** | **0** | **0** |
| ПОТРЕБНОСТИ | **0** | **0** | **0** | **0** | **0** | **0** | **892** |

Число базисных переменных должно быть m + n – 1, т.е. 11.

Первое допустимое решение:

x14 = 5, x16 = 173, x22 = 118, x25 = 6, x31 = 153, x35 = 7, x43 = 81, x44 = 88, x54 = 110, x63 = 60, x65 = 92.

Значение функции цели:

Z = 5×13 + 173×11 + 118×10 + 6×17 + 153×11 + 7×12 + 81×20 + 88×13 + 110×10 + 60×0 + 92×0 = 65 + 1903 + 1180 + 102 + 1683 + 84 + 1620 + 1144 + 1100 = 8 881

* **Метод потенциалов.**

В методе потенциалов каждой строке i и каждому столбцу j транспортной таблицы ставятся в соответствие числа (потенциалы) *ui* (поставщики)и *vj* (потребители). Для каждой базисной переменной xij потенциалы *ui* и *vj*удовлетворяют уравнению

*ui* + *vj* = *сij*

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| ПОТРЕБИТЕЛИ  ПОСТАВЩИКИ | 1 | 2 | 3 | 4 | 5 | 6 | ЗАПАСЫ |
| 1 | **22** | **12** | **16** | **13|5** | **21** | **11|173** | **178** |
| 2 | **20** | **10|118** | **18** | **15** | **17|6** | **23** | **123** |
| 3 | **11|153** | **15** | **21** | **18** | **12|7** | **21** | **160** |
| 4 | **14** | **20** | **20|81** | **13|88** | **23** | **12** | **169** |
| 5 | **13** | **21** | **19** | **10|110** | **20** | **14** | **110** |
| 6 | **0** | **0** | **0|60** | **0** | **0|92** | **0|92** | **152** |
| ПОТРЕБНОСТИ | **153** | **117** | **141** | **203** | **105** | **173** | **892** |

Потенциалы: *ui ,i =* 1,6, *vj ,j* =1,6

Определяем потенциалы для всех базисных переменных:

*u*1+ *v*4= 13

*u*1+ *v*6= 11

*u*2+ *v*2= 10

*u*2+ *v*5= 17

*u*3+ *v*1= 11

*u*3+ *v*5= 12

*u*4+ *v*3= 20

*u*4+ *v*4= 13

*u*5+ *v*4= 10

*u*6+ *v*3= 0

*u*6+ *v*5= 0

Уравнений 11, неизвестных 12.

Присваиваем *u*1 = 0. Значит,

*v*4 = 13, *u*4 = 0, *u*5 = -3,

*v*3 = 20, *u*6 = -20, *v*5 = 20,

*u*3 = -8, *v*1 = 19,

*u*2 = -3, *v*2 = 13, *v*6 = 11

|  |  |
| --- | --- |
| Небазисная переменная |  |
| x11 | *u*1+ *v*1– *с*11= 0 + 19 – 22 = -3 |
| x12 | *u*1+ *v*2– *с*12 = 0 + 13 – 12 = 1 |
| x13 | *u*1+ *v*3– *с*13 = 0 + 20 – 16 = 4 |
| x15 | *u*1+ *v*5– *с*15 = 0 + 20 – 21= -1 |
| x21 | *u*2+ *v*1– *с*21 = -3 + 19 – 20 = -4 |
| x23 | *u*2+ *v*3– *с*23 = -3 + 20 – 18= -1 |
| x24 | *u*2+ *v*4– *с*24 = -3 + 13 – 15 = -5 |
| x26 | *u*2+ *v*6– *с*26 = -3 + 11 – 23 = -15 |
| x32 | *u*3+ *v*2– *с*32 = -8 + 13 – 15 = -10 |
| x33 | *u*3+ *v*3– *с*33 = -8 + 20 – 21 = -9 |
| x34 | *u*3+ *v*4– *с*34 = -8 + 13 – 18 = -13 |
| x36 | *u*3+ *v*6– *с*36 = -8 + 11 – 21= -18 |
| x41 | *u*4+ *v*1– *с*41 = 0 + 19 – 14 = 5 |
| x42 | *u*4+ *v*2– *с*42 = 0 + 13 – 20 = -7 |
| x45 | *u*4+ *v*5– *с*45 = 0 + 20 – 23 = -3 |
| x46 | *u*4+ *v*6– *с*46 = 0 + 11 – 12= -1 |
| x51 | *u*5+ *v*1– *с*51 = -3 + 19 – 13 = 3 |
| x52 | *u*5+ *v*2– *с*52 = -3 + 13 – 21 = -11 |
| x53 | *u*5+ *v*3– *с*53 = -3 + 20 – 19 = -2 |
| x55 | *u*5+ *v*5– *с*55 = -3 + 20 – 20 = -3 |
| x56 | *u*5+ *v*6– *с*56 = -3 + 11 – 14= -6 |
| x61 | *u*6+ *v*1– *с*61 = -20 + 19 – 0 = -1 |
| x62 | *u*6+ *v*2– *с*62 = -20 + 13 – 0 = -7 |
| x64 | *u*6+ *v*4– *с*64 = -20 + 13 – 0 = -7 |
| x66 | *u*6+ *v*6– *с*66 = -20 + 11 – 0 = -9 |

Вводимой в базис будет переменная, имеющая наибольшее положительное значение – х41.

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| ПОТРЕБИТЕЛИ  ПОСТАВЩИКИ | 1 | 2 | 3 | 4 | 5 | 6 | ЗАПАСЫ |
| 1 | **22** | **12** | **16** | **13|5** | **21** | **11|173** | **178** |
| 2 | **20** | **10|118** | **18** | **15** | **17|6** | **23** | **123** |
| 3 | **11|153** | **15** | **21** | **18** | **12|7** | **21** | **160** |
| 4 | **14** | **20** | **20|81** | **13|88** | **23** | **12** | **169** |
| 5 | **13** | **21** | **19** | **10|110** | **20** | **14** | **110** |
| 6 | **0** | **0** | **0|60** | **0** | **0|92** | **0|92** | **152** |
| ПОТРЕБНОСТИ | **153** | **117** | **141** | **203** | **105** | **173** | **892** |

81-наименьшее число в -

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| ПОТРЕБИТЕЛИ  ПОСТАВЩИКИ | 1 | 2 | 3 | 4 | 5 | 6 | ЗАПАСЫ |
| 1 | **22** | **12** | **16** | **13|5** | **21** | **11|173** | **178** |
| 2 | **20** | **10|118** | **18** | **15** | **17|6** | **23** | **123** |
| 3 | **11|72** | **15** | **21** | **18** | **12|88** | **21** | **160** |
| 4 | **14|81** | **20** | **20|0** | **13|88** | **23** | **12** | **169** |
| 5 | **13** | **21** | **19** | **10|110** | **20** | **14** | **110** |
| 6 | **0** | **0** | **0|141** | **0** | **0|11** | **0|92** | **152** |
| ПОТРЕБНОСТИ | **153** | **117** | **141** | **203** | **105** | **173** | **892** |

Допустимое решение:

x14 = 5, x16 = 173, x22 = 118, x25 = 6, x31 = 72, x35 = 88, x41 = 81, x44 = 88, x54 = 110, x63 = 141, x65 = 11.

Значение функции цели:

Z = 5×13 + 173×11 + 118×10 + 6×17 + 72×11 + 88×12 + 81×14 + 88×13 + 110×10 = 65 + 1903 + 1180 + 102 + 792 + 1056 + 1134 + 1144 + 1100 = 8476

8 881-8476=405

Определяем потенциалы для всех базисных переменных:

*u*1+ *v*4= 13

*u*1+ *v*6= 11

*u*2+ *v*2= 10

*u*2+ *v*5= 17

*u*3+ *v*1= 11

*u*3+ *v*5= 12

*u*4+ *v*1= 14

*u*4+ *v*4= 13

*u*5+ *v*4= 10

*u*6+ *v*3= 0

*u*6+ *v*5= 0

Уравнений 11, неизвестных 12.

Присваиваем *u*1 = 0.

Значит, *v*4 = 13, *v*6 = 11, *u*4 = 0, *u*5 = -3, *v*1 = 14, *u*3 = -3, *v*5 = 15, *u*2 = 2, *v*2 = 8,

*u*6 = -15, *v*3 = 15

|  |  |
| --- | --- |
| Небазисная переменная |  |
| x11 | *u*1+ *v*1– *с*11= 0 + 14 – 22 = -8 |
| x12 | *u*1+ *v*2– *с*12 = 0 + 8 – 12 = -4 |
| x13 | *u*1+ *v*3– *с*13 = 0 + 15 – 16 = -1 |
| x15 | *u*1+ *v*5– *с*15 = 0 + 15 – 21 = -6 |
| x21 | *u*2+ *v*1– *с*21 = 2 + 14 – 20 = -4 |
| x23 | *u*2+ *v*3– *с*23 = 2 + 15 – 18 = -1 |
| x24 | *u*2+ *v*4– *с*24 = 2 + 13 – 15 = 0 |
| x26 | *u*2+ *v*6– *с*26 = 2 + 11 – 23 = -10 |
| x32 | *u*3+ *v*2– *с*32 = -3 + 8 – 15 = -10 |
| x33 | *u*3+ *v*3– *с*33 = -3 + 15 – 21 = -9 |
| x34 | *u*3+ *v*4– *с*34 = -3 + 13 – 18 = -8 |
| x36 | *u*3+ *v*6– *с*36 = -3 + 11 – 21 = -13 |
| x42 | *u*4+ *v*2– *с*42 = 0 + 8 – 20 = -12 |
| x45 | *u*4+ *v*5– *с*45 = 0 + 15 – 23 = -8 |
| x46 | *u*4+ *v*6– *с*46 = 0 + 11 – 12 = -1 |
| x51 | *u*5+ *v*1– *с*51 = -3 + 14 – 13 = -2 |
| x52 | *u*5+ *v*2– *с*52 = -3 + 8 – 21 = -16 |
| x53 | *u*5+ *v*3– *с*53 = -3 + 15 – 19 = -7 |
| x55 | *u*5+ *v*5– *с*55 = -3 + 15 – 20 = -8 |
| x56 | *u*5+ *v*6– *с*56 = -3 + 11 – 14 = -6 |
| x61 | *u*6+ *v*1– *с*61 = -15 + 14 – 0 = -1 |
| x62 | *u*6+ *v*2– *с*62 = -15 + 8 – 0 = -7 |
| x64 | *u*6+ *v*4– *с*64 = -15 + 13 – 0 = -2 |
| x66 | *u*6+ *v*6– *с*66 = -15 + 11 – 0 = -4 |

Остановка, т.к. нет положительных.