МИНИСТЕРСТВО ОБРАЗОВАНИЯ РЕСПУБЛИКИ БЕЛАРУСЬ  
Учреждение образования «БЕЛОРУССКИЙ ГОСУДАРСТВЕННЫЙ   
ТЕХНОЛОГИЧЕСКИЙ УНИВЕРСИТЕТ»

Факультет Информационных технологий

Кафедра Информационных систем и технологий

Специальность 1—98 01 03 «Программное обеспечение информационной

безопасности мобильных систем»

**Исполнитель**

студент 2 курса 8 группы Высоцкий Ф.С.

Оглавление

[Лабораторная работа 1. Вспомогательные функции 1](#_Toc166694637)

[Лабораторная работа 2. Комбинаторные алгоритмы решения оптимизационных задач 9](#_Toc166694638)

[Лабораторная работа 3. Метод ветвей и границ. Задача коммивояжера и методы её решения. 31](#_Toc166694639)

[Лабораторная работа 4. ДИНАМИЧЕСКОЕ ПРОГРАММИРОВАНИЕ 40](#_Toc166694640)

[Лабораторная работа 5. ТРАНСПОРТНАЯ ЗАДАЧА 53](#_Toc166694641)

[Лабораторная работа 6. АЛГОРИТМЫ НА ГРАФАХ 63](#_Toc166694642)

[Лабораторная работа 7. Сетевые модели 82](#_Toc166694643)

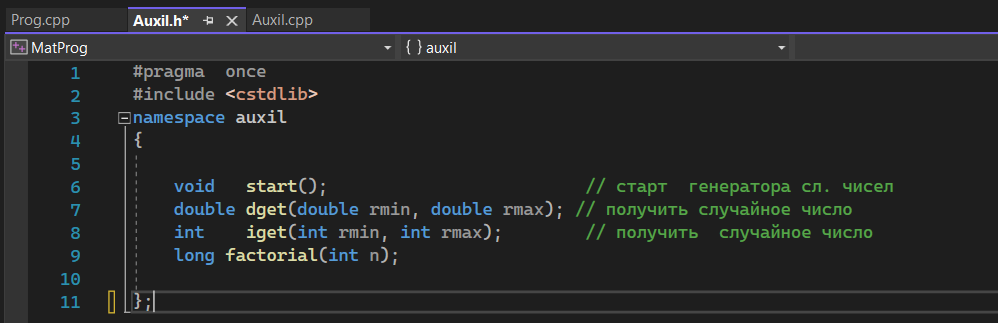
[Лабораторная работа 8. Графический метод решения оптимизационных задач 85](#_Toc166694644)

# Лабораторная работа 1. Вспомогательные функции

Вариант 3

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

***Задание 1.*** Разработайте три функции (start, dget и iget):



Auxil.cpp:

**#include "Auxil.h"**

**#include <ctime>**

**namespace auxil**

**{**

**void start()**

**{**

**srand((unsigned)time(NULL));**

**};**

**double dget(double rmin, double rmax)**

**{**

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

**};**

**int iget(int rmin, int rmax)**

**{**

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

**};**

**long factorial(int n) {**

**if (n == 0 || n == 1) {**

**return 1;**

**}**

**else {**

**return n \* factorial(n - 1);**

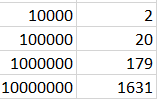
**}**

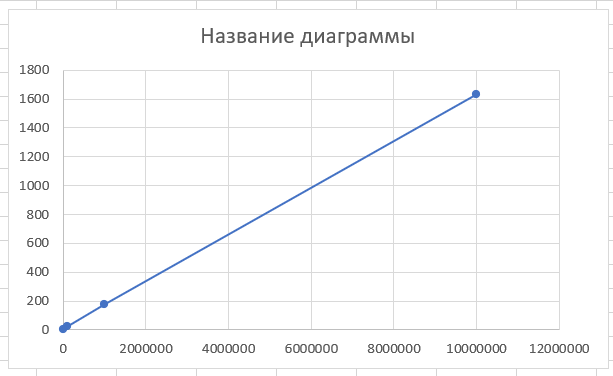
**}**

**}**

***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.





#include "Auxil.h" // вспомогательные функции

#include <iostream>

#include <ctime>

#include <locale>

#define CYCLE 1000000 // количество циклов

int main()

{

double av1 = 0, av2 = 0;

clock\_t t1 = 0, t2 = 0;

setlocale(LC\_ALL, "rus");

auxil::start(); // старт генерации

t1 = clock(); // фиксация времени

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

{

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

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

}

t2 = clock(); // фиксация времени

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

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

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

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

std::cout << std::endl << " (сек): "

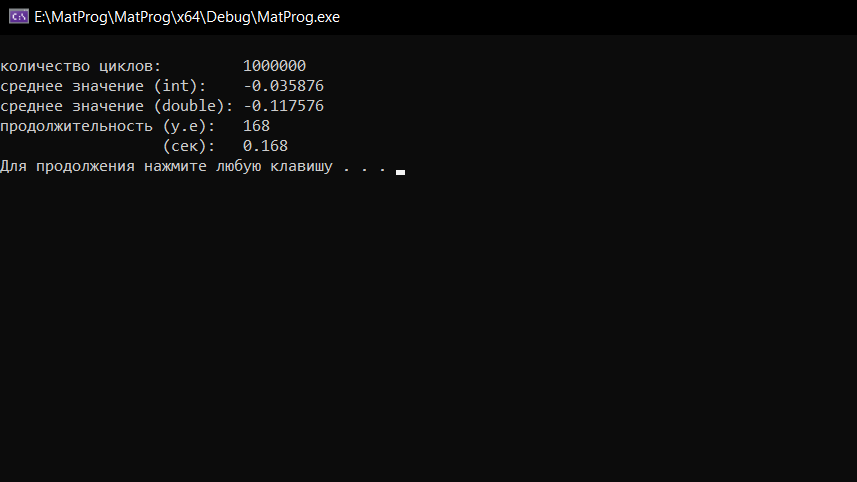
<< ((double)(t2 - t1)) / ((double)CLOCKS\_PER\_SEC);

std::cout << std::endl;

system("pause");

return 0;

}



**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.

#include "Auxil.h" // вспомогательные функции

#include <iostream>

#include <ctime>

#include <locale>

#define CYCLE 1000000 // количество циклов

int main()

{

double av1 = 0, av2 = 0;

clock\_t t1 = 0, t2 = 0;

setlocale(LC\_ALL, "rus");

auxil::start(); // старт генерации

t1 = clock(); // фиксация времени

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

{

auxil::factorial(100); // сумма случайных чисел

}

t2 = clock();

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

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

std::cout << std::endl << " (сек): "

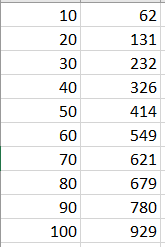
<< ((double)(t2 - t1)) / ((double)CLOCKS\_PER\_SEC);

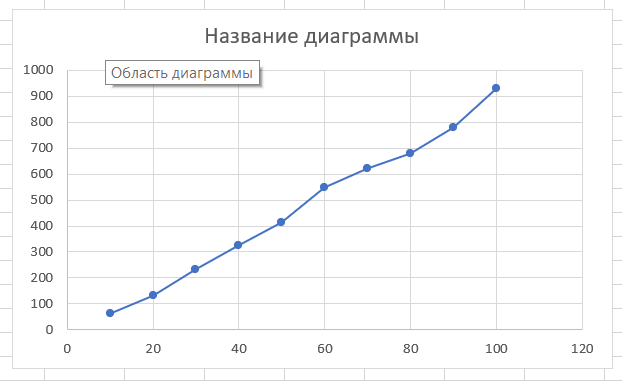
std::cout << std::endl;

system("pause");

return 0;

}





Линейная зависимость

Вывод: я приобрёл навыки составления и отладки программ с использованием пользовательских функций для замера продолжительности процесса вычисления.

# Лабораторная работа 2. Комбинаторные алгоритмы решения оптимизационных задач

Вариант 3

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

**Задание 1.** 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)

**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

**};**

**};**

// Combi.cpp

**#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);};**

**};**

// Main

**#include <iostream>**

**#include "Combi.h"**

**int main()**

**{**

**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

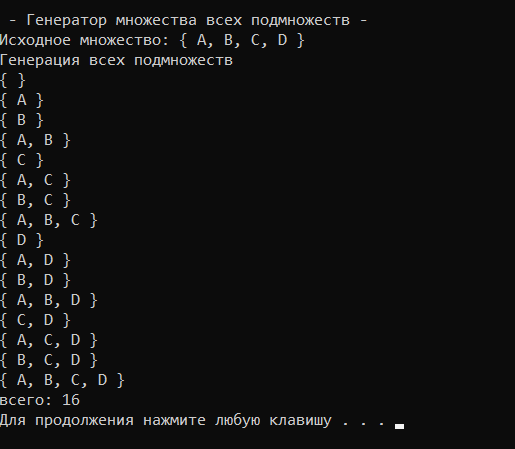
**};**

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

**system("pause");**

**return 0;**

**}**



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

// 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

**};**

**};**

// Combi.cpp

**#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;**

**};**

**};**

// Main

**#include <iostream>**

**#include "Combi.h"**

**int main()**

**{**

**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();**

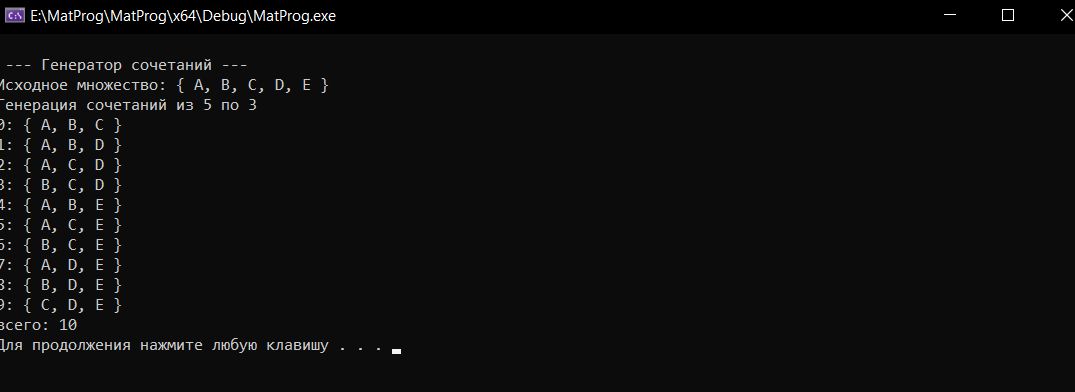
**};**

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

**system("pause");**

**return 0;**

**}**



**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

**};**

**};**

// 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); };**

**}**

// --- Main

**#include <iostream>**

**#include "Combi.h"**

**#include <iomanip>**

**int main()**

**{**

**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();**

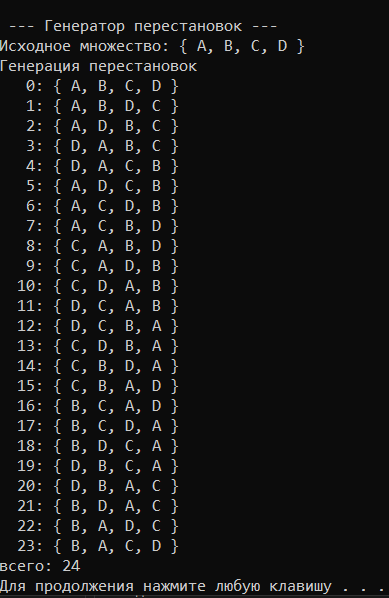
**};**

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

**system("pause");**

**return 0;**

**}**



**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**

**};**

**}**

**};**

// Combi.cpp

**#include "stdafx.h"**

**#include "Combi.h"**

**namespace combi**

**{**

**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]; };**

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;**

**};**

**}**

**};**

// --- main

**#include <iostream>**

**#include <iomanip>**

**#include "Combi.h"**

**#define N (sizeof(AA)/2)**

**#define M 3**

**int main()**

**{**

**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();**

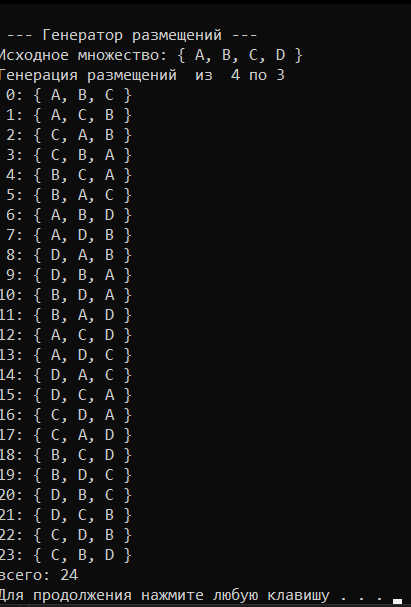
**};**

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

**system("pause");**

**return 0;**

**}**

****

**Задание 5.**  Решить в соответствии с вариантом задачу и результат занести в отчет:

об оптимальной загрузке судна (веса контейнеров сгенерировать случайным образом: ограничение по общему весу – 1500 кг., количество мест на судне для контейнеров – 5, количество контейнеров 25, веса контейнеров 100 – 900 кг., доход от перевозки 10 – 150 у.е.);

// --- main

**#include <iostream>**

**#include <iomanip>**

**#include <cstdlib>**

**#include "Boat.h"**

**#define NN 25**

**#define MM 5**

**int main()**

**{**

**setlocale(LC\_ALL, "rus");**

**int V = 1500,**

**v[NN] = {},**

**c[NN] = {};**

**for (int i = 0; i < NN; i++) {**

**v[i] = rand() % 801 + 100; // Генерация случайного числа от 100 до 900**

**}**

**for (int i = 0; i < NN; i++) {**

**c[i] = rand() % 141 + 10; // Генерация случайного числа от 10 до 150**

**}**

**short r[MM];**

**int cc = boat(**

**V, // [in] максимальный вес груза**

**MM, // [in] количество мест для контейнеров**

**NN, // [in] всего контейнеров**

**v, // [in] вес каждого контейнера**

**c, // [in] доход от перевозки каждого контейнера**

**r // [out] результат: индексы выбранных контейнеров**

**);**

**std::cout << std::endl << "- Задача о размещении контейнеров на судне";**

**std::cout << std::endl << "- общее количество контейнеров : " << NN;**

**std::cout << std::endl << "- количество мест для контейнеров : " << MM;**

**std::cout << std::endl << "- ограничение по суммарному весу : " << V;**

**std::cout << std::endl << "- вес контейнеров : ";**

**for (int i = 0; i < NN; i++) std::cout << std::setw(3) << v[i] << " ";**

**std::cout << std::endl << "- доход от перевозки : ";**

**for (int i = 0; i < NN; i++) std::cout << std::setw(3) << c[i] << " ";**

**std::cout << std::endl << "- выбраны контейнеры (0,1,...,m-1): ";**

**for (int i = 0; i < MM; i++) std::cout << r[i] << " ";**

**std::cout << std::endl << "- доход от перевозки : " << cc;**

**std::cout << std::endl << "- общий вес выбранных контейнеров : ";**

**int s = 0; for (int i = 0; i < MM; i++) s += v[r[i]]; std::cout << s;**

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

**system("pause");**

**return 0;**

**}**

// 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**

**};**

**};**

// Combi.cpp

**#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;**

**};**

**};**

// Boat.h

**// --- Вoat.h**

**// -- решение задачи об оптимальной загрузке судна**

**// функция возвращает доход от перевози выбранных контейнеров**

**#pragma once**

**#include "Combi.h"**

**int boat(**

**int V, // [in] максимальный вес груза**

**short m, // [in] количество мест для контейнеров**

**short n, // [in] всего контейнеров**

**const int v[], // [in] вес каждого контейнера**

**const int c[], // [in] доход от перевозки каждого контейнера**

**short r[] // [out] результат: индексы выбранных контейнеров**

**);**

**// --- Вoat.cpp**

**#include "Boat.h"**

**namespace boatfnc**

**{**

**int calcv(combi::xcombination s, const int v[]) // вес**

**{**

**int rc = 0;**

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

**return rc;**

**};**

**int calcc(combi::xcombination s, const int c[]) // доход**

**{**

**int rc = 0;**

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

**return rc;**

**};**

**void copycomb(short m, short\* r1, const short\* r2) // копировать**

**{**

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

**};**

**}**

**int boat(**

**int V, // [in] максимальный вес груза**

**short m, // [in] количество мест для контейнеров**

**short n, // [in] всего контейнеров**

**const int v[], // [in] вес каждого контейнера**

**const int c[], // [in] доход от перевозки каждого контейнера**

**short r[] // [out] результат: индексы выбранных контейнеров**

**)**

**{**

**combi::xcombination xc(n, m);**

**int rc = 0, i = xc.getfirst(), cc = 0;**

**while (i > 0)**

**{**

**if (boatfnc::calcv(xc, v) <= V)**

**if ((cc = boatfnc::calcc(xc, c)) > rc)**

**{**

**rc = cc; boatfnc::copycomb(m, r, xc.sset);**

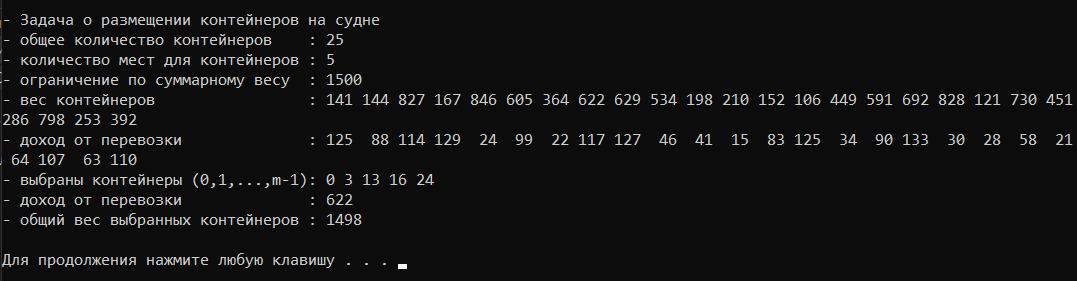
**}**

**i = xc.getnext();**

**};**

**return rc;**

**};**



**Задание 6.** Исследовать зависимость времени вычисления необходимое для решения задачи (в соответствии с вариантом) от размерности задачи и результат в виде графика с небольшим пояснением занести в отчет:

3, 7, 11, 15) об оптимальной загрузке судна (количество мест на судне для контейнеров – 6, количество контейнеров 25 – 35)

// --- main

**#include <iostream>**

**#include <iomanip>**

**#include <time.h>**

**#include "Boat.h"**

**#define SPACE(n) std::setw(n)<<" "**

**#define NN 11**

**int main()**

**{**

**setlocale(LC\_ALL, "rus");**

**std::cout << std::endl << "-- Задача о размещении контейнеров -- ";**

**std::cout << std::endl << "-- всего мест: " << 6;**

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

**std::cout << std::endl << " контейнеров вычисления ";**

**clock\_t t1, t2;**

**for (int i = 25; i < 36; i++)**

**{**

**short r[6];**

**int\* v = new int[i];**

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

**v[j] = rand() % 801 + 100;**

**}**

**int\* c = new int[i];**

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

**c[j] = rand() % 141 + 10;**

**}**

**t1 = clock();**

**boat(1500, 6, i, v, c, r);**

**t2 = clock();**

**std::cout << std::endl << SPACE(7) << std::setw(2) << i**

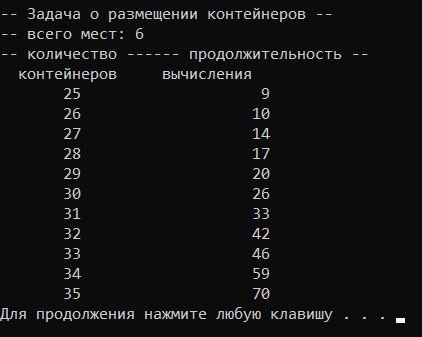
**<< SPACE(15) << std::setw(6) << (t2 - t1);**

**}**

**std::cout << std::endl; system("pause");**

**return 0;**

**}**

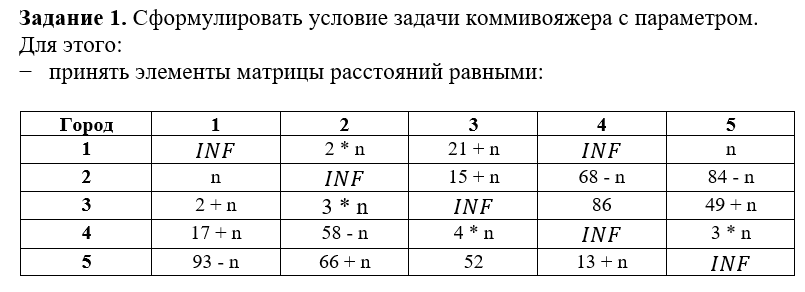




# Лабораторная работа 3. Метод ветвей и границ. Задача коммивояжера и методы её решения.

**Вариант 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.





|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| 1)1)- | 2)6 | 3)24 | 4)- | 5)3 |
| 2)3 | - | 18 | 65 | 81 |
| 3)5 | 9 | - | 86 | 52 |
| 4)20 | 55 | 12 | - | 9 |
| 5)90 | 69 | 52 | 16 | - |

Приводим матрицу по строкам:

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| - | 3 | 21 | - | 0 |
| 0 | - | 15 | 62 | 78 |
| 0 | 4 | - | 81 | 47 |
| 11 | 46 | 3 | - | 0 |
| 74 | 53 | 36 | 0 | - |

Теперь приводим по столбцам:

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| - | 0 | 18 | - | 0 |
| 0 | - | 12 | 62 | 78 |
| 0 | 1 | - | 81 | 47 |
| 11 | 43 | 0 | - | 0 |
| 74 | 50 | 33 | 0 | - |

Сумма констант приведения определяет нижнюю границу H:  
H = 3+3+5+9+16+0+3+3+0+0 = 42.  
Причем каждая строка и столбец входят в маршрут только один раз с элементом dij.  
Шаг №1.  
Определяем ребро ветвления и разобьем все множество маршрутов относительно этого ребра на два подмножества (i,j) и (i\*,j\*).

Для каждого нуля определяем константу приведения:

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| - | 01 | 18 | - | 00 |
| 012 | - | 12 | 62 | 78 |
| 01 | 1 | - | 81 | 47 |
| 11 | 43 | 012 | - | 00 |
| 74 | 50 | 33 | 095 | - |

Наибольшая сумма констант приведения равна 95 для ребра (5,4), следовательно, множество разбивается на два подмножества (5,4) и (5\*,4\*).  
Исключение ребра (5,4) проводим путем замены элемента d54 = 0 на M, после чего осуществляем очередное приведение матрицы расстояний для образовавшегося подмножества (5\*,4\*), в результате получим редуцированную матрицу.

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| - | 01 | 18 | - | 00 |
| 012 | - | 12 | 62 | 78 |
| 01 | 1 | - | 81 | 47 |
| 11 | 43 | 012 | - | 00 |
| 74 | 50 | 33 | - | - |

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| - | 01 | 18 | - | 00 |
| 012 | - | 12 | 0 | 78 |
| 01 | 1 | - | 19 | 47 |
| 11 | 43 | 012 | - | 00 |
| 41 | 19 | 0 | - | - |

Нижняя граница гамильтоновых циклов этого подмножества:  
H(5\*,4\*) = 42 + 95 = 137  
Включение ребра (5,4) проводится путем исключения всех элементов 5-ой строки и 4-го столбца, в которой элемент d45 заменяем на -, для исключения образования негамильтонова цикла.  
В результате получим другую сокращенную матрицу (4 x 4), которая подлежит операции приведения.

|  |  |  |  |
| --- | --- | --- | --- |
| 1)- | 2)01 | 3)18 | 5)00 |
| 2)012 | - | 12 | 78 |
| 3)01 | 1 | - | 47 |
| 4)11 | 43 | 012 | - |

Сумма констант приведения сокращенной матрицы:  
∑di + ∑dj = 0  
Нижняя граница подмножества (5,4) равна:  
H(5,4) = 42 + 0 = 42 ≤ 137  
Поскольку нижняя граница этого подмножества (5,4) меньше, чем подмножества (5\*,4\*), то ребро (5,4) включаем в маршрут с новой границей H = 42.  
**Шаг №2**.  
**Определяем ребро ветвления**.

|  |  |  |  |
| --- | --- | --- | --- |
| - | 01 | 18 | 047 |
| 012 | - | 12 | 78 |
| 01 | 1 | - | 47 |
| 11 | 43 | 023 | - |

max: d(1,5)=47.  
**Исключение ребра** (1,5): d15=M.

|  |  |  |  |
| --- | --- | --- | --- |
| - | 01 | 18 | - |
| 012 | - | 12 | 78 |
| 01 | 1 | - | 47 |
| 11 | 43 | 023 | - |

|  |  |  |  |
| --- | --- | --- | --- |
| - | 0 | 18 | - |
| 0 | - | 12 | 31 |
| 0 | 1 | - | 0 |
| 11 | 43 | 0 | - |

H(1\*,5\*) = 42 + 47 = 89

**Включение ребра** (1,5): d51=М.

|  |  |  |
| --- | --- | --- |
| 2)1)0 | 2)- | 3)12 |
| 3)0 | 1 | - |
| 4)11 | 43 | 0 |

|  |  |  |
| --- | --- | --- |
| 0 | - | 12 |
| 0 | 0 | - |
| 11 | 42 | 0 |

∑di + ∑dj = 1  
H(1,5) = 42 + 1 = 43 ≤ 89  
Чтобы исключить подциклы, запретим следующие переходы: (4,1),  
Ребро (1,5) включаем в маршрут с новой границей H=43.  
**Шаг №3**.  
**Определяем ребро ветвления**.

|  |  |  |
| --- | --- | --- |
| 012 | - | 12 |
| 00 | 042 | - |
| - | 42 | 054 |

max: d(4,3)=54.  
**Исключение ребра** (4,3): d43=M.

|  |  |  |
| --- | --- | --- |
| 0 | - | 12 |
| 0 | 0 | - |
| - | 42 | - |

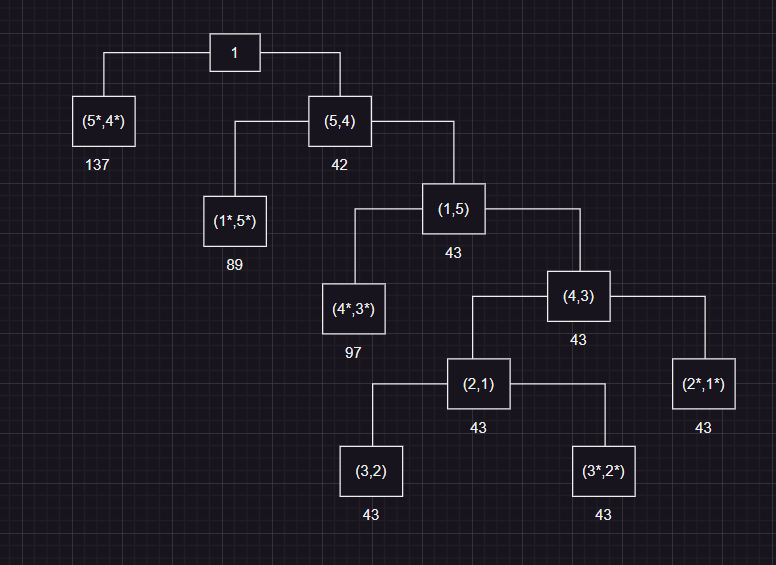
|  |  |  |
| --- | --- | --- |
| 0 | - | 0 |
| 0 | 0 | - |
| - | 0 | - |

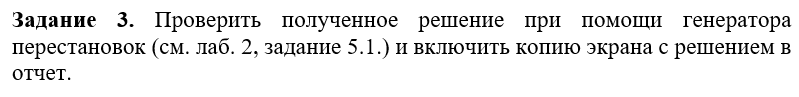
H(4\*,3\*) = 43 + 54 = 97

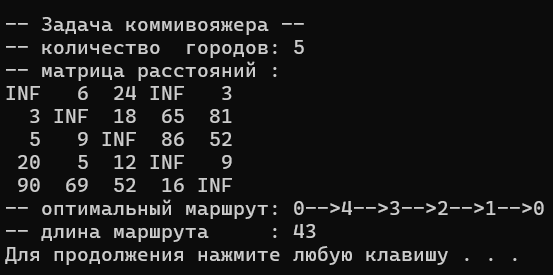
**Включение ребра** (4,3): d34=М.

|  |  |
| --- | --- |
| 2)1)0 | 2)- |
| 3)0 | 0 |

∑di + ∑dj = 0  
H(4,3) = 43 + 0 = 43 ≤ 97  
Ребро (4,3) включаем в маршрут с новой границей H=43.  
В соответствии с этой матрицей включаем в гамильтонов маршрут ребра (2,1) и (3,2).  
В результате по дереву ветвлений гамильтонов цикл образуют ребра:  
(5,4), (4,3), (3,2), (2,1), (1,5),  
Длина маршрута равна F(Mk) = 43







# Лабораторная работа 4. ДИНАМИЧЕСКОЕ ПРОГРАММИРОВАНИЕ

**Вариант 3**

***Задание 1.***

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

#include <iostream>

#include <string>

#include <random>

// Функция для генерации случайной строки заданной длины

std::string generateRandomString(int length) {

static const char alphabet[] =

"ABCDEFGHIJKLMNOPQRSTUVWXYZ"

"abcdefghijklmnopqrstuvwxyz";

std::random\_device rd;

std::mt19937 gen(rd());

std::uniform\_int\_distribution<> dis(0, sizeof(alphabet) - 2);

std::string result;

result.reserve(length);

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

result += alphabet[dis(gen)];

}

return result;

}

int main() {

// Генерация строк

std::string S1 = generateRandomString(300);

std::string S2 = generateRandomString(200);

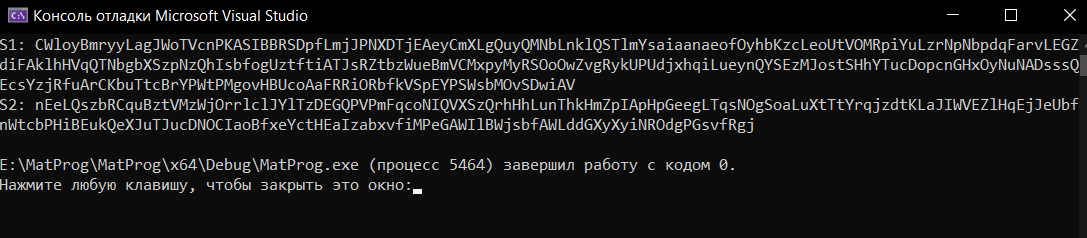
// Вывод строк

std::cout << "S1: " << S1 << std::endl;

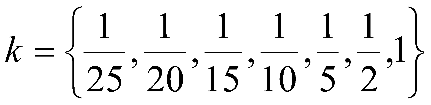
std::cout << "S2: " << S2 << std::endl;

return 0;

}

****

***Задание 2.***

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

//levenshtein.h

#pragma once

int levenshtein(

int lx, // длина слова x

const char x[], // слово длиной lx

int ly, // длина слова y

const char y[] // слово y

);

// -- дистанции Левенштeйна (рекурсия)

int levenshtein\_r(

int lx, // длина строки x

const char x[], // строка длиной lx

int ly, // длина строки y

const char y[] // строка y

);

//levenshtein.cpp

#include <iomanip>

#include <algorithm>

#include "Levenshtein.h"

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

using namespace std;

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

{

return min(min(x1, x2), x3);

}

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

{

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

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

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

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

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

{

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

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

}

return DD(lx, ly);

}

int levenshtein\_r(

int lx, const char x[],

int ly, const char y[]

)

{

int rc = 0;

if (lx == 0) rc = ly;

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

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

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

else rc = min3(

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

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

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

);

return rc;

};

//main.cpp

#include <iostream>

#include <string>

#include <random>

#include <algorithm>

#include <ctime>

#include <iomanip>

#include "Levenshtein.h"

#include <cstring>

void generate\_random\_string(char\* random\_string, int length) {

std::string alphabet = "abcdefghijklmnopqrstuvwxyzABCDEFGHIJKLMNOPQRSTUVWXYZ";

std::random\_device rd;

std::mt19937 generator(rd());

std::uniform\_int\_distribution<> distribution(0, alphabet.size() - 1);

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

random\_string[i] = alphabet[distribution(generator)];

}

random\_string[length] = '\0'; // Добавляем нулевой символ в конец строки

}

#include <cstring>

char\* getFirstNChars(char x[301], int n) {

char\* result = new char[n + 1];

strncpy\_s(result, n + 1, x, n);

result[n] = '\0';

return result;

}

// Ваши функции levenshtein и levenshtein\_r здесь...

int main() {

char x[121];

char y[81];

generate\_random\_string(x, 120);

generate\_random\_string(y, 80);

float k\_values[] = { 1.0 / 25, 1.0 / 20, 1.0 / 15, 1.0 / 10, 1.0 / 5, 1.0 / 2, 1.0 };

int num\_k\_values = sizeof(k\_values) / sizeof(float);

setlocale(LC\_ALL, "rus");

std::cout << std::endl << "-- расстояние Левенштейна -----" << std::endl;

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

for (int i = 0; i < num\_k\_values; i++) {

int len\_x = strlen(x) \* k\_values[i];

int len\_y = strlen(y) \* k\_values[i];

std::cout << getFirstNChars(x, len\_x) << '\n';

clock\_t t1 = clock();

levenshtein\_r(len\_x, getFirstNChars(x, len\_x), len\_x, getFirstNChars(y, len\_y));

clock\_t t2 = clock();

clock\_t t3 = clock();

levenshtein(len\_x, getFirstNChars(x, len\_x), len\_x, getFirstNChars(y, len\_y));

clock\_t t4 = clock();

int distance\_r = levenshtein\_r(len\_x, getFirstNChars(x, len\_x), len\_x, getFirstNChars(y, len\_y));

int distance = levenshtein(len\_x, getFirstNChars(x, len\_x), len\_x, getFirstNChars(y, len\_y));

std::cout << "Расстояние Левенштейна (рекурсивно): " << distance\_r << std::endl;

std::cout << "Расстояние Левенштейна (дин. программирование): " << distance << std::endl;

std::cout << std::right << std::setw(2) << len\_y << "/" << std::setw(2) << len\_x

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

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

}

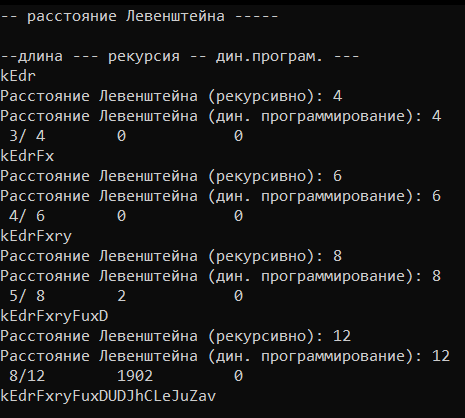
system("pause");

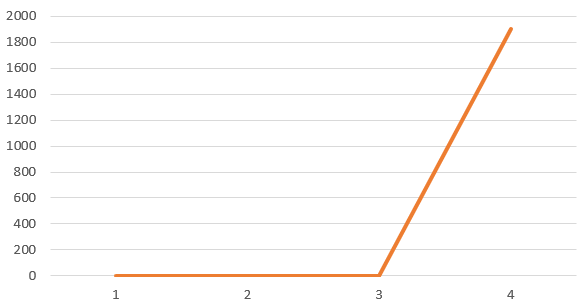
return 0;

}

***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).



****

***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).**

1. **Лом, Гомон = min(Ло, Гомон+1)**

**( Лом, Гомо+1)**

**(Ло, Гомо+1)**

1. **Ло, Гомон = min(Л, Гомон+1)**

**(Ло, Гомо+1)**

**(Л, Гомо+1)**

1. **Лом, Гомо = min(Ло, Гомо+1)**

**(Лом, Гом+1)**

**(Ло, Гом+1)**

1. **Ло, Гомо = min(Л, Гомо+1)**

**(Ло, Гом+1)**

**(Л, Гом)**

1. **Л, Гомон = min(« », Гомон) = 5**

**(Л, Гомо+1)**

**(«», Гомо) = 4**

1. **Л, Гомо = min («», Гомо) = 4**

**(Л, Гом+1)**

**(«», Гом) = 3**

1. **Лом, Гом = min(Ло, Гом+1)**

**(Лом, Го+1)**

**(Ло, Го)**

1. **Ло, Гом = min(Л, Гом+1)**

**(Ло, Го+1)**

**(Л, Го+1)**

1. **Л, Гом = min(«», Гом) = 3**

**(Л, Го+1)**

**(«», Го) = 2**

1. **Лом, Го = min(Ло, Го+1)**

**(Лом, Г+1)**

**(Ло, Г+1)**

1. **Ло, Го = min(Л, Го+1)**

**(Ло, Г+1)**

**(Л, Г)**

1. **Л, Го = min(«», Го) = 2**

**(Л, Г+1)**

**(«», Г) = 1**

1. **Лом, Г = min(Ло, Г+1)**

**(Лом, «») = 3**

**(Ло, «») = 2**

1. **Ло, Г = min(Л, Г+1)**

**(Ло, «») = 2**

**(Л, «») = 1**

1. **Л, Г = min(«», Г) = 1**

**(Л, «») = 1**

**(«», «») =0**

1. **Л, Г = min (2, 2, 1) = 1**
2. **Ло, Г = min (2, 3, 2) = 2**
3. **Лом, Г = min (3, 4, 3) = 3**
4. **Л, Го = min (3, 2, 2) = 2**
5. **Ло, Го = min (3, 3, 1) = 1**
6. **Лом, Го = min (2, 4, 3) = 2**
7. **Л, Гом = min (4, 3, 3) = 3**
8. **Ло, Гом = min (4, 2, 3) = 2**
9. **Лом, Гом = min (3, 3, 1) = 1**
10. **Л, Гомо = min (5, 4, 4) = 4**
11. **Л, Гомон = min (6, 5, 5) = 5**
12. **Ло, Гомо = min (5, 3, 4) = 3**
13. **Лом, Гомо = min (5, 4, 2) = 2**
14. **Ло, Гомон = min (6, 3, 4) = 3**
15. **Лом, Гомон = min (4, 3, 4) = 3**

***Задание 5.***

**Нечетные варианты. Выполнить сравнительный анализ времени затраченного на решение задачи о наибольшей общей подпоследовательности для двух методов решения (рекурсивное решение, динамическое программирование). Две последовательности взять в соответствии с вариантом. Построить графики зависимости времени вычисления от . Отобразить ход решения в отчете(по примеру из лекции) + код и копии экрана.**

****

// - LCS.h

// -- рекурсивное вычисление длины LCS

int lcs ( int lenx, // длина последовательности X

const char x[], // последовательность X

int leny, // длина последовательности Y

const char y[] // последовательность Y

);

// -- динамическое вычисление LCS

int lcsd( const char x[], // последовательность X

const char y[], // последовательность Y

char z[] // наибольшая общая подпоследовательность

);

// - LCS.cpp

// -- рекурсивное вычисление длины LCS

#include "stdafx.h"

#include <algorithm>

#include "LCS.h"

int lcs (int lenx, const char x[],

int leny, const char y[])

{

int rc = 0;

if (lenx > 0 && leny > 0)

{

if (x[lenx-1] == y[leny-1]) rc = 1 + lcs(lenx-1, x,leny-1, y);

else rc = std::max(lcs(lenx, x,leny-1, y), lcs(lenx-1, x,leny, y));

}

return rc; //длина LCS

}

#define LCS\_C(x1,x2) (C[(x1)\*(leny+1)+(x2)])

#define LCS\_B(x1,x2) (B[(x1)\*(leny+1)+(x2)])

#define LCS\_X(i) (x[(i)-1])

#define LCS\_Y(i) (y[(i)-1])

#define LCS\_Z(i) (z[(i)-1])

enum Dart{TOP,LEFT,LEFTTOP};

void getLCScontent( int lenx, int leny, const char x[],

const Dart\* B,

int n, int i, int j, char z[])

{

if ((i > 0 && j > 0 && n > 0 ))

{

if (LCS\_B(i,j) == LEFTTOP)

{

getLCScontent(lenx, leny,x, B, n-1, i-1, j-1, z);

LCS\_Z(n) = LCS\_X(i);

LCS\_Z(n+1) = 0;

}

else if (LCS\_B(i,j)== TOP)

getLCScontent(lenx, leny,x, B, n, i-1, j, z);

else getLCScontent(lenx, leny,x, B, n, i, j-1, z);

}

};

int lcsd(const char x[], const char y[], char z[])

{

int n;

int lenx = strlen(x), leny = strlen(x),

\*C = new int[(lenx+1)\*(leny+1)];

Dart\* B = new Dart[(lenx+1)\*(leny+1)];

memset(C,0,sizeof(int)\*(lenx+1)\*(leny+1));

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

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

if (LCS\_X(i) == LCS\_Y(j))

{LCS\_C(i,j) = LCS\_C(i-1,j-1)+1;

LCS\_B(i,j) = LEFTTOP;}

else if (LCS\_C(i-1,j) >= LCS\_C(i, j-1))

{

LCS\_C(i,j) = LCS\_C(i-1, j);

LCS\_B(i,j) = TOP;

}

else

{

LCS\_C(i,j) = LCS\_C(i, j-1);

LCS\_B(i,j) = LEFT;

}

getLCScontent(lenx, leny, x, B, LCS\_C(lenx,leny), lenx, leny, z);

return LCS\_C(lenx,leny);

}

#undef LCS\_Z

#undef LCS\_C

#undef LCS\_B

#undef LCS\_X

#undef LCS\_Y

// - main

// -- вычисления длины LCS

#include "stdafx.h"

#include <iostream>

#include "LCS.h"

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

{

setlocale(LC\_ALL, "rus");

char z[100]="";

char X[] = "ABCDFGI";

char Y[] = "EATUFI";

std::cout<<std::endl<<"-- вычисление длины LCS для X и Y(рекурсия)";

std::cout<<std::endl<<"-- последовательность X: "<< X;

std::cout<<std::endl<<"-- последовательность Y: "<< Y;

int s = lcs(sizeof(X)-1, " BXWAFRE", sizeof(Y)-1, " XCDUFR" );

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

// наибольшая общая подпоследовательность

int l = lcsd(x, y, z);

std::cout<<std::endl

<< "-- наибольшая общая подпоследовательость - LCS(динамическое"

<<"программирование)"<< std::endl;

std::cout<<std::endl<<"последовательость X: " << x;

std::cout<<std::endl<<"последовательость Y: " << x;

std::cout<<std::endl<<" LCS: " << z;

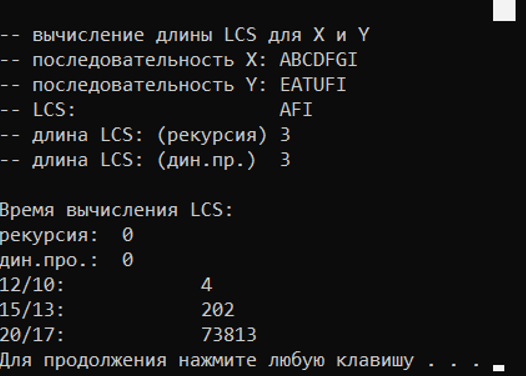
std::cout<<std::endl<<" длина LCS: " << l;

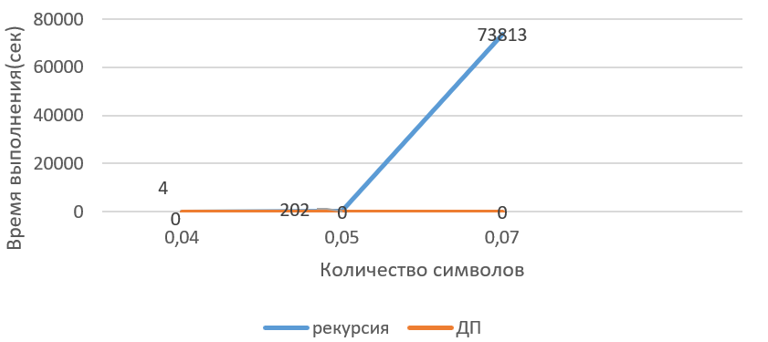
std::cout<<std::endl;

system("pause");

return 0;

}

****



|  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- |
|  |  | A -1 | B -2 | C - 3 | D - 4 | F - 5 | G - 6 | I - 7 |
|  | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| E -1 | 0 | ↑0 | ↑0 | ↑0 | ↑0 | ↑0 | ↑0 | ↑0 |
| A - 2 | 0 | ↖1 | ←1 | ←1 | ←1 | ←1 | ←1 | ←1 |
| T - 3 | 0 | ↑1 | ←1 | ←1 | ←1 | ←1 | ←1 | ↑1 |
| U - 4 | 0 | ↑1 | ←1 | ←1 | ←1 | ←1 | ←1 | ↑1 |
| F - 5 | 0 | ↑1 | ←1 | ←1 | ←1 | ↖2 | ←2 | ←2 |
| I - 6 | 0 | ↑1 | ←1 | ←1 | ←1 | ↑2 | ↑2 | ↖3 |

1. С 7,6 = c 6,5 +1
2. C 6,5 = max (c 5,5, c 6,4)
3. C 5,5 = c 4,4 +1
4. C 6,4 = max (c 6,3, c 5,4)
5. C 4,4 = max (c 3,4, c 4,3)
6. C 6,3 = max (c 6,2, c 5,3)
7. C 5,4 = max (c 5,3, c 4,4)
8. C 3,4 = max (c 3,3, c 2,4)
9. C 4,3 = max (c 4,2, c 3,3)
10. C 6,2 = max (c 6,1, c 5,2)
11. C 5,3 = max (c 5,2, c 4,3)
12. C 3,3 = max (c 3,2, c 2,3)
13. C 2,4 = max (c 2,3, c 1,4)
14. C 4,2 = max (c 4,1, c 3,2)
15. C 6,1 = max (c 6,0, c 5,1) = c 5,1
16. C 5,2 = max (5,1 4,2)
17. C 3,2 = max (3,1 2,2)
18. C 2,3 = max (2,2 1,3)
19. C 1,4 = max (1,3 0,4) = c 1,3
20. C 5,1 = max (5,0 4,1) = c 4,1
21. C 4,1 = max (4,0 3,1) = 3,1
22. C 3,1 = max (3,0 2,1) = c 2,1
23. C 2,2 = max (2,1 1,2)
24. C 1,3 = max (1,2 0,3) = c 1,2
25. C 2,1 = max (2,0 1,1) = c 1,1
26. C 1,2 = c 0,1 +1 = 1
27. C 1,1 = max (0,1 1,0) = 0
28. C 2,1 = 0
29. C 1,3 = 1
30. C 2,2 = max (1,0) = 1
31. C 3,1 = 0
32. C 4,1 = 0
33. C 5,1 = 0
34. C 1,4 = 1
35. C 2,3 = max (1,1) =1
36. C 3,2 = max (1, 0) = 1
37. C 5,2 = max (1, 0) = 1
38. C 6,1 = 0
39. C 4,2 = max (1, 0) = 1
40. C 2,4 = max (1, 1) = 1
41. C 33 = max (1, 1) = 1
42. C 53 = max (1, 1) = 1
43. C 62 = max (1, 0) = 1
44. C 43 = max (1, 1) = 1
45. C 34 = max (1, 1) = 1
46. C 54 = max (1, 1) = 1
47. C 63 = max (1, 1) = 1
48. C 44 = max (1, 1) = 1
49. C 64 = max (1, 1) = 1
50. C 55 = 1+1 = 2
51. C 65 = max (2, 1) = 2
52. C 76 = 2+1 = 3

# Лабораторная работа 5. ТРАНСПОРТНАЯ ЗАДАЧА

Вариант 3

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| 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** |  |

Задача является открытой, так как суммарное количество потребности груза превышает запасы поставщиков

Запасы: 171+116+153+162+103=705

Потребители: 146+110+134+196+98+116=850

**Опорный план:**

1. Построение начального базисного решения: метод наименьшей стоимости (минимального элемента)
2. Итеративный процесс поиска оптимального решения (метод потенциалов).

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| ПОТРЕБИТЕЛИ  ПОСТАВЩИКИ | 1 | 2 | 3 | 4 | 5 | 6 | ЗАПАСЫ |
| 1 | **15** | **5** | **9** | **6** | **14** | **4** | **171** |
| 2 | **13** | **3** | **11** | **8** | **10** | **16** | **116** |
| 3 | **4** | **8** | **14** | **11** | **5** | **14** | **153** |
| 4 | **7** | **13** | **13** | **6** | **16** | **5** | **162** |
| 5 | **6** | **14** | **12** | **3** | **13** | **7** | **103** |
| 6 | **0** | **0** | **0** | **0** | **0** | **0** | **145** |
| ПОТРЕБНОСТИ | **146** | **110** | **134** | **196** | **98** | **166** |  |

X2,2 = min(110, 116) = 110

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| ПОТРЕБИТЕЛИ  ПОСТАВЩИКИ | 1 | 2 | 3 | 4 | 5 | 6 | ЗАПАСЫ |
| 1 | **15** | **5** | **9** | **6** | **14** | **4** | **171** |
| 2 | **13** | **3|110** | **11** | **8** | **10** | **16** | **6** |
| 3 | **4** | **8** | **14** | **11** | **5** | **14** | **153** |
| 4 | **7** | **13** | **13** | **6** | **16** | **5** | **162** |
| 5 | **6** | **14** | **12** | **3** | **13** | **7** | **103** |
| 6 | **0** | **0** | **0** | **0** | **0** | **0** | **145** |
| ПОТРЕБНОСТИ | **146** | **0** | **134** | **196** | **98** | **166** |  |

X5,4

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| ПОТРЕБИТЕЛИ  ПОСТАВЩИКИ | 1 | 2 | 3 | 4 | 5 | 6 | ЗАПАСЫ |
| 1 | **15** | **5** | **9** | **6** | **14** | **4** | **171** |
| 2 | **13** | **3|110** | **11** | **8** | **10** | **16** | **6** |
| 3 | **4** | **8** | **14** | **11** | **5** | **14** | **153** |
| 4 | **7** | **13** | **13** | **6** | **16** | **5** | **162** |
| 5 | **6** | **14** | **12** | **3|103** | **13** | **7** | **0** |
| 6 | **0** | **0** | **0** | **0** | **0** | **0** | **145** |
| ПОТРЕБНОСТИ | **146** | **0** | **134** | **93** | **98** | **166** |  |

X1,6

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| ПОТРЕБИТЕЛИ  ПОСТАВЩИКИ | 1 | 2 | 3 | 4 | 5 | 6 | ЗАПАСЫ |
| 1 | **15** | **5** | **9** | **6** | **14** | **4|166** | **5** |
| 2 | **13** | **3|110** | **11** | **8** | **10** | **16** | **6** |
| 3 | **4** | **8** | **14** | **11** | **5** | **14** | **153** |
| 4 | **7** | **13** | **13** | **6** | **16** | **5** | **162** |
| 5 | **6** | **14** | **12** | **3|103** | **13** | **7** | **0** |
| 6 | **0** | **0** | **0** | **0** | **0** | **0** | **145** |
| ПОТРЕБНОСТИ | **146** | **0** | **134** | **93** | **98** | **0** |  |

X3,1

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| ПОТРЕБИТЕЛИ  ПОСТАВЩИКИ | 1 | 2 | 3 | 4 | 5 | 6 | ЗАПАСЫ |
| 1 | **15** | **5** | **9** | **6** | **14** | **4|166** | **5** |
| 2 | **13** | **3|110** | **11** | **8** | **10** | **16** | **6** |
| 3 | **4|146** | **8** | **14** | **11** | **5** | **14** | **7** |
| 4 | **7** | **13** | **13** | **6** | **16** | **5** | **162** |
| 5 | **6** | **14** | **12** | **3|103** | **13** | **7** | **0** |
| 6 | **0** | **0** | **0** | **0** | **0** | **0** | **145** |
| ПОТРЕБНОСТИ | **0** | **0** | **134** | **93** | **98** | **0** |  |

X3,5

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| ПОТРЕБИТЕЛИ  ПОСТАВЩИКИ | 1 | 2 | 3 | 4 | 5 | 6 | ЗАПАСЫ |
| 1 | **15** | **5** | **9** | **6** | **14** | **4|166** | **5** |
| 2 | **13** | **3|110** | **11** | **8** | **10** | **16** | **6** |
| 3 | **4|146** | **8** | **14** | **11** | **5|7** | **14** | **0** |
| 4 | **7** | **13** | **13** | **6** | **16** | **5** | **162** |
| 5 | **6** | **14** | **12** | **3|103** | **13** | **7** | **0** |
| 6 | **0** | **0** | **0** | **0** | **0** | **0** | **145** |
| ПОТРЕБНОСТИ | **0** | **0** | **134** | **93** | **91** | **0** |  |

X1,4

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| ПОТРЕБИТЕЛИ  ПОСТАВЩИКИ | 1 | 2 | 3 | 4 | 5 | 6 | ЗАПАСЫ |
| 1 | **15** | **5** | **9** | **6|5** | **14** | **4|166** | **0** |
| 2 | **13** | **3|110** | **11** | **8** | **10** | **16** | **6** |
| 3 | **4|146** | **8** | **14** | **11** | **5|7** | **14** | **0** |
| 4 | **7** | **13** | **13** | **6** | **16** | **5** | **162** |
| 5 | **6** | **14** | **12** | **3|103** | **13** | **7** | **0** |
| 6 | **0** | **0** | **0** | **0** | **0** | **0** | **145** |
| ПОТРЕБНОСТИ | **0** | **0** | **134** | **88** | **91** | **0** |  |

X4,4

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| ПОТРЕБИТЕЛИ  ПОСТАВЩИКИ | 1 | 2 | 3 | 4 | 5 | 6 | ЗАПАСЫ |
| 1 | **15** | **5** | **9** | **6|5** | **14** | **4|166** | **0** |
| 2 | **13** | **3|110** | **11** | **8** | **10** | **16** | **6** |
| 3 | **4|146** | **8** | **14** | **11** | **5|7** | **14** | **0** |
| 4 | **7** | **13** | **13** | **6|88** | **16** | **5** | **74** |
| 5 | **6** | **14** | **12** | **3|103** | **13** | **7** | **0** |
| 6 | **0** | **0** | **0** | **0** | **0** | **0** | **145** |
| ПОТРЕБНОСТИ | **0** | **0** | **134** | **0** | **91** | **0** |  |

X2,5

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| ПОТРЕБИТЕЛИ  ПОСТАВЩИКИ | 1 | 2 | 3 | 4 | 5 | 6 | ЗАПАСЫ |
| 1 | **15** | **5** | **9** | **6|5** | **14** | **4|166** | **0** |
| 2 | **13** | **3|110** | **11** | **8** | **10|6** | **16** | **0** |
| 3 | **4|146** | **8** | **14** | **11** | **5|7** | **14** | **0** |
| 4 | **7** | **13** | **13** | **6|88** | **16** | **5** | **74** |
| 5 | **6** | **14** | **12** | **3|103** | **13** | **7** | **0** |
| 6 | **0** | **0** | **0** | **0** | **0** | **0** | **145** |
| ПОТРЕБНОСТИ | **0** | **0** | **134** | **0** | **85** | **0** |  |

X4,3

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| ПОТРЕБИТЕЛИ  ПОСТАВЩИКИ | 1 | 2 | 3 | 4 | 5 | 6 | ЗАПАСЫ |
| 1 | **15** | **5** | **9** | **6|5** | **14** | **4|166** | **0** |
| 2 | **13** | **3|110** | **11** | **8** | **10|6** | **16** | **0** |
| 3 | **4|146** | **8** | **14** | **11** | **5|7** | **14** | **0** |
| 4 | **7** | **13** | **13|74** | **6|88** | **16** | **5** | **0** |
| 5 | **6** | **14** | **12** | **3|103** | **13** | **7** | **0** |
| 6 | **0** | **0** | **0** | **0** | **0** | **0** | **145** |
| ПОТРЕБНОСТИ | **0** | **0** | **60** | **0** | **85** | **0** |  |

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| ПОТРЕБИТЕЛИ  ПОСТАВЩИКИ | 1 | 2 | 3 | 4 | 5 | 6 | ЗАПАСЫ |
| 1 | **15** | **5** | **9** | **6|5** | **14** | **4|166** | **0** |
| 2 | **13** | **3|110** | **11** | **8** | **10|6** | **16** | **0** |
| 3 | **4|146** | **8** | **14** | **11** | **5|7** | **14** | **0** |
| 4 | **7** | **13** | **13|74** | **6|88** | **16** | **5** | **0** |
| 5 | **6** | **14** | **12** | **3|103** | **13** | **7** | **0** |
| 6 | **0** | **0** | **0** | **0** | **0** | **0** | **145** |
| ПОТРЕБНОСТИ | **0** | **0** | **60** | **0** | **85** | **0** |  |

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| ПОТРЕБИТЕЛИ  ПОСТАВЩИКИ | 1 | 2 | 3 | 4 | 5 | 6 | ЗАПАСЫ |
| 1 | **15** | **5** | **9** | **6|5** | **14** | **4|166** | **0** |
| 2 | **13** | **3|110** | **11** | **8** | **10|6** | **16** | **0** |
| 3 | **4|146** | **8** | **14** | **11** | **5|7** | **14** | **0** |
| 4 | **7** | **13** | **13|74** | **6|88** | **16** | **5** | **0** |
| 5 | **6** | **14** | **12** | **3|103** | **13** | **7** | **0** |
| 6 | **0** | **0** | **0|60** | **0** | **0|85** | **0** | **145** |
| ПОТРЕБНОСТИ | **0** | **0** | **60** | **0** | **85** | **0** |  |

Должно быть m + n – 1 = 6 + 6 – 1 = 11 переменных.

Заполнено 11 переменных.

Значение функции цели: 4\*146 + 3\*110 + 13\*74 + 6\*5 + 6\*88 + 3\*103 + 10\*6 + 5\*7 + 4\*166 = 3502.

Метод потенциалов:

u1+v4=6; 0+v4=6; v4=6  
u4+v4=6; 6+u4=6; u4=0  
u4+v3=13; 0+v3=13; v3=13  
u6+v3=0; 13+u6=0; u6=-13  
u6+v5=0; -13+v5=0; v5=13  
u2+v5=10; 13+u2=10; u2=-3  
u2+v2=3; -3+v2=3; v2=6  
u3+v5=5; 13+u3=5; u3=-8  
u3+v1=4; -8+v1=4; v1=12  
u5+v4=3; 6+u5=3; u5=-3  
u1 + v6 = 4; 0 + v6 = 4; v6 = 4

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
|  | v1=12 | v2=6 | v3=13 | v4=6 | v5=13 | v6=4 |
| u1=0 | 15 | 5 | 9 | 6[5] | 14 | 4[166] |
| u2=-3 | 13 | 3[110] | 11 | 8 | 10[6] | 16 |
| u3=-8 | 4[146] | 8 | 14 | 11 | 5[7] | 14 |
| u4=0 | 7 | 13 | 13[74] | 6[88] | 16 | 5 |
| u5=-3 | 6 | 14 | 12 | 3[103] | 13 | 7 |
| u6=-13 | 0 | 0 | 0[60] | 0 | 0[85] | 0 |

Опорный план не является оптимальным, так как существуют оценки свободных клеток, для которых

ui+vj>cij  
(1;2): 0+6 > 5; X12 = 0 + 6 - 5 = 1 > 0  
(1;3): 0 + 13 > 9; X13 = 0 + 13 - 9 = 4 > 0  
(4;1): 0 + 12 > 7; X41 = 0 + 12 - 7 = 5 > 0  
(5;1): -3 + 12 > 6; X51 = -3 + 12 - 6 = 3 > 0  
max(1,4,5,3) = 5

Выбираем максимальную оценку свободной клетки (4;1): 7  
Для этого в перспективную клетку (4;1) поставим знак «+», а в остальных вершинах многоугольника чередующиеся знаки «-», «+», «-».

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| ПОТРЕБИТЕЛИ  ПОСТАВЩИКИ | 1 | 2 | 3 | 4 | 5 | 6 | Запасы |
| 1 | 15 | 5 | 9 | 6[5] | 14 | 4[166] | 171 |
| 2 | 13 | 3[110] | 11 | 8 | 10[6] | 16 | 116 |
| 3 | 4[146][-] | 8 | 14 | 11 | 5[7][+] | 14 | 153 |
| 4 | 7[+] | 13 | 13[74][-] | 6[88] | 16 | 5 | 162 |
| 5 | 6 | 14 | 12 | 3[103] | 13 | 7 | 103 |
| 6 | 0 | 0 | 0[60][+] | 0 | 0[85]-] | 0 | 145 |
| ПОТРЕБНОСТИ | 146 | 110 | 134 | 196 | 98 | 166 |  |

Цикл приведен в таблице (4,1 → 4,3 → 6,3 → 6,5 → 3,5 → 3,1).  
Из грузов хij стоящих в минусовых клетках, выбираем наименьшее, т.е. у = min (4, 3) = 74. Прибавляем 74 к объемам грузов, стоящих в плюсовых клетках и вычитаем 74 из Хij, стоящих в минусовых клетках. В результате получим новый опорный план.

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| ПОТРЕБИТЕЛИ  ПОСТАВЩИКИ | 1 | 2 | 3 | 4 | 5 | 6 | ЗАПАСЫ |
| 1 | **15** | **5** | **9** | **6|5** | **14** | **4|166** | **171** |
| 2 | **13** | **3 |110** | **11** | **8** | **10|6** | **16** | **116** |
| 3 | **4|72** | **8** | **14** | **11** | **5|81** | **14** | **153** |
| 4 | **7|74** | **13** | **13** | **6|88** | **16** | **5** | **162** |
| 5 | **6** | **14** | **12** | **3|103** | **13** | **7** | **103** |
| 6 | **0** | **0** | **0|134** | **0** | **0|11** | **0** | **145** |
| ПОТРЕБНОСТИ | **0** | **0** | **60** | **0** | **85** | **0** |  |

Заново находим потенциалы:

u1+v4=6; 0+v4=6; v4=6  
u4+v4=6; 6+u4=6; u4=0  
u4+v1=7; 0+v1=7; v1=7  
u3+v1=4; 7+u3=4; u3=-3  
u3+v5=5; -3+v5=5; v5=8  
u2+v5=10; 8+u2=10; u2=2  
u2+v2=3; 2+v2=3; v2=1  
u6+v5=0; 8+u6=0; u6=-8  
u6+v3=0; -8+v3=0; v3=8  
u5+v4=3; 6+u5=3; u5=-3  
u1 + v6 = 4; 0 + v6 = 4; v6 = 4

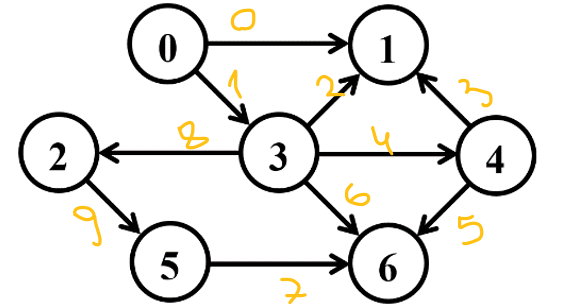
|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
|  | v1=7 | v2=1 | v3=8 | v4=6 | v5=8 | v6=4 |
| u1=0 | 15 | 5 | 9 | 6[5] | 14 | 4[166] |
| u2=2 | 13 | 3[110] | 11 | 8 | 10[6] | 16 |
| u3=-3 | 4[72] | 8 | 14 | 11 | 5[81] | 14 |
| u4=0 | 7[74] | 13 | 13 | 6[88] | 16 | 5 |
| u5=-3 | 6 | 14 | 12 | 3[103] | 13 | 7 |
| u6=-8 | 0 | 0 | 0[134] | 0 | 0[11] | 0 |

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

F(x) = 4\*72+7\*74+3\*110+6\*5+6\*88+3\*103+10\*6+5\*81+4\*166 = 3132

# Лабораторная работа 6. АЛГОРИТМЫ НА ГРАФАХ

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



Матрица инцидентности:

|  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
|  | 0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 |
| 0 | 1 | 1 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| 1 | -1 | 0 | -1 | -1 | 0 | 0 | 0 | 0 | 0 | 0 |
| 2 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | -1 | 1 |
| 3 | 0 | -1 | 2 | 0 | 1 | 0 | 1 | 0 | 1 | 0 |
| 4 | 0 | 0 | 0 | 1 | -1 | 1 | 0 | 0 | 0 | 0 |
| 5 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 1 | 0 | -1 |
| 6 | 0 | 0 | 0 | 0 | 0 | -1 | -1 | -1 | 0 | 0 |

Строки = Кол-во вершин

Столбцы = Кол-во рёбер

0 – вершина не инцидентна ребру

1 – дуга выходит из вершины

-1 – дуга входит в вершину

Матрица смежности:

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

Список смежных вершин:

0 → {1,3}

1 → {}

2 → {5}

3 → {1,2,4,6}

4 → {1,6}

5 → {6}

6 → {}

***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.

**Поиск в ширину**



Шаг 1

Посещенные вершины: {}

Очередь: {1, 3}

Текущая вершина: 0

Шаг 2

Посещенные вершины: {0}

Очередь: {3}

Текущая вершина: 1

Шаг 3

Посещенные вершины: {0, 1}

Очередь: {2,4,6}

Текущая вершина: 3

Шаг 4

Посещенные вершины: {0, 1, 3}

Очередь: {4,6,5}

Текущая вершина: 2

Шаг 5

Посещенные вершины: {0, 1, 3, 2}

Очередь: {6, 5}

Текущая вершина: 4

Шаг 6

Посещенные вершины: {0, 1, 3, 2, 4}

Очередь: {5}

Текущая вершина: 6

Шаг 7

Посещенные вершины: {0, 1, 3, 2, 4, 6}

Очередь: {}

Текущая вершина: 5

Шаг 8

Посещенные вершины: {0, 1, 3, 2, 4, 6, 5}

Очередь: {}

Текущая вершина: -

**Поиск в глубину**

Шаг 1

Посещенные вершины: {}

Стек: {1,3}

Текущая вершина: 0

Шаг 2

Посещенные вершины: {0}

Стек: {3}

Текущая вершина: 1

Шаг 3

Посещенные вершины: {0,1}

Стек: {2,4,6}

Текущая вершина: 3

Шаг 4

Посещенные вершины: {0,1,3}

Стек: {5,4,6}

Текущая вершина: 2

Шаг 5

Посещенные вершины: {0,1,3,2}

Стек: {4,6}

Текущая вершина: 5

Шаг 5

Посещенные вершины: {0,1,3,2,5}

Стек: {6}

Текущая вершина: 4

Шаг 6

Посещенные вершины: {0,1,3,2,5,4}

Стек: {}

Текущая вершина: 6

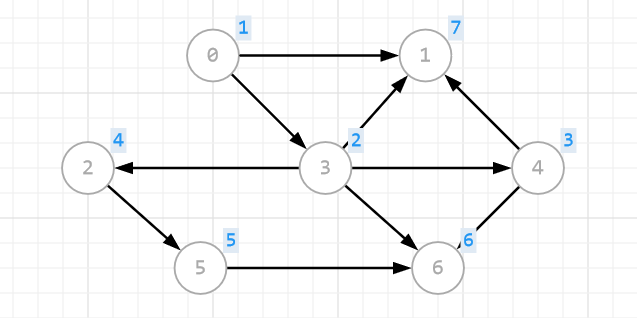
Шаг 7

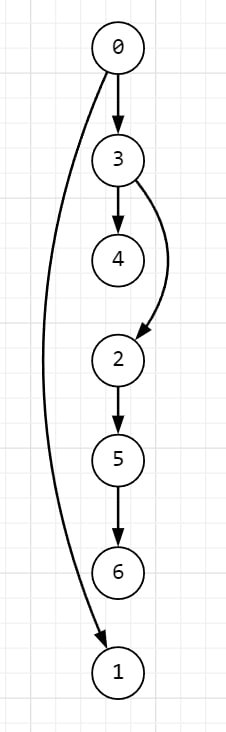
Посещенные вершины: {0,1,3,2,5,4,6}

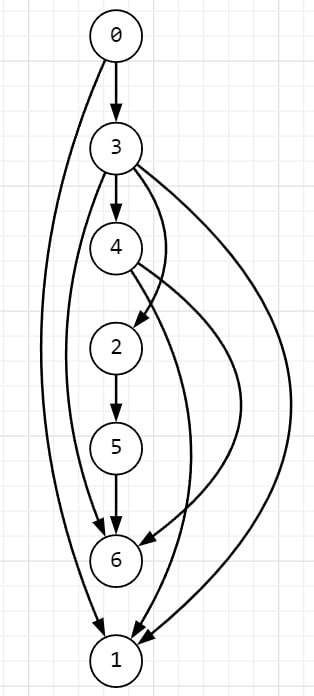
Стек: {}

Текущая вершина: -

**Топологическая сортировка**

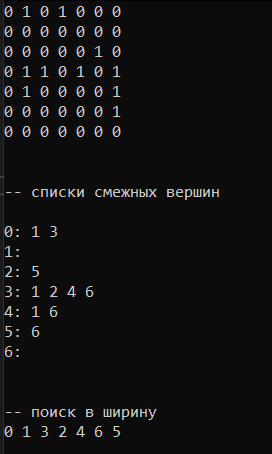




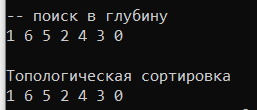


Результирующая последовательность: 0 3 4 2 5 6 1

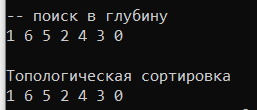
***Задание 3.*** Осуществить программную реализацию алгоритмов на C++. Разработать структуры **AMatrix** и **АList**  для представления ориентированного графа матричным и списковым способом. Разработать функции преобразования из одного способа представления в другой. Разработать функцию **BFS** обхода вершин графа, используя метод поиска в ширину. Продемонстрировать работу функции. Копии экрана вставить в отчет.



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



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



***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;

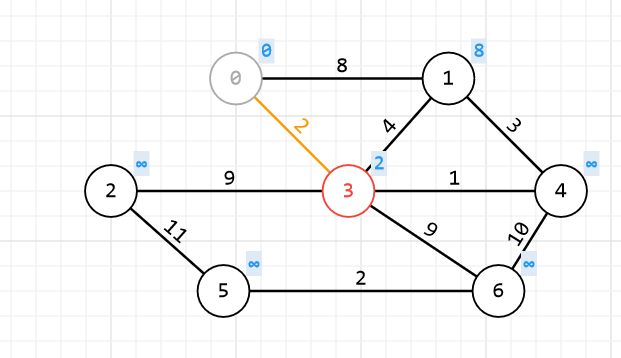
W(e4,3)=4; W(e3,4)=1;

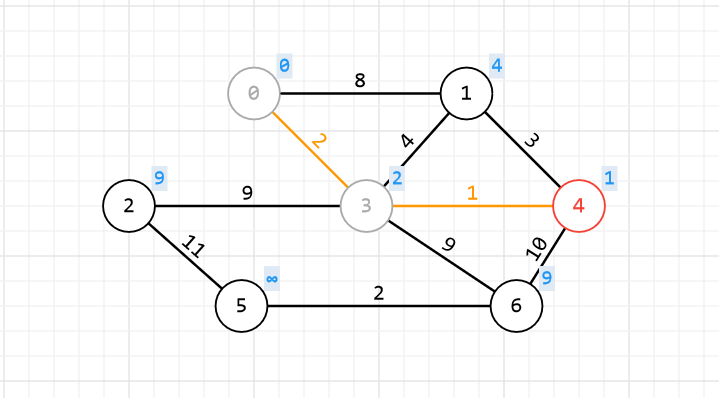
W(e4,6)=10; W(e6,4)=2;

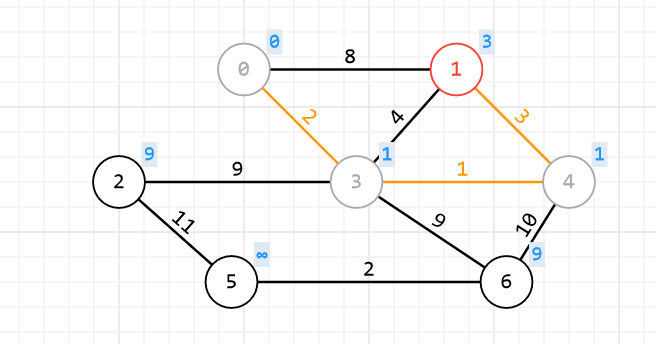
W(e5,6)=2; W(e6,5)=6;

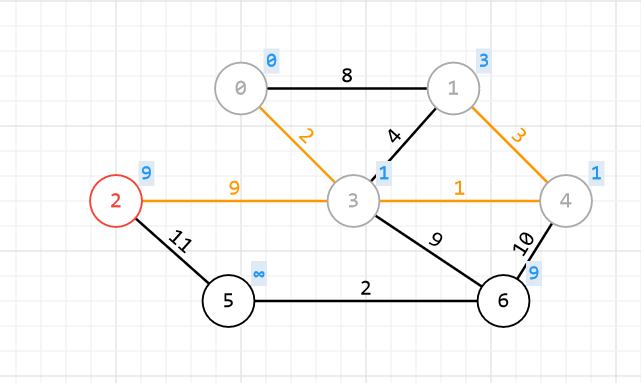
W(e5,3)=3; W(e3,5)=6;

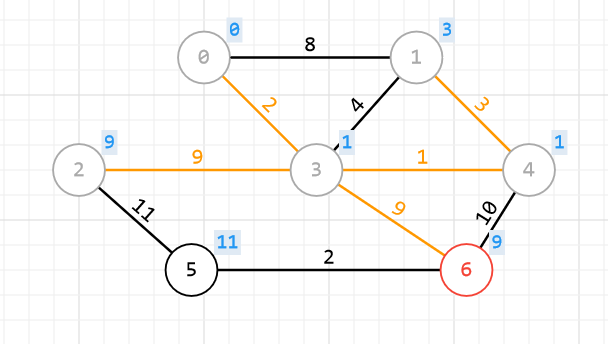
W(e6,3)=7; W(e3,6)=9;

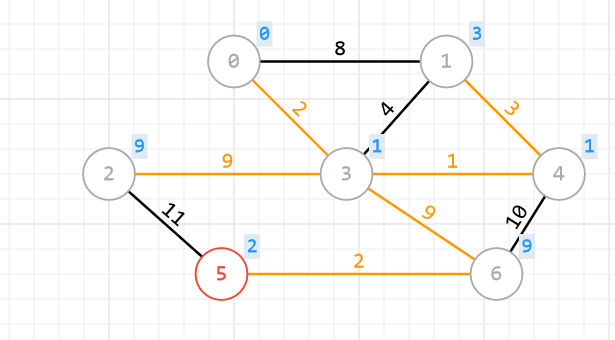


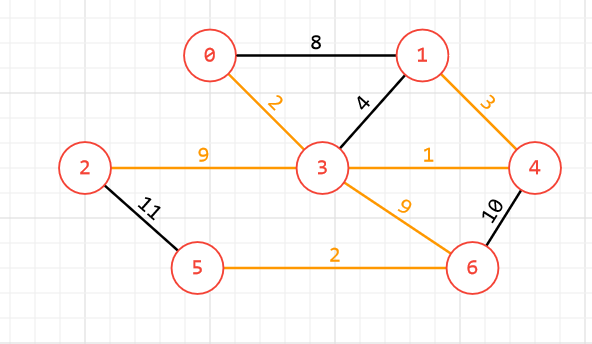












***Задание 7.*** По графу, соответствующему варианту составить минимальное остовное дерево по алгоритму Крускала. Шаги построения отразить в отчете.

W(e3,4)=1;

W(e0,3)=2;

W(e5,6)=2;

W(e4,1)=3;

W(e3,1)=4;

W(e0,1)=8;

W(e3,2)=9;

W(e3,6)=9;

W(e4,6)=10;

W(e2,5)=11;

Шаги решения:

1)3,4

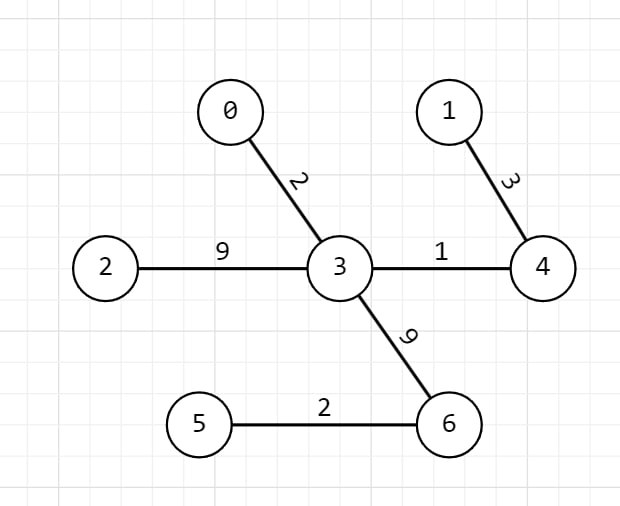
2)0,3

3)5,6

4)4,1

5)3,2

6)3,6



# Лабораторная работа 7. Сетевые модели

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

**Задание для выполнения:**

Лабораторная работа базируется на исследовании различных тематик в проектировании программных продуктов, составлении сетевых графиков для разных тем, нахождении критических путей в составленных графиках. Каждый проект принять условным или обобщенным, но допустимо делать упор на конкретные примеры.

|  |  |  |
| --- | --- | --- |
| 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 |

**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.

Вариант 3. «Создание банковского приложения». Время выполнения всех задач – 65 дней

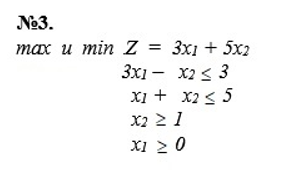
|  |  |  |  |
| --- | --- | --- | --- |
| Код операции | Наименование операции | Предшествующие операции | t |
| I. АНАЛИЗ | | | |
| Z1 | Системный анализ |  | 10 |
| Z2 | Анализ требований | Z1 | 5 |
| II. ПРОЕКТИРОВАНИЕ | | | |
| Z3 | Проектирование базы данных | Z2 | 8 |
| Z4 | Проектирование архитектуры приложения | Z2 | 7 |
| Z5 | Проектирование пользовательского интерфейса | Z2 | 6 |
| III. КОДИРОВАНИЕ | | | |
| Z6 | Разработка серверной части приложения | Z3, Z4 | 12 |
| Z7 | Разработка клиентской части приложения | Z5 | 10 |
| Z8 | Реализация функций работы с банковскими данными | Z6 | 9 |
| IV. ТЕСТИРОВАНИЕ | | | |
| Z9 | Функциональное тестирование | Z6, Z7, Z8, Z15, Z16, Z17 | 8 |
| Z10 | Структурное тестирование | Z6, Z7, Z8, Z15, Z16, Z17 | 8 |
| V. ВНЕДРЕНИЕ | | | |
| Z11 | Разработка документации | Z6, Z7, Z8, Z9 | 6 |
| Z12 | Установка и конфигурирование приложения на сервере | Z11 | 5 |
| Z13 | Тестирование после внедрения | Z11, Z12 | 6 |
| Z14 | Обучение пользователей | Z11, Z12, Z13 | 4 |
| VI. ДОПОЛНИТЕЛЬНЫЕ РАБОТЫ | | | |
| Z15 | Создание дизайна и брендинга | Z5 | 4 |
| Z16 | Обеспечение безопасности приложения | Z6 | 7 |
| Z17 | Резервное копирование и восстановление данных | Z6 | 4 |

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

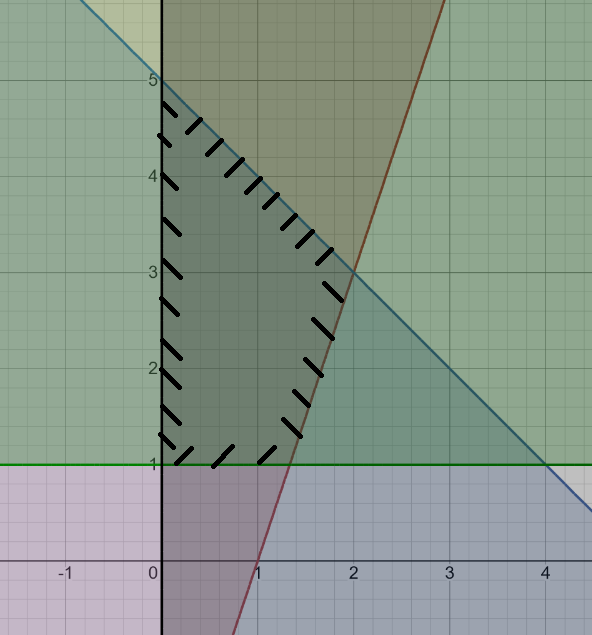
|  |  |  |  |
| --- | --- | --- | --- |
| Начальное событие | Код операции | Предшествующие операции | Конечное событие |
| 0 | Z1 |  | 1 |
| 2 | Z2 | Z1, | 3 |
| 4 | Z3 | Z2 | 5 |
| 6 | Z4 | Z2 | 7 |
| 8 | Z5 | Z2 | 9 |
| 10 | Z6 | Z3, Z4 | 11 |
| 12 | Z7 | Z5 | 13 |
| 14 | Z8 | Z6 | 15 |
| 16 | Z9 | Z6, Z7, Z8 | 17 |
| 18 | Z10 | Z6, Z7, Z8 | 19 |
| 20 | Z11 | Z6, Z7, Z8, Z9 | 21 |
| 22 | Z12 | Z11 | 23 |
| 24 | Z13 | Z12 | 25 |
| 26 | Z14 | Z11, Z12 | 27 |
| 28 | Z15 | Z5 | 29 |
| 30 | Z16 | Z6 | 31 |
| 32 | Z17 | Z6 | 33 |

# Лабораторная работа 8. Графический метод решения оптимизационных задач

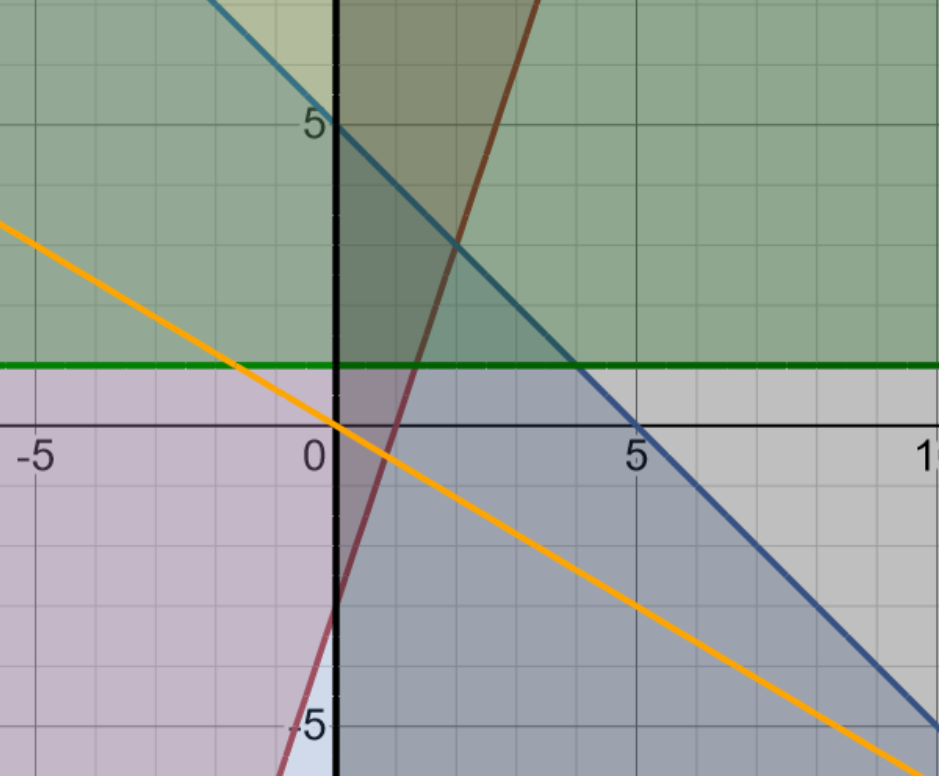
**Цель работы:** Освоить решение задач графическим методом.



1. Строим область допустимых решений, т.е. решаем графически систему неравенств. Для этого строим каждую прямую и определяем полуплоскости, заданные неравенствами (полуплоскости обозначаем штрихом).



1. Строим прямую, соответствующую задаче, или целевой функции, приравненной к нулю. Область допустимых решений может представлять бесконечное множество. Поэтому ищем max и min в области ограничений, если это возможно.



Нет max и min.