Учреждение образования

«Белорусский государственный технологический университет»

Факультет информационных технологий

Специальность программная инженерия

Дисциплина: Математическое программирование

**Отчет по лабораторным работам**

**По дисциплине «Математическое программирование»**

Исполнитель:

Студент 2 курса 10 группы

Ярохович С.А.

Руководитель:

Ромыш А.С.

Минск 2025

Оглавление

[Лабораторная работа №1 3](#_Toc196230020)

[Лабораторная работа №2 10](#_Toc196230021)

[Лабораторная работа №3 30](#_Toc196230022)

[Лабораторная работа №4 51](#_Toc196230023)

[Лабораторная работа №5 55](#_Toc196230024)

[Лабораторная работа №6 65](#_Toc196230025)

[Лабораторная работа №7 80](#_Toc196230026)

[Лабораторная работа №8 83](#_Toc196230027)

# Лабораторная работа №1

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

***Задание 1.*** Разработать три функции (start, dget и iget), используя следующие спецификации:

start - функция установки начального числа как текущего значения для генератора.

dget - функция возвращает действительное число.

iget - функция возвращает целое число.

Для начала работы нам нужно подключить предварительно скомпилированные заголовки.

Для использования предварительно скомпилированных заголовков их необходимо включить параметром /Y или через страницу свойств проекта **Configuration Properties > C/C++ > Precompiled Headers**. После этого в каждом cpp-файле в самом начале (до любых директив препроцессора или строк кода) нужно добавить #include "pch.h".

В примере 1 у нас были даны прототипы функций и сама их реализация, в листинге 1.1 приведен код.

#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

};

Листинг 1.1 – Файл Auxil.h

#include "pch.h"

#include "Auxil.h"

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);

};

}

Листинг 1.2 – Файл Auxil.cpp

***Задание 2***

1. Реализовать пример 2.
2. Для проверки работоспособности разработанных функций и приобретения навыков замера продолжительности процесса вычисления реализуйте программу, приведенную в примере 2.

Код реализованного задания представлен в листинге 2.

#include "pch.h"

#include "Auxil.h" // вспомогательные функции

#define CYCLE 1000000 // количество циклов

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

{

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, 200); // сумма случайных чисел

av2 += auxil::dget(100, 200); // 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;

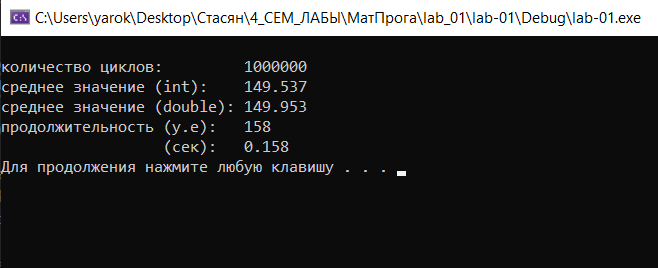
system("pause");

return 0;

}

Листинг 2 – Проверка функций

После многочисленных настроек, проект был успешно запущен. Результат работы программы представлен на рисунке 1.

Рисунок 1 – Результат работы программы

***Задание 3***

Проведите необходимые эксперименты и постройте график зависимости (Excel) продолжительности процесса вычисления от количества циклов в примере 2. Проанализируйте характер зависимости. Проведите исследование любого другого рекурсивного алгоритма, например, вычисления факториала или генератора чисел Фибоначчи (прим. – например вычислите каким будет 100-е, 200-е, 300-е и т.д число), и включите в отчет график.

**Примечание:** продолжительность вычисления измерять в условных единицах процессорного времени (функция clock).

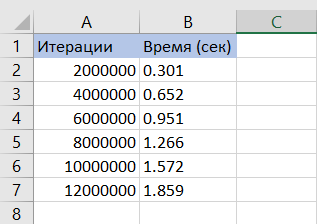
Для создания графика зависимости количества итераций от времени (сек.) нужны экспериментальные данные. В ходе проверок были установлены значения, которые отображены на рисунке 2.

Рисунок 2 – Экспериментальные данные

 На основе полученных данных теперь можно составить график зависимости количества итераций от времени (сек.). Для построения графика будет использоваться программа Exсel. Полученный график представлен на рисунке 3.

Рисунок 3 – Зависимость итераций от времени

Как видно из графика, зависимость является линейной, так как время выполнения программы зависит от количества итераций.

Те же действия можно проделать при проверке алгоритма чисел Фибоначчи. Код программы, реализующей алгоритм вычисления чисел Фибоначчи представлен в листинге 3.

#include "pch.h"

using namespace std;

static long long fibonacci(int n)

{

if (n <= 1) return n;

return fibonacci(n - 1) + fibonacci(n - 2);

}

int main()

{

SetConsoleCP(1251);

SetConsoleOutputCP(1251);

for (int n = 5; n <= 45; n += 5)

{

clock\_t t1 = clock();

long long result = fibonacci(n);

clock\_t t2 = clock();

cout << "Фибоначии(" << n << ") = " << result << ", Время: " << (t2 - t1) << "\n";

}

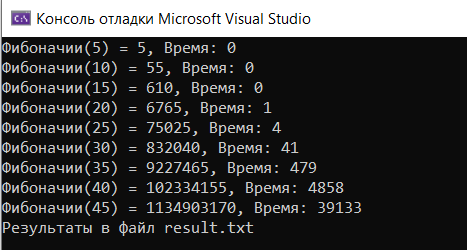
cout << "Результаты в файл result.txt\n";

return 0;

}

Листинг 3 – Программа Фибоначчи

Результат выполнения программы представлен на рисунке 4.

Рисунок 4 – Результат программы

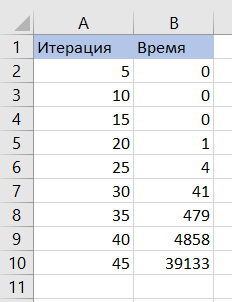
Данные были перенесены в Excel в виде таблицы и графика. Таблица данных представлена на рисунке 5. График представлен на рисунке 6.

Рисунок 5 – Экспериментальные данные

Рисунок 6 – График зависимости времени от номера числа Фибоначчи

Как видно из графика, зависимость является экспоненциальной. Время выполнения программы резко возрастает при большом номере числа Фибоначчи.

**Выводы:**

**Исходя из вышеизложенного, можно сделать вывод о том, что в зависимости от реализованного алгоритма, скорость выполнения программы может сильно отличаться. Это хорошо наблюдается при анализе графиков, полученных в результате проведения лабораторной работы. В первой программе мы получили линейную зависимость (О-большое равно n). Во второй программе была получена экспоненциальная зависимость (О-большое равно 2^n).**

# Лабораторная работа №2

**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.

***Вариант 14***

***Задание 1.*** Разобрать и разработать генератор подмножеств заданного множества.

// Combi.h

#pragma once

namespace combi

{

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(количество элементов исходного множества)

void reset();

short getfirst(); // сформормировать массив индексов по битовой маске

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

short ntx(short i);

unsigned \_\_int64 count();

};

}

Листинг 1.1 – файл Combi.h

// Combi.cpp

#include "stdafx.h"

#include "Combi.h"

#include <algorithm>

namespace combi

{

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);

};

};

Листинг 1.2 – файл Combi.cpp

// Main.cpp

#include "stdafx.h"

#include <iostream>

#include "Combi.h"

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

{

setlocale(LC\_ALL, "rus");

char AA[][2] = { "A", "B", "C", "D"};

std::cout << std::endl << " - Генератор множества всех подмножеств -";

std::cout << std::endl << "Исходное множество: ";

std::cout << "{ ";

for (int i = 0; i < sizeof(AA) / 2; i++)

std::cout << AA[i] << ((i < sizeof(AA) / 2 - 1) ? ", " : " ");

std::cout << "}";

std::cout << std::endl << "Генерация всех подмножеств ";

combi::subset s1(sizeof(AA) / 2); // создание генератора

int n = s1.getfirst(); // первое (пустое) подмножество

while (n >= 0) // пока есть подмножества

{

std::cout << std::endl << "{ ";

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

std::cout << AA[s1.ntx(i)] << ((i < n - 1) ? ", " : " ");

std::cout << "}";

n = s1.getnext(); // cледующее подмножество

};

std::cout << std::endl << "всего: " << s1.count() << std::endl;

system("pause");

return 0;

}

Листинг 1.3 – файл Main.cpp

***Задание 2.*** Разобрать и разработать генератор сочетаний.

// Combi.h

#pragma once

namespace combi

{

struct xcombination // генератор сочетаний (эвристика)

{

short n, // количество элементов исходного множества

nc,

m, // количество элементов в сочетаниях

\* sset; // массив индексов текущего сочетания

xcombination(

short n = 1, //количество элементов исходного множества

short m = 1 // количество элементов в сочетаниях

);

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

short getfirst(); // сформировать первый массив индексов

short getnext(); // сформировать следующий массив индексов

short ntx(short i); //

unsigned \_\_int64 count() const;

};

}

Листинг 2.1 – файл Combi.h

// Combi.cpp

#include "stdafx.h"

#include "Combi.h"

#include <algorithm>

namespace combi

{

xcombination::xcombination(short n, short m)

{

this->n = n;

this->m = m;

this->sset = new short[m + 2];

this->reset();

}

void xcombination::reset() // сбросить генератор, начать сначала

{

this->nc = 0;

for (int i = 0; i < this->m; i++) this->sset[i] = i;

this->sset[m] = this->n;

this->sset[m + 1] = 0;

};

short xcombination::getfirst()

{

return (this->n >= this->m) ? this->m : -1;

};

short xcombination::getnext() // сформировать следующий массив индексов

{

short rc = getfirst();

if (rc > 0)

{

short j;

for (j = 0; this->sset[j] + 1 == this->sset[j + 1]; ++j)

this->sset[j] = j;

if (j >= this->m) rc = -1;

else {

this->sset[j]++;

this->nc++;

};

}

return rc;

};

short xcombination::ntx(short i)

{

return this->sset[i];

};

unsigned \_\_int64 fact(unsigned \_\_int64 x) { return(x == 0) ? 1 : (x \* fact(x - 1)); };

unsigned \_\_int64 xcombination::count() const

{

return (this->n >= this->m) ?

fact(this->n) / (fact(this->n - this->m) \* fact(this->m)) : 0;

};

}

Листинг 2.2 – файл Combi.cpp

// Main

#include "stdafx.h"

#include <iostream>

#include "Combi.h"

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

{

setlocale(LC\_ALL, "rus");

char AA[][2] = { "A", "B", "C", "D", "E" };

std::cout << std::endl << " --- Генератор сочетаний ---";

std::cout << std::endl << "Исходное множество: ";

std::cout << "{ ";

for (int i = 0; i < sizeof(AA) / 2; i++)

std::cout << AA[i] << ((i < sizeof(AA) / 2 - 1) ? ", " : " ");

std::cout << "}";

std::cout << std::endl << "Генерация сочетаний ";

combi::xcombination xc(sizeof(AA) / 2, 3);

std::cout << "из " << xc.n << " по " << xc.m;

int n = xc.getfirst();

while (n >= 0)

{

std::cout << std::endl << xc.nc << ": { ";

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

std::cout << AA[xc.ntx(i)] << ((i < n - 1) ? ", " : " ");

std::cout << "}";

n = xc.getnext();

}

std::cout << std::endl << "всего: " << xc.count() << std::endl;

system("pause");

return 0;

}

Листинг 2.3 – файл Main.cpp

***Задание 3.*** Разобрать и разработать генератор перестановок.

// Combi.h

#pragma once

namespace combi

{

struct permutation // генератор перестановок

{

const static bool L = true; // левая стрелка

const static bool R = false; // правая стрелка

short n, // количество элементов исходного множества

\* sset; // массив индексов текущей перестановки

bool\* dart; // массив стрелок (левых-L и правых-R)

permutation(short n = 1); // конструктор (количество элементов исходного множества)

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

\_\_int64 getfirst(); // сформировать первый массив индексов

\_\_int64 getnext(); // сформировать случайный массив индексов

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

unsigned \_\_int64 np; // номер перествновки 0,... count()-1

unsigned \_\_int64 count() const; // вычислить общее кол. перестановок

};

}

Листинг 3.1 – файл Combi.h

// Combi.cpp

#include "stdafx.h"

#include "Combi.h"

#include <algorithm>

#define NINF ((short)0x8000)

namespace combi

{

permutation::permutation(short n)

{

this->n = n;

this->sset = new short[n];

this->dart = new bool[n];

this->reset();

};

void permutation::reset()

{

this->getfirst();

};

\_\_int64 permutation::getfirst()

{

this->np = 0;

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

{

this->sset[i] = i; this->dart[i] = L;

};

return (this->n > 0) ? this->np : -1;

};

\_\_int64 permutation::getnext() //

{

\_\_int64 rc = -1;

short maxm = NINF, idx = -1;

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

{

if (i > 0 &&

this->dart[i] == L &&

this->sset[i] > this->sset[i - 1] &&

maxm < this->sset[i]) maxm = this->sset[idx = i];

if (i < (this->n - 1) &&

this->dart[i] == R &&

this->sset[i] > this->sset[i + 1] &&

maxm < this->sset[i]) maxm = this->sset[idx = i];

};

if (idx >= 0)

{

std::swap(this->sset[idx],

this->sset[idx + (this->dart[idx] == L ? -1 : 1)]);

std::swap(this->dart[idx],

this->dart[idx + (this->dart[idx] == L ? -1 : 1)]);

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

if (this->sset[i] > maxm) this->dart[i] = !this->dart[i];

rc = ++this->np;

}

return rc;

};

short permutation::ntx(short i) { return this->sset[i]; };

unsigned \_\_int64 fact(unsigned \_\_int64 x) { return (x == 0) ? 1 : (x \* fact(x - 1)); };

unsigned \_\_int64 permutation::count() const { return fact(this->n); };

}

Листинг 3.2 – файл Combi.cpp

// --- Main

#include "stdafx.h"

#include <iostream>

#include "Combi.h"

#include <iomanip>

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

{

setlocale(LC\_ALL, "rus");

char AA[][2] = { "A", "B", "C", "D" };

std::cout << std::endl << " --- Генератор перестановок ---";

std::cout << std::endl << "Исходное множество: ";

std::cout << "{ ";

for (int i = 0; i < sizeof(AA) / 2; i++)

std::cout << AA[i] << ((i < sizeof(AA) / 2 - 1) ? ", " : " ");

std::cout << "}";

std::cout << std::endl << "Генерация перестановок ";

combi::permutation p(sizeof(AA) / 2);

\_\_int64 n = p.getfirst();

while (n >= 0)

{

std::cout << std::endl << std::setw(4) << p.np << ": { ";

for (int i = 0; i < p.n; i++)

std::cout << AA[p.ntx(i)] << ((i < p.n - 1) ? ", " : " ");

std::cout << "}";

n = p.getnext();

};

std::cout << std::endl << "всего: " << p.count() << std::endl;

system("pause");

return 0;

}

Листинг 3.3 – файл Main.cpp

***Задание 4.*** Разобрать и разработать генератор размещений.

// Combi.h

#pragma once

namespace combi

{

struct xcombination // генератор сочетаний (эвристика)

{

short n, // количество элементов исходного множества

m, // количество элементов в сочетаниях

\* sset; // массив индексов текущего сочетания

xcombination(

short n = 1, // количество элементов исходного множества

short m = 1 // количество элементов в сочетаниях

);

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

short getfirst(); // сформировать первый массив индексов

short getnext(); // сформировать следующий массив индексов

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

unsigned \_\_int64 nc; // номер сочетания 0,..., count()-1

unsigned \_\_int64 count() const; // вычислить количество сочетаний

};

struct permutation // генератор перестановок

{

const static bool L = true; // левая стрелка

const static bool R = false; // правая стрелка

short n, // количество элементов исходного множества

\* sset; // массив индексов текущей перестановки

bool\* dart; // массив стрелок (левых-L и правых-R)

permutation(short n = 1); // конструктор (количество элементов исходного множества)

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

\_\_int64 getfirst(); // сформировать первый массив индексов

\_\_int64 getnext(); // сформировать случайный массив индексов

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

unsigned \_\_int64 np; // номер перествновки 0,... count()-1

unsigned \_\_int64 count() const; // вычислить общее кол. перестановок

};

struct accomodation // генератор размещений

{

short n, // количество элементов исходного множества

m, // количество элементов в размещении

\* sset; // массив индесов текущего размещения

xcombination\* cgen; // указатель на генератор сочетаний

permutation\* pgen; // указатель на генератор перестановок

accomodation(short n = 1, short m = 1); // конструктор

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

short getfirst(); // сформировать первый массив индексов

short getnext(); // сформировать следующий массив индексов

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

unsigned \_\_int64 na; // номер размещения 0, ..., count()-1

unsigned \_\_int64 count() const; // общее количество размещений

};

}

Листинг 4.1 – Файл Combi.h

// Combi.cpp

#include "Combi.h"

#include <algorithm>

#define NINF ((short)0x8000)

namespace combi

{

// ================================== ACCOMODATION ==================================

accomodation::accomodation(short n, short m)

{

this->n = n;

this->m = m;

this->cgen = new xcombination(n, m);

this->pgen = new permutation(m);

this->sset = new short[m];

this->reset();

}

void accomodation::reset()

{

this->na = 0;

this->cgen->reset();

this->pgen->reset();

this->cgen->getfirst();

};

short accomodation::getfirst()

{

short rc = (this->n >= this->m) ? this->m : -1;

if (rc > 0)

{

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

this->sset[i] = this->cgen->sset[this->pgen->ntx(i)];

};

return rc;

};

short accomodation::getnext()

{

short rc;

this->na++;

if ((this->pgen->getnext()) > 0) rc = this->getfirst();

else if ((rc = this->cgen->getnext()) > 0)

{

this->pgen->reset(); rc = this->getfirst();

};

return rc;

};

short accomodation::ntx(short i)

{

return this->sset[i];

};

// факториал из структуры accomodation

unsigned \_\_int64 fact(unsigned \_\_int64 x) { return (x == 0) ? 1 : (x \* fact(x - 1)); };

unsigned \_\_int64 accomodation::count() const

{

return (this->n >= this->m) ?

fact(this->n) / fact(this->n - this->m) : 0;

};

// ================================== PERMUTATION ==================================

permutation::permutation(short n)

{

this->n = n;

this->sset = new short[n];

this->dart = new bool[n];

this->reset();

};

void permutation::reset()

{

this->getfirst();

};

\_\_int64 permutation::getfirst()

{

this->np = 0;

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

{

this->sset[i] = i; this->dart[i] = L;

};

return (this->n > 0) ? this->np : -1;

};

\_\_int64 permutation::getnext()

{

\_\_int64 rc = -1;

short maxm = NINF, idx = -1;

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

{

if (i > 0 &&

this->dart[i] == L &&

this->sset[i] > this->sset[i - 1] &&

maxm < this->sset[i]) maxm = this->sset[idx = i];

if (i < (this->n - 1) &&

this->dart[i] == R &&

this->sset[i] > this->sset[i + 1] &&

maxm < this->sset[i]) maxm = this->sset[idx = i];

};

if (idx >= 0)

{

std::swap(this->sset[idx],

this->sset[idx + (this->dart[idx] == L ? -1 : 1)]);

std::swap(this->dart[idx],

this->dart[idx + (this->dart[idx] == L ? -1 : 1)]);

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

if (this->sset[i] > maxm) this->dart[i] = !this->dart[i];

rc = ++this->np;

}

return rc;

};

short permutation::ntx(short i) { return this->sset[i]; };

unsigned \_\_int64 permutation::count() const { return fact(this->n); };

// ================================== XCOMBINATION ==================================

xcombination::xcombination(short n, short m)

{

this->n = n;

this->m = m;

this->sset = new short[m + 2];

this->reset();

}

void xcombination::reset() // ñáðîñèòü ãåíåðàòîð, íà÷àòü ñíà÷àëà

{

this->nc = 0;

for (int i = 0; i < this->m; i++) this->sset[i] = i;

this->sset[m] = this->n;

this->sset[m + 1] = 0;

};

short xcombination::getfirst()

{

return (this->n >= this->m) ? this->m : -1;

};

short xcombination::getnext() // ñôîðìèðîâàòü ñëåäóþùèé ìàññèâ èíäåêñîâ

{

short rc = getfirst();

if (rc > 0)

{

short j;

for (j = 0; this->sset[j] + 1 == this->sset[j + 1]; ++j)

this->sset[j] = j;

if (j >= this->m) rc = -1;

else

{

this->sset[j]++;

this->nc++;

}

}

return rc;

};

short xcombination::ntx(short i)

{

return this->sset[i];

};

unsigned \_\_int64 xcombination::count() const

{

return (this->n >= this->m) ?

fact(this->n) / (fact(this->n - this->m) \* fact(this->m)) : 0;

};

}

Листинг 4.2 – Файл Combi.cpp

// --- main

#include "stdafx.h"

#include <iostream>

#include <iomanip>

#include "Combi.h"

#define N (sizeof(AA)/2)

#define M 3

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

{

setlocale(LC\_ALL, "rus");

char AA[][2] = { "A", "B", "C", "D" };

std::cout << std::endl << " --- Генератор размещений ---";

std::cout << std::endl << "Исходное множество: ";

std::cout << "{ ";

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

std::cout << AA[i] << ((i < N - 1) ? ", " : " ");

std::cout << "}";

std::cout << std::endl << "Генерация размещений из " << N << " по " << M;

combi::accomodation s(N, M);

int n = s.getfirst();

while (n >= 0)

{

std::cout << std::endl << std::setw(2) << s.na << ": { ";

for (int i = 0; i < 3; i++)

std::cout << AA[s.ntx(i)] << ((i < n - 1) ? ", " : " ");

std::cout << "}";

n = s.getnext();

};

std::cout << std::endl << "всего: " << s.count() << std::endl;

system("pause");

return 0;

}

Листинг 4.3 – Файл Main.cpp

***Задание 5.***  Решить в соответствии с вариантом упрощенную задачу 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.).

#pragma once

namespace combi

{

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(количество элементов исходного множества)

short getfirst(); // сформормировать 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 количество подмножеств

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

};

};

Листинг 5.1 – Файл Combi.h

#include "Combi.h"

#include <algorithm>

namespace combi

{

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);

};

};

Листинг 5.2 – Файл Combi.cpp

// backpack.h

#pragma once

#include "Combi.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

);

Листинг 5.3 – Файл backpack.h

// backpack.cpp

#include "backpack.h"

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

int calcv(combi::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(combi::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(combi::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}

)

{

combi::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;

};

Листинг 5.4 – Файл backpack.cpp

// Main

#include <iostream>

#include "Combi.h"

#include "backpack.h"

#include <ctime>

#define NN 20

int main()

{

setlocale(LC\_ALL, "rus");

int V = 300, // вместимость рюкзака

v[] = { 25, 30, 60, 20, 100, 20, 34, 53, 34, 56, 68, 46, 56, 180, 230, 240, 245, 260, 270, 280}, // размер предмета каждого типа

c[] = { 25, 10, 20, 30, 23, 43, 23, 43, 22, 42, 11, 38, 29, 25, 46, 43, 45, 50, 49, 60}; // стоимость предмета каждого типа

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

clock\_t t1 = 0, t2 = 0;

t1 = clock();

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

);

t2 = clock();

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 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;

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;

}

Листинг 5.5 – Файл Main.cpp

***Задание 6.*** Исследовать зависимость времени вычисления необходимое для решения задачи. Упрощенная задача о рюкзаке (количество предметов 12 – 20 шт.).

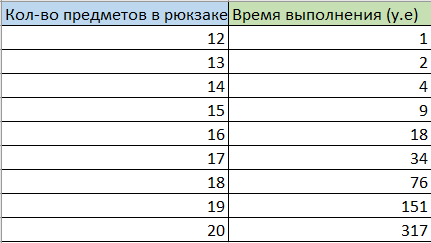
Исходя из результатов выполнения программы были получены следующие данные, которые отображены на рисунке 1.

Рисунок 1 – Данные эксперимента

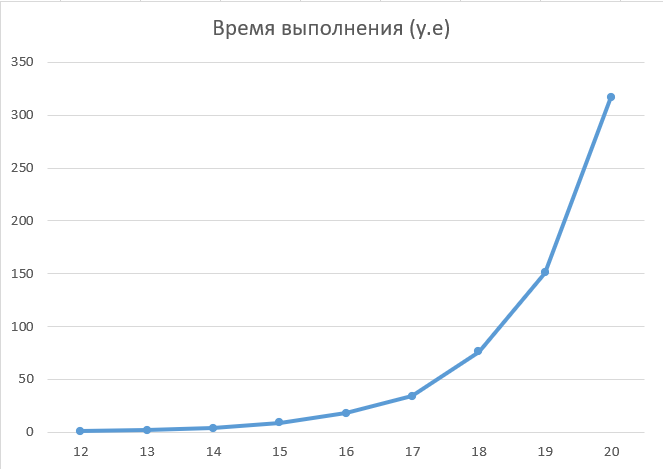
Исходя из полученных данных был построен график зависимости количества предметов в рюкзаке от времени выполнения. Данный график представлен на рисунке 2.

Рисунок 2 – График зависимости

Как видно из графика, зависимость получилась экспоненциальной.

**Выводы:**

**В результате выполнения лабораторной работы, были разобраны и разработаны различные генераторы. К таким относятся: генератор сочетаний, генератор перестановок, генератор размещений и генератор подмножеств заданного множества.**

**Также, был проведен анализ упрощенной задачи о рюкзаке. В результате анализа, был построен график зависимости времени выполнения от количества предметов в рюкзаке.**

# Лабораторная работа №3

**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.

***Вариант 14***

***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 (n = 14)* – номер варианта или номер по журналу;

Исходя из номера журнала получается следующая таблица:

Таблица 1.

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **Город** | **1** | **2** | **3** | **4** | **5** |
| **1** |  | 28 | 35 |  | 14 |
| **2** | 14 |  | 29 | 54 | 70 |
| **3** | 16 | 42 |  | 86 | 63 |
| **4** | 31 | 44 | 56 |  | 42 |
| **5** | 79 | 80 | 52 | 27 |  |

***Задание №2.*** Решить сформулированную задачу методом ветвей и границ.

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| **Город** | **1** | **2** | **3** | **4** | **5** | **di** |
| **1** | INF | 28 | 35 |  | 14 | 14 |
| **2** | 14 |  | 29 | 54 | 70 | 14 |
| **3** | 16 | 42 |  | 86 | 63 | 16 |
| **4** | 31 | 44 | 56 |  | 42 | 31 |
| **5** | 79 | 80 | 52 | 27 |  | 27 |

1. Находим минимальное значение в каждой строке и записываем его в отдельный столбец di
2. Затем производим **редукцию строк** – из каждого значения в строку вычитаем минимальный элемент соответствующей строки (Cij = Cij — di). Получаем таблицу с новыми данными.

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| **Город** | **1** | **2** | **3** | **4** | **5** | **di** |
| **1** | INF | 14 | 21 |  | 0 | 14 |
| **2** | 0 |  | 15 | 40 | 56 | 14 |
| **3** | 0 | 26 |  | 70 | 47 | 16 |
| **4** | 0 | 13 | 25 |  | 11 | 31 |
| **5** | 52 | 53 | 25 | 0 |  | 27 |

1. Далее находим минимальные значения в каждом столбце (dj). Эти минимумы (включая нули) выписываем в отдельную строку.

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **Город** | **1** | **2** | **3** | **4** | **5** |
| **1** | INF | 14 | 21 |  | 0 |
| **2** | 0 |  | 15 | 40 | 56 |
| **3** | 0 | 26 |  | 70 | 47 |
| **4** | 0 | 13 | 25 |  | 11 |
| **5** | 52 | 53 | 25 | 0 |  |
| **dj** | 0 | 13 | 15 | 0 | 0 |

Найденные значение dj называются **константами приведения для столбцов.**

1. Производим **редукцию столбцов** — вычитаем из каждого элемента каждого столбца соответствующее ему значение минимума (Cij = Cij — dj).

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **Город** | **1** | **2** | **3** | **4** | **5** |
| **1** | INF | 3 | 6 |  | 0 |
| **2** | 0 |  | 0 | 40 | 56 |
| **3** | 0 | 15 |  | 70 | 47 |
| **4** | 0 | 0 | 10 |  | 11 |
| **5** | 52 | 40 | 10 | 0 |  |
| **dj** | 0 | 13 | 15 | 0 | 0 |

1. На этом этапе следует провести небольшое, но крайне важное вычисление, а именно определить корневую локальную нижнюю границу (Hk) длины (стоимости, длительности) маршрута. Для этого нужно суммировать константы приведения di и dj.

Формула для вычисления локальной нижней границы в корневой (стартовой) точке решения следующая (далее мы будем использовать несколько отличающиеся формулы): H0 = ∑di + ∑dj.

Найденное значение **H = 130** является текущей или локальной нижней границей. Вообще, при последующих вычислениях H к предыдущему значению локальной нижней границы прибавляется новое значение локальной нижней границы. В же первый раз, нижняя граница вычисляется просто как сумма констант приведения.

1. Ход решения задачи коммивояжера удобно представить в виде графа, где вершинами будут ключевые решения по включению / невключению в итоговый маршрут тех или иных отрезков пути, а ребра, показывающие ход решения, образуют ветви альтернативных вариантов маршрута. Пока у нас есть только одна вершина — «корень» дерева решения, куда мы вписываем значение локальной нижней границы **H = 130**.

Отметим, что, по сути, значение H, это ни что иное как текущая минимальная длина маршрута. Т. е. короче вычисляемый маршрут быть не может, длиннее — возможно

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| **Город** | **1** | **2** | **3** | **4** | **5** | **di** |
| **1** | INF | 3 | 6 |  | 0 | 14 |
| **2** | 0 |  | 0 | 40 | 56 | 14 |
| **3** | 0 | 15 |  | 70 | 47 | 16 |
| **4** | 0 | 0 | 10 |  | 11 | 31 |
| **5** | 52 | 40 | 10 | 0 |  | 27 |
| **dj** | 0 | 13 | 15 | 0 | 0 | 130 |

1. Для каждой нулевой клетки преобразованной матрицы находим «оценку» (pij). Ею будет сумма минимума по строке и минимума по столбцу, на пересечении которых находится данная клетка с нулем. При этом сама нулевая клетка, для которой вычисляется оценка и найденные ранее di и dj НЕ учитываются (но другие нулевые клетки, если они есть в рассматриваемых строке и столбце, учитывать нужно; также учитываются ячейки с M, которые означают бесконечно большое число). Полученную оценку записываем рядом с нулем, в скобках.

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **Город** | **1** | **2** | **3** | **4** | **5** |
| **1** | INF | 3 | 6 |  | 0(15) |
| **2** | 0(0) |  | 0(6) | 40 | 56 |
| **3** | 0(15) | 15 |  | 70 | 47 |
| **4** | 0(0) | 0(3) | 10 |  | 11 |
| **5** | 52 | 40 | 10 | 0(50) |  |

Вычисленные нами оценки также называются штрафами за неиспользование тех отрезков маршрута, которым соответствуют клетки с нулями (напоминаем, что каждая клетка «указывает» на два города: по своей строке — на город, из которого выезжаем, и по своему столбцу — на город, в который приезжаем). Штраф указывают на затраты расстояния, времени, стоимости (или иного параметра, в целях оптимизации которого мы решаем задачу коммивояжера), которые появятся если НЕ выбрать данную клетку.

1. Следующий важный шаг — выбор среди нулевых клеток матрицы той, что имеет наибольшую оценку. Это делается для того, чтобы избежать максимального удлинения маршрута, которое появится если НЕ выбрать такую ячейку.

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **Город** | **1** | **2** | **3** | **4** | **5** |
| **1** | INF | 3 | 6 |  | 0(9) |
| **2** | 0(0) |  | 0(6) | 40 | 56 |
| **3** | 0(15) | 15 |  | 70 | 47 |
| **4** | 0(0) | 0(3) | 10 |  | 11 |
| **5** | 52 | 40 | 10 | 0(50) |  |

В нашем случае максимальную оценку имеет клетку 5-4. Этой ячейке соответствует отрезок пути от города 5 к городу 4.

Стоит отметить, что, когда не одна, а несколько нулевых ячеек имеют одинаковую максимальную оценку, можно выбрать любую из них.

1. На этом этапе решение задачи начинает ветвиться. Возникает развилка с парой альтернативных вариантов: ветвь решения, где мы включаем в маршрут выбранный отрезок пути (5-4) и ветвь, где мы его в маршрут НЕ включаем.

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **Город** | **1** | **2** | **3** | **4** | **5** |
| **1** | INF | 3 | 6 |  | 0(9) |
| **2** | 0(0) |  | 0(6) | 40 | 56 |
| **3** | 0(15) | 15 |  | 70 | 47 |
| **4** | 0(0) | 0(3) | 10 |  | 11 |
| **5** | 52 | 40 | 10 | *INF* |  |

Для начала рассмотрим первую ветвь решения, предусматривающую включение отрезка в маршрут. Для проверки этого варианта необходимо провести редукцию матрицы. Делается это следующим образом: выбрав клетку с максимальной оценкой, вписываем в нее букву M, и затем полностью вычеркиваем соответствующие ей строку и столбец.

В нашем примере мы включаем в итоговый маршрут движение из города 5 в город 4. Следовательно, из города 5 в другие города выезжать уже не будем, а в город 4 по условиям задачи дважды заезжать нельзя. Поэтому данные строка и столбец нам более не понадобятся.

1. Важный момент! В клетку, соответствующую обратному пути (4-5), ставим еще один INF (т. к. мы уже не будем возвращаться обратно из города 4 в город 5).

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Город** | **1** | **2** | **3** | **5** |
| **1** | INF | 3 | 6 | 0 |
| **2** | 0 |  | 0 | 56 |
| **3** | 0 | 15 |  | 47 |
| **4** | 0 | 0 | 10 | INF |

В результате редукции (исключения строки и столбца нулевой клетки с максимальной оценкой) получаем новую уменьшенную таблицу.

1. Теперь необходимо вычислить локальную нижнюю границу для первой ветви решения задачи, предполагающей включение в итоговый маршрут выбранного отрезка пути .

Чтобы это сделать, для новой таблицы вновь находим минимумы по строкам и проводим редукцию строк.

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **Город** | **1** | **2** | **3** | **5** | **di** |
| **1** | INF | 3 | 6 | 0 | 0 |
| **2** | 0 |  | 0 | 56 | 0 |
| **3** | 0 | 15 |  | 47 | 0 |
| **4** | 0 | 0 | 10 | INF | 0 |

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Город** | **1** | **2** | **3** | **5** |
| **1** | INF | 3 | 6 | 0 |
| **2** | 0 |  | 0 | 56 |
| **3** | 0 | 15 |  | 47 |
| **4** | 0 | 0 | 10 | INF |
| **dj** | 0 | 0 | 0 | 0 |

1. Далее находим минимумы по столбцам и проводим редукцию.

Затем вычисляем значение локальной нижней границы по формуле: Hk = Hk-1 + ∑di + ∑dj.

**Получаем: H = 130.**

1. Далее необходимо проверить вторую ветвь — вычислим для нее значение локальной нижней границы. В этом случае мы НЕ исключаем из таблицы отрезок 5-4 и локальная нижняя граница будет равна сумме предыдущей локальной нижней границы и максимальной оценки (т. е. оценки той нулевой клетки, которую мы выбрали на предыдущем шаге):

Hk\* = Hk-1 + pij.

Таким образом локальная нижняя граница для второй ветви (пометим ее звездочкой) составит

**H1\* = 130 + 40 = 170.**

Как видно данная ветвь решения увеличивает минимально возможную длину итогового маршрута до значения 168.



130

(5\*-4\*) =170

(5-4) = 130

Прежде всего, для наглядности, добавим обе рассмотренных ветви решения задачи в граф. Здесь рассматриваемая пара ветвей исходит из корневой вершины. Запишем для каждой ветви относящийся к ней отрезок пути (звездочками помечена ветвь в которой отрезок пути НЕ включен в итоговый маршрут) и значение локальной нижней границы.

Теперь из всех (!) НЕ ветвившихся вершин графа выбираем ту, что имеет наименьшее значение локальной нижней границы.

Важный нюанс — при выборе следует рассматривать не только проверяемую в текущий момент пару ветвей, но и все другие ветви, даже откинутые на предыдущих этапах (пока таких нет, но далее по ходу решения задачи подобный случай будет разобран подробнее).

Поскольку в нашем примере пока всего две ветви и та, что включает в маршрут отрезок 5 → 4, имеет значение локальной нижней границы меньше, чем в случае альтернативного варианта (130 < 170), выбираем ее.

Если мы еще не нашли все отрезки пути, то продолжаем решение задачи и здесь возможны 3 варианта:

* + - выбор ветви, включающей рассматриваемый отрезок — в этом случае решение задачи продолжается с пункта 7. Вновь находить минимумы по строкам и столбцам, а также проводить редукцию строк и столбцов не нужно, т. к. все это уже было сделано при вычислении локальной нижней границы в пункте 10. Поэтому сразу переходим к этапу вычисления оценок нулевых клеток;
    - выбор ветви, НЕ включающей рассматриваемый отрезок — такой вариант предусматривает исключение из итогового маршрута искомого отрезка, для чего в соответствующую ему клетку таблицы необходимо поставить INF. Затем возвращаемся к пункту 2 и продолжаем решение задачи;
    - выбор другой ветви — здесь мы возвращаемся к соответствующим этой ветви этапу решения и таблице с данными. В нашем случае мы выбрали вариант с включением в итоговый маршрут ветви, содержащей отрезок пути 5-4, а значит вновь находить минимумы по строкам и столбцам, а также проводить их редукцию не нужно — это мы уже делали ранее. Поэтому сразу вычисляем оценки для нулевых клеток и определяем ячейку с максимальной.

1. В нашем случае мы выбрали вариант с включением в итоговый маршрут ветви, содержащей отрезок пути 5-4, а значит вновь находить минимумы по строкам и столбцам, а также проводить их редукцию не нужно — это мы уже делали ранее. Поэтому сразу вычисляем оценки для нулевых клеток и определяем ячейку с максимальной (1-5).

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Город** | **1** | **2** | **3** | **5** |
| **1** | INF | 3 | 6 | 0(50) |
| **2** | 0(0) |  | 0(6) | 56 |
| **3** | 0(15) | 15 |  | 47 |
| **4** | 0(0) | 0(3) | 10 | INF |

Рассматриваем два варианта: с включением ветви 1 → 5 в маршрут и без этой ветви. В последнем случае, как мы подсчитали выше, локальная минимальная нижняя граница будет равна **H\* = 130 (предыдущая локальная нижняя граница) + 50 (максимальная оценка нулевой клетки) = 180.**

1. Получаем следующую матрицу:

|  |  |  |  |
| --- | --- | --- | --- |
| **Город** | **1** | **2** | **3** |
| **2** | 0 |  | 0 |
| **3** | 0 | 15 |  |
| **4** | 0 | 0 | 10 |

1. Производим редукцию:

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Город** | **1** | **2** | **3** | **di** |
| **2** | 0 |  | 0 | 0 |
| **3** | 0 | 15 |  | 0 |
| **4** | 0 | 0 | 10 | 0 |

1. Производим редукцию:

|  |  |  |  |
| --- | --- | --- | --- |
| **Город** | **1** | **2** | **3** |
| **2** | 0 |  | 0 |
| **3** | 0 | 15 |  |
| **4** | 0 | 0 | 10 |
| **dj** | 0 | 0 | 0 |

1. Локальная нижняя граница (H) = 130.

130

(5-4) = 130

(5\*-4\*) =170

(1-5) = 130

(1\*-5\*) = 180

Теперь выбираем из **всех** еще не ветвившихся вершин ту, что имеет наименьшее значение локальной нижней границы. Здесь этому условию отвечает вершина 1 — 5, которая имеет локальную нижнюю минимальную границу равную 130. Поэтому нам необходимо вернуться и продолжить развитие этой ветви решения задачи.

1. В нашем случае мы выбрали вариант с включением в итоговый маршрут ветви, содержащей отрезок пути 5-4, а значит вновь находить минимумы по строкам и столбцам, а также проводить их редукцию не нужно — это мы уже делали ранее. Поэтому сразу вычисляем оценки для нулевых клеток и определяем ячейку с максимальной (4-2).

|  |  |  |  |
| --- | --- | --- | --- |
| **Город** | **1** | **2** | **3** |
| **2** | 0(0) |  | 0(10) |
| **3** | 0(15) | 15 |  |
| **4** | 0(0) | 0(15) | 10 |

1. Получаем следующую матрицу:

|  |  |  |
| --- | --- | --- |
| **Город** | **1** | **3** |
| **2** | 0 | 0 |
| **3** | 0 |  |

|  |  |  |  |
| --- | --- | --- | --- |
| **Город** | **1** | **3** | **di** |
| **2** | 0 | 0 | 0 |
| **3** | 0 |  | 0 |

1. Производим редукцию по строкам:

|  |  |  |
| --- | --- | --- |
| **Город** | **1** | **3** |
| **2** | 0 | 0 |
| **3** | 0 |  |
| **dj** | 0 | 0 |

1. Производим редукцию по столбцам:

Найдем для новой таблицы минимумы по строкам (везде 0) и столбцам (также 0). Данные, в результате редукции строк и столбцов, не изменятся.

Вычислим локальную нижнюю границу для ветви, включающей отрезок 4-2: H = 130 + (0 + 0) + (0 + 0) = 130.

В то же время локальная нижняя граница для ветви, не включающей отрезок B-A равна: H\* = 130 + 15 = 145.

1. Локальная нижняя граница (H) = 130.

**130**

(5-4) = **130**

(5\*-4\*) = **170**

(4-2) **130**

(4\*-2\*) **180**

(1-5) **130**

(1\*-5\*) **145**

1. Вычислим оценки для нулевых клеток и найдем среди них максимальную.

|  |  |  |
| --- | --- | --- |
| **Город** | **1** | **3** |
| **2** | 0(0) | 0(INF) |
| **3** | 0(INF) |  |

Здесь две клетки (2-3 и 3-1) имеют максимальное значение оценки, которое бесконечно велико (INF означает бесконечно большое число, и, к примеру, в строке 2 оно является минимумом, т. к. саму нулевую клетку при подсчете оценки не учитываем, а других клеток в этой строке нет). Выбираем любую из этих двух ячеек, допустим, 2-3.

1. Проводим редукцию матрицы. Обратного пути в таблице нет, поэтому исключать его нужды нет.

|  |  |
| --- | --- |
| **Город** | **1** |
| **3** | 0 |

Минимумы по строкам и столбцам — нулевые, редукция строк и столбцов данные не изменит. Определим локальную нижнюю границу для ветви, включающей отрезок 3-1: H = 130 + (0) + (0) = 130.

**130**

(5-4) = **130**

(5\*-4\*) = **170**

(4-2) **130**

(4\*-2\*) **180**

(1-5) **130**

(1\*-5\*) **145**

(2-3) **130**

(2\*-3\*) **INF**

1. Найдя все отрезки пути, остается только соединить их между собой и рассчитать общую длину пути (стоимость поездки по этому маршруту, затраченное время и т. д.). Расстояния между городами берем из самой первой таблицы с исходными данными.

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **Город** | **1** | **2** | **3** | **4** | **5** |
| **1** |  | 28 | 35 |  | 14 |
| **2** | 14 |  | 29 | 54 | 70 |
| **3** | 16 | 42 |  | 86 | 63 |
| **4** | 31 | 44 | 56 |  | 42 |
| **5** | 79 | 80 | 52 | 27 |  |

В нашем примере в ходе решения задачи коммивояжера мы нашли следующие участки пути: 4-2 (44), 1-5 (14), 2-3 (29), 5-4 (27) и 3-1 (16). В скобках указано расстояние между городами. Упорядочим и соединим эти отрезки, тогда маршрут получится следующий:

**(1-5) → (5-4) → (4-2) → (2-3) → (3-1).**

**Длинна пути составляет 130.**

***Задание №3.***

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

Листинги программы представлены ниже:

//main.cpp

#include <iostream>

#include <iomanip>

#include "Traveler.h"

#define N 5

int main()

{

setlocale(LC\_ALL, "rus");

int d[N][N] = {

{ INF, 28, 35, INF, 14 },

{ 14, INF, 29, 54, 70},

{ 16, 42, INF, 86, 63 },

{ 31, 44, 56, INF, 42 },

{ 79, 80, 52, 27, INF}

};

int r[N]; // результат

int s = traveler(

N, // [in] количество городов

(int\*)d, // [in] массив [n\*n] расстояний

r // [out] массив [n] маршрут 0 x x x x

);

std::cout << std::endl << "-- Задача коммивояжера -- ";

std::cout << std::endl << "-- количество городов: " << N;

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

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

{

std::cout << std::endl;

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

if (d[i][j] != INF) std::cout << std::setw(3) << d[i][j] << " ";

else std::cout << std::setw(3) << "INF" << " ";

}

std::cout << std::endl << "-- оптимальный маршрут: ";

for (int i = 0; i < N; i++) std::cout << r[i] << "-->"; std::cout << 0;

std::cout << std::endl << "-- длина маршрута : " << s;

std::cout << std::endl;

system("pause");

return 0;

}

Листинг 1 – main.cpp

// Combi.h

#pragma once

namespace combi

{

struct permutation // генератор перестановок

{

const static bool L = true; // левая стрелка

const static bool R = false; // правая стрелка

short n, // количество элементов исходного множества

\* sset; // массив индексов текущей перестановки

bool\* dart; // массив стрелок (левых-L и правых-R)

permutation(short n = 1); // конструктор (количество элементов исходного множества)

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

\_\_int64 getfirst(); // сформировать первый массив индексов

\_\_int64 getnext(); // сформировать случайный массив индексов

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

unsigned \_\_int64 np; // номер перествновки 0,... count()-1

unsigned \_\_int64 count() const; // вычислить общее кол. перестановок

};

};

Листинг 2 – Combi.h

//-- Traveler.h

// -- Решение задачи коммивояжера перебором вариантов

#define INF 0x7fffffff // бесконечность

#include "Combi.h"

int traveler( // функция возвращает длину оптимального маршрута

int n, // [in] количество городов

const int\* d, // [in] массив [n\*n] расстояний

int\* r // [out] массив [n] маршрут 0 x x x x

);

Листинг 3 – Traveler.h

// Combi.cpp

#include "Combi.h"

#include <algorithm>

#define NINF ((short)0x8000)

namespace combi

{

permutation::permutation(short n)

{

this->n = n;

this->sset = new short[n];

this->dart = new bool[n];

this->reset();

};

void permutation::reset()

{

this->getfirst();

};

\_\_int64 permutation::getfirst()

{

this->np = 0;

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

{

this->sset[i] = i; this->dart[i] = L;

}

return (this->n > 0) ? this->np : -1;

};

\_\_int64 permutation::getnext()

{

\_\_int64 rc = -1;

short maxm = NINF, idx = -1;

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

{

if (i > 0 &&

this->dart[i] == L &&

this->sset[i] > this->sset[i - 1] &&

maxm < this->sset[i]) maxm = this->sset[idx = i];

if (i < (this->n - 1) &&

this->dart[i] == R &&

this->sset[i] > this->sset[i + 1] &&

maxm < this->sset[i]) maxm = this->sset[idx = i];

}

if (idx >= 0)

{

std::swap(this->sset[idx],

this->sset[idx + (this->dart[idx] == L ? -1 : 1)]);

std::swap(this->dart[idx],

this->dart[idx + (this->dart[idx] == L ? -1 : 1)]);

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

if (this->sset[i] > maxm) this->dart[i] = !this->dart[i];

rc = ++this->np;

}

return rc;

};

short permutation::ntx(short i) { return this->sset[i]; };

unsigned \_\_int64 fact(unsigned \_\_int64 x) { return (x == 0) ? 1 : (x \* fact(x - 1)); };

unsigned \_\_int64 permutation::count() const { return fact(this->n); };

}

Листинг 4 – Combi.cpp

// -- Traveler.cpp

#include "Traveler.h"

int sum(int x1, int x2) // суммирование с учетом бесконечности

{

return (x1 == INF || x2 == INF) ? INF : (x1 + x2);

};

int\* firstpath(int n) // формирование 1го маршрута 0,1,2,..., n-1, 0

{

int\* rc = new int[n + 1]; rc[n] = 0;

for (int i = 0; i < n; i++) rc[i] = i;

return rc;

};

int\* source(int n) // формирование исходного массива 1,2,..., n-1

{

int\* rc = new int[n - 1];

for (int i = 1; i < n; i++) rc[i - 1] = i;

return rc;

};

void copypath(int n, int\* r1, const int\* r2) // копировать маршрут

{

for (int i = 0; i < n; i++) r1[i] = r2[i];

};

int distance(int n, int\* r, const int\* d) // длина маршрута

{

int rc = 0;

for (int i = 0; i < n - 1; i++) rc = sum(rc, d[r[i] \* n + r[i + 1]]);

return sum(rc, d[r[n - 1] \* n + 0]); //+ последняя дуга (n-1,0)

};

void indx(int n, int\* r, const int\* s, const short\* ntx)

{

for (int i = 1; i < n; i++) r[i] = s[ntx[i - 1]];

}

int traveler(

int n, // [in] количество городов

const int\* d, // [in] массив [n\*n] расстояний

int\* r // [out] массив [n] маршрут 0 x x x x

)

{

int\* s = source(n), \* b = firstpath(n), rc = INF, dist = 0;

combi::permutation p(n - 1);

int k = p.getfirst();

while (k >= 0) // цикл генерации перестановок

{

indx(n, b, s, p.sset); // новый маршрут

if ((dist = distance(n, b, d)) < rc)

{

rc = dist; copypath(n, r, b);

}

k = p.getnext();

}

return rc;

}

Листинг 5 – Traveler.cpp

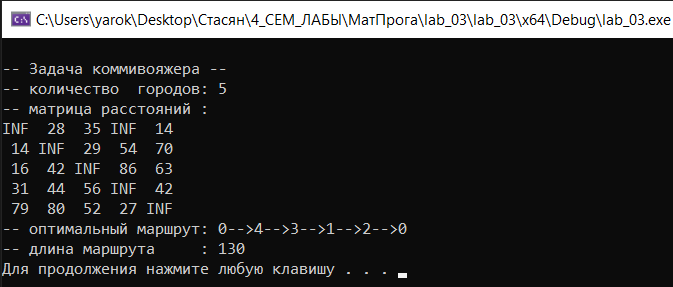
Результат работы программы представлен на рисунке 1.

Рисунок 1 – Результат работы программы

Ответы на вопросы:

1. Задача коммивояжера: найти кратчайший маршрут, проходящий через заданные города ровно один раз и возвращающийся в исходный пункт.

2. Методы решения: динамическое программирование, ветвей и границ, жадные алгоритмы, генетические алгоритмы, муравьиные алгоритмы.

3. Симметричная vs. несимметричная: В симметричной задаче расстояние между двумя городами одинаково в обоих направлениях; в несимметричной — нет.

4. Замкнутая vs. незамкнутая: в замкнутой маршруты возвращаются в начальную точку, в незамкнутой — нет.

5. Метод ветвей и границ: Разделение задачи на подзадачи, исключение неэффективных путей и поиск оптимального.

6. Процедуры метода: ветвление (разделение на подзадачи) и ограничение (исключение заведомо невыгодных решений).

7. Применение метода: оптимизация маршрутов, логистика, задачи в графах.

8. Жадный алгоритм: метод выбора локально оптимальных решений на каждом этапе.

9. Муравьиный алгоритм: имитация поведения муравьев при выборе пути, включая обновление феромонов.

10. Генетический алгоритм: основан на механизмах эволюции; применяется в оптимизации и моделировании сложных систем.

# Лабораторная работа №4

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

***Задание 1.*** На языке С++ сгенерировать случайным образом строку букв латинского алфавита S1 длиной 300 символов и S2 длиной 200.

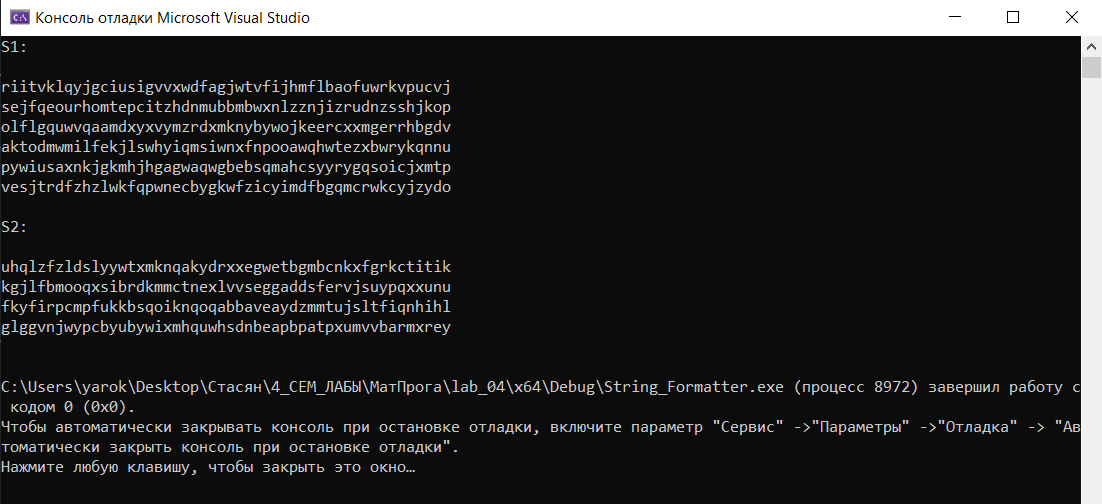
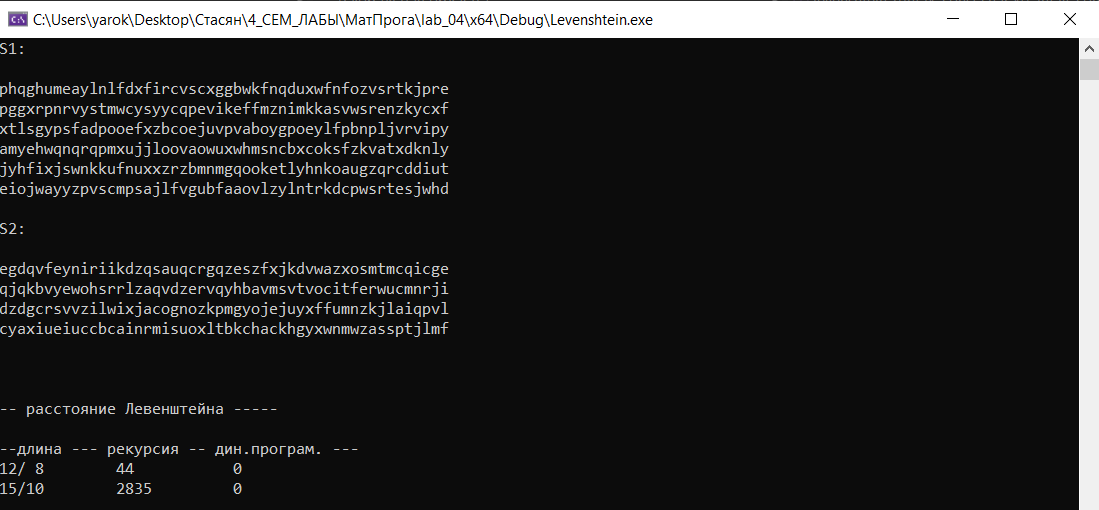
Результат работы программы показан на рисунке 1.

Рисунок 1 – Результат вывода строк

***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).

Результат работы данной программы показан на рисунке 2.

Рисунок 2 – Вывод расстояния Левенштейна

***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).

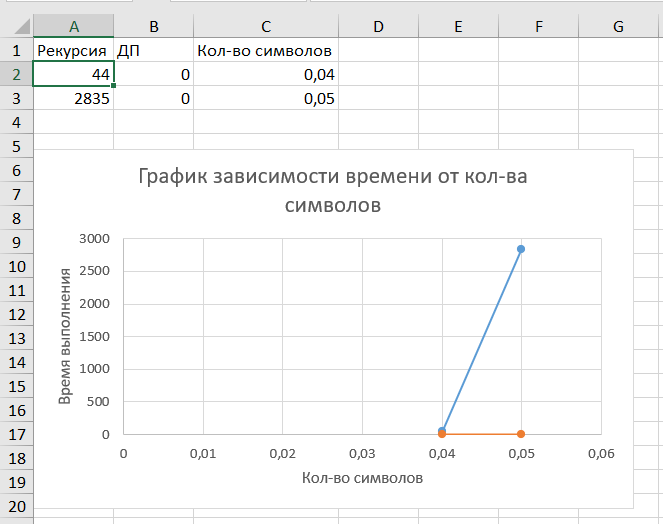
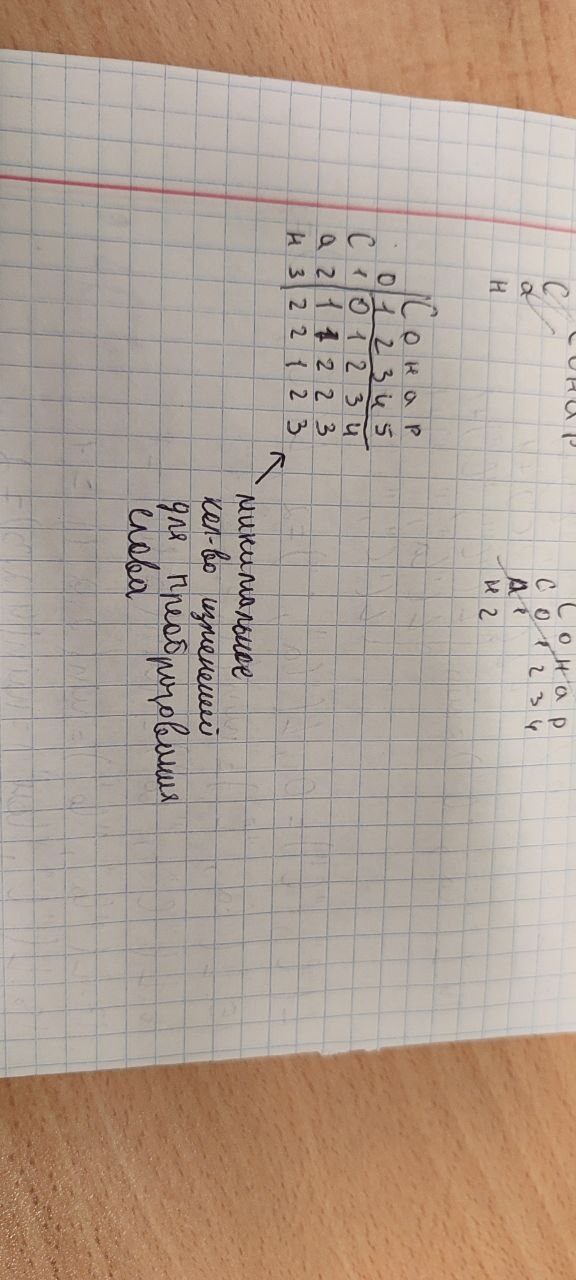
Анализ скорости работы программы динамического и рекурсивного решения показан на рисунке 3.

Рисунок 3 – Анализ скорости работы программы

***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).



***Задание 5.*** 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.

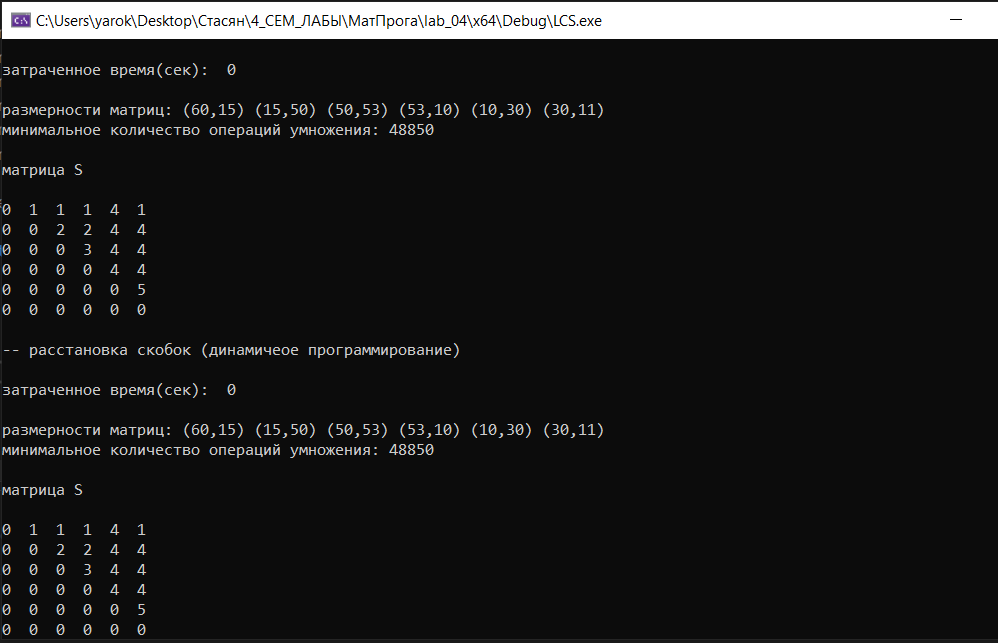
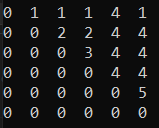
Результат работы данной программы показан на рисунке 4.  


Рисунок 4 – Результат работы программы

Рассмотрим принцип расставления скобок по итоговой матрице.

Итоговая матрица:



Цифры – точки разбиения матрицы.

Начинаем разделение с правой стороны. Выносим за скобки элементы, которые идут после взятого элемента. Повторяем процесс и получаем результат.

***(A1​×(((A2​×A3​)×(A4​×(A5​×A6​)))))***

# Лабораторная работа №5

**Цель работы:** Приобретение навыков решения открытой транспортной задачи

***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.

Исходная таблица:

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| ПОТРЕБИТЕЛИ  ПОСТАВЩИКИ | 1 | 2 | 3 | 4 | 5 | 6 | ЗАПАСЫ |
| 1 | **N+12** | **N+2** | **N+6** | **N+3** | **N+11** | **N+1** | **168+N** |
| 2 | **N+10** | **N** | **N+8** | **N+5** | **N+7** | **N+13** | **113+N** |
| 3 | **N+1** | **N+5** | **N+11** | **N+8** | **N+2** | **N+11** | **150+N** |
| 4 | **N+4** | **N+10** | **N+10** | **N+3** | **N+13** | **N+2** | **159+N** |
| 5 | **N+3** | **N+11** | **N+9** | **N** | **N+10** | **N+4** | **100+N** |
| ПОТРЕБНОСТИ | **143+N** | **107+N** | **131+N** | **193+N** | **95+N** | **163+N** |  |

Полученная таблица в соответствии с вариантом (14).

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| ПОТРЕБИТЕЛИ  ПОСТАВЩИКИ | 1 | 2 | 3 | 4 | 5 | 6 | ЗАПАСЫ |
| 1 | **26** | **16** | **20** | **17** | **25** | **15** | **182** |
| 2 | **24** | **14** | **22** | **19** | **21** | **27** | **127** |
| 3 | **15** | **19** | **25** | **22** | **16** | **25** | **164** |
| 4 | **18** | **24** | **24** | **17** | **27** | **16** | **173** |
| 5 | **17** | **25** | **23** | **14** | **24** | **18** | **114** |
| ПОТРЕБНОСТИ | **157** | **121** | **145** | **207** | **109** | **177** | **916/760** |

Задача открытого типа, так как сумма потребителей не равна сумме запасов:

∑запасы = 182 + 127 + 164 + 173 + 114 = 760

∑потребности = 157 + 121 + 145 + 207 + 109 + 177 =916

Исходя из того, что потребностей больше, чем запасов (916 – 760 = 156), нужно ввести дополнительную строку в матрицу, чтобы привести задачу к закрытой.

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| ПОТРЕБИТЕЛИ  ПОСТАВЩИКИ | 1 | 2 | 3 | 4 | 5 | 6 | ЗАПАСЫ |
| 1 | **26** | **16** | **20** | **17** | **25** | **15** | **182** |
| 2 | **24** | **14** | **22** | **19** | **21** | **27** | **127** |
| 3 | **15** | **19** | **25** | **22** | **16** | **25** | **164** |
| 4 | **18** | **24** | **24** | **17** | **27** | **16** | **173** |
| 5 | **17** | **25** | **23** | **14** | **24** | **18** | **114** |
| 6 | **0** | **0** | **0** | **0** | **0** | **0** | **156** |
| ПОТРЕБНОСТИ | **157** | **121** | **145** | **207** | **109** | **177** | **916** |

***Решение:***

*Метод наименьшей стоимости (минимального элемента)*

Пусть m = 6 (поставщики), n = 6 (потребители).

1. Выбираем ячейку с наименьшим значением. В таблице есть несколько наименьших значений. Выбираем любую ячейку из наименьших. Например, c2,2:

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| ПОТРЕБИТЕЛИ  ПОСТАВЩИКИ | 1 | 2 | 3 | 4 | 5 | 6 | ЗАПАСЫ |
| 1 | **26** | **16** | **20** | **17** | **25** | **15** | **182** |
| 2 | **24** | **14** | **22** | **19** | **21** | **27** | **127** |
| 3 | **15** | **19** | **25** | **22** | **16** | **25** | **164** |
| 4 | **18** | **24** | **24** | **17** | **27** | **16** | **173** |
| 5 | **17** | **25** | **23** | **14** | **24** | **18** | **114** |
| 6 | **0** | **0** | **0** | **0** | **0** | **0** | **156** |
| ПОТРЕБНОСТИ | **157** | **121** | **145** | **207** | **109** | **177** | **916** |

1. X2,2 = min(127, 121) = 121.

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| ПОТРЕБИТЕЛИ  ПОСТАВЩИКИ | 1 | 2 | 3 | 4 | 5 | 6 | ЗАПАСЫ |
| 1 | **26** | **16** | **20** | **17** | **25** | **15** | **182** |
| 2 | **24** | **14|121** | **22** | **19** | **21** | **27** | **6** |
| 3 | **15** | **19** | **25** | **22** | **16** | **25** | **164** |
| 4 | **18** | **24** | **24** | **17** | **27** | **16** | **173** |
| 5 | **17** | **25** | **23** | **14** | **24** | **18** | **114** |
| 6 | **0** | **0** | **0** | **0** | **0** | **0** | **156** |
| ПОТРЕБНОСТИ | **157** | **0** | **145** | **207** | **109** | **177** | **916** |

Далее продолжаем, пока все ячейки запасы и потребности не станут нулевыми.

1. X5,4 = min(114, 207) = 114.

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| ПОТРЕБИТЕЛИ  ПОСТАВЩИКИ | 1 | 2 | 3 | 4 | 5 | 6 | ЗАПАСЫ |
| 1 | **26** | **16** | **20** | **17** | **25** | **15** | **182** |
| 2 | **24** | **14|121** | **22** | **19** | **21** | **27** | **6** |
| 3 | **15** | **19** | **25** | **22** | **16** | **25** | **164** |
| 4 | **18** | **24** | **24** | **17** | **27** | **16** | **173** |
| 5 | **17** | **25** | **23** | **14|114** | **24** | **18** | **0** |
| 6 | **0** | **0** | **0** | **0** | **0** | **0** | **156** |
| ПОТРЕБНОСТИ | **157** | **0** | **145** | **93** | **109** | **177** | **925** |

1. X1,6 = min(182, 177) = 177.

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| ПОТРЕБИТЕЛИ  ПОСТАВЩИКИ | 1 | 2 | 3 | 4 | 5 | 6 | ЗАПАСЫ |
| 1 | **26** | **16** | **20** | **17** | **25** | **15|177** | **5** |
| 2 | **24** | **14|121** | **22** | **19** | **21** | **27** | **6** |
| 3 | **15** | **19** | **25** | **22** | **16** | **25** | **164** |
| 4 | **18** | **24** | **24** | **17** | **27** | **16** | **173** |
| 5 | **17** | **25** | **23** | **14|114** | **24** | **18** | **0** |
| 6 | **0** | **0** | **0** | **0** | **0** | **0** | **156** |
| ПОТРЕБНОСТИ | **157** | **0** | **145** | **93** | **109** | **0** | **925** |

1. X3,1 = min(164, 157) = 157.

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| ПОТРЕБИТЕЛИ  ПОСТАВЩИКИ | 1 | 2 | 3 | 4 | 5 | 6 | ЗАПАСЫ |
| 1 | **26** | **16** | **20** | **17** | **25** | **15|177** | **5** |
| 2 | **24** | **14|121** | **22** | **19** | **21** | **27** | **6** |
| 3 | **15|157** | **19** | **25** | **22** | **16** | **25** | **7** |
| 4 | **18** | **24** | **24** | **17** | **27** | **16** | **173** |
| 5 | **17** | **25** | **23** | **14|114** | **24** | **18** | **0** |
| 6 | **0** | **0** | **0** | **0** | **0** | **0** | **156** |
| ПОТРЕБНОСТИ | **0** | **0** | **145** | **93** | **109** | **0** | **925** |

1. X1,4 = min(5, 93) = 5.

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| ПОТРЕБИТЕЛИ  ПОСТАВЩИКИ | 1 | 2 | 3 | 4 | 5 | 6 | ЗАПАСЫ |
| 1 | **26** | **16** | **20** | **17|5** | **25** | **15|177** | **0** |
| 2 | **24** | **14|121** | **22** | **19** | **21** | **27** | **6** |
| 3 | **15|157** | **19** | **25** | **22** | **16** | **25** | **7** |
| 4 | **18** | **24** | **24** | **17** | **27** | **16** | **173** |
| 5 | **17** | **25** | **23** | **14|114** | **24** | **18** | **0** |
| 6 | **0** | **0** | **0** | **0** | **0** | **0** | **156** |
| ПОТРЕБНОСТИ | **0** | **0** | **145** | **86** | **109** | **0** | **925** |

1. X3,5 = min(7, 109) = 7.

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| ПОТРЕБИТЕЛИ  ПОСТАВЩИКИ | 1 | 2 | 3 | 4 | 5 | 6 | ЗАПАСЫ |
| 1 | **26** | **16** | **20** | **17|5** | **25** | **15|177** | **0** |
| 2 | **24** | **14|121** | **22** | **19** | **21** | **27** | **6** |
| 3 | **15|157** | **19** | **25** | **22** | **16|7** | **25** | **0** |
| 4 | **18** | **24** | **24** | **17** | **27** | **16** | **173** |
| 5 | **17** | **25** | **23** | **14|114** | **24** | **18** | **0** |
| 6 | **0** | **0** | **0** | **0** | **0** | **0** | **156** |
| ПОТРЕБНОСТИ | **0** | **0** | **145** | **86** | **102** | **0** | **925** |

1. X4,4 = min(173, 86) = 86.

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| ПОТРЕБИТЕЛИ  ПОСТАВЩИКИ | 1 | 2 | 3 | 4 | 5 | 6 | ЗАПАСЫ |
| 1 | **26** | **16** | **20** | **17|5** | **25** | **15|177** | **0** |
| 2 | **24** | **14|121** | **22** | **19** | **21** | **27** | **6** |
| 3 | **15|157** | **19** | **25** | **22** | **16|7** | **25** | **0** |
| 4 | **18** | **24** | **24** | **17|86** | **27** | **16** | **87** |
| 5 | **17** | **25** | **23** | **14|114** | **24** | **18** | **0** |
| 6 | **0** | **0** | **0** | **0** | **0** | **0** | **156** |
| ПОТРЕБНОСТИ | **0** | **0** | **145** | **0** | **102** | **0** | **925** |

1. X2,5 = min(6, 102) = 6.

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| ПОТРЕБИТЕЛИ  ПОСТАВЩИКИ | 1 | 2 | 3 | 4 | 5 | 6 | ЗАПАСЫ |
| 1 | **26** | **16** | **20** | **17|5** | **25** | **15|177** | **0** |
| 2 | **24** | **14|121** | **22** | **19** | **21|6** | **27** | **0** |
| 3 | **15|157** | **19** | **25** | **22** | **16|7** | **25** | **0** |
| 4 | **18** | **24** | **24** | **17|86** | **27** | **16** | **87** |
| 5 | **17** | **25** | **23** | **14|114** | **24** | **18** | **0** |
| 6 | **0** | **0** | **0** | **0** | **0** | **0** | **156** |
| ПОТРЕБНОСТИ | **0** | **0** | **145** | **0** | **96** | **0** | **925** |

1. X4,3 = min(87, 145) = 87.

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| ПОТРЕБИТЕЛИ  ПОСТАВЩИКИ | 1 | 2 | 3 | 4 | 5 | 6 | ЗАПАСЫ |
| 1 | **26** | **16** | **20** | **17|5** | **25** | **15|177** | **0** |
| 2 | **24** | **14|121** | **22** | **19** | **21|6** | **27** | **0** |
| 3 | **15|157** | **19** | **25** | **22** | **16|7** | **25** | **0** |
| 4 | **18** | **24** | **24|87** | **17|86** | **27** | **16** | **0** |
| 5 | **17** | **25** | **23** | **14|114** | **24** | **18** | **0** |
| 6 | **0** | **0** | **0** | **0** | **0** | **0** | **156** |
| ПОТРЕБНОСТИ | **0** | **0** | **58** | **0** | **96** | **0** | **925** |

1. X6,3 = min(157, 58) = 58.

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| ПОТРЕБИТЕЛИ  ПОСТАВЩИКИ | 1 | 2 | 3 | 4 | 5 | 6 | ЗАПАСЫ |
| 1 | **26** | **16** | **20** | **17|5** | **25** | **15|177** | **0** |
| 2 | **24** | **14|121** | **22** | **19** | **21|6** | **27** | **0** |
| 3 | **15|157** | **19** | **25** | **22** | **16|7** | **25** | **0** |
| 4 | **18** | **24** | **24|87** | **17|86** | **27** | **16** | **0** |
| 5 | **17** | **25** | **23** | **14|114** | **24** | **18** | **0** |
| 6 | **0** | **0** | **0|58** | **0** | **0** | **0** | **99** |
| ПОТРЕБНОСТИ | **0** | **0** | **0** | **0** | **96** | **0** | **925** |

1. X6,5 = min(107, 96) = 96.

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| ПОТРЕБИТЕЛИ  ПОСТАВЩИКИ | 1 | 2 | 3 | 4 | 5 | 6 | ЗАПАСЫ |
| 1 | **26** | **16** | **20** | **17|5** | **25** | **15|177** | **0** |
| 2 | **24** | **14|121** | **22** | **19** | **21|6** | **27** | **0** |
| 3 | **15|157** | **19** | **25** | **22** | **16|7** | **25** | **0** |
| 4 | **18** | **24** | **24|87** | **17|86** | **27** | **16** | **0** |
| 5 | **17** | **25** | **23** | **14|114** | **24** | **18** | **0** |
| 6 | **0** | **0** | **0|58** | **0** | **0|96** | **0** | **3** |
| ПОТРЕБНОСТИ | **0** | **0** | **0** | **0** | **0** | **0** | **925** |

X2,2 = 121, X5,4 = 14, X5,4 = 114, X1,6 = 177, X3,1 = 157, X1,4 = 5, X3,5 = 7, X4,4 = 86, X2,5 = 6, X4,3 = 87, X6,3 = 58, X6,5 = 96.

Значение функции цели:

Z = 17\*5 + 14\*121 +21\*6 + 15\*157 + 16\*7 + 24\*87 + 17\*86 + 14\*114 + 0 + 0 = **9 518.**

**Метод потенциалов**

Каждому поставщику ai ставим в соответствие некоторое число - ui, называемое потенциалом поставщика. Каждому потребителю bj ставим в соответствие некоторое число - vj, называемое потенциалом потребителя. Для базисной ячейки (задействованного маршрута), сумма потенциалов поставщика и потребителя должна быть равна тарифу данного маршрута.

ui + vj = cij

Найдем предварительные потенциалы ui, vj, по занятым клеткам таблицы, полагая, что u1 = 0:

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| ПОТРЕБИТЕЛИ  ПОСТАВЩИКИ | 1 | 2 | 3 | 4 | 5 | 6 | ui |
| 1 | **26** | **16** | **20** | **17|5** | **25** | **15|177** | **0** |
| 2 | **24** | **14|121** | **22** | **19** | **21|6** | **27** | **0** |
| 3 | **15|157** | **19** | **25** | **22** | **16|7** | **25** | **0** |
| 4 | **18** | **24** | **24|87** | **17|86** | **27** | **16** | **0** |
| 5 | **17** | **25** | **23** | **14|114** | **24** | **18** | **0** |
| 6 | **0** | **0** | **0|58** | **0** | **0|96** | **0** |  |
| vj | **0** | **0** | **0** | **0** | **0** | **0** |

Теперь мы можем найти v4 = a14 – u1 = 17 – 0 = 17.

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| ПОТРЕБИТЕЛИ  ПОСТАВЩИКИ | 1 | 2 | 3 | 4 | 5 | 6 | ui |
| 1 | **26** | **16** | **20** | **17|5** | **25** | **15|177** | **0** |
| 2 | **24** | **14|121** | **22** | **19** | **21|6** | **27** | **0** |
| 3 | **15|157** | **19** | **25** | **22** | **16|7** | **25** | **0** |
| 4 | **18** | **24** | **24|87** | **17|86** | **27** | **16** | **0** |
| 5 | **17** | **25** | **23** | **14|114** | **24** | **18** | **0** |
| 6 | **0** | **0** | **0|58** | **0** | **0|96** | **0** |  |
| vj | **0** | **0** | **0** | **17** | **0** | **0** |

Зная v4 мы теперь можем найти u4 = 17 -17 = 0.

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| ПОТРЕБИТЕЛИ  ПОСТАВЩИКИ | 1 | 2 | 3 | 4 | 5 | 6 | ui |
| 1 | **26** | **16** | **20** | **17|5** | **25** | **15|177** | **0** |
| 2 | **24** | **14|121** | **22** | **19** | **21|6** | **27** | **0** |
| 3 | **15|157** | **19** | **25** | **22** | **16|7** | **25** | **0** |
| 4 | **18** | **24** | **24|87** | **17|86** | **27** | **16** | **0** |
| 5 | **17** | **25** | **23** | **14|114** | **24** | **18** | **0** |
| 6 | **0** | **0** | **0|58** | **0** | **0|96** | **0** |  |
| vj | **0** | **0** | **0** | **17** | **0** | **0** |

Зная v4 можем найти u5 = 14 – 17 = -3.

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| ПОТРЕБИТЕЛИ  ПОСТАВЩИКИ | 1 | 2 | 3 | 4 | 5 | 6 | ui |
| 1 | **26** | **16** | **20** | **17|5** | **25** | **15|177** | **0** |
| 2 | **24** | **14|121** | **22** | **19** | **21|6** | **27** | **0** |
| 3 | **15|157** | **19** | **25** | **22** | **16|7** | **25** | **0** |
| 4 | **18** | **24** | **24|87** | **17|86** | **27** | **16** | **0** |
| 5 | **17** | **25** | **23** | **14|114** | **24** | **18** | **-3** |
| 6 | **0** | **0** | **0|58** | **0** | **0|96** | **0** |  |
| vj | **0** | **0** | **0** | **17** | **0** | **0** |

Продолжаем процесс и получаем…

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| ПОТРЕБИТЕЛИ  ПОСТАВЩИКИ | 1 | 2 | 3 | 4 | 5 | 6 | ui |
| 1 | **26** | **16** | **20** | **17|5** | **25** | **15|177** | **0** |
| 2 | **24** | **14|121** | **22** | **19** | **21|6** | **27** | **-3** |
| 3 | **15|157** | **19** | **25** | **22** | **16|7** | **25** | **-8** |
| 4 | **18** | **24** | **24|87** | **17|86** | **27** | **16** | **0** |
| 5 | **17** | **25** | **23** | **14|114** | **24** | **18** | **-3** |
| 6 | **0** | **0** | **0|58** | **0** | **0|96** | **0** | **-24** |
| vj | **23** | **17** | **24** | **17** | **24** | **15** |

Затем вычисляем max:

a12: 17 + 0 – 16 = 1,

a13: 24 + 0 – 20 = 4,

a41: 23 + 0 – 18 = 5.

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| ПОТРЕБИТЕЛИ  ПОСТАВЩИКИ | 1 | 2 | 3 | 4 | 5 | 6 | ui |
| 1 | **26** | **16** | **20** | **17|5** | **25** | **15|177** | **0** |
| 2 | **24** | **14|121** | **22** | **19** | **21|6** | **27** | **-3** |
| 3 | **15|157** | **19** | **25** | **22** | **16|7** | **25** | **-8** |
| 4 | **18** | **24** | **24|87** | **17|86** | **27** | **16** | **0** |
| 5 | **17** | **25** | **23** | **14|114** | **24** | **18** | **-3** |
| 6 | **0** | **0** | **0|58** | **0** | **0|96** | **0** | **-24** |
| vj | **23** | **17** | **24** | **17** | **24** | **15** |

Выбираем максимальную оценку свободной клетки (4;1): 18. Для этого в перспективную клетку (4;1) поставим знак «+», а в остальных вершинах многоугольника чередующиеся знаки «-», «+», «-».

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| ПОТРЕБИТЕЛИ  ПОСТАВЩИКИ | 1 | 2 | 3 | 4 | 5 | 6 | ui |
| 1 | **26** | **16** | **20** | **17|5** | **25** | **15|177** | **0** |
| 2 | **24** | **14|121** | **22** | **19** | **21|6** | **27** | **-3** |
| 3 | **15|157[-]** | **19** | **25** | **22** | **16|7[+]** | **25** | **-8** |
| 4 | **18[+]** | **24** | **24|87[-]** | **17|86** | **27** | **16** | **0** |
| 5 | **17** | **25** | **23** | **14|114** | **24** | **18** | **-3** |
| 6 | **0** | **0** | **0|58[+]** | **0** | **0|96[-]** | **0** | **-24** |
| vj | **23** | **17** | **24** | **17** | **24** | **15** |

Из грузов хij стоящих в минусовых клетках, выбираем наименьшее, т.е. прибавляем 7 к объемам грузов, стоящих в плюсовых, и вычитаем 7 из xij, стоящих в минусовых клетках. В результате получим новый опорный план.

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| ПОТРЕБИТЕЛИ  ПОСТАВЩИКИ | 1 | 2 | 3 | 4 | 5 | 6 | ui |
| 1 | **26** | **16** | **20** | **17|5** | **25** | **15|177** | **0** |
| 2 | **24** | **14|121** | **22** | **19** | **21|6** | **27** | **-3** |
| 3 | **15|70[-]** | **19** | **25** | **22** | **16|94[+]** | **25** | **-8** |
| 4 | **18|87[+]** | **24** | **24[-]** | **17|86** | **27** | **16** | **0** |
| 5 | **17** | **25** | **23** | **14|114** | **24** | **18** | **-3** |
| 6 | **0** | **0** | **0|147[+]** | **0** | **0|9[-]** | **0** | **-24** |
| vj | **23** | **17** | **24** | **17** | **24** | **15** |

Проверим оптимальность опорного плана. Найдем предварительные потенциалы ui, vj по занятым клеткам таблицы, в которых ui + vj = cij, полагая, что u1 = 0.

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| ПОТРЕБИТЕЛИ  ПОСТАВЩИКИ | 1 | 2 | 3 | 4 | 5 | 6 | ui |
| 1 | **26** | **16** | **20** | **17|5** | **25** | **15|177** | **0** |
| 2 | **24** | **14|121** | **22** | **19** | **21|6** | **27** | **-3** |
| 3 | **15|70[-]** | **19** | **25** | **22** | **16|94[+]** | **25** | **-8** |
| 4 | **18|87[+]** | **24** | **24[-]** | **17|86** | **27** | **16** | **0** |
| 5 | **17** | **25** | **23** | **14|114** | **24** | **18** | **-3** |
| 6 | **0** | **0** | **0|147[+]** | **0** | **0|9[-]** | **0** | **-24** |
| vj | **23** | **17** | **24** | **17** | **24** | **15** |

u1 + v4 = 14; 0 + v4 = 14; v4 = 14

u1 + v6 = 12; 0 + v6 = 12; v6 = 12

u4 + v4 = 14; 14 + u4 = 14; u4 = 0

u5 + v4 = 11; 14 + u5 = 11; u5 = -3

u4 + v1 = 15; 0 + v1 = 15; v1 = 15

u3 + v1 = 12; 15 + u3 = 12; u3 = -3

u3 + v5 = 13; -3 + v5 = 13; v5 = 16

u2 + v5 = 18; 16 + u2 = 18; u2 = 2

u2 + v2 = 11; 2 + v2 = 11; v2 = 9

u6 + v5 = 0; 16 + u6 = 0; u6 = -16

u6 + v3 = 0; -16 + v3 = 0; v3 = 16

Опорный план является оптимальным, так все оценки свободных клеток удовлетворяют условию ui + vj ≤ cij.

Минимальные затраты составят: Z = 4\*111+ 4\*104+ 5\*167+ 5\*72+ 6\*82+ 7\*5+ + 7\*88+ 11\*6+ 0\*135+ 0\*11+ 8\*75 = 8 864;

9 518– 8 864 = 654 – оптимизировано на 654 ед.

*Анализ оптимального плана.*

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| ПОТРЕБИТЕЛИ  ПОСТАВЩИКИ | 1 | 2 | 3 | 4 | 5 | 6 | ui |
| 1 | **26** | **16** | **20** | **17|5** | **25** | **15|177** | **0** |
| 2 | **24** | **14|121** | **22** | **19** | **21|6** | **27** | **0** |
| 3 | **15|70** | **19** | **25** | **22** | **16|94** | **25** | **0** |
| 4 | **18|87** | **24** | **24** | **17|86** | **27** | **16** | **0** |
| 5 | **17** | **25** | **23** | **14|114** | **24** | **18** | **0** |
| 6 | **0** | **0** | **0|147** | **0** | **0|9** | **0** | **0** |
| vj | **0** | **0** | **0** | **0** | **0** | **0** |

Из 1-го склада необходимо доставить 5 ед. товара 4-му потребителю и 177 ед. товара 6-му потребителю.

Из 2-го склада необходимо доставить 121 ед. товара 2-му потребителю и 6 ед. товара 5-му потребителю.

Из 3-го склада необходимо доставить 70 ед. товара 1-му потребителю и 94 ед. товара 5-му потребителю.

Из 4-го склада необходимо доставить 87 ед. товара 1-му потребителю и 86 ед. товара 4-му потребителю.

Из 5-го склада необходимо доставить 114 ед. товара 4-му потребителю.

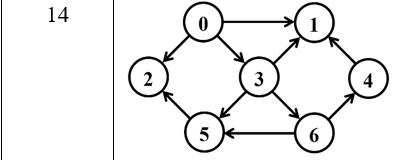
Потребность 3-го потребителя остается неудовлетворенной на 147 ед.

Потребность 5-го потребителя остается неудовлетворенной на 9 ед.

# Лабораторная работа №6

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

***Задание 1.*** Ориентированный граф **G** взять в соответствии с вариантом. Представить его в отчете в виде матрицы смежности, матрицы инцидентности, списка смежных вершин.

Исходный граф:

**Матрица смежности:**

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

**Матрица инцидентности:**

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

**Список смежных вершин:**

**0: [1, 2, 3]**

**1: []**

**2: []**

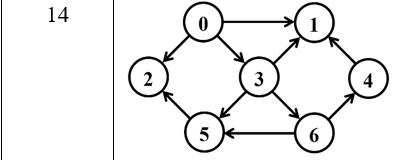
**3: [1, 5, 6]**

**4: [1]**

**5: [2]**

**6: [4, 5]**

***Задание 2.*** 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.

Исходный граф:

**Поиск в ширину:**

Начинаем со стартовой вершины. Предполагается, что берется вершина с минимальным номером. В данном случае этой вершиной будет вершина 0.

Обход в ширину основан на структуре данных **очередь.**

Начинаем обход:

Очередь: [0]

Текущая вершина:

Пройденные вершины: []

2)

Очередь: [1, 2, 3]

Текущая вершина: 0

Пройденные вершины: []

3)

Очередь: [2, 3]

Текущая вершина: 1

Пройденные вершины: [0]

4)

Очередь: [3]

Текущая вершина: 2

Пройденные вершины: [0, 1]

5) Вершину 1 не добавляем в очередь, т.к. она уже пройдена

Очередь: [5, 6]

Текущая вершина: 3

Пройденные вершины: [0, 1, 2]

6) Вершину 2 не добавляем в очередь, т.к. она уже пройдена

Очередь: [6]

Текущая вершина: 5

Пройденные вершины: [0, 1, 2, 3]

7) Вершину 5 не добавляем в очередь, т.к. она уже пройдена

Очередь: [4]

Текущая вершина: 6

Пройденные вершины: [0, 1, 2, 3, 5]

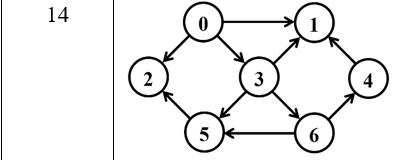
8) Вершину 1 не добавляем в очередь, т.к. она уже пройдена

Очередь: []

Текущая вершина: 4

Пройденные вершины: [0, 1, 2, 3, 5, 6]

**Результат**: обход вершин будет таким – **[0, 1, 2, 3, 5, 6, 4]**

**Исходный граф:**

**Поиск в глубину:**

Начинаем со стартовой вершины. Предполагается, что берется вершина с минимальным номером. В данном случае этой вершиной будет вершина 0.

Обход в ширину основан на структуре данных **стек**.

1)

Стек: [0]

Текущая вершина:

Пройденные вершины: []

2)

Стек: [1, 2, 3]

Текущая вершина: 0

Пройденные вершины: []

3)

Стек: [1, 2, 5, 6]

Текущая вершина: 3

Пройденные вершины: [0]

4)

Стек: [1, 2, 5, 4]

Текущая вершина: 6

Пройденные вершины: [0, 3]

5)

Стек: [1, 2, 5]

Текущая вершина: 4

Пройденные вершины: [0, 3, 6]

6)

Стек: [1, 2]

Текущая вершина: 5

Пройденные вершины: [0, 3, 6, 4]

7)

Стек: [1]

Текущая вершина: 2

Пройденные вершины: [0, 3, 6, 4, 5]

8)

Стек: []

Текущая вершина: 1

Пройденные вершины: [0, 3, 6, 4, 5, 2]

**Результат**: обход вершин будет таким – **[0, 3, 6, 4, 5, 2, 1]**

**Топологическая сортировка**

Выполним топологическую сортировку графа с помощью алгоритма поиска в глубину.

Для этого нужно сделать следующее: запустить алгоритм DFS, при выходе из вершины добавляя вершину в конец списка с ответом, а после окончания алгоритма список с ответом развернуть в противоположном порядке.

Так как у нас уже есть результат обхода графа в глубину, то просто можно развернуть ответ и получить результат топологической сортировки.

Получаем: [0, 3, 6, 4, 5, 2, 1] 🡪 **[1, 2, 5, 4, 6, 3, 0]**

***Задание 3.*** 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.

// --- Graph.h

//

#pragma once

#include <list>

#include "Graph.h"

namespace graph

{

struct AList;

struct AMatrix // матрица смежности

{

int n\_vertex; // количество вершин

int\* mr; // матрица

AMatrix(int n); // создать нулевую матрицу n\*n

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

AMatrix(const AMatrix& am); // создать подобную матрицу

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 n\_vertex; // количество вершин

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

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

AList(int n); // создать массив пустых списков

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

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

AList(const AList& al); // создать подобную структуру

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

};

};

Листинг 1 – реализация структур AMatrix и АList

// ---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

};

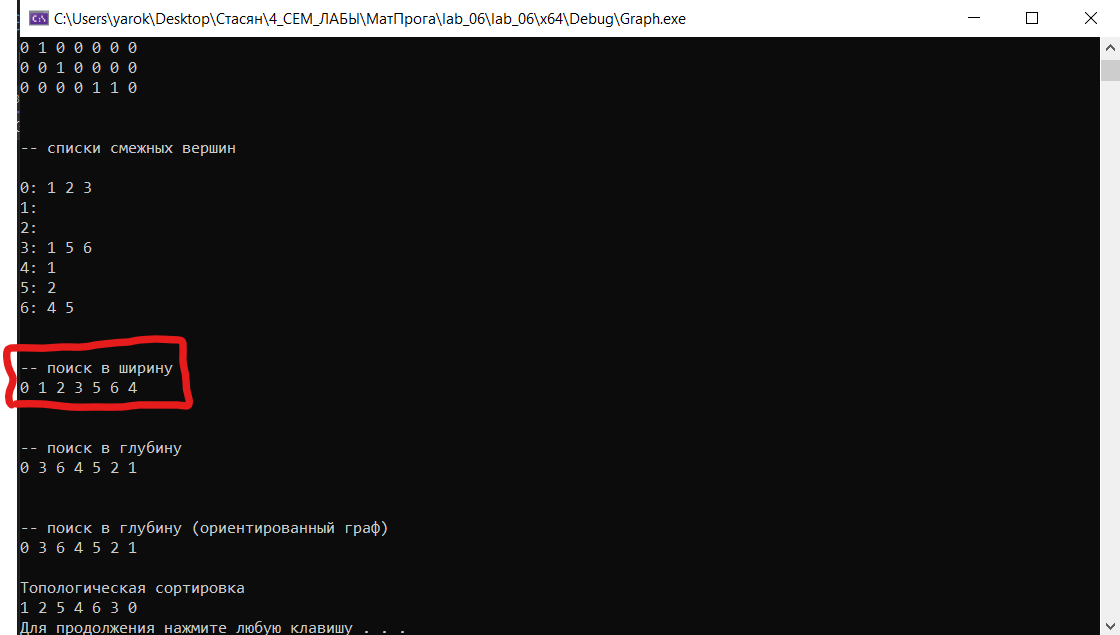
Листинг 2 – реализация BFS

Рисунок 1 – поиск в ширину

***Задание 4.*** Разработать функцию **DFS**  обхода вершин графа, используя метод поиска глубину. Продемонстрировать работу функции. Копии экрана вставить в отчет.

// ---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);

void visit(int v);

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

int get(int i); // получить вершину

};

Листинг 3 – реализация DFS

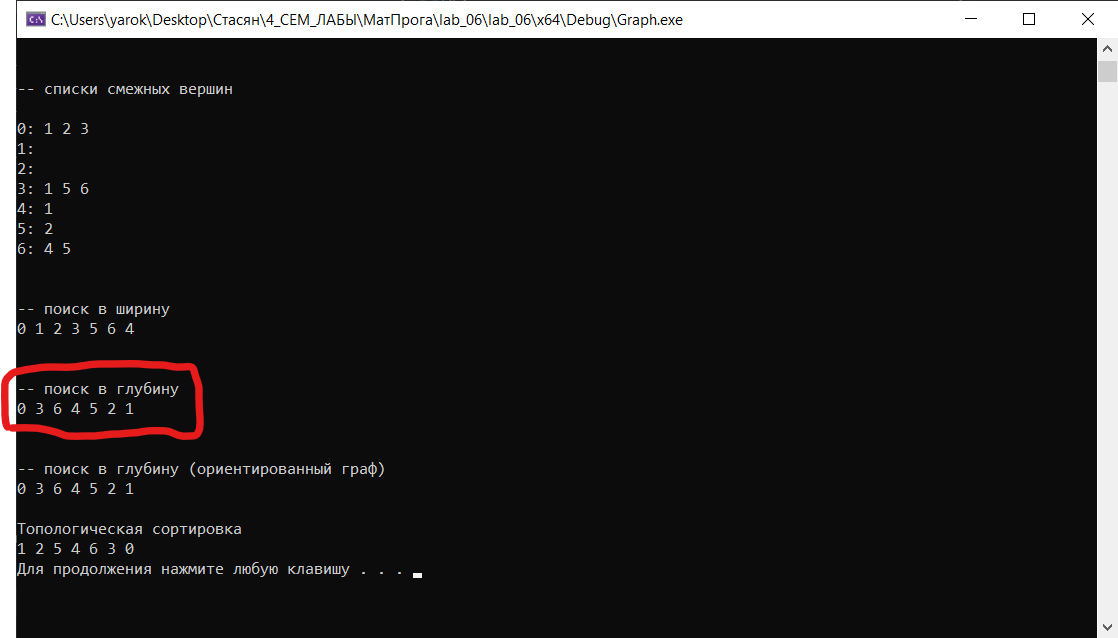


Рисунок 2 – поиск в глубину

***Задание 5.*** Доработайте функцию **DFS**,для выполнения топологической сортировки графа. Продемонстрировать работу функции. Копии экрана вставить в отчет.

// ---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

};

Листинг 4 – измененный DFS

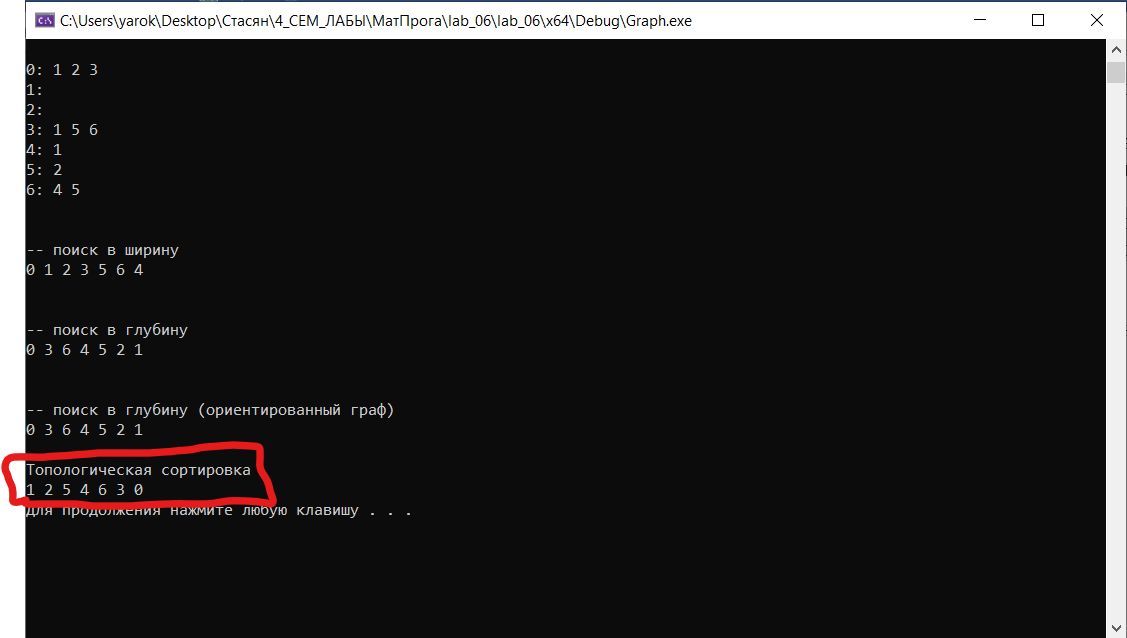


Рисунок 3 – топологическая сортировка

***Задание 6.*** По графу, соответствующему варианту составить минимальное остовное дерево по алгоритму Прима. Шаги построения отразить в отчете.

Веса ребер принять:

W:

W(e0,1)=8; W(e1,0)=5;

W(e0,2)=1; W(e2,0)=3;

W(e0,3)=2; W(e3,0)=8;

W(e1,3)=11; W(e3,1)=4;

W(e1,4)=5; W(e4,1)=3;

W(e2,3)=7; W(e3,2)=9;

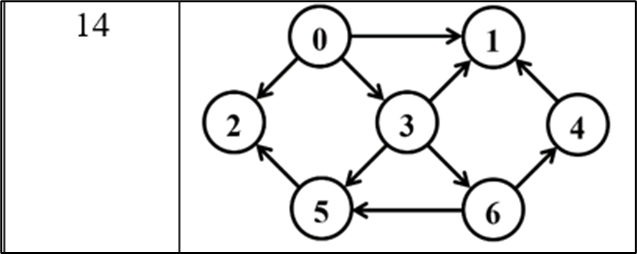
W(e2,5)=11; W(e5,2)=10;

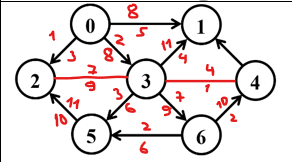
W(e4,3)=4; W(e3,4)=1;

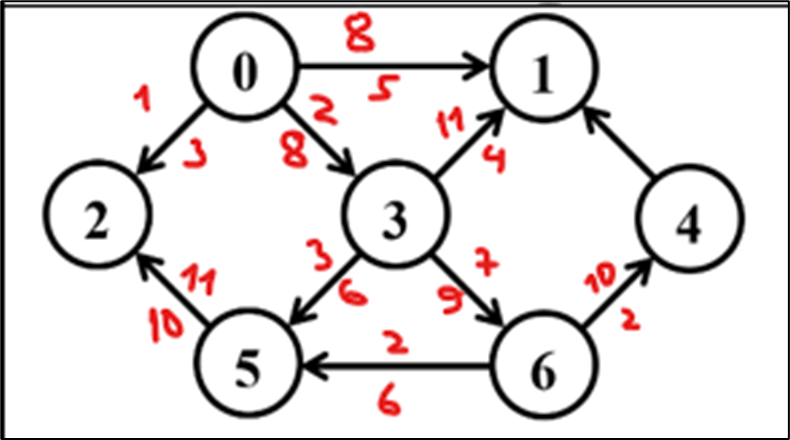
W(e4,6)=10; W(e6,4)=2;

W(e5,6)=2; W(e6,5)=6;

W(e5,3)=3; W(e3,5)=6;

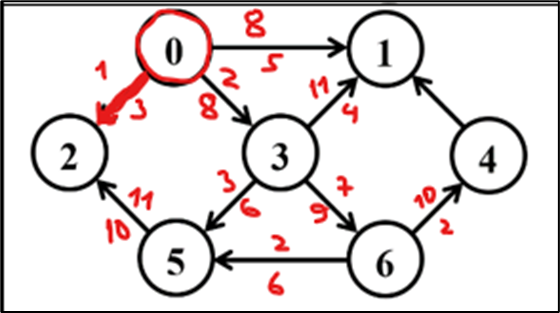
W(e6,3)=7; W(e3,6)=9;

Исходя из того, что данных ребер в графе нет, тогда их можно и не писать, т.к. ребра отсутствуют.

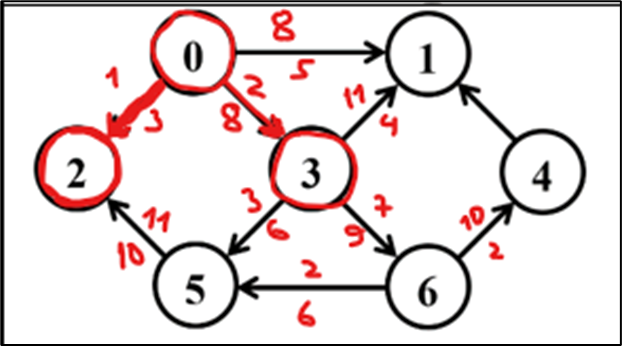


В алгоритме Прима мы должны двигаться следующим образом: рассматриваются только те ребра, которые инцидентны пройденным ребрам.

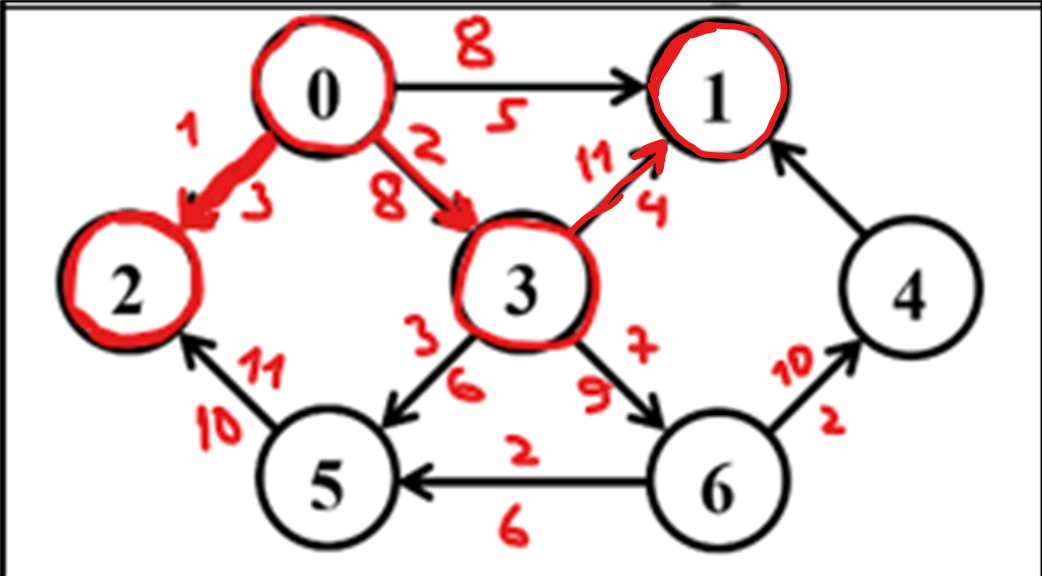
1. Выбираем вершину, с которой только выходят ребра. Это вершина 0.



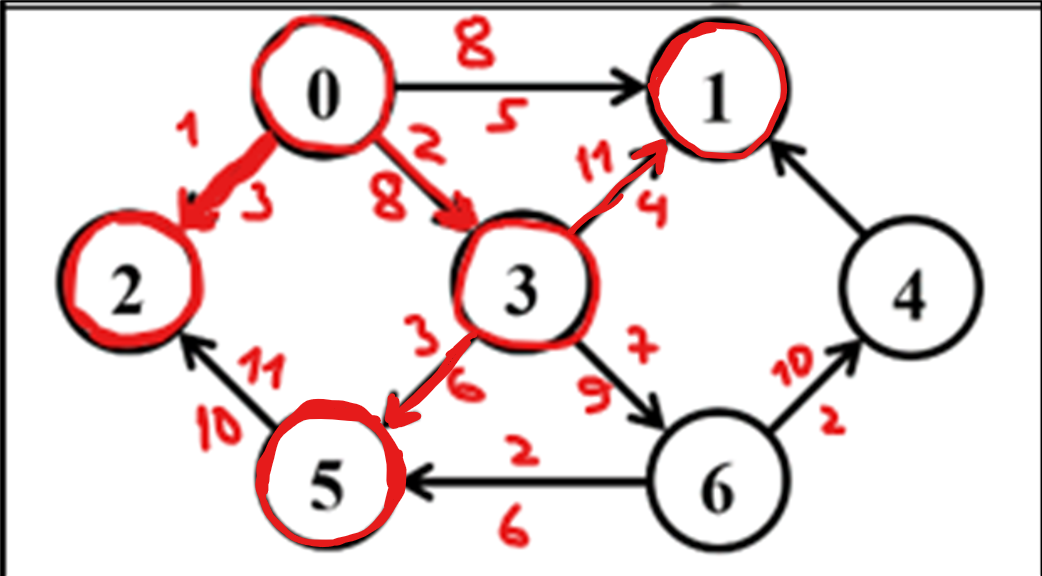
1. Затем берем вершину 3.



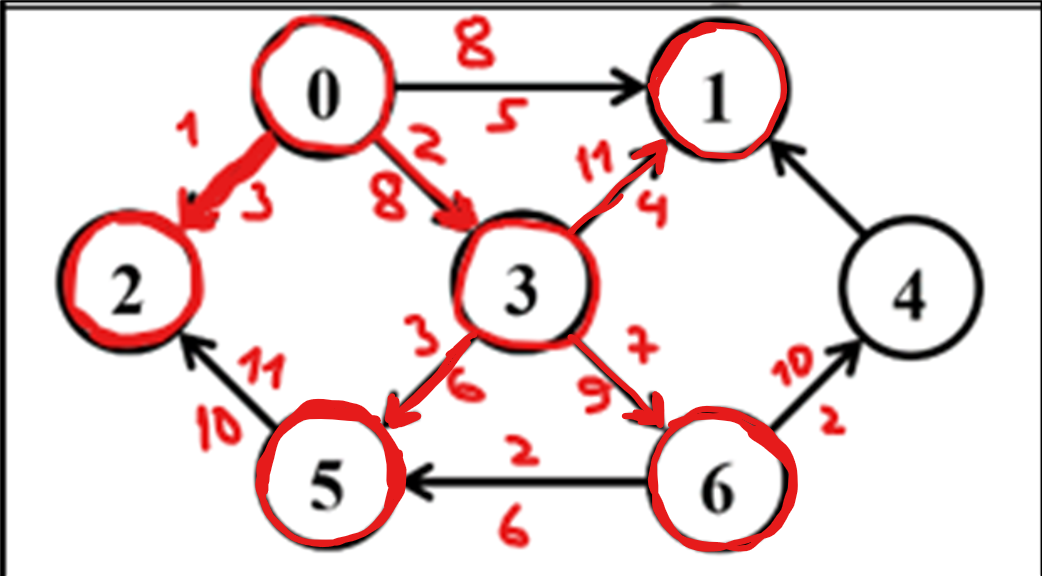
1. Берем вершину 1.



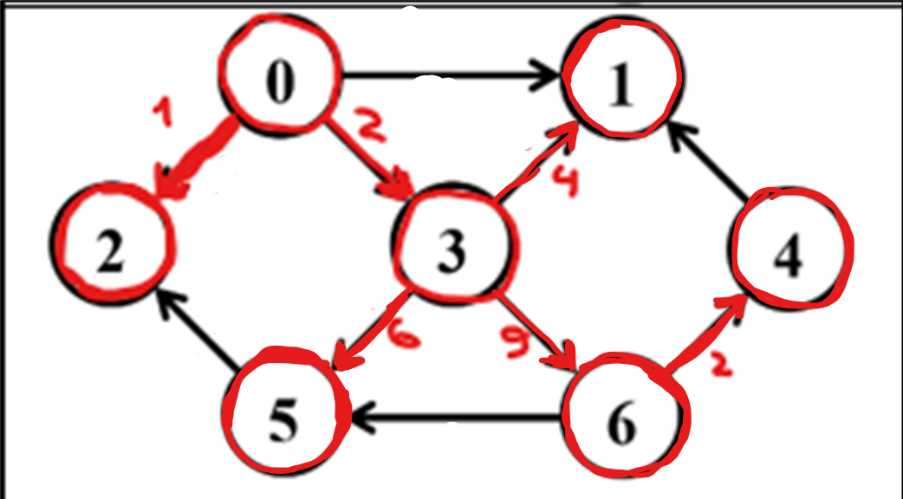
1. Затем берем минимальное ребро из предложенных: 6 или 7. Очевидно, что выбираем ребро с весом 6.



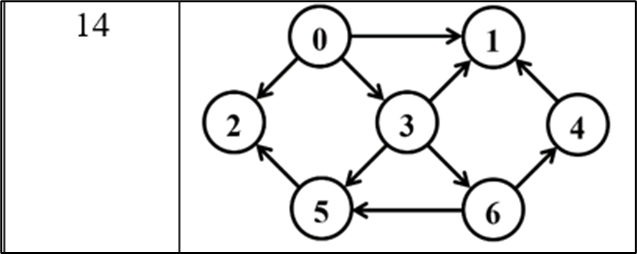
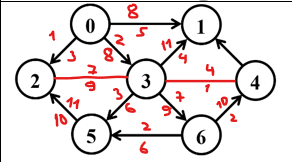
1. Так как выбора у нас особо нет, то выбираем вершину 6.

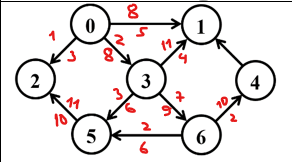


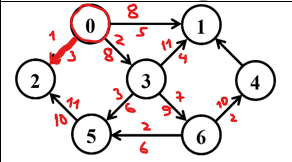
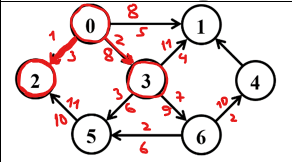
1. Остается последняя вершина – это 4 вершина.

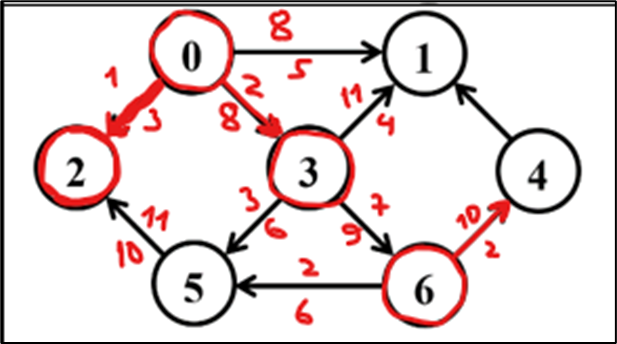


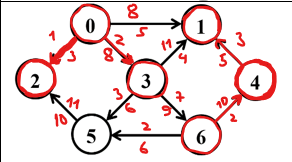
***Задание 7.*** По графу, соответствующему варианту составить минимальное остовное дерево по алгоритму Крускала. Шаги построения отразить в отчете.

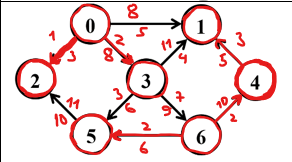


Исходя из того, что данных ребер в графе нет, тогда их можно и не писать, т.к. ребра отсутствуют.

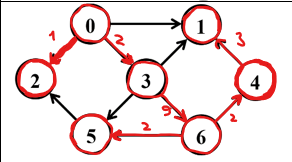
1. Начинаем с вершины, у которой вес ребра минимальный. В данном случае получаем, что вершина 0 имеет минимальное вес ребра (1).
2. По такой аналогии в вершине 0, есть ребро с весом 2. Отмечаем ребро и вершину.
3. Находим вершину 6, у которой есть ребро, соединяющее его с вершиной 4 и имеющее вес 2.



1. В вершине 4 есть ребро с весом равным 3.
2. В вершине 6 есть ребро с весом 2, которое ведет к вершине 5. Отметим это ребро и вершину.



1. Осталось соединить полученные множества. Получаем следующую картину.



|  |  |  |
| --- | --- | --- |
| Вариант | Проект для исследования | Время выполнения всех задач |
| Вариант 1, 7, 13 | «Создание десктопного приложения» | 65 дней |
| Вариант 2, 8, 14 | «Создание мобильной игры» | 50 дней |
| Вариант 3, 9, 15 | «Создание банковского приложения» | 65 дней |
| Вариант 4, 10, 16 | «Создание облачного хранилища» | 60 дней |
| Вариант 5, 11, 17 | «Создание веб-приложения» | 70 дней |
| Вариант 6, 12, 18 | «Создание компьютерной игры» | 90 дней |

# Лабораторная работа №7

**Цель работы:** Приобретение навыков сетевого планирования и составления сетевых графиков, приобретение опыта нахождения критического пути.

***Задание 1. Структурное планирование.***

Подумайте и выделите в проекте, согласно вашему варианту не менее 4 этапов работ. Также разбейте полученные этапы на задачи, их количество в совокупности по этапам должно быть не менее 12. Пример оформления задания смотрите в приложении ниже и в лекционном материале по теме.

***Задание 2. Календарное планирование.***

Распределите время, отпущенное на ваш проект согласно вариантам, на выделенные вами этапы. Скорректируйте сформулированные вами задачи, если это необходимо.

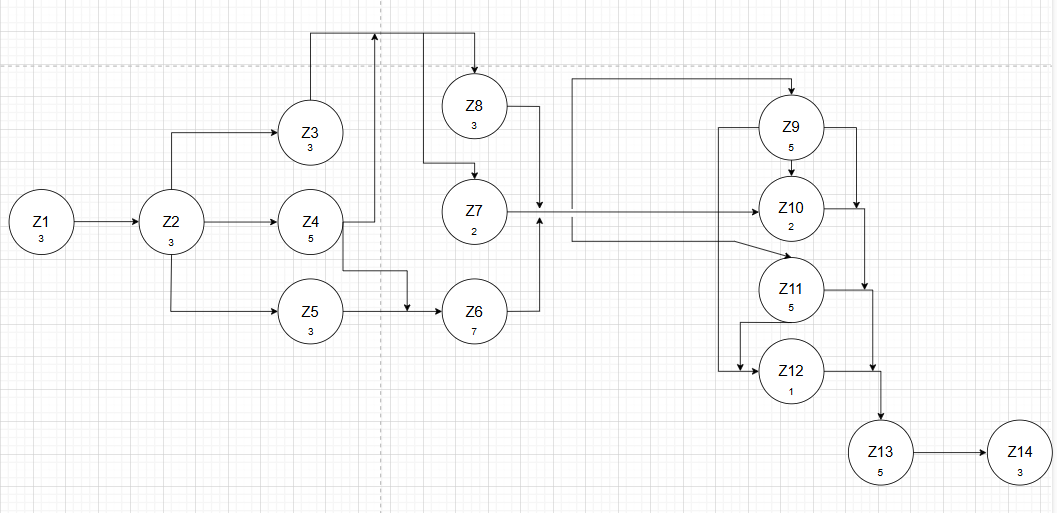
Выполнения 2-ух вышеперечисленных заданий представлено ниже в таблице.

***Тема «Создание мобильной игры»***

|  |  |  |  |
| --- | --- | --- | --- |
| Код операции | Наименование операции | Предшествующие операции | Время (в днях) |
| I. АНАЛИЗ | | | |
| Z1 | Системный анализ |  | 3 |
| Z2 | Анализ требований | Z1 | 3 |
| II. ПРОЕКТИРОВАНИЕ | | | |
| Z3 | Проектирование базы данных | Z2 | 3 |
| Z4 | Проектирование классов | Z2 | 5 |
| Z5 | Проектирование интерфейсов пользователей | Z2 | 3 |
| III. КОДИРОВАНИЕ | | | |
| Z6 | Кодирование интерфейсов пользователей | Z4, Z5 | 7 |
| Z7 | Кодирование процедур СУБД | Z3, Z4 | 2 |
| Z8 | Кодирование классов | Z3, Z4 | 3 |
| IV. ТЕСТИРОВАНИЕ | | | |
| Z9 | Функциональное тестирование | Z6, Z7, Z8 | 5 |
| Z10 | Структурное тестирование | Z6, Z7, Z8 | 2 |
| V. ВНЕДРЕНИЕ | | | |
| Z11 | Разработка документации | Z6, Z7, Z8, Z9 | 5 |
| Z12 | Обучение пользователей | Z9, Z11 | 1 |
| Z13 | Испытание | Z9, Z10, Z11, Z12 | 5 |
| Z14 | Завершение работ | Z13 | 3 |

***Задание 3. Сетевой график, нахождение критического пути.***

Согласно составленному перечню задач и распределённому времени составьте сетевой график вашего проекта. Помните о правилах составления графика и вводите фиктивные операции и операции ожидания если это необходимо.



Длина критического пути: 33 дня. Последовательность задач, принадлежащих критическому пути: **Z1 → Z2 → Z4 → Z6 → Z9 → Z10 → Z13 → Z14.**

**Вывод:** В процессе выполнения лабораторной работы был разработан структурный и календарный план для создания мобильной игры. Также был создан сетевой график и определён критический путь.

# Лабораторная работа №8

**Цель работы:** Освоить решение задач графическим методом.

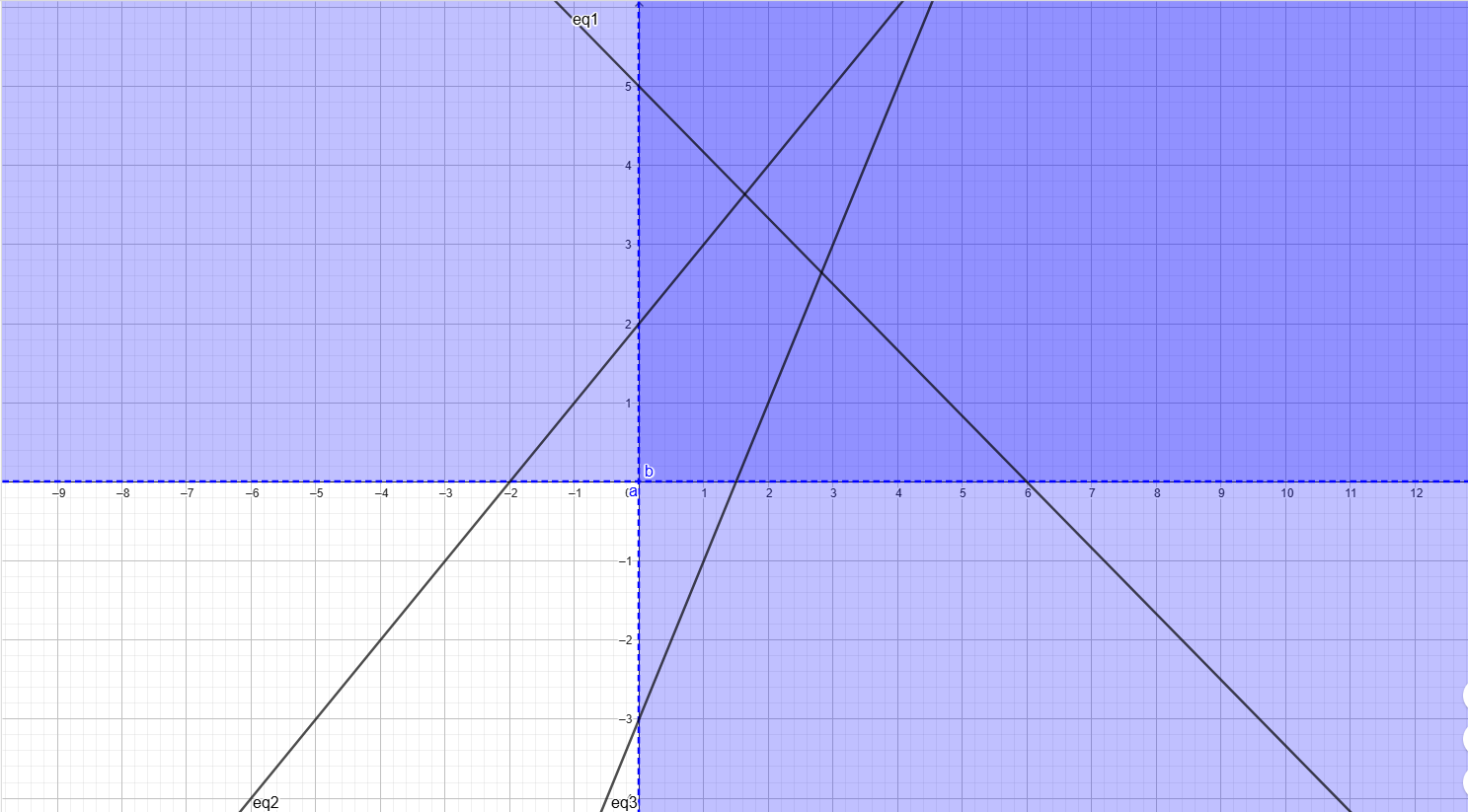
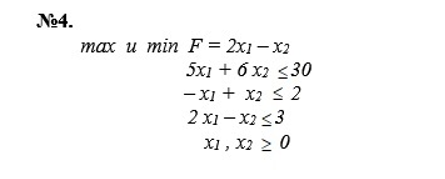
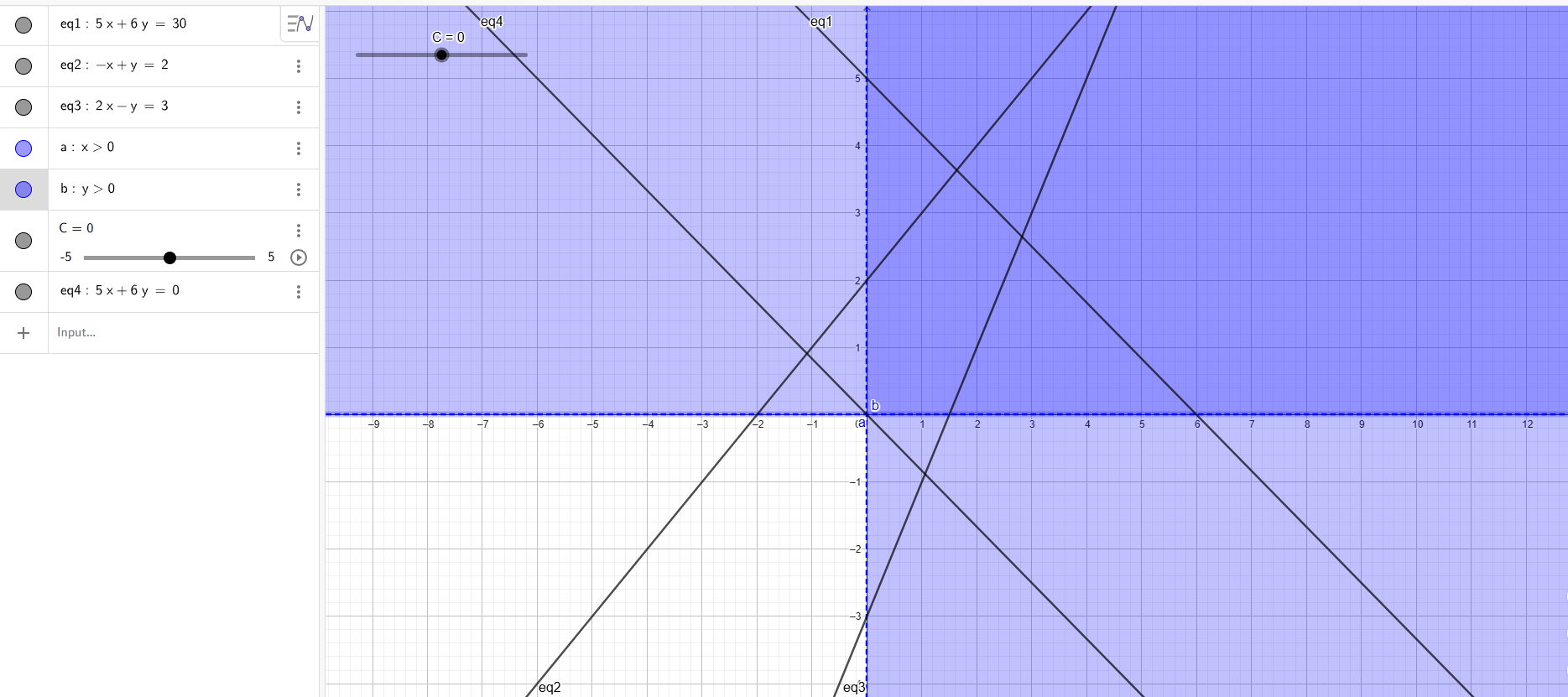
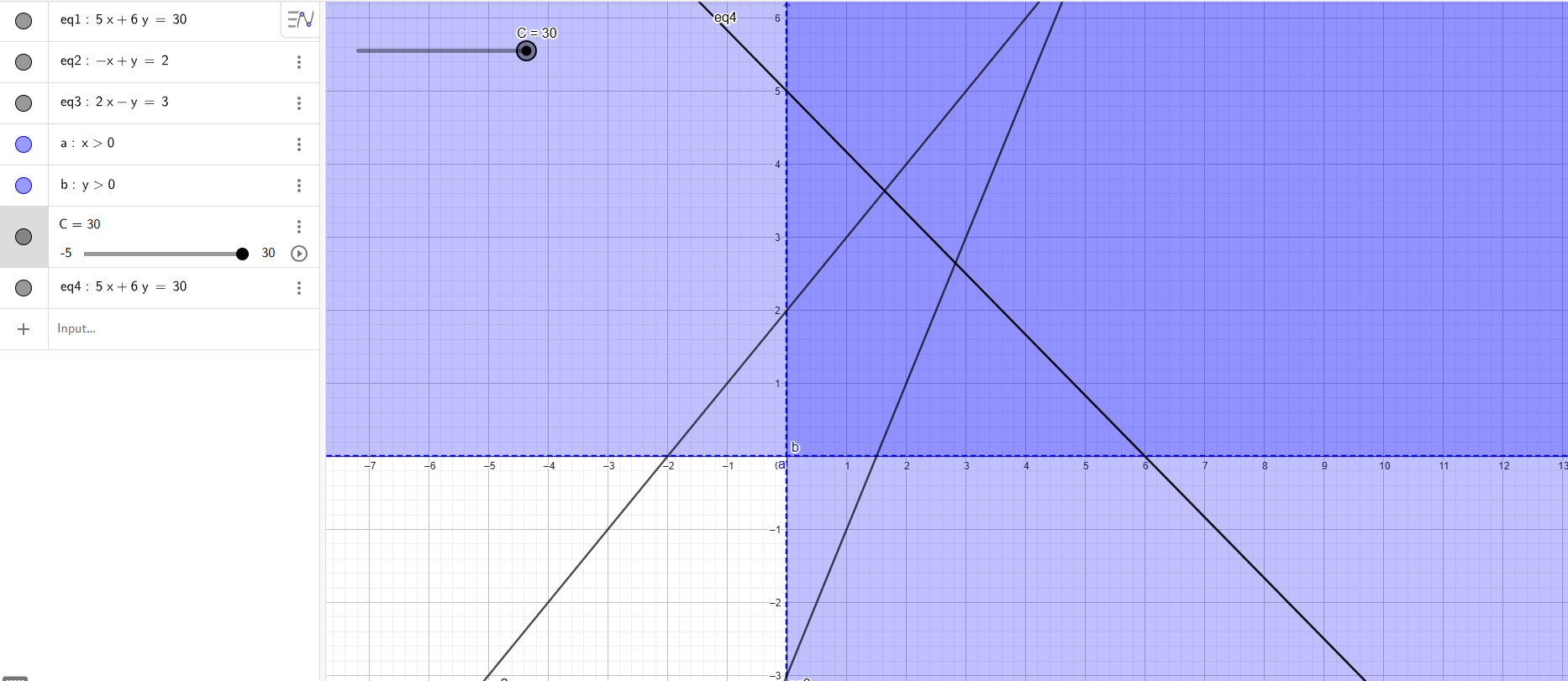
****

Рисунок 1 – Область ограничений

Рисунок 2 – Нахождение минимального значения

Чтобы определить минимальное значение в заданной области, необходимо построить график прямой, заданной уравнением 5𝑥 + 6𝑦 = С и затем сдвигать эту прямую влево к крайней точке выделенной фигуры. Мы ищем координаты точки, в которой достигается минимальное значение. Эта точка может быть (0; 0) или (0; 2), выберем 1. Подставив выбранные координаты в выражение 5𝑥 + 6𝑦, получаем: 0 + 0 = 0. Таким образом, минимальное значение функции равно 0 и совпадает с графическим решением.

Рисунок 3 – Нахождение максимального значения

Чтобы определить максимальное значение в заданной области, необходимо построить график прямой, заданной уравнением 5𝑥 + 6𝑦 = С и затем сдвигать эту прямую вправо к крайней точке выделенной фигуры. Мы ищем координаты точки, в которой достигается максимальное значение. Эта точка (2.82; 2.47). Подставив выбранные координаты в выражение 5𝑥 + 6𝑦, получаем: 5 ⋅ 2.82 + 6 ⋅ 2.47 = 30. Таким образом, максимальное значение функции равно 30 и совпадает с графическим решением.

**Вывод:** Мы научились решать задачи графическим способом и проверять корректность решения с помощью графика.