Белорусский государственный технологический университет

Факультет информационных технологий

Кафедра программной инженерии

Отчеты

По дисциплине «Математическое программирование»

Выполнил:

Студент 2 курса 9 группы

Павлович Ян Андреевич

Преподаватель: асс. Ромыш А.С.

2025, Минск

Оглавление

Лабораторная работа №1. Вспомогательные функции

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

Ход работы

Задание 1. Разработайте три функции (start, dget и iget), используя следующие спецификации.

Реализация функций представлена в листингах 1, 2.

```
#include "stdafx.h"
#include "Auxil.h"
#include <ctime>
namespace auxil
{
    void start()                            // старт  генератора сл. чисел
    {
        srand((unsigned)time(NULL));
    };
    double dget(double rmin, double rmax) // получить случайное число
    {
        return ((double)rand() / (double)RAND_MAX) * (rmax - rmin) + rmin;
    };
    int iget(int rmin, int rmax)          // получить случайное число

    {
        return (int)dget((double)rmin, (double)rmax);
    };
}
```

Листинг 1. Код файла Auxil.cpp

```
#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 2. Код файла Auxil.h

Задание 2

1. Реализовать пример 2.
2. Для проверки работоспособности разработанных функций и приобретения навыков замера продолжительности процесса вычисления реализуйте программу, приведенную в листинге 3.

```cpp
#include "stdafx.h"
#include <tchar.h>
#include "Auxil.h"                              // вспомогательные функции
#include <iostream>
#include <ctime>
#include <locale>

#define  CYCLE  1000000                         // количество циклов

int _tmain(int argc, _TCHAR* 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, 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;
}
```

Листинг 3. Выполнение примера

Результат работы данного кода приведен на рисунке 1.

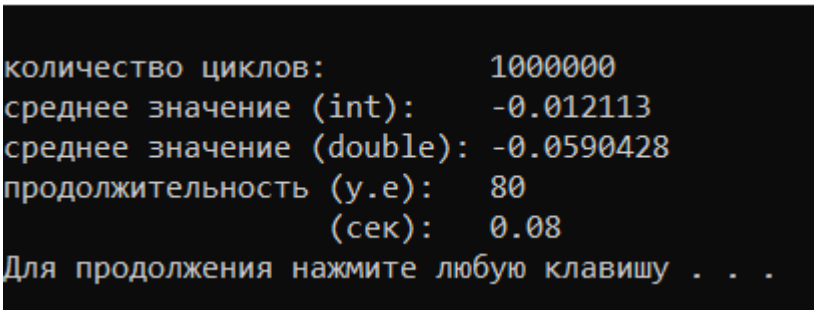


Рисунок 1. Выполнение примера

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.

Код генерации числа Фибоначчи представлен в листинге 4, а его результат на рисунке 2.

```
#include "stdafx.h"
#include <tchar.h>
#include <iostream>
#include <ctime>
#include "Auxil.h"
using namespace  std;
#define N 100

clock_t SS[N];

unsigned long long getFibonacci(int n) {
    if (n <= 1) return n;
    return getFibonacci(n - 1) + getFibonacci(n - 2);
}


int _tmain(int argc, _TCHAR* argv[])
{
    setlocale(LC_ALL, "rus");

    auxil::start();

    for (int n = 10; n <= 45; n++)
    {
        SS[n] = clock();
        unsigned long long F = getFibonacci(n);
        SS[n] = -(SS[n] - clock());
        cout << n << "-ое число Фибоначчи = " << F << " за время: " << SS[n] <<
endl;
    }

    system("pause");
    return 0;
}
```

Листинг 4. Код генерации чисел Фибоначчи

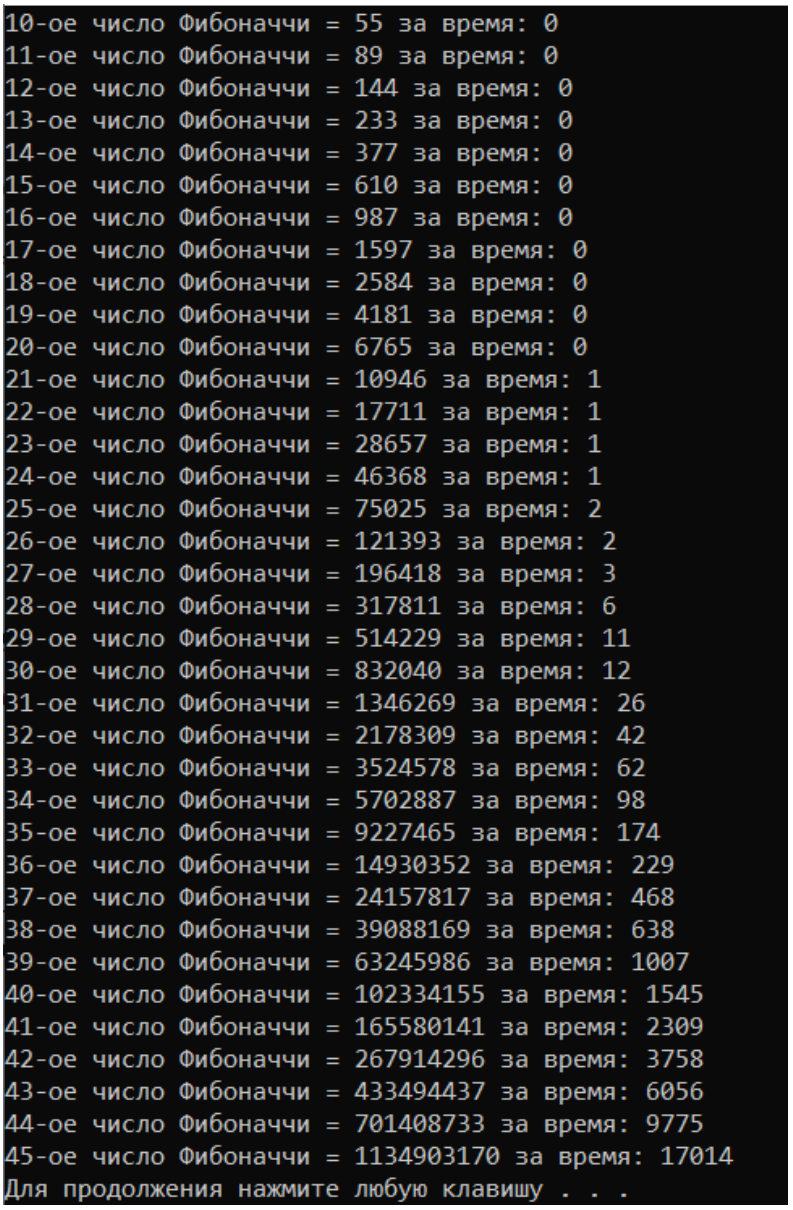


Рисунок 2. Выполнение генерации чисел Фибоначчи

Результаты экспериментов, а также график зависимости продолжительности процесса вычисления от количества циклов представлен на рисунке 3.

Количество операций	Время(у.е.)	Время(сек.)
1000000	74	0,074
2000000	134	0,134
3000000	209	0,209
4000000	268	0,268
5000000	316	0,316
6000000	360	0,36
7000000	467	0,467
8000000	556	0,556
9000000	604	0,604
10000000	633	0,633

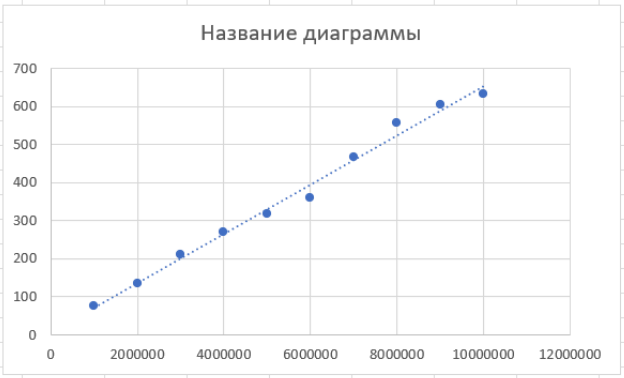


Рисунок 3. Результаты экспериментов и график

Таким образом, зависимость продолжительности процесса вычисления от количества циклов является линейной.

Результаты подсчета времени вычисления n-го числа Фибоначчи, а также график зависимости представлен на рисунке 4.

n-ое число Фибоначчи	Время(у.е.)
10	0
11	0
12	0
13	0
14	0
15	0
16	0
17	0
18	0
19	0
20	0
21	1
22	1
23	1
24	1
25	2
26	2
27	3
28	6
29	11
30	12
31	26
32	42
33	62
34	98
35	174
36	229
37	468
38	638
39	1007
40	1545
41	2309
42	3758
43	6056
44	9775
45	17014

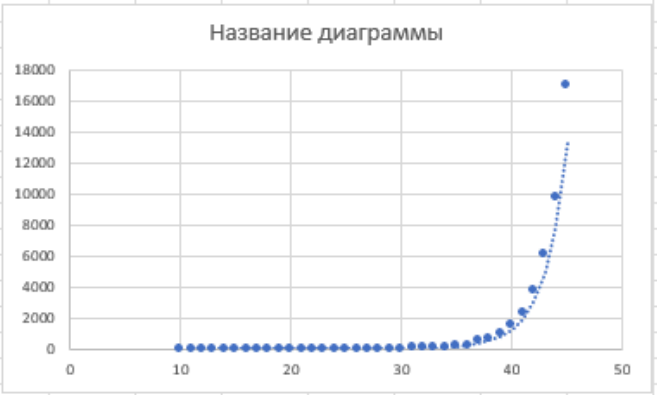


Рисунок 4. Результаты экспериментов и график

Зависимость рекурсивного нахождения числа Фибоначчи от затрачиваемого времени является экспоненциальной.

Вывод: В результате выполнения лабораторной работы №1 были приобретены навыки составления и отладки программ с использованием пользовательских функций для замера продолжительности процесса вычисления.

Лабораторная работа 2. Комбинаторные алгоритмы решения оптимизационных задач

ЦЕЛЬ РАБОТЫ: приобрести навыки разработки генераторов подмножеств, перестановок, сочетаний и размещений на C++; 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. Реализация функций показана в листинге 1, 2.

Листинг 1 – Прототипы функций Combi.h

```
// 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
   };

   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 перестановки 0,... count()-1

    unsigned __int64 count() const;  // вычислить общее кол. перестановок
  };

};
```

Листинг 2 – Код файла Combi.cpp

```
// Combi.cpp
//#include "stdafx.h"
#include "Combi.h"
#include <algorithm>
#define NINF  ((short)0x8000)
namespace combi
{
  unsigned __int64 fact(unsigned __int64 x) { return(x == 0) ? 1 : (x * fact(x - 1)); };

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

  unsigned __int64 permutation::count() const { return fact(this->n); };

  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);
  };

  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 xcombination::count() const
{
    return (this->n >= this->m) ?
        fact(this->n) / (fact(this->n - this->m) * fact(this->m)) : 0;
};

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;
  };

};
```

Задание 2. Разобрать и разработать генератор сочетаний.
Задание под номером 2 показано в листинге 1,2.
Задание 3. Разобрать и разработать генератор перестановок.
Задание под номером **3** показано в листинге 1,2.
Задание 4. Разобрать и разработать генератор размещений.
Задание под номером **4** показано в листинге 1,2.
Задание 5. Разобрать и разработать генератор размещений.
Решить задачу и результат занести в отчет.
Задача коммивояжера (расстояния сгенерировать случайным образом: 10 городов, расстояния 10 – 300 км, 3 расстояния между городами задать бесконечными);
Задание под номером 5 показано в листинге 3.
Листинг 3 – Код файла Algorithms.cpp

```
// Main
//#include "stdafx.h"
#include <iostream>
#include "Combi.h"
#include <time.h>
#include <iomanip>
#include "Knapsack.h"
#include "Boat.h"
#include "Auxil.h"
#include "Salesman.h"
```

```
// Определите макросы для включения/выключения заданий
#define TASK_SUBSET_GENERATOR 0
#define TASK_KNAPSACK 0
#define TASK_COMBINATION_GENERATOR 0
#define SHIP_LOAD 0
#define PERMUTATION 0
#define SALESMAN 1
#define SALESMAN_TIME 1

#define NN (sizeof(c)/sizeof(int))
#define MM 3
#define SPACE(n) std::setw(n)<<" "
#define N 12

int main()
{
   setlocale(LC_ALL, "rus");

#if TASK_SUBSET_GENERATOR
   // Генератор множества подмножеств
   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();                      // следующее подмножество
   };
   std::cout << std::endl << "всего: " << s1.count() << std::endl;
#endif

#if TASK_KNAPSACK
   // Задача о рюкзаке
   int V = 600,              // вместимость рюкзака
      v[] = { 25, 56, 67, 40, 20, 27, 37, 33, 33, 44, 53, 12,
            60, 75, 12, 55, 54, 42, 43, 14, 30, 37, 31, 12 },
      c[] = { 15, 26, 27, 43, 16, 26, 42, 22, 34, 12, 33, 30,
            12, 45, 60, 41, 33, 11, 14, 12, 25, 41, 30, 40 };
   short m[NN];
   int maxcc = 0;
   clock_t t1, t2;
   std::cout << std::endl << "-------- Задача о рюкзаке --------- ";
   std::cout << std::endl << "- вместимость рюкзака  : " << V;
   std::cout << std::endl << "-- количество ------ продолжительность -- ";
   std::cout << std::endl << "    предметов         вычисления  ";
   for (int i = 14; i <= NN; i++)
   {
      t1 = clock();
      maxcc = knapsack_s(V, i, v, c, m);
      t2 = clock();
      std::cout << std::endl << "       " << std::setw(2) << i
         << "              " << std::setw(5) << (t2 - t1);
   }
```

```
   std::cout << std::endl << std::endl;
#endif

#if TASK_COMBINATION_GENERATOR
   // Генератор сочетаний
   char  AAComb[][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(AAComb) / 2; i++)
      std::cout << AAComb[i] << ((i < sizeof(AAComb) / 2 - 1) ? ", " : " ");

   std::cout << "}";
   std::cout << std::endl << "Генерация сочетаний ";
   combi::xcombination xc(sizeof(AAComb) / 2, 3);
   std::cout << "из " << xc.n << " по " << xc.m;
   int  nComb = xc.getfirst();
   while (nComb >= 0)
   {

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

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


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

      std::cout << "}";

      nComb = xc.getnext();
   };
   std::cout << std::endl << "всего: " << xc.count() << std::endl;
#endif

#if SHIP_LOAD
   int V = 1000,
      v[] = { 250, 560, 670, 400, 200, 270, 370, 330, 330, 440, 530, 120,
            200, 270, 370, 330, 330, 440, 700, 120, 550, 540, 420, 170,
            600, 700, 120, 550, 540, 420, 430, 140, 300, 370, 310, 120 };
   int c[] = { 15,26,  27,  43,  16,  26,  42,  22,  34,  12,  33,  30,
            42,22,  34,  43,  16,  26,  14,  12,  25,  41,  17,  28,
            12,45,  60,  41,  33,  11,  14,  12,  25,  41,  30,  40 };
   short r[MM];
   int maxcc = 0;
   clock_t t1, t2;
   std::cout << std::endl << "-- Задача об оптимальной загрузке судна -- ";
   std::cout << std::endl << "-  ограничение по весу    : " << V;
   std::cout << std::endl << "-  количество мест        : " << MM;
   std::cout << std::endl << "-- количество ------ продолжительность -- ";
   std::cout << std::endl << "   контейнеров       вычисления  ";
   for (int i = 24; i <= NN; i++)
   {
      t1 = clock();
      int maxcc = boat(V, MM, i, v, c, r);
      t2 = clock();
      std::cout << std::endl << SPACE(7) << std::setw(2) << i
         << SPACE(15) << std::setw(5) << (t2 - t1);
   }

   std::cout << std::endl << std::endl;
#endif

#if PERMUTATION
   char  AAPermut[][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(AAPermut) / 2; i++)

      std::cout << AAPermut[i] << ((i < sizeof(AAPermut) / 2 - 1) ? ", " : " ");
   std::cout << "}";
   std::cout << std::endl << "Генерация перестановок ";
   combi::permutation p(sizeof(AAPermut) / 2);
   __int64  nPermut = p.getfirst();
   while (nPermut >= 0)
   {
      std::cout << std::endl << std::setw(4) << p.np << ": { ";

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

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

      std::cout << "}";

      nPermut = p.getnext();
   };
   std::cout << std::endl << "всего: " << p.count() << std::endl;
#endif

#if SALESMAN
   int d[N][N] = { //0   1    2    3     4
            {  0,  45, INF,  25,   50},    //  0
            { 45,   0,  55,  20,  100},    //  1
            { 70,  20,   0,  10,   30},    //  2
            { 80,  10,  40,   0,   10},    //  3
            { 30,  50,  20,  10,    0} };   //  4
   int r[N];                  // результат
   int s = salesman(
      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;
#endif

#if SALESMAN_TIME
   int dSalesmanTime[N * N + 1], rSalesmanTime[N];
   auxil::start();
   for (int i = 0; i <= N * N; i++) {
      dSalesmanTime[i] = auxil::iget(10, 100);
   }
   std::cout << std::endl << "-- Задача коммивояжера -- ";
```

```
    std::cout << std::endl << "-- количество ------ продолжительность -- ";
    std::cout << std::endl << "      городов          вычисления  ";
    clock_t t1, t2;
    for (int i = 7; i <= N; i++)
    {
        t1 = clock();
        salesman(i, (int*)dSalesmanTime, rSalesmanTime);
        t2 = clock();
        std::cout << std::endl << SPACE(7) << std::setw(2) << i
            << SPACE(15) << std::setw(5) << (t2 - t1);
    }
    std::cout << std::endl;
#endif

    system("pause");
    return 0;
}
```

Результат работы программы представлен на рисунке 1.1.

```
 - Генератор множества всех подмножеств -
Исходное множество: { A, B, C, D }
Генерация всех подмножеств
{ }
{ A }
{ B }
{ A, B }
{ C }
{ A, C }
{ B, C }
{ A, B, C }
{ D }
{ A, D }
{ B, D }
{ A, B, D }
{ C, D }
{ A, C, D }
{ B, C, D }
{ A, B, C, D }
всего: 16
```

```
 --- Генератор сочетаний ---
Исходное множество: { A, B, C, D, E }
Генерация сочетаний из 5 по 3
0: { A, B, C }
1: { A, B, D }
2: { A, C, D }
3: { B, C, D }
4: { A, B, E }
5: { A, C, E }
6: { B, C, E }
7: { A, D, E }
8: { B, D, E }
9: { C, D, E }
всего: 10
```

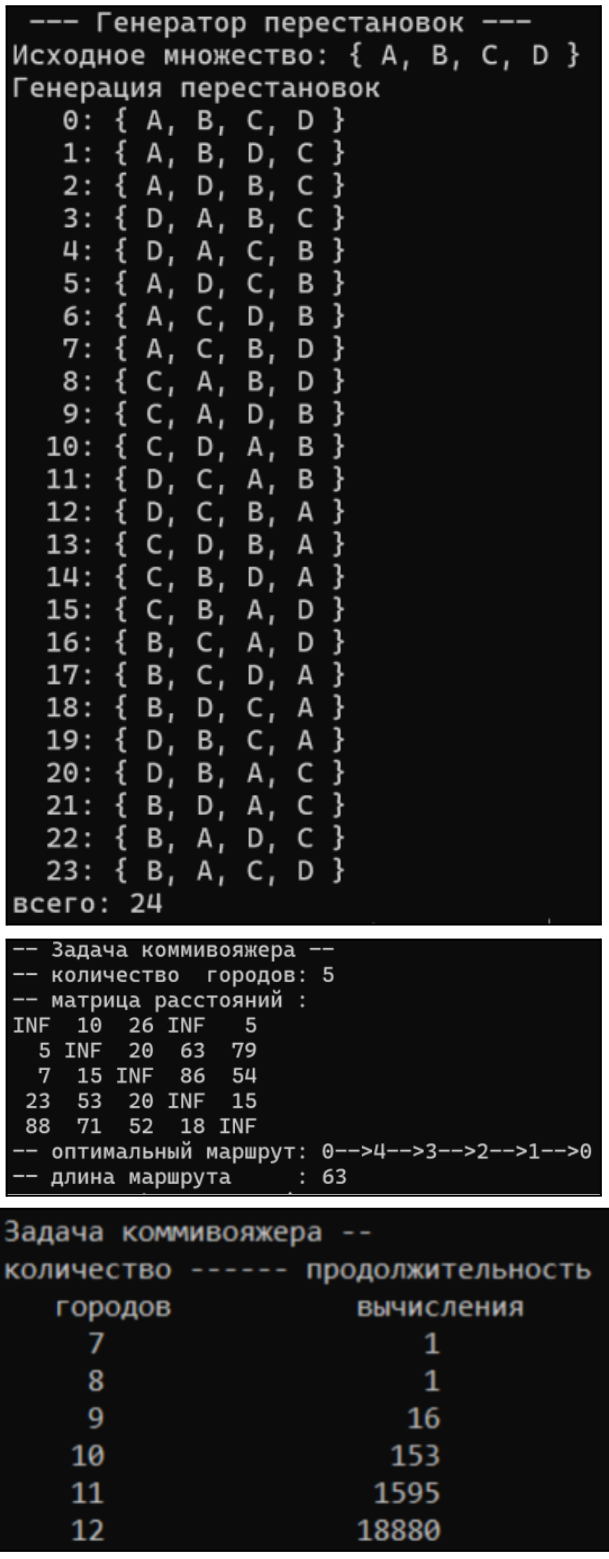


Рисунок 1.1 Результаты работы программы

Задание 6.

Исследовать зависимость времени вычисления необходимое для решения задачи коммивояжера (6 – 12 городов) от размерности задачи и результат в виде графика с небольшим пояснением занести в отчет:

Измерение скорости выполнения функции генерации случайных чисел. Результаты измерений и соответствующий график приведены на рисунке 1.2.

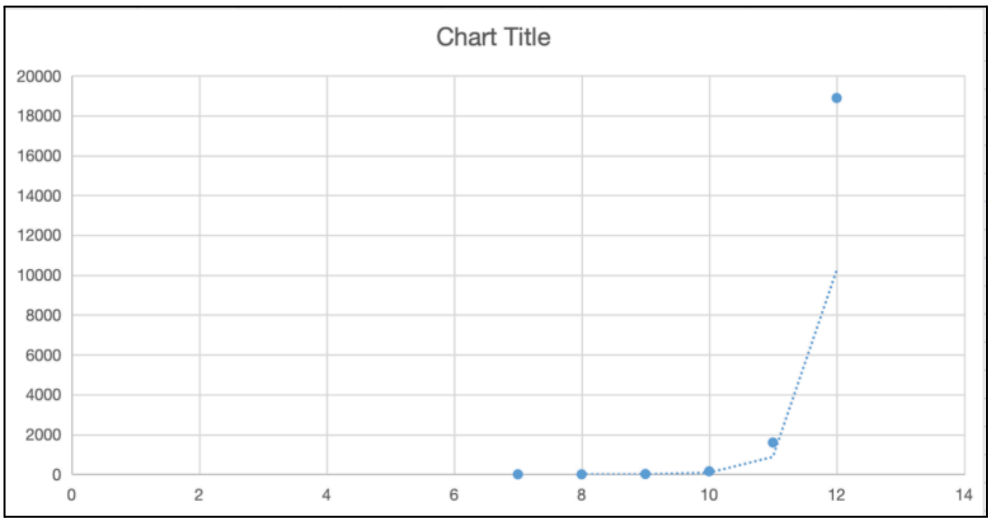


Рисунок 1.2 Результаты измерений и их график

Вывод: скорость выполнения программы не линейно зависит от количества итераций цикла. Похоже на факториал.

Вывод

Разработаны генераторы подмножеств, сочетаний, перестановок и размещений. Реализовано решение задачи коммивояжера для 10 городов с учетом случайных расстояний и бесконечных маршрутов. Проведено исследование зависимости времени вычисления от размерности задачи. Установлено, что время выполнения задачи коммивояжера растет экспоненциально с увеличением количества городов, что подтверждает NP-сложность задачи. Приобретены навыки работы с комбинаторными алгоритмами и их применения для решения задач оптимизации.

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

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.

Задание 1. Сформулировать условие задачи коммивояжера с параметром. Для этого: − принять элементы матрицы расстояний равными:

Город	1	2	3	4	5
1		18	30		9
2	9		24	59	75
3	11	27		86	58
4	26	49	36		27
5	84	75	52	22	

где *n* – номер варианта или номер по журналу;
n = 9;
Задачу следует решить с использованием метода ветвей и границ.
Ход решения:
Имеем 5 городов, построим матрицу расстояний между городами:

	1	2	3	4	5

1		18	30		9
2	9		24	59	75
3	11	27		86	58
4	26	49	36		27
5	84	75	52	22	

Находим минимальное значение в каждой строке (di) и выписываем его в отдельный столбец:

	1	2	3	4	5	
1		18	30		9	9
2	9		24	59	75	9
3	11	27		86	58	11

4	26	49	36		27	26
5	84	75	52	22		22

Производим приведение строк – из каждого элемента в строке вычитаем соответствующее значение найденного минимума (di).

	1	2	3	4	5	
1		9	21		0	9
2	0		15	50	66	9
3	0	16		75	47	11
4	0	23	10		1	26
5	62	*53*	30	0		22
					77	

Находим минимальные значения в каждом столбце (dj). Эти минимумы выписываем в отдельную строку.

	1	2	3	4	5	
1		9	21		0	
2	0		15	50	66	
3	0	16		75	47	
4	0	23	10		1	
5	62	*53*	30	0		
	0	9	10	0	0	19

Вычитаем из каждого элемента матрицы соответствующее ему минимальные значения в каждом столбце dj.

	1	2	3	4	5

1		0	11		0
2	0		5	50	66
3	0	7		75	47
4	0	14	0		1
5	62	*44*	20	0	

Тогда корневой вершиной будет

$$f=77+19=96.$$

Для каждой нулевой клетки получившейся преобразованной матрицы находим «оценку». Полученную оценку записываем рядом с нулем, в скобках.

	1	2	3	4	5
1		0(0)	11		0(0)
2	0(5)		5	50	66
3	0(7)	7		75	47

4	0(0)	14	0(0)		1
5	62	*44*	20	0(20)	

Выбираем нулевую клетку с наибольшей оценкой. Будем рассматривать дугу (5,4). Так как удаление дуги (5,4) позволяет получить саму большую константу приведения, т.е. увеличение нижней границы. Для этого заменим вес дуги (5,4) на знак “INF.

	1	2	3	4	5
1		0(0)	11		0(0)
2	0(5)		5	50	66
3	0(7)	7		75	47
4	0(0)	14	0(0)		1
5	62	*44*	20	0(20)	

	1	2	3	4	5

1		0	11		0
2	0		5	0	66
3	0	7		25	47
4	0	14	0		1
5	42	*24*	0		

Локальная нижняя граница: H2=96+20+50=166
Выбираем ветвь 1, тк 166>96
Получаем матрицу.

	1	2	3	5
1		0	11	0
2	0		5	66
3	0	7		47
4	0	14	0	1

Вычисляем оценки для нулевых клеток.

	1	2	3	5
1		0(0)	11	0(0)
2	0(5)		5	66
3	0(7)	7		47
4	0(0)	14	0(0)	1

Исключаем строку 3 и столбец 1. Запрещаем обратный путь.

	2	3	5
1	0	11	0
2		5	66
4	14	0	1

Проводим редукцию

	2	3	5
1	0	11	0
2		0	61
4	14	0	1

Локальная нижняя границв 96+5 = 101

Исключаем путь 3,1

	1	2	3	5
1		0	11	0
2	0		5	66
3		7		47
4	0	14	0	1

Проводим редукцию

	1	2	3	5
1		0	11	0
2	0		5	66
3		0		40
4	0	14	0	1

Локальная нижняя граница 96+7 = 103
выбираем 1.1, тк 103>101

	2	3	5

1	0(0)	11	0(0)
2		0(61)	61
4	14	0(1)	1

Исключаем строку 2 и столбец 3.

	2	5
1	0	0
4	14	1

Проводим редукцию

	2	5
1	0	0
4	13	0

Локальная нижняя граница 101+1=102

Исключаем путь (2,3)

	2	3	5
1	0	11	0
2			61
4	14	0	1

Проводим редукцию

	2	3	5
1	0	11	0

2			0
4	14	0	1

Локальная нижняя граница 101+61=162

Выбираем ветвь 1.1.1 тк 162>102

	2	5
1	0(0)	0(0)
4	13	0(13)

Исключаем строку 4 и столбец 5

	2
1	0

Локальная нижняя граница 102+0=102

Исключаем путь 4,5

	2	5
1	0	0
4	13	INF

Проводим редукцию

	2	5
1	0	0
4	0	INF

Локальная нижняя граница 102+13=115

Выбираем ветвь 1.1.1.1 тк 115>102

Итоговый маршрут
5-4 22

4-3 36

3-2 27

2-1 9

1-5 9

Общая длина 22+36+27+9+9=103

Корень ($H_0 = 96$)

├── (5,4) включено (H= 96)

│ ├── (3,1) включено (H = 101)

│ │ ├── (2,3) включено (H = 102)

│ │ │ ├── (4,5)

│ │ │ └── (4,5) исключено (H= 115)

│ │ └── (2,3) исключено (H= 162)

│ └── (3,1) исключено (H= 103)

└── (5,4) исключено (H= 166)

Вывод

В процессе выполнения лабораторной работы были изучены основные принципы решения задач методом ветвей и границ. Была решена задача коммивояжера с использованием этого метода, а также проведено сравнение полученного результата с решением, найденным методом полного перебора (комбинаторным методом перестановок).

Лабораторная работа 4. ДИНАМИЧЕСКОЕ ПРОГРАММИРОВАНИЕ

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

Задание 1.

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

Код функции для генерации случайных строк представлен в листинге 1.

```
char* GenerateRandomString(int size)
{
    char* str = (char*)malloc(sizeof(char) * (size + 1));
    for (int i = 0; i < size; i++) {
        str[i] = rand() % 26 + 'a'; // 26 букв в алфавите
    }
    str[size] = '\0';
    return str;
}
```

Листинг 1

Результат генерации строк представлен на рисунке 1.

```
S1:

phqghumeaylnlfdxfircvscxggbwkfnqduxwfnfozvsrtkjpre
pggxrpnrvystmwcysyycqpevikeffmznimkkasvwsrenzkycxf
xtlsgypsfadpooefxzbcoejuvpvaboygpoeylfpbnpljvrvipy
amyehwqnqrqpmxujjloovaowuxwhmsncbxcoksfzkvatxdknly
jyhfixjswnkkufnuxxzrzbmnmgqooketlyhnkoaugzqrcddiut
eiojwayyzpvscmpsajlfvgubfaaovlzylntrkdcpwsrtesjwhd

S2:

lckyljakromawkouvmxtgxhnumeoorcrpxzvwnnppdfffiudvy
lzobwvhlskizdrbymzadgzlxkcwqpdahbhzjahcxnwcqvzgnuf
yxfoewdmdzmkghfcvjvagvfntlvbfeciuzblqyeeurftlqozpd
vqgcuuqsjbqlnhahsrgzvrwtcoximodvhoufjqivnisvqzmpqv
```

Рисунок 1

Задание 2.

Вычислить двумя способами (рекурсивно и с помощью динамического программирования) $L(prefix(S_1, k \times len(S_1)), prefix(S_2, k \times len(S_2))$ – дистанцию

Левенштейна для $k = \left\{\frac{1}{25}, \frac{1}{20}, \frac{1}{15}, \frac{1}{10}, \frac{1}{5}, \frac{1}{2}, 1\right\}$, где $len(S)$ - длина строки S, $prefix(S,k)$ - строка состоящая из первых k символов строки S.

Код функций для вычисления дистанции Левенштейна для строк, сгенерированных в задании 1, представлен в листингах 2-3.

```
#pragma once
// -- дистанции   Левенштейна (динамическое программирование)
int levenshtein(
      int lx,           // длина слова x
      const char x[],   // слово длиной lx
      int ly,           // длина слова y
      const char y[]    // слово y
);
// -- дистанции   Левенштейна (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
);
```

Листинг 2 – Код файла Levenshtein.h

```
#include "stdafx.h"
#include <iomanip>
#include <algorithm>
#include "Levenshtein.h"
#define DD(i,j) d[(i)*(ly+1)+(j)]

int min3(int x1, int x2, int x3)
{
	return std::min(std::min(x1, x2), x3);
}

int levenshtein(int lx, const char x[], int ly, const char y[])
{
	int* d = new int[(lx + 1) * (ly + 1)];
	for (int i = 0; i <= lx; i++) DD(i, 0) = i;
	for (int j = 0; j <= ly; j++) DD(0, j) = j;
	for (int i = 1; i <= lx; i++)
		for (int j = 1; j <= ly; j++)
		{
			DD(i, j) = min3(DD(i - 1, j) + 1, DD(i, j - 1) + 1,
				DD(i - 1, j - 1) + (x[i - 1] == y[j - 1] ? 0 : 1));
		}
	return DD(lx, ly);
}

int levenshtein_r(
	int lx, const char x[],
	int ly, const char y[]
)
{
	int rc = 0;
	if (lx == 0) rc = ly;
	else  if (ly == 0) rc = lx;
	else  if (lx == 1 && ly == 1 && x[0] == y[0]) rc = 0;
	else  if (lx == 1 && ly == 1 && x[0] != y[0]) rc = 1;
	else  rc = min3(
		levenshtein_r(lx - 1, x, ly, y) + 1,
		levenshtein_r(lx, x, ly - 1, y) + 1,
		levenshtein_r(lx - 1, x, ly - 1, y) + (x[lx - 1] == y[ly - 1] ? 0 : 1)
	);
	return rc;
};
```

Листинг 3 – Код файла Levenshtein.cpp

Задание 3.

Выполнить сравнительный анализ времени затраченного на вычисление дистанции Левенштейна для двух методов решения. Построить графики зависимости времени вычисления от k.

Результат выполнений вычислений с вычислением времени в милисекундах представлен на рисунке 2.

```
-- расстояние Левенштейна -----

--длина --- рекурсия -- дин.програм. ---
12/ 8        87           0
15/10        3571         0
```

Рисунок 2

Исходя из результатов затраченного времени можно сделать вывод, что рекурсивный метод более времязатратный.

График зависимости времени вычисления представлен на рисунке 3.

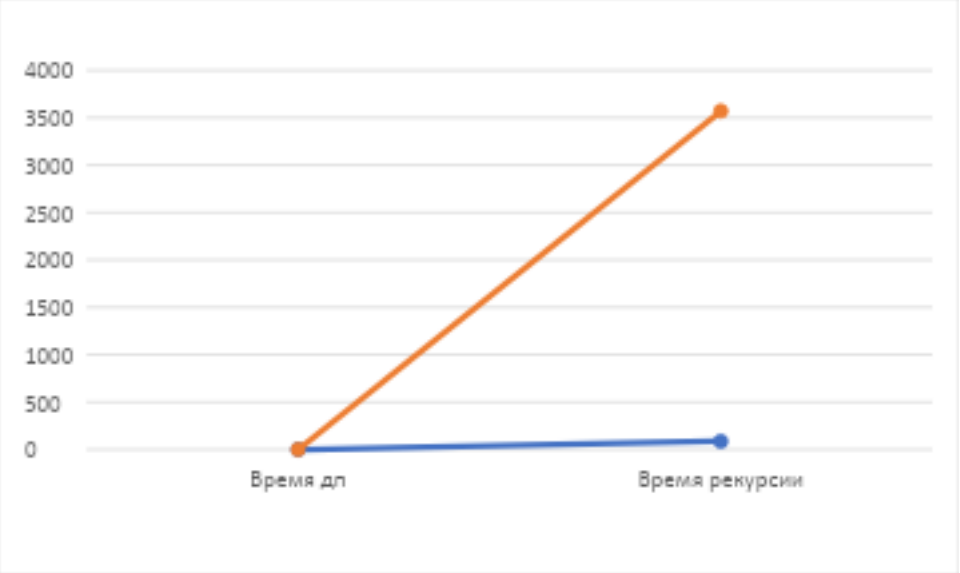


Рисунок 3

Задание 4.

Реализовать вручную пример вычисления дистанции Левенштейна при помощи рекурсивного алгоритма (в соответствии с вариантом).

8	Вар	Баран

1. $L(\text{"Вар"}, \text{"Баран"})$
$= \{L(\text{"Ва"}, \text{"Баран"}) + 1,\ L(\text{"Вар"}, \text{"Бара"}) + 1,\ L(\text{"Ва"}, \text{"Бара"}) + 1.$

2. $L(\text{"Ва"}, \text{"Баран"})$
$= \{L(\text{"В"}, \text{"Баран"}) + 1,\ L(\text{"Ва"}, \text{"Бара"}) + 1,\ L(\text{"В"}, \text{"Бара"}) + 1.$

3. $L(\text{"Вар"}, \text{"Бара"})$
$= \{L(\text{“Ва”}, \text{”Бара”}) + 1,\ L(\text{“Вар”}, \text{”Бар”}) + 1,\ L(\text{“Ва”}, \text{”Бар”}) + 1$

4. L("Ва", "Бара")

$= \{L$("В", "Бара") $+ 1,\ L$("Ва", "Бар") $+ 1,\ L$("В", "Бар") $+ 0.$

5. L("В", "Баран")

$= \{L$("", "Баран") $+ 1,\ L$("В", "Бара") $+ 1,\ L$("", "Бара") $+ 1.$

L("", "Баран") = 5.

L("", "Бара") = 4.

6. L("В", "Бара")

$= \{L$("", "Бара") $+ 1,\ L$("В", "Бар") $+ 1,\ L$("", "Бар") $+ 1.$

L("", "Бара") = 4.

L("", "Бар") = 3.

7. L("Вар", "Бар")

$= \{L$("Ва", "Бар") $+ 1,\ L$("Вар", "Ба") $+ 1,\ L$("Ва", "Ба") $+ 0.$

8. L("Ва", "Бар")

$= \{L$("В", "Бар") $+ 1,\ L$("Ва", "Ба") $+ 1,\ L$("В", "Ба") $+ 1.$

9. L("В", "Бар")

$= \{L$("", "Бар") $+ 1,\ L$("В", "Ба") $+ 1,\ L$("", "Ба") $+ 1.$

L("", "Бар") = 3.

L("", "Ба") = 2.

10. L("Вар", "Ба")

$= \{L$("Ва", "Ба") $+ 1,\ L$("Вар", "Б") $+ 1,\ L$("Ва", "Б")

11. L("Ва", "Ба")

$= \{L$("В", "Ба") $+ 1,\ L$("Ва", "Б") $+ 1,\ L$("В", "Б") $+ 0.$

12. L("В", "Ба") $= \{L$("", "Ба") $+ 1,\ L$("В", "Б") $+ 1,\ L$("", "Б") $+ 1.$

L("", "Ба") = 2.

L("", "Б") = 1.

13. L("Вар", "Б")

$= \{L$("Ва", "Б") $+ 1,\ L$("Вар", "") $+ 1,\ L$("Ва", "") $+ 1.$

L("Вар", "") = 3.

L("Ва", "") = 2.

14. L("Ва", "Б") $= \{L$("В", "Б") $+ 1,\ L$("Ва", "") $+ 1,\ L$("В", "") $+ 1$

L("Ва", "") = 2.

L("В, "") = 1.
L(“В”, “Б”) = 0

15. L(“Ва”, “Б”) = min(2,3,2) =2
16. L(“Вар”, “Б”) = min(3,4,3) = 3
17. L(“В”, “Ба”) = min(3,2,2) = 2
18. L(“Ва”, “Ба”) = min(3,3,1) = 1
19. L(“Вар”, “Ба”) = min(2,4,3) = 2
20. L(“В”, “Бар”) = min(4,3,3) = 3
21. L(“Ва”, “Бар”) = min(4,2,3) = 2
22. L(“Вар”, “Бар”) = min(3,3,1) = 1
23. L(“В”, “Бара”) = min(5,4,4) = 4
24. L(“В”, “Баран”) = min(6,5,5) = 5
25. L(“Ва”, “Бара”) = min(5,3,3) = 3
26. L(“Вар”, “Бара”) = min(5,3,3) = 3
27. L(“Ва”, “Баран”) = min(6,4,5) = 4
28. L(“Вар”, “Баран”) = min(5,3,4) = 3

Таким образом самая короткая дистанция Левенштейна равна 3. Результат представлен на рисунке 4.

```
Строка 1: Вар (3 символа)
Строка 2: Баран (5 символов)
Рекурсивно: расстояние Левенштейна = 3
Динамически: расстояние Левенштейна = 3
Для продолжения нажмите любую клавишу . . .
```

Рисунок 4

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.

8	10*15, 15*80, 80*23, 23*50, 50*40, 40*71

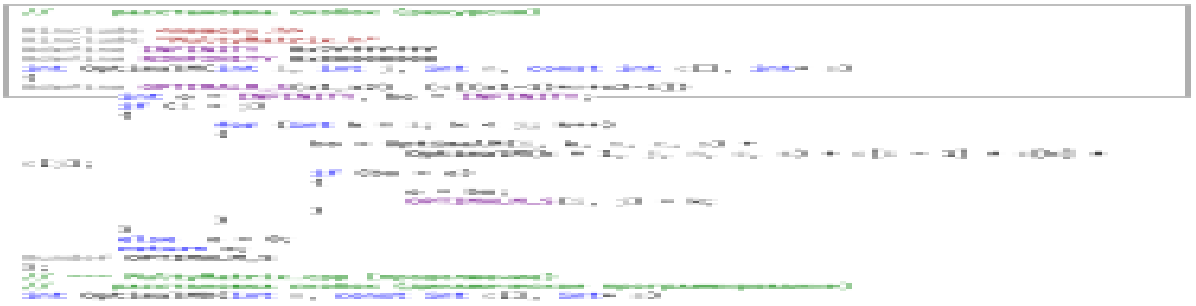
Листинги кода представлены ниже.

```h
#pragma once
//    расстановка скобок
#pragma once
// расстановка скобок при умножении матриц
// функции возвращают минимальное количество операций умножения
#define OPTIMALM_PARM(x) ((int*)x)    // для представления 2мерного
массива

int OptimalM(              // рекурсия
      int i,           //  [in] номер первой матрицы
      int j,           //  [in] номер последней матрицы
      int n,           //  [in] количество матриц
      const int c[],   //  [in] массив размерностей
      int* s           //  [out] результат: позиции скобок
);

int OptimalMD(                // динамическое программирование
      int n,           //  [in] количество матриц
      const int c[],   //  [in] массив размерностей
      int* s           //  [out] результат: позиции скобок
);
```

Листинг 4 – Код файла MultyMatrix.h



Листинг 5 – Код файла MultyMatrix.cpp

Результат выполнения представлен на рисунке 5.

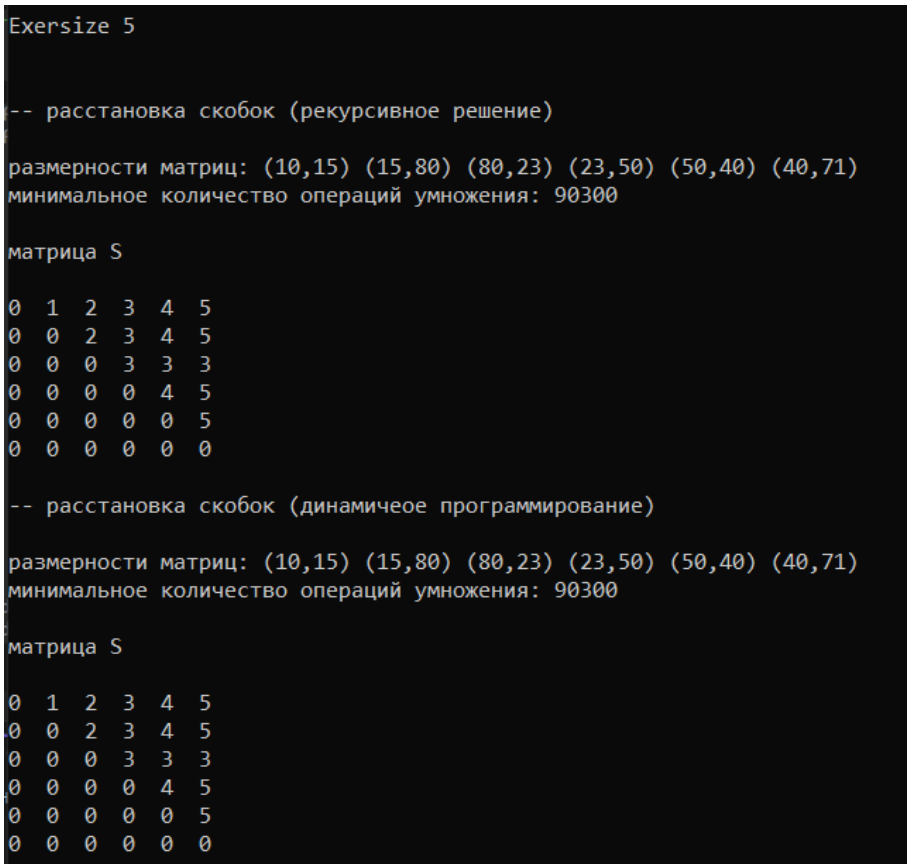


Рисунок 5

Принцип расстановки скобок по итоговой матрице:

Скобки расставляются по принципу «сначала внешние – затем внутренние». Имеется 6 матриц, вот их размерность:

A1=10*15,
A2=15*80,
A3=80*23,
A4 =23*50,
A5 =50*40,
A6 =40*71.

Матрица S:

	1	**2**	**3**	**4**	**5**	**6**
1	0	1	2	3	4	5
2	0	0	2	3	4	5
3	0	0	0	3	3	3
4	0	0	0	0	4	5
5	0	0	0	0	0	5
6	0	0	0	0	0	0

Оптимальное разбиение матриц выглядит так (((((A1A2)A3)A4)A5)A6)

Вывод: динамический подход к решению задач позволяет выполнять их значительно быстрее, чем рекурсивный, особенно это будет заметно при решении задач с большим объёмом информации.

Лабораторная работа 5. ТРАНСПОРТНАЯ ЗАДАЧА

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*).

ПОТРЕБИТЕЛИ ПОСТАВЩИКИ	1	2	3	4	5	6	ЗАПАСЫ
1	**20**	**10**	**14**	**11**	**19**	**9**	**176**
2	**18**	**8**	**16**	**13**	**15**	**21**	**121**
3	**9**	**13**	**19**	**16**	**10**	**19**	**158**
4	**12**	**18**	**18**	**11**	**21**	**10**	**167**
5	**11**	**19**	**17**	**8**	**18**	**12**	**108**
ПОТРЕБНОСТИ	**151**	**115**	**139**	**201**	**103**	**171**	

Для проверки на открытость или закрытость задачи необходимо посчитать сумму потребностей груза и сумму запасов груза:

Запасы груза: ∑a = 176+121+158+167+108=730

Потребность груза: ∑b = 151+115+139+201+103+171=880

$\sum_{i=1}^{m} a_i = \sum_{j=1}^{n} b_j$ – задача называется ***закрытой***

$\sum_{i=1}^{m} a_i \neq \sum_{j=1}^{n} b_j$ – задача называется ***открытой(с нарушенным балансом)***.

Решение открытой задачи сводится к решению закрытой

Таким образом, так как ∑a != ∑b, значит задача является открытой.

Для решения задачи необходимо привести ее к закрытому виду, для этого введем дополнительного фиктивного поставщика с запасом равным 150.

I. Опорный план(метод наименьшей стоимости)

Суть метода заключается в том, что из всей таблицы стоимостей выбирают наименьшую, и в клетку, которая ей соответствует, помещают меньшее из чисел a_i, или b_j.

Затем, из рассмотрения исключают либо строку, соответствующую поставщику, запасы которого полностью израсходованы, либо столбец, соответствующий потребителю, потребности которого полностью удовлетворены, либо и строку и столбец, если израсходованы запасы поставщика и удовлетворены потребности потребителя.

Из оставшейся части таблицы стоимостей снова выбирают наименьшую стоимость, и процесс распределения запасов продолжают, пока все запасы не будут распределены, а потребности удовлетворены.

Выбираем ячейку с наименьшим значением: c(22) = 8

ПОТРЕБИТЕЛИ ПОСТАВЩИКИ	1	2	3	4	5	6	ЗАПАСЫ
1	**20**	**10**	**14**	**11**	**19**	**9**	**176**
2	**18**	**8**	**16**	**13**	**15**	**21**	**121**
3	**9**	**13**	**19**	**16**	**10**	**19**	**158**
4	**12**	**18**	**18**	**11**	**21**	**10**	**167**
5	**11**	**19**	**17**	**8**	**18**	**12**	**108**
6	**0**	**0**	**0**	**0**	**0**	**0**	**150**
ПОТРЕБНОСТИ	**151**	**115**	**139**	**201**	**103**	**171**	

x(22)=min(121, 115) = 115

Выбираем ячейку с наименьшим значением: c(54) = 8

ПОТРЕБИТЕЛИ ПОСТАВЩИКИ	1	2	3	4	5	6	ЗАПАСЫ
1	**20**	**10**	**14**	**11**	**19**	**9**	**176**
2	**18**	**8| 115**	**16**	**13**	**15**	**21**	**6**
3	**9**	**13**	**19**	**16**	**10**	**19**	**158**
4	**12**	**18**	**18**	**11**	**21**	**10**	**167**
5	**11**	**19**	**17**	**8**	**18**	**12**	**108**
6	**0**	**0**	**0**	**0**	**0**	**0**	**150**
ПОТРЕБНОСТИ	**151**	**0**	**139**	**201**	**103**	**171**	

x(54)=min(108, 201) = 108

Выбираем ячейку с наименьшим значением: c(16) = 9

ПОТРЕБИТЕЛИ ПОСТАВЩИКИ	1	2	3	4	5	6	ЗАПАСЫ
1	**20**	**10**	**14**	**11**	**19**	**9**	**176**
2	**18**	**8| 115**	**16**	**13**	**15**	**21**	**6**
3	**9**	**13**	**19**	**16**	**10**	**19**	**158**
4	**12**	**18**	**18**	**11**	**21**	**10**	**167**
5	**11**	**19**	**17**	**8| 108**	**18**	**12**	**0**
6	**0**	**0**	**0**	**0**	**0**	**0**	**150**
ПОТРЕБНОСТИ	**151**	**0**	**139**	**93**	**103**	**171**	

x(16)=min(176, 171) = 171

Выбираем ячейку с наименьшим значением: c(31) = 9

ПОТРЕБИТЕЛИ ПОСТАВЩИКИ	1	2	3	4	5	6	ЗАПАСЫ

1	**20**	**10**	**14**	**11**	**19**	**9| 171**	**5**
2	**18**	**8| 115**	**16**	**13**	**15**	**21**	**6**
3	**9**	**13**	**19**	**16**	**10**	**19**	**158**
4	**12**	**18**	**18**	**11**	**21**	**10**	**167**
5	**11**	**19**	**17**	**8| 108**	**18**	**12**	**0**
6	**0**	**0**	**0**	**0**	**0**	**0**	**150**
ПОТРЕБНОСТИ	**151**	**0**	**139**	**93**	**103**	**0**	

x(31)=min(158, 151) = 151

Выбираем ячейку с наименьшим значением: c(35) = 10

ПОТРЕБИТЕЛИ ПОСТАВЩИКИ	1	2	3	4	5	6	ЗАПАСЫ
1	**20**	**10**	**14**	**11**	**19**	**9| 171**	**5**
2	**18**	**8| 115**	**16**	**13**	**15**	**21**	**6**
3	**9| 151**	**13**	**19**	**16**	**10**	**19**	**7**
4	**12**	**18**	**18**	**11**	**21**	**10**	**167**
5	**11**	**19**	**17**	**8| 108**	**18**	**12**	**0**
6	**0**	**0**	**0**	**0**	**0**	**0**	**150**
ПОТРЕБНОСТИ	**0**	**0**	**139**	**93**	**103**	**0**	

x(35)=min(7, 103) = 7

Выбираем ячейку с наименьшим значением: c(14) = 11

ПОТРЕБИТЕЛИ ПОСТАВЩИКИ	1	2	3	4	5	6	ЗАПАСЫ
1	**20**	**10**	**14**	**11**	**19**	**9| 171**	**5**
2	**18**	**8| 115**	**16**	**13**	**15**	**21**	**6**
3	**9| 151**	**13**	**19**	**16**	**10| 7**	**19**	**0**
4	**12**	**18**	**18**	**11**	**21**	**10**	**167**
5	**11**	**19**	**17**	**8| 108**	**18**	**12**	**0**
6	**0**	**0**	**0**	**0**	**0**	**0**	**150**
ПОТРЕБНОСТИ	**0**	**0**	**139**	**93**	**96**	**0**	

x(14)=min(5, 93) = 5

Выбираем ячейку с наименьшим значением: c(44) = 11

ПОТРЕБИТЕЛИ ПОСТАВЩИКИ	1	2	3	4	5	6	ЗАПАСЫ
1	**20**	**10**	**14**	**11| 5**	**19**	**9| 171**	**0**
2	**18**	**8| 115**	**16**	**13**	**15**	**21**	**6**
3	**9| 151**	**13**	**19**	**16**	**10| 7**	**19**	**0**
4	**12**	**18**	**18**	**11**	**21**	**10**	**167**
5	**11**	**19**	**17**	**8| 108**	**18**	**12**	**0**
6	**0**	**0**	**0**	**0**	**0**	**0**	**150**

ПОТРЕБНОСТИ	**0**	**0**	**139**	**88**	**96**	**0**	

x(44)=min(167, 88) = 88

Выбираем ячейку с наименьшим значением: c(25) = 15

ПОТРЕБИТЕЛИ ПОСТАВЩИКИ	1	2	3	4	5	6	ЗАПАСЫ
1	**20**	**10**	**14**	**11| 5**	**19**	**9| 171**	**0**
2	**18**	**8| 115**	**16**	**13**	**15**	**21**	**6**
3	**9| 151**	**13**	**19**	**16**	**10| 7**	**19**	**0**
4	**12**	**18**	**18**	**11| 88**	**21**	**10**	**79**
5	**11**	**19**	**17**	**8| 108**	**18**	**12**	**0**
6	**0**	**0**	**0**	**0**	**0**	**0**	**150**
ПОТРЕБНОСТИ	**0**	**0**	**139**	**0**	**96**	**0**	

x(25)=min(6, 96) = 6

Выбираем ячейку с наименьшим значением: c(43) = 18

ПОТРЕБИТЕЛИ ПОСТАВЩИКИ	1	2	3	4	5	6	ЗАПАСЫ
1	**20**	**10**	**14**	**11| 5**	**19**	**9| 171**	**0**
2	**18**	**8| 115**	**16**	**13**	**15| 6**	**21**	**0**
3	**9| 151**	**13**	**19**	**16**	**10| 7**	**19**	**0**
4	**12**	**18**	**18**	**11| 88**	**21**	**10**	**79**
5	**11**	**19**	**17**	**8| 108**	**18**	**12**	**0**
6	**0**	**0**	**0**	**0**	**0**	**0**	**150**
ПОТРЕБНОСТИ	**0**	**0**	**139**	**0**	**90**	**0**	

x(43)=min(79, 139) = 79

Выбираем ячейку с наименьшим значением: c(63) = 0

ПОТРЕБИТЕЛИ ПОСТАВЩИКИ	1	2	3	4	5	6	ЗАПАСЫ
1	**20**	**10**	**14**	**11| 5**	**19**	**9| 171**	**0**
2	**18**	**8| 115**	**16**	**13**	**15| 6**	**21**	**0**
3	**9| 151**	**13**	**19**	**16**	**10| 7**	**19**	**0**
4	**12**	**18**	**18| 79**	**11| 88**	**21**	**10**	**0**
5	**11**	**19**	**17**	**8| 108**	**18**	**12**	**0**
6	**0**	**0**	**0**	**0**	**0**	**0**	**150**
ПОТРЕБНОСТИ	**0**	**0**	**60**	**0**	**90**	**0**	

x(63)=min(150, 60) = 60

Выбираем ячейку с наименьшим значением: c(65) = 0

ПОТРЕБИТЕЛИ ПОСТАВЩИКИ	1	2	3	4	5	6	ЗАПАСЫ
1	**20**	**10**	**14**	**11| 5**	**19**	**9| 171**	**0**

2	**18**	**8| 115**	**16**	**13**	**15| 6**	**21**	**0**
3	**9| 151**	**13**	**19**	**16**	**10| 7**	**19**	**0**
4	**12**	**18**	**18| 79**	**11| 88**	**21**	**10**	**0**
5	**11**	**19**	**17**	**8| 108**	**18**	**12**	**0**
6	**0**	**0**	**0| 60**	**0**	**0**	**0**	**90**
ПОТРЕБНОСТИ	**0**	**0**	**0**	**0**	**90**	**0**	

x(65)=min(90, 90) = 90

ПОТРЕБИТЕЛИ ПОСТАВЩИКИ	1	2	3	4	5	6	ЗАПАСЫ
1	**20**	**10**	**14**	**11| 5**	**19**	**9| 171**	**0**
2	**18**	**8| 115**	**16**	**13**	**15| 6**	**21**	**0**
3	**9| 151**	**13**	**19**	**16**	**10| 7**	**19**	**0**
4	**12**	**18**	**18| 79**	**11| 88**	**21**	**10**	**0**
5	**11**	**19**	**17**	**8| 108**	**18**	**12**	**0**
6	**0**	**0**	**0| 60**	**0**	**0| 90**	**0**	**0**
ПОТРЕБНОСТИ	**0**	**0**	**0**	**0**	**0**	**0**	

В результате получен первый опорный план, который является допустимым, так как все запасы распределены, а потребности удовлетворены, план соответствует системе ограничений транспортной задачи. Должно быть m + n - 1 = 6 + 6 - 1=11 переменных. В таблице их 11, значит план невырожденный.

Первое допустимое решение:

X14=5, X16=171, X22=115, X25=6, X31=151, X35=7, X43=79, X44=88, X54=108, X63=60, X65=90

Значение функции цели:

Z=11*5+9*171+8*115+15*6+9*151+10*7+18*79+11*88+8*108+0*60+0*90=7287

II. Метод потенциалов

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

Проверим оптимальность опорного плана. Найдем предварительные потенциалы ui, vj. по занятым клеткам таблицы, в которых ui + vj = cij. Уравнений 11 неизвестных 12. Присваиваем одному из них произвольное значение (обычно u1 = 0).

u1+v4=11; 0+v4=11; v4=11
u1+v6=9; 0+v6=9; v6=9
u2+v2=8; -3+v2=8; v2=11
u2+v5=15; u2+18=15; u2=-3
u3+v1=9; -8+v1=9; v1=17
u3+v5=10; u3+18=10; u3=-8

u4+v3=18; 0+v3=18; v3=18
u4+v4=11; u4+11=11; u4=0
u5+v4=8; u5+11=8; u5=-3
u6+v3=0; u6+18=0; u6=-18
u6+v5=0; -18+v5=0; v5=18

ПОТРЕБИТЕЛИ ПОСТАВЩИКИ	v1=17	v2=11	v3=18	v4=11	v5=18	v6=9	ЗАПАСЫ
u1=0	**20**	**10**	**14**	**11| 5**	**19**	**9| 171**	**0**
u2=-3	**18**	**8| 115**	**16**	**13**	**15| 6**	**21**	**0**
u3=-8	**9| 151**	**13**	**19**	**16**	**10| 7**	**19**	**0**
u4=0	**12**	**18**	**18| 79**	**11| 88**	**21**	**10**	**0**
u5=-3	**11**	**19**	**17**	**8| 108**	**18**	**12**	**0**
u6=-18	**0**	**0**	**0| 60**	**0**	**0| 90**	**0**	**0**
ПОТРЕБНОСТИ	**0**	**0**	**0**	**0**	**0**	**0**	

Опорный план не является оптимальным, так как существуют оценки свободных клеток, для которых $(u_i + v_j) - c_{ij} > 0$

Для свободных клеток $x_{ij} = u_i + v_j - c_{ij}$

Небазисная переменная	$u_i + v_j - c_{ij} = x_{ij}$
x_{11}	0+17-20=-3
x_{12}	0+11-10=1
x_{13}	0+18-14=4
x_{15}	0+18-19=-1
x_{21}	-3+17-18=-4
x_{23}	-3+18-16=-1
x_{24}	-3+11-13=-5
x_{26}	-3+9-21=-15
x_{32}	-8+11-13=-10
x_{33}	-8+18-19=-9
x_{34}	-8+11-16=-13
x_{36}	-8+9-19=-18
x_{41}	0+17-12=5
x_{42}	0+11-18=-7
x_{45}	0+18-21=-3
x_{46}	0+9-10=-1
x_{51}	-3+17-11=3
x_{52}	-3+11-19=-11
x_{53}	-3+18-17=-2
x_{55}	-3+18-18=-3
x_{56}	-3+9-12=-6

Вводимой в базис будет переменная имеющая наибольшее положительное значение -$x_{41.}$

Создаем цикл, который представляет собой замкнутую линию, содержащую исключительно вертикальные и горизонтальные линии, которые соединяют выбранную ячейку и ячейки, которые входят в решение транспортной задачи. 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	**20**	**10**	**14**	**11| 5**	**19**	**9| 171**	**176**
2	**18**	**8| 115**	**16**	**13**	**15| 6**	**21**	**121**
3	**9| 151[-]**	**13**	**19**	**16**	**10| 7[+]**	**19**	**158**
4	**12[+]**	**18**	**18| 79[-]**	**11| 88**	**21**	**10**	**167**
5	**11**	**19**	**17**	**8| 108**	**18**	**12**	**108**
6	**0**	**0**	**0| 60[+]**	**0**	**0| 90[-]**	**0**	**150**
ПОТРЕБНОСТИ	**151**	**115**	**139**	**201**	**103**	**107**	

Цикл приведен в таблице (4,1 → 4,3 → 6,3 → 6,5 → 3,5 → 3,1).

Из грузов xij стоящих в минусовых клетках, выбираем наименьшее, т.е. y = min (4, 3) = 79. Прибавляем 79 к объемам грузов, стоящих в плюсовых клетках и вычитаем 79 из X_{ij}, стоящих в минусовых клетках.

ПОТРЕБИТЕЛИ ПОСТАВЩИКИ	1	2	3	4	5	6	ЗАПАСЫ
1	**20**	**10**	**14**	**11| 5**	**19**	**9| 171**	**176**
2	**18**	**8| 115**	**16**	**13**	**15| 6**	**21**	**121**
3	**9[151-79=72]**	**13**	**19**	**16**	**10[7+79=86]**	**19**	**158**
4	**12[0+79=79]**	**18**	**18[79-79=0]**	**11| 88**	**21**	**10**	**167**
5	**11**	**19**	**17**	**8| 108**	**18**	**12**	**108**
6	**0**	**0**	**0[60+79=139]**	**0**	**0[90-79=11]**	**0**	**150**
ПОТРЕБНОСТИ	**151**	**115**	**139**	**201**	**103**	**107**	

В результате получим новый опорный план.

ПОТРЕБИТЕЛИ ПОСТАВЩИКИ	1	2	3	4	5	6	ЗАПАСЫ
1	**20**	**10**	**14**	**11| 5**	**19**	**9| 171**	**176**
2	**18**	**8| 115**	**16**	**13**	**15| 6**	**21**	**121**
3	**9| 72**	**13**	**19**	**16**	**10| 86**	**19**	**158**
4	**12| 79**	**18**	**18**	**11| 88**	**21**	**10**	**167**
5	**11**	**19**	**17**	**8| 108**	**18**	**12**	**108**
6	**0**	**0**	**0| 139**	**0**	**0| 11**	**0**	**150**
ПОТРЕБНОСТИ	**151**	**115**	**139**	**201**	**103**	**107**	

Проверим оптимальность опорного плана. Найдем *предварительные потенциалы* u_i, v_j. по занятым клеткам таблицы, в которых $u_i + v_j = c_{ij}$, полагая, что $u_1 = 0$.

u1+v4=11; 0+v4=11; v4=11
u1+v6=9; 0+v6=9; v6=9
u2+v2=8; 2+v2=8; v2=6
u2+v5=15; u2+13=15; u2=2
u3+v1=9; u3+12=9; u3=-3
u3+v5=10; -3+v5=10; v5=13
u4+v1=12; 0+v1=12; v1=12
u4+v4=11; u4+11=11; u4=0
u5+v4=8; u5+11=8; u5=-3
u6+v3=0; -13+v3=0; v3=13
u6+v5=0; u6+13=0; u6=-13

ПОТРЕБИТЕЛИ ПОСТАВЩИКИ	v1=12	v2=6	v3=13	v4=11	v5=13	v6=9	ЗАПАСЫ
u1=0	**20**	**10**	**14**	**11| 5**	**19**	**9| 171**	**176**
u2=2	**18**	**8| 115**	**16**	**13**	**15| 6**	**21**	**121**
u3=-3	**9| 72**	**13**	**19**	**16**	**10| 86**	**19**	**158**
u4=0	**12| 79**	**18**	**18**	**11| 88**	**21**	**10**	**167**
u5=-3	**11**	**19**	**17**	**8| 108**	**18**	**12**	**108**
u6=-13	**0**	**0**	**0| 139**	**0**	**0| 11**	**0**	**150**
ПОТРЕБНОСТИ	**151**	**115**	**139**	**201**	**103**	**107**	

Небазисная переменная	$u_i + v_j - c_{ij} = x_{ij}$
x_{11}	0+12-20=-8
x_{12}	0+6-10=-4
x_{13}	0+13-14=-1
x_{15}	0+13-19=-6
x_{21}	2+12-18=-4
x_{23}	2+13-16=-1

x_{24}	2+11-13=0
x_{26}	2+9-21=-10
x_{32}	-3+6-13=-10
x_{33}	-3+13-19=-9
x_{34}	-3+11-16=-8
x_{36}	-3+9-19=-13
x_{42}	0+6-18=-12
x_{43}	0+13-18=-5
x_{45}	0+13-21=-8
x_{46}	0+9-10=-1
x_{51}	-3+12-11=-2
x_{52}	-3+6-19=-16
x_{53}	-3+13-17=-7
x_{55}	-3+13-18=-8
x_{56}	-3+9-12=-6

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

Минимальные затраты составят: F(x) = 11*5 + 9*171 + 8*115 + 15*6 + 9*72 + 10*86 + 12*79 + 11*88 + 8*108 + 0*139 + 0*11 = 6892.

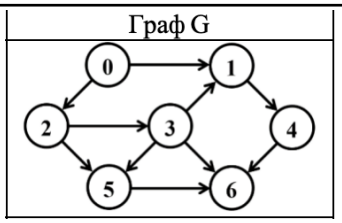
Анализ оптимального плана.
Из 1-го склада необходимо груз направить к 4-у потребителю (5 ед.), к 6-у потребителю (171 ед.)
Из 2-го склада необходимо груз направить к 2-у потребителю (115 ед.), к 5-у потребителю (6 ед.)
Из 3-го склада необходимо груз направить к 1-у потребителю (72 ед.), к 5-у потребителю (86 ед.)
Из 4-го склада необходимо груз направить к 1-у потребителю (79 ед.), к 4-у потребителю (88 ед.)
Из 5-го склада необходимо весь груз направить к 4-у потребителю.
Потребность 3-го потребителя остается неудовлетворенной на 139 ед.
Оптимальный план является вырожденным, так как базисная переменная x63=0.
Потребность 5-го потребителя остается неудовлетворенной на 11 ед.
Оптимальный план является вырожденным, так как базисная переменная x65=0.
Вывод: научился решать транспортную задачу.

Лабораторная работа 6. АЛГОРИТМЫ НА ГРАФАХ

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 ДЛЯ ВЫПОЛНЕНИЯ:

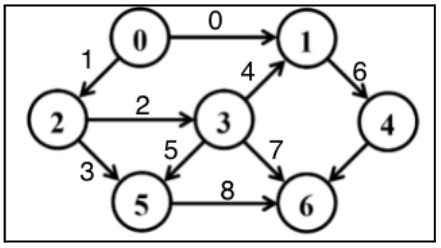
Задание 1. Ориентированный граф **G** взять в соответствии с вариантом. Представить его в отчете в виде матрицы смежности, матрицы инцидентности, списка смежных вершин.

9	

Матрица смежности:

Вершина\вершина	0	1	2	3	4	5	6
0	0	1	1	0	0	0	0
1	0	0	0	0	1	0	0
2	0	0	0	1	0	1	0
3	0	0	0	0	0	1	1
4	0	0	0	0	0	0	1
5	0	0	0	0	0	0	1
6	0	0	0	0	0	0	0

Пронумеруем ребра графа:



Матрица инцидентности:

Вершина\ребро	0	1	2	3	4	5	6	7	8
0	1	1	0	0	0	0	0	0	0
1	-1	0	0	0	-1	0	1	0	0
2	0	-1	1	1	0	0	0	0	0
3	0	0	-1	0	1	1	0	1	0
4	0	0	0	0	0	0	-1	0	0
5	0	0	0	-1	0	-1	0	0	1

6	0	0	0	0	0	0	0	-1	-1

Список смежных вершин:
0: {1, 2}
1: {4}
2: {3, 5}
3: {1, 5, 6}
4: {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.

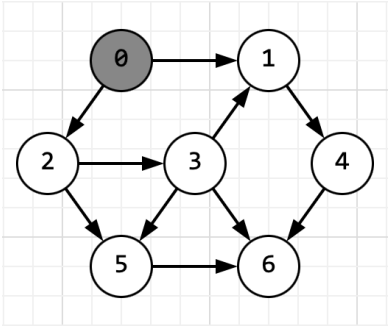
Обход графа в ширину (BFS)

Граф задан на изображении. Начальная вершина: 0.

Шаг	Очередь	Обрабатываем	Добавляем в очередь	Цвета (после шага)	Родители (P)
1	[0]	0	1, 2	Ч, С, С, Б, Б, Б, Б	-, 0, 0, -, -, -, -
2	[1, 2]	1	4	Ч, Ч, С, Б, С, Б, Б	-, 0, 0, -, 1, -, -
3	[2, 4]	2	3, 5	Ч, Ч, Ч, С, С, С, Б	-, 0, 0, 2, 1, 2, -
4	[4, 3, 5]	4	6	Ч, Ч, Ч, С, Ч, С, С	-, 0, 0, 2, 1, 2, 4
5	[3, 5, 6]	3	—	Ч, Ч, Ч, Ч, Ч, С, С	-, 0, 0, 2, 1, 2, 4
6	[5, 6]	5	—	Ч, Ч, Ч, Ч, Ч, Ч, С	-, 0, 0, 2, 1, 2, 4
7	[6]	6	—	Ч, Ч, Ч, Ч, Ч, Ч, Ч	-, 0, 0, 2, 1, 2, 4

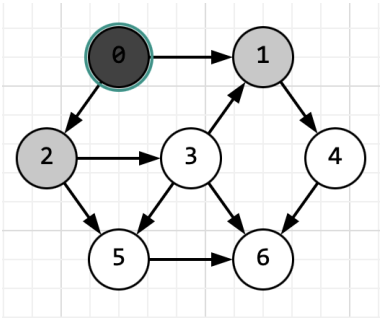
Шаг 1

Начинаем обход с вершины 0. Добавляем её в очередь.



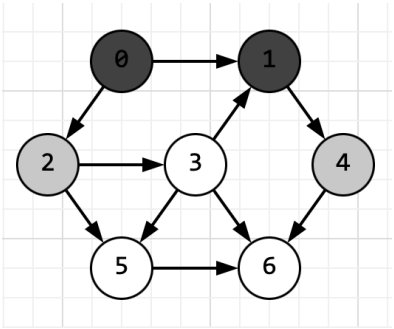
Шаг 2

Вершина 0 имеет смежные вершины 1 и 2. Добавляем их в очередь, помечаем серым, 0 — в чёрный.



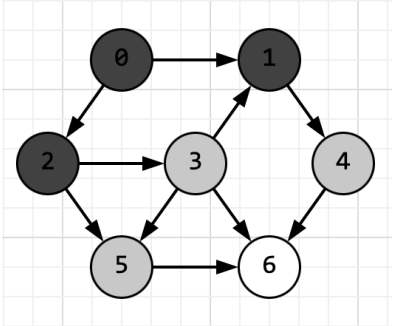
Шаг 3

Обрабатываем вершину 1. Она имеет смежную вершину 4, которую добавляем.



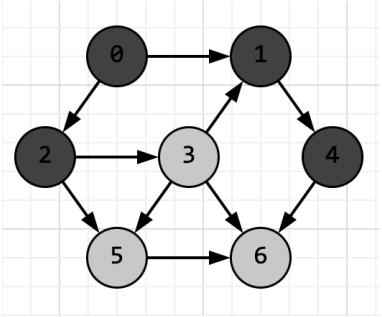
Шаг 4

Обрабатываем вершину 2. Смежные — 3 и 5. Добавляем.



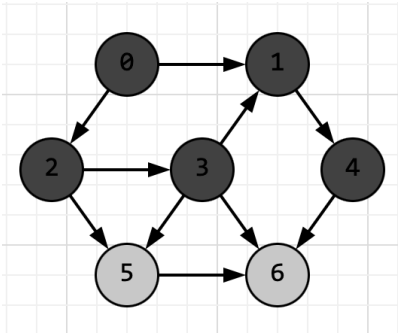
Шаг 5

Обрабатываем вершину 4. Смежная — 6. Добавляем.



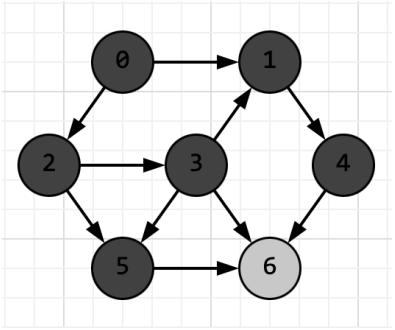
Шаг 6

Обрабатываем вершину 3. Все ее соседи уже посещены.



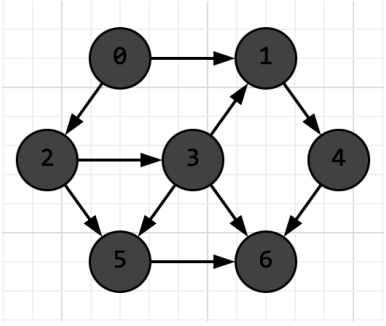
Шаг 7

Обрабатываем вершину 5. Сосед 6 уже посещён.



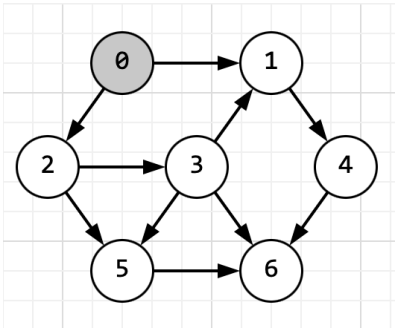
Шаг 8

Обрабатываем вершину 6. Смежных нет.



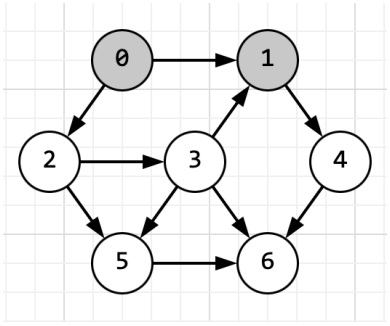
Обход графа в глубину (DFS)

Шаг	Стек	Обрабатываем	Переходим в	Цвета	Родители
1	[0]	0	1	С, Б, Б, Б, Б, Б, Б	-, -, -, -, -, -, -
2	[0,1]	1	4	С, С, Б, Б, Б, Б, Б	-, 0, -, -, -, -, -
3	[0,1,4]	4	6	С, С, Б, Б, С, Б, Б	-, 0, -, -, 1, -, -
4	[0,1,4,6]	6	—	С, С, Б, Б, С, Б, С	-, 0, -, -, 1, -, 4
5	[0,1,4]	6	—	С, С, Б, Б, С, Б, Ч	-, 0, -, -, 1, -, 4
6	[0,1]	4	—	С, С, Б, Б, Ч, Б, Ч	-, 0, -, -, 1, -, 4
7	[0]	1	—	С, Ч, Б, Б, Ч, Б, Ч	-, 0, -, -, 1, -, 4
8	[0]	0	2	С, Ч, Б, Б, Ч, Б, Ч	-, 0, -, -, 1, -, 4
9	[0,2]	2	3	С, Ч, С, Б, Ч, Б, Ч	-, 0, 0, -, 1, -, 4
10	[0,2,3]	3	5	С, Ч, С, С, Ч, Б, Ч	-, 0, 0, 2, 1, -, 4
11	[0,2,3,5]	5	—	С, Ч, С, С, Ч, С, Ч	-, 0, 0, 2, 1, 3, 4
12	[0,2,3]	5	—	С, Ч, С, С, Ч, Ч, Ч	-, 0, 0, 2, 1, 3, 4
13	[0,2]	3	—	С, Ч, С, Ч, Ч, Ч, Ч	-, 0, 0, 2, 1, 3, 4
14	[0]	2	—	С, Ч, Ч, Ч, Ч, Ч, Ч	-, 0, 0, 2, 1, 3, 4
15	[]	0	—	Ч, Ч, Ч, Ч, Ч, Ч, Ч	-, 0, 0, 2, 1, 3, 4



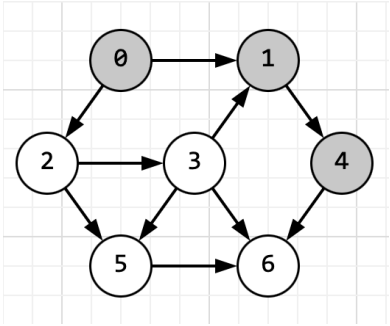
Шаг 1

Начинаем обход с вершины 0. Переходим к её первому соседу — 1.



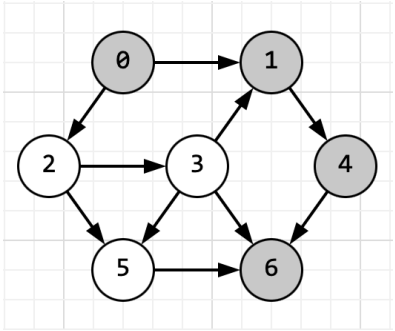
Шаг 2

Вершина 1 ведёт к 4. Переходим к 4.



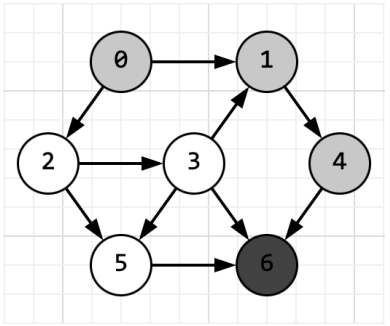
Шаг 3

Вершина 4 ведёт к 6. Переходим к 6.



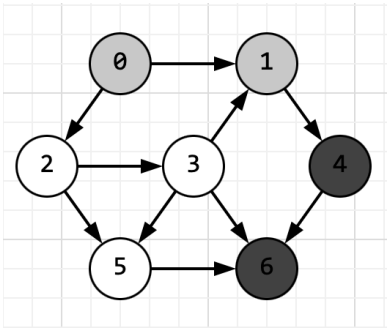
Шаг 4

Вершина 6 не имеет соседей. Возвращаемся назад и добавляем 6 в результат.



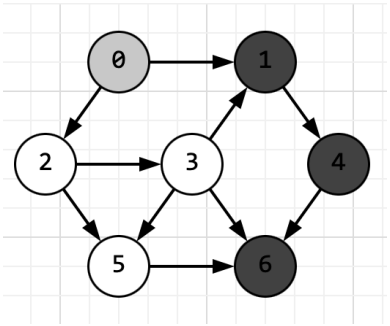
Шаг 5

Возвращаемся к вершине 4. Все соседи посещены. Добавляем 4 в результат.



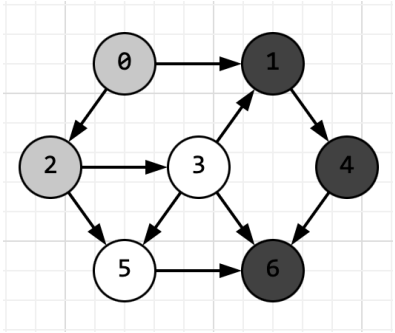
Шаг 6

Возвращаемся к вершине 1. Все соседи посещены. Добавляем 1 в результат.



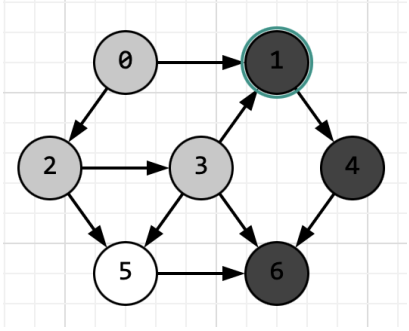
Шаг 7

Возвращаемся к вершине 0 и переходим ко второму соседу — 2.



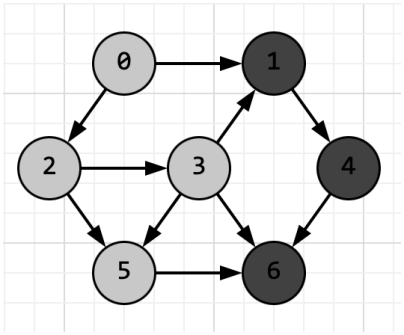
Шаг 8

Вершина 2 ведёт к 3. Переходим к 3.



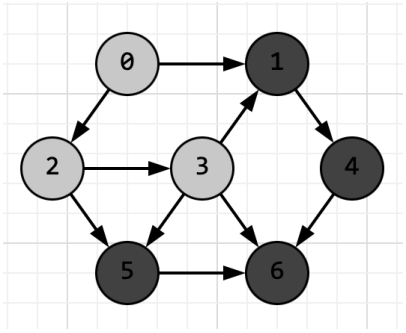
Шаг 9

Вершина 3 ведёт к 1, но она уже посещена. Далее — 5. Переходим к 5.



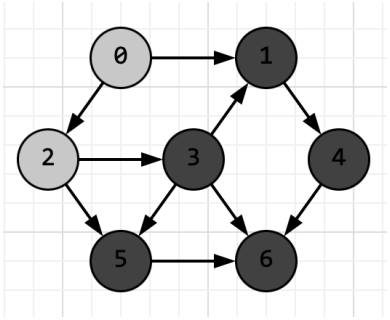
Шаг 10

Вершина 5 ведёт к 6, но она уже посещена. Добавляем 5 в результат.



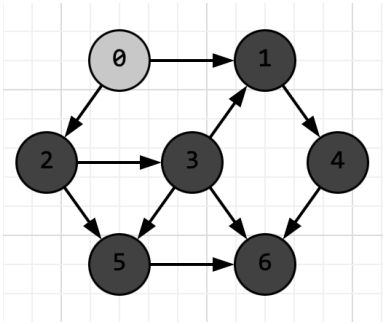
Шаг 11

Возвращаемся к вершине 3. Остался сосед 6, но он уже посещён. Добавляем 3 в результат.



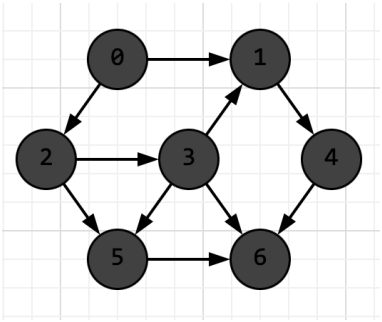
Шаг 12

Возвращаемся к вершине 2. Все соседи посещены. Добавляем 2 в результат.



Шаг 13

Возвращаемся к вершине 0. Все соседи посещены. Добавляем 0 в результат.



Поиск в глубину: 6 4 1 5 3 2 0

Задание 3. Осуществить программную реализацию алгоритмов на C++. Разработать структуры **AMatrix** и **AList** 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.

Результаты преобразования функций из одного представления в другое и обратно, а также функции поиска в ширину представлен на рисунке 1.

```
Матрица смежности:
0 1 1 0 0 0 0
0 0 0 0 1 0 0
0 0 0 1 0 1 0
0 1 0 0 0 1 1
0 0 0 0 0 0 1
0 0 0 0 0 0 1
0 0 0 0 0 0 0

Списки смежных вершин:
0: 1 2
1: 4
2: 3 5
3: 1 5 6
4: 6
5: 6
6:

Преобразованная матрица смежности:
0 1 1 0 0 0 0
0 0 0 0 1 0 0
0 0 0 1 0 1 0
0 1 0 0 0 1 1
0 0 0 0 0 0 1
0 0 0 0 0 0 1
0 0 0 0 0 0 0

Преобразованные списки смежных вершин:
0: 1 2
1: 4
2: 3 5
3: 1 5 6
4: 6
5: 6
6:

Поиск в ширину:
0 1 2 4 3 5 6
```

Рисунок 1

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

Результат выполнения функции поиска в глубину представлен на рисунке 2.

```
Поиск в глубину:
6 4 1 5 3 2 0
```

Рисунок 2

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

Результат выполнения топологической сортировки представлен на рисунке 3.

```
Топологическая сортировка:
6 4 1 5 3 2 0
```

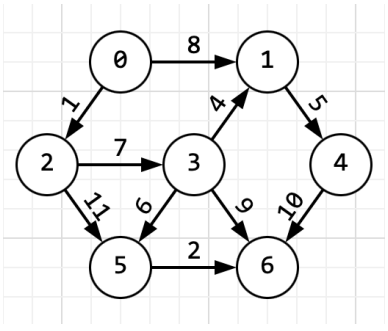
Рисунок 3

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

Веса ребер принять:

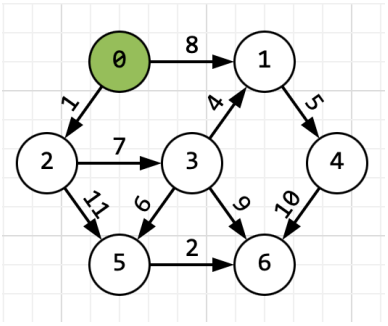
W:

$W(e_{0,1})=8$; $W(e_{1,0})=5$;
$W(e_{0,2})=1$; $W(e_{2,0})=3$;
$W(e_{0,3})=2$; $W(e_{3,0})=8$;
$W(e_{1,3})=11$; $W(e_{3,1})=4$;
$W(e_{1,4})=5$; $W(e_{4,1})=3$;
$W(e_{2,3})=7$; $W(e_{3,2})=9$;
$W(e_{2,5})=11$; $W(e_{5,2})=10$;
$W(e_{4,3})=4$; $W(e_{3,4})=1$;
$W(e_{4,6})=10$; $W(e_{6,4})=2$;
$W(e_{5,6})=2$; $W(e_{6,5})=6$;
$W(e_{5,3})=3$; $W(e_{3,5})=6$;
$W(e_{6,3})=7$; $W(e_{3,6})=9$;



Шаг 1

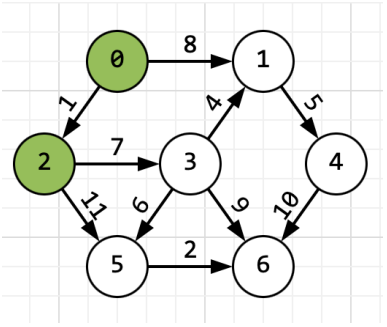
Выберем в качестве начальной точки вершину номер 0.



Суммарная длина дерева = 0

Шаг 2

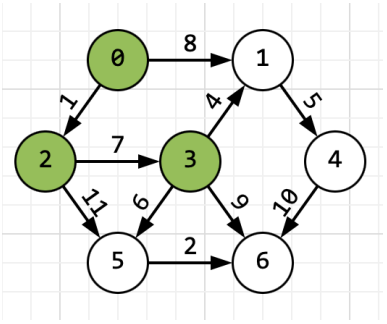
Выбираем ребро, которое является смежным для вершины 0, с самым минимальным весом и так, чтобы вершина была не посещенной.



Суммарная длина дерева = 1

Шаг 3

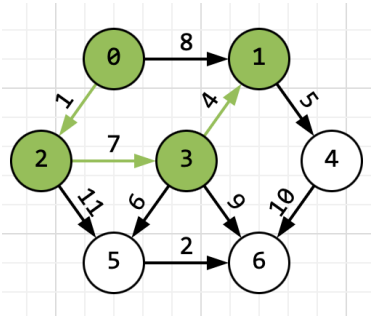
Выбираем ребро, которое является смежным для всех выделенных вершин, с самым минимальным весом и так, чтобы вершина была не посещенной.



Суммарная длина дерева = 8

Шаг 4

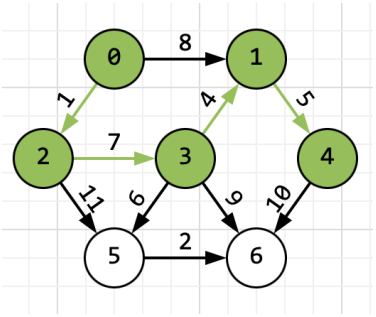
Выбираем ребро, которое является смежным для всех выделенных вершин, с самым минимальным весом и так, чтобы вершина была не посещенной.



Суммарная длина дерева = 12

Шаг 5

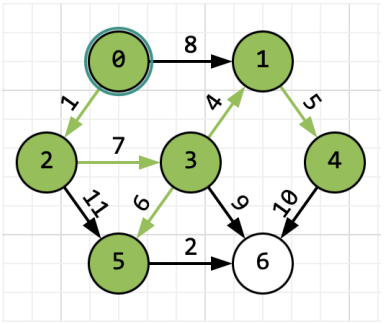
Выбираем ребро, которое является смежным для всех выделенных вершин, с самым минимальным весом и так, чтобы вершина была не посещенной.



Суммарная длина дерева = 17

Шаг 6

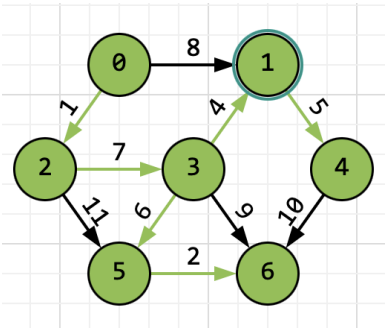
Выбираем ребро, которое является смежным для всех выделенных вершин, с самым минимальным весом и так, чтобы вершина была не посещенной.



Суммарная длина дерева = 23

Шаг 7

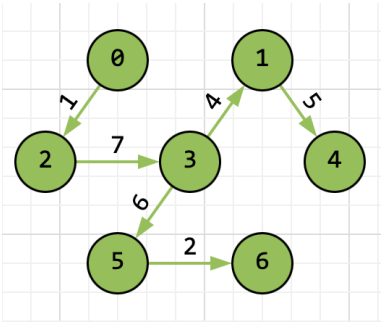
Выбираем ребро, которое является смежным для всех выделенных вершин, с самым минимальным весом и так, чтобы вершина была не посещенной.



Суммарная длина дерева = 25

Шаг 8

Удаляем лишние ребра и получаем соответствующее остовное дерево.

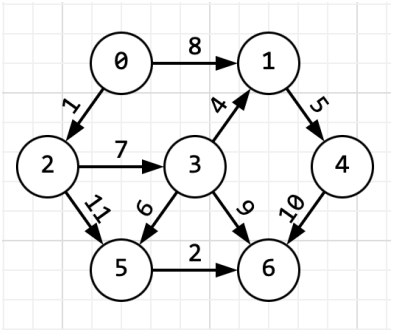


Суммарная длина дерева = 25

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

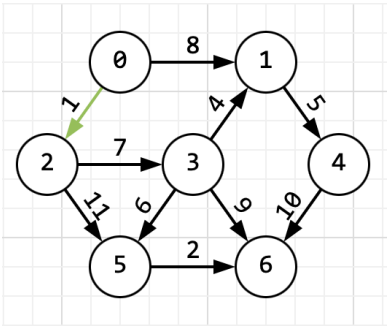
Шаг 1

Построим остовной подграф, содержащий только изолированные вершины.



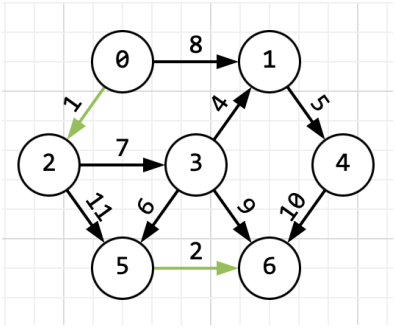
Шаг 2

Найдем ребро минимального веса и добавим его в граф.



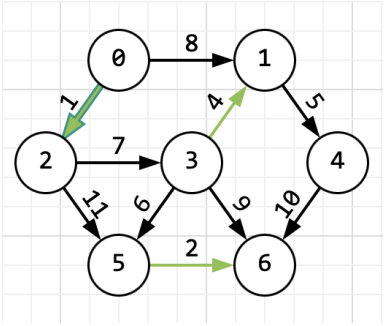
Шаг 3

Среди оставшихся ребер находим ребро с минимальным весом и смежное с хотя бы одной вершиной, которая еще не посещена(не имеет смежных ребер).



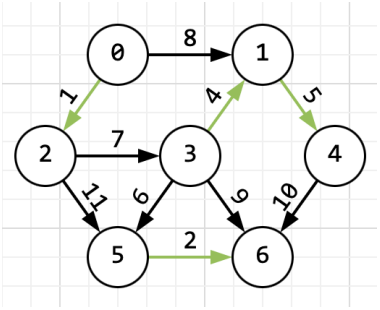
Шаг 4

Среди оставшихся ребер находим ребро с минимальным весом и смежное с хотя бы одной вершиной, которая еще не посещена(не имеет смежных ребер).



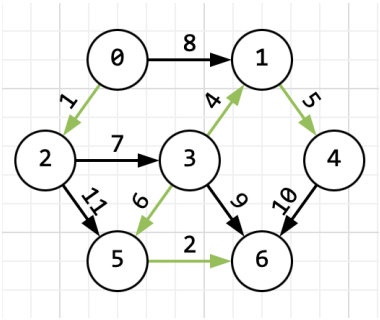
Шаг 5

Среди оставшихся ребер находим ребро с минимальным весом и смежное с хотя бы одной вершиной, которая еще не посещена(не имеет смежных ребер).



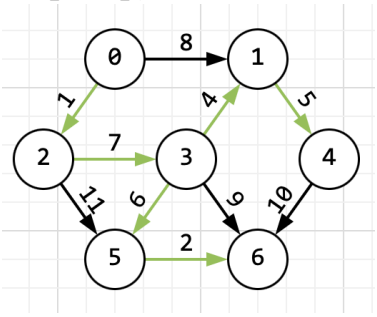
Шаг 6

Среди оставшихся ребер находим ребро с минимальным весом и смежное с хотя бы одной вершиной, которая еще не посещена(не имеет смежных ребер).



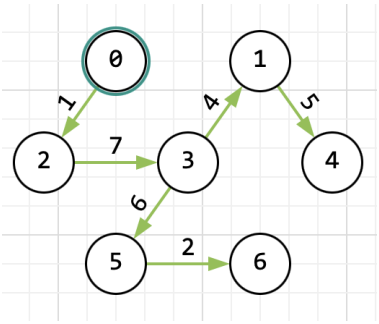
Шаг 7

Среди оставшихся ребер находим ребро с минимальным весом и смежное с хотя бы одной вершиной, которая еще не посещена(не имеет смежных ребер).



Шаг 8

Удаляем оставшиеся ненужные ребра и получаем остовное дерево.



Минимальный вес = 25.

Вывод: были освоены сущность и программная реализация: а) способов представления графов; б) алгоритмов поиска в ширину и глубину; в) алгоритма топологической сортировки графов. Разобран алгоритм Прима и алгоритм Крускала.

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 3, 9, 15	«Создание банковского приложения»	65 дней

Задание 1. Структурное планирование - Задание 2. Календарное планирование.

Код операции	Наименование операции	Предшествующие операции	t
I. АНАЛИЗ			
Z1	Исследование рынка и конкурентов		3
Z2	Определение целевой аудитории	Z1	1

Z3	Сбор и анализ требований	Z2	2
II. ПРОЕКТИРОВАНИЕ			
Z4	Разработка архитектуры приложения	Z3	6
Z5	Проектирование базы данных	Z4	5
Z6	Проектирование интерфейса	Z4	4
III. РАЗРАБОТКА			
Z7	Разработка серверной части	Z4, Z5	8
Z8	Разработка клиентской части	Z6	7
Z9	Интеграция модулей	Z7, Z8	4
IV. ТЕСТИРОВАНИЕ И ПУБЛИКАЦИЯ			
Z10	Тестирование функционала и безопасности	Z9	6
Z11	Подготовка документации и обучение	Z10	3

Z12	Публикация в Google Play / App Store	Z11	1

Общее время:

3 (Z1) + 1 (Z2) + 2 (Z3) + 6 (Z4) + 5 (Z5) + 4 (Z6) + 8 (Z7) + 7 (Z8) + 4 (Z9) + 6 (Z10) + 3 (Z11) + 1 (Z12) = 50 дней

Корректировка под 65 дней:

Увеличить время на разработку серверной части (Z7) с 8 до 15 дней.

Увеличить время на тестирование (Z10) с 6 до 10 дней.

Добавить этап "Доработка по отзывам (Z13)" после публикации (3 дня).

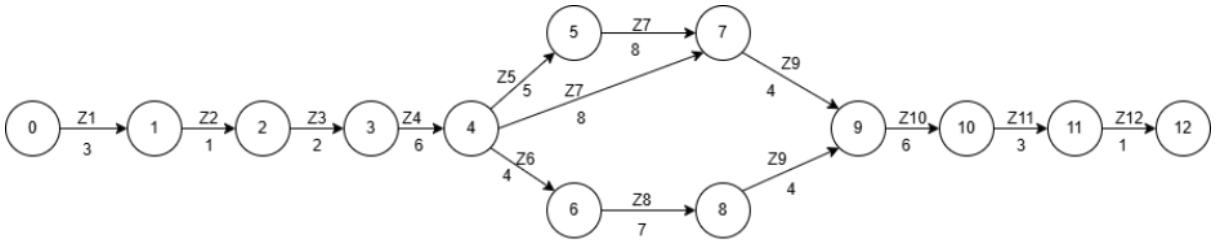
Итого: 3 + 1 + 2 + 6 + 5 + 4 + 15 + 7 + 4 + 10 + 3 + 1 + 3 = 65 дней.

Задание 2. Календарное планирование

Скорректированная таблица с новыми временными рамками:

Код операции	Время (дни)
Z1	3
Z2	1
Z3	2
Z4	6
Z5	5
Z6	4
Z7	15
Z8	7
Z9	4
Z10	10
Z11	3
Z12	1
Z13	3

Задание 3. Сетевой график и критический путь



Критический путь:

Z1 → Z2 → Z3 → Z4 → Z7 → Z9 → Z10 → Z11 → Z12 → Z13

Суммарное время: 3 + 1 + 2 + 6 + 15 + 4 + 10 + 3 + 1 + 3 = 48 дней

Операции критического пути:

Z1, Z2, Z3, Z4, Z7, Z9, Z10, Z11, Z12, Z13

Примечание: Критический путь меньше 65 дней, так как часть задач выполняется параллельно (например, Z5 и Z6).

Задание 4. Оптимизация

Ускорение разработки: Назначить дополнительных backend-разработчиков для сокращения Z7 (с 15 до 10 дней).

Автоматизация тестирования: Внедрить инструменты для уменьшения Z10 с 10 до 7 дней.

Гибкая публикация: Выпустить MVP без Z13, сократив срок до 62 дней.

Вывод: в процессе выполнения лабораторной работы были приобретены навыки сетевого планирования и составлениях сетевых графиков, также был приобретен опыт нахождения критического пути.

Лабораторная работа 8. Графический метод решения оптимизационных задач

Цель работы: Освоить решение задач графическим методом.
Задание для выполнения:

№9.

$$max \text{ и } min\ Z = x_1 + 6x_2$$
$$2x_1 + x_2 \geq 12$$
$$x_1 + 2x_2 \geq 12$$
$$x_1 \geq 2$$
$$x_2 \geq 3$$

1. Построение области допустимых решений

1.1 Прямая $2x_1 + x_2 = 12$

- Пересечение с осью X ($x_2=0$):
  $2x_1 = 12 \Rightarrow x_1=6 \rightarrow$ точка (6; 0)
- Пересечение с осью Y ($x_1=0$):
  $x_2=12 \rightarrow$ точка (0; 12)
- Неравенство $\geq \Rightarrow$ выбираем полуплоскость выше прямой.

1.2 Прямая $x_1 + 2x_2 = 12$

- Пересечение с осью X ($x_2=0$):
  $x_1=12 \rightarrow$ точка (12; 0)
- Пересечение с осью Y ($x_1=0$):
  $2x_2=12 \Rightarrow x_2=6 \rightarrow$ точка (0; 6)
- Неравенство $\geq \Rightarrow$ выбираем полуплоскость выше прямой.

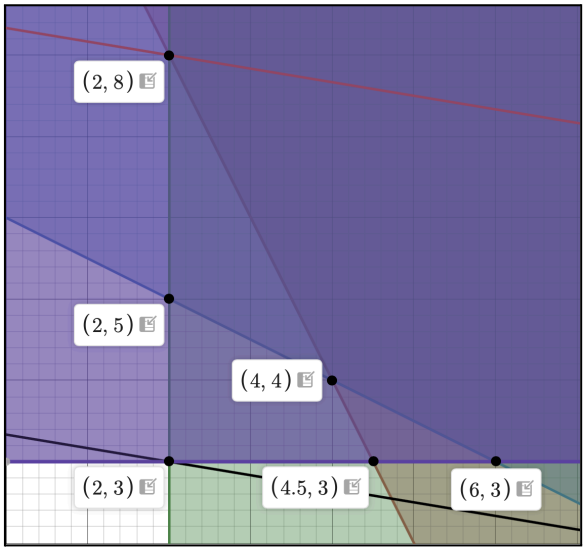
1.3 Прямая $x_1 = 2$

- Вертикальная прямая через $x_1=2$.
- Неравенство $\geq \Rightarrow$ выбираем полуплоскость правее прямой.

1.4 Прямая $x_2 = 3$

- Горизонтальная прямая через $x_2=3$.
- Неравенство $\geq \Rightarrow$ выбираем полуплоскость выше прямой.

График:



Область допустимых решений - пересечение всех полуплоскостей. Это многоугольник с вершинами:

- A (2; 8) - пересечение $x_1=2$ и $2x_1 + x_2=12$
- B (4; 4) - пересечение $2x_1 + x_2=12$ и $x_1 + 2x_2=12$
- C (6; 3) - пересечение $x_1 + 2x_2=12$ и $x_2=3$
- D (2; 3) - пересечение $x_1=2$ и $x_2=3$

2. Поиск max и min целевой функции $Z = x_1 + 6x_2$

2.1 Вычисление значений Z в вершинах:

- A (2;8): $Z = 2 + 6\times8 = 50$
- B (4;4): $Z = 4 + 6\times4 = 28$
- C (6;3): $Z = 6 + 6\times3 = 24$
- D (2;3): $Z = 2 + 6\times3 = 20$

2.2 Результаты:

- Максимальное значение: Zmax = 50 в точке A (2;8)
- Минимальное значение: Zmin = 20 в точке D (2;3)

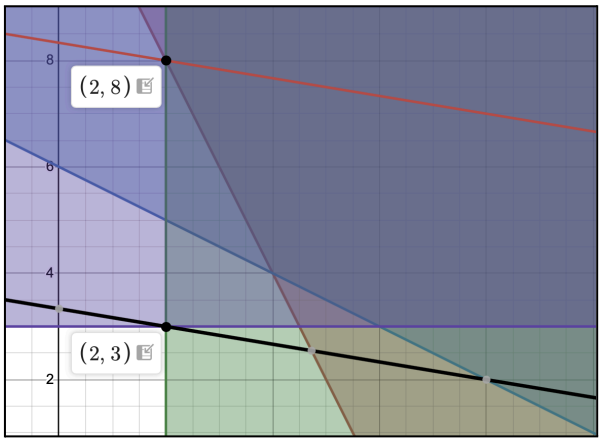
3. Графическое подтверждение

3.1 Целевая функция:

$Z = x_1 + 6x_2$ представляется семейством прямых с угловым коэффициентом -1/6.

3.2 Граничные положения:

- Для max Z: прямая проходит через точку A (2;8)
- Для min Z: прямая проходит через точку D (2;3)



Общий вид графика:

Вывод

В ходе решения задачи графическим методом:

- Найдена область допустимых решений в виде многоугольника
- Определены экстремальные значения целевой функции:
  - Максимальное: Zmax = 50 (в точке A (2;8))
  - Минимальное: Zmin = 20 (в точке D (2;3))