Учреждение образования

«Белорусский государственный технологический университет»

**Отчет по лабораторным работам по дисциплине**

**“Математическое программирование”**

Выполнила:

Студентка 2 курса 8 группы ИТ

Казакова Виктория Васильевна

Проверила Ромыш А.С.

2025 г.

**Оглавление**

[**Лабораторная работа №1. Вспомогательные функции** 3](#_Toc194395807)

[**Лабораторная работа 2. Комбинаторные алгоритмы решения оптимизационных задач** 7](#_Toc194395808)

[**Лабораторная работа №3. Метод ветвей и границ. Задача коммивояжера и методы ее решения.** 22](#_Toc194395809)

[**Лабораторная работа №4. Динамическое программирование** 30](#_Toc194395810)

[**Лабораторная работа №5.** **Транспортная задача** 41](#_Toc194395811)

[**Лабораторная работа №6. Алгоритмы на графах** 52](#_Toc194395812)

[**Лабораторная работа №7. Сетевые модели** 67](#_Toc194395813)

[**Лабораторная работа №8. Графический метод решения оптимизационных задач.** 71](#_Toc194395814)

# **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 показана в листинге 1, 2.

Листинг 1 – Код файла Auxil.cpp

|  |
| --- |
| #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 2 – Прототипы функций Auxil.h

|  |
| --- |
| #pragma once  #include <cstdlib>  namespace auxil  {  void start(); // старт генератора сл. чисел  double dget(double rmin, double rmax); // получить случайное число  int iget(int rmin, int rmax); // получить случайное число  }; |

Разработанные функции располагаются в файле **Auxil.cpp**,  а в файле **Auxil.h –** прототипы функций

**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.

Задание под номером 2 показано в листинге 3.

Листинг 3 – вызов функции и измерение времени выполнения lab1.cpp

|  |
| --- |
| #include "stdafx.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;  } |

Результат работы программы представлен на рисунке 1.1.

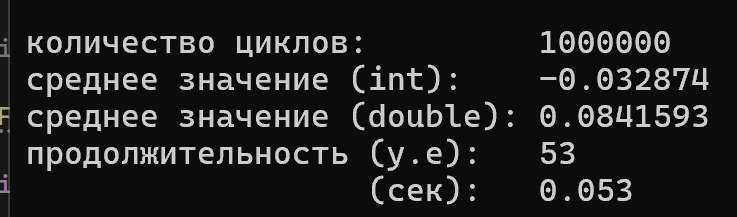


Рисунок 1.1 Результаты работы программы

**Задание 3.**

Проведите необходимые эксперименты и постройте график зависимости (Excel) продолжительности процесса вычисления от количества циклов в примере 2.

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

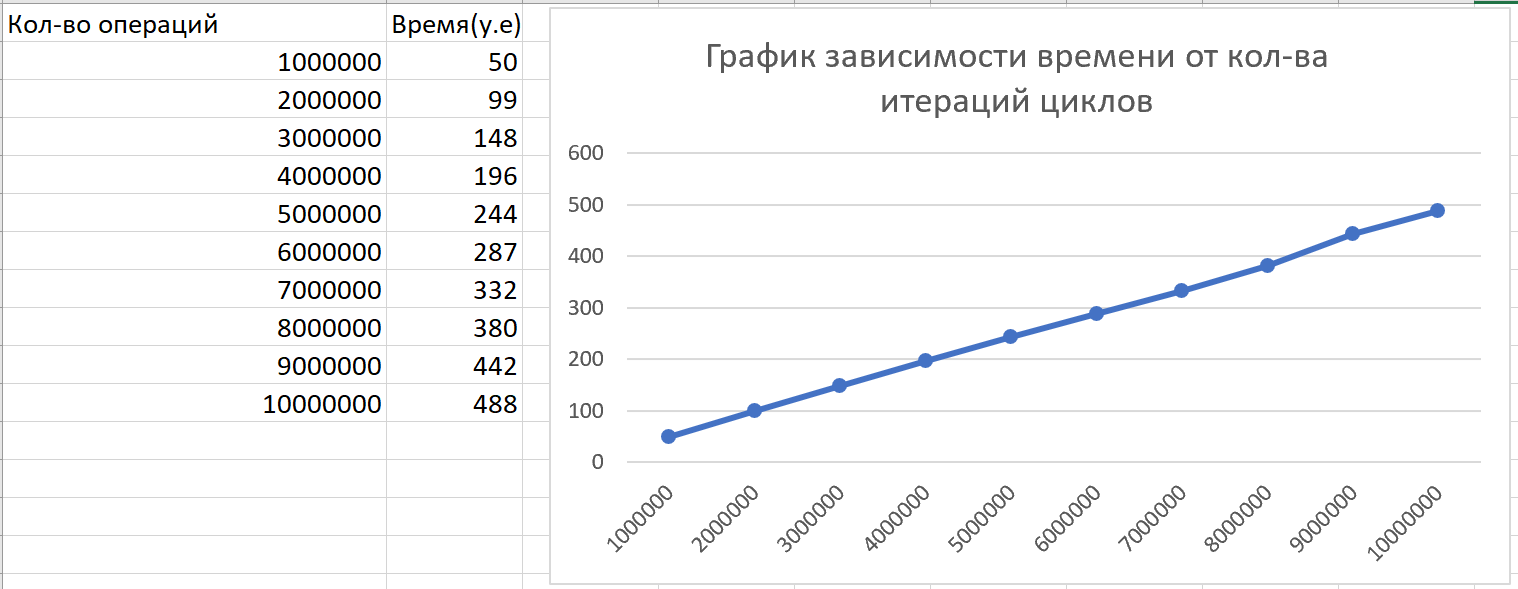


Рисунок 1.2 Результаты измерений и их график

**Вывод**: скорость выполнения программы линейно зависит от количества итераций цикла.

Реализация Фибоначчи представлена в листенге 4.

|  |
| --- |
| int Fib(int i)  {  if (i < 1)  return 0;  if (i == 1)  return 1;  return Fib(i - 1) + Fib(i - 2);  }  for (int n = 25; n < 43; n++)  {  t1 = clock();  int num = Fib(n);  t2 = clock();  std::cout << std::endl << n << "-е число Фибоначчи " << (t2 - t1) << " у.е.";  } |

Измерение скорости выполнения функции подсчета n-го числа Фибоначчи.

Результаты измерений и соответствующий график приведены на рисунке 1.3.

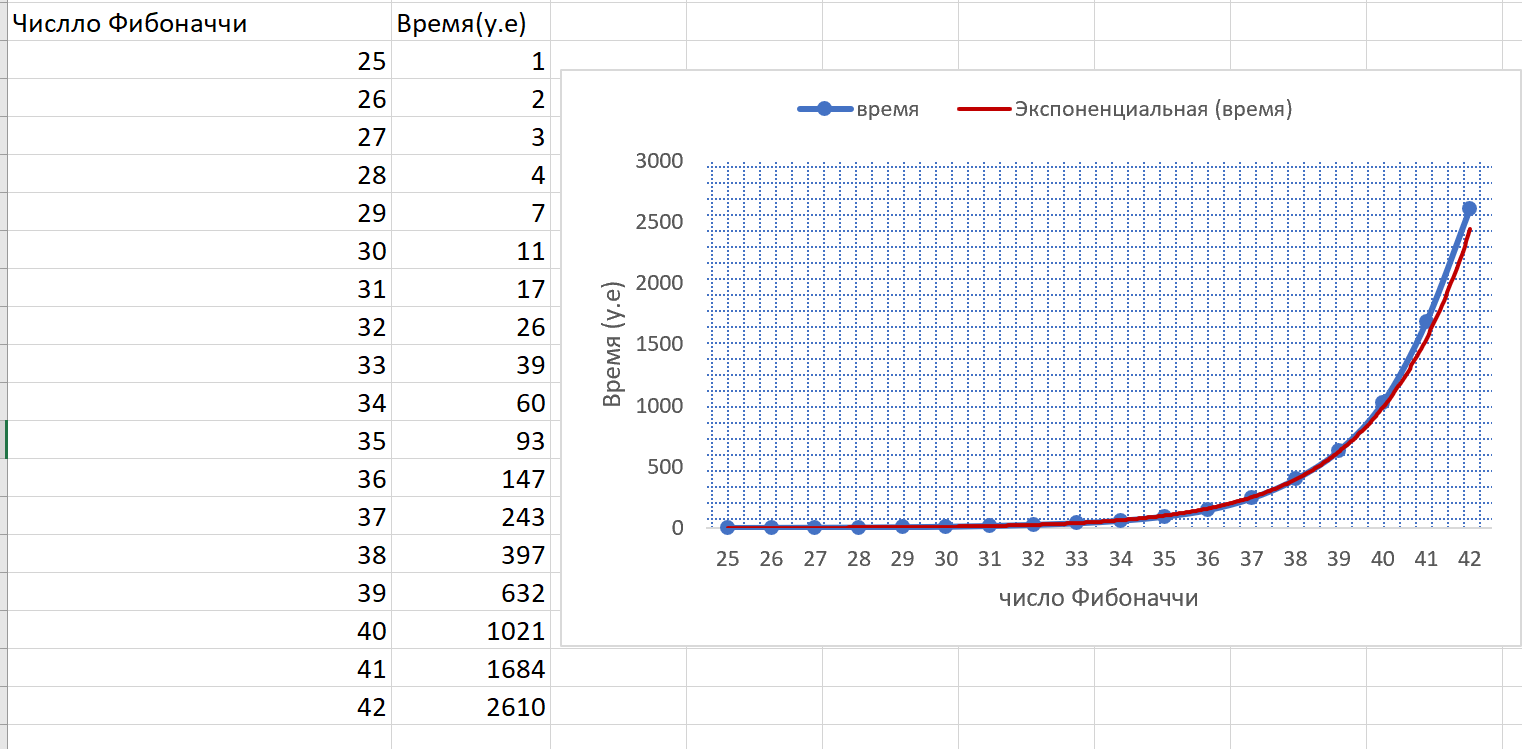


Рисунок 1.3. Результаты измерений и их график

**Вывод**: скорость выполнения программы экспоненциально зависит от порядкового номера числа Фибоначчи. Такая зависимость обусловлена особенностями алгоритма расчета последовательности чисел Фибоначчи.

**Вывод:** В лабораторной работе №1 были приобретены навыки составления и отладки программ с использованием пользовательских функций для замера продолжительности процесса вычисления. Этот процесс позволяет создавать эффективные и оптимизированные программы, которые максимально ускоряют вычисления и снижают временные затраты. При разработке пользовательских функций для замера продолжительности процесса вычисления важно учитывать особенности конкретной задачи и требования программы. Некоторые процессы могут выполняться весьма быстро и требовать высокой точности измерения времени, в то время как другие могут быть длительными и требовать более грубой оценки.

# **Лабораторная работа 2. Комбинаторные алгоритмы решения оптимизационных задач**

**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.

**Ход работы**

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

Был разобран генератор подмножества заданного множества из методического пособия и представлен код файла main.cpp, Combi.cpp представлены в листингах 1.1, 1.2 соответственно.

Листинг 1.1 – Код файла main.cpp

|  |
| --- |
| #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;  } |

Листинг 1.2 – Код файла 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);  };  }; |

Так же был реализован заголовочный файл для генератора подмножества заданного множества, который представлен в листинге 1.3.

Листинг 1.3 – Код заголовочного файла 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(кол-во эл-ов исх. мн-ва)  short getfirst(); // сформ. массив индексов по битовой маске  short getnext(); // ++маска и сформировать массив индексов  short ntx(short i); // получить i-й элемент массива индексов  unsigned \_\_int64 count(); // вычислить общее кол-во подмножеств  void reset(); // сбросить генератор, начать сначала  };  }; |

На рисунке 1 представлен вывод в консоль генератора подмножества заданного множества.

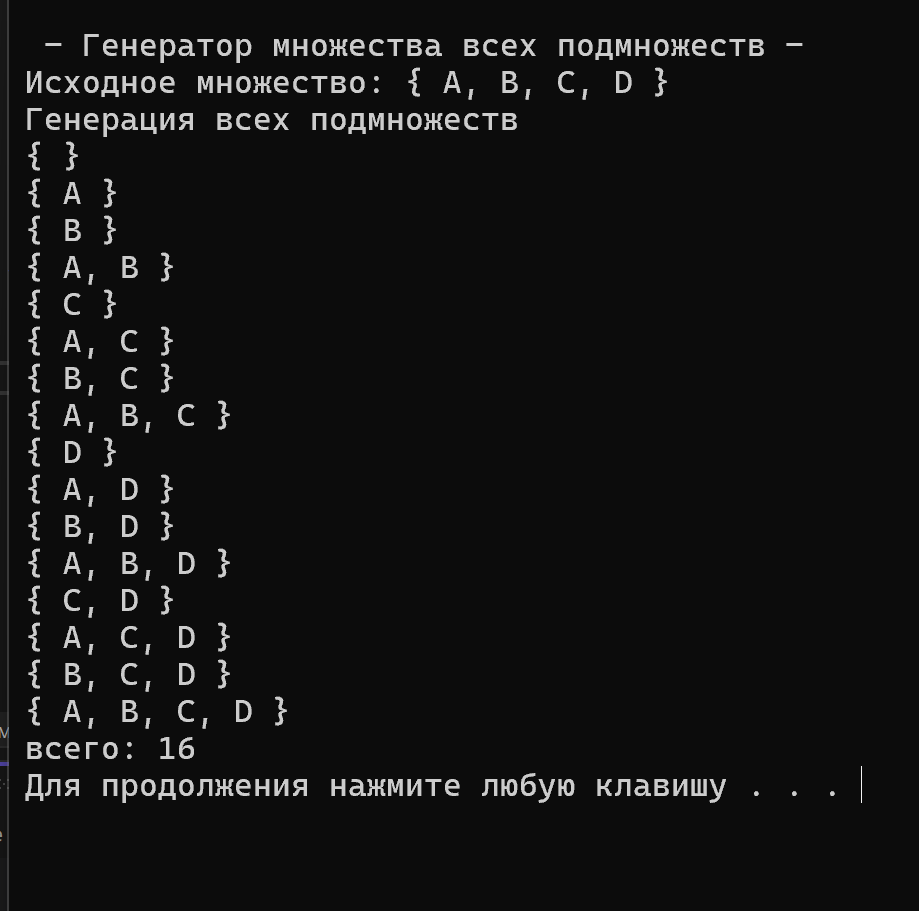


Рисунок 1- Результат выполнения программы

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

Был разобран генератор сочетаний из методического пособия и представлен код файла Main2.cpp, Combi.cpp представлены в листингах 2.1, 2.2 соответственно.

Листинг 2.1 – Код файла Main2.cpp

|  |
| --- |
| #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 ";  combi2::xcombination xc(sizeof(AA) / 2, 3);  std::cout << "из " << xc.n << " по " << xc.m;  int n = xc.getfirst();  while (n >= 0)  {  std::cout << std::endl << xc.nc << ": { ";  for (int i = 0; i < n; i++)  std::cout << AA[xc.ntx(i)] << ((i < n - 1) ? ", " : " ");  std::cout << "}";  n = xc.getnext();  };  std::cout << std::endl << "всего: " << xc.count() << std::endl;  system("pause");  return 0;  } |

Листинг 2.2 – Код файла Combi.cpp

|  |
| --- |
| #include "Combi.h"  #include <algorithm>  namespace combi2  {  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;  };  }; |

Так же был реализован заголовочный файл для генератора сочетаний, который представлен в листинге 2.3.

Листинг 2.3 – Код файла Combi.h

|  |
| --- |
| #pragma once  namespace combi2  {  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  };  } |

На рисунке 2 представлен вывод в консоль генератора сочетаний.

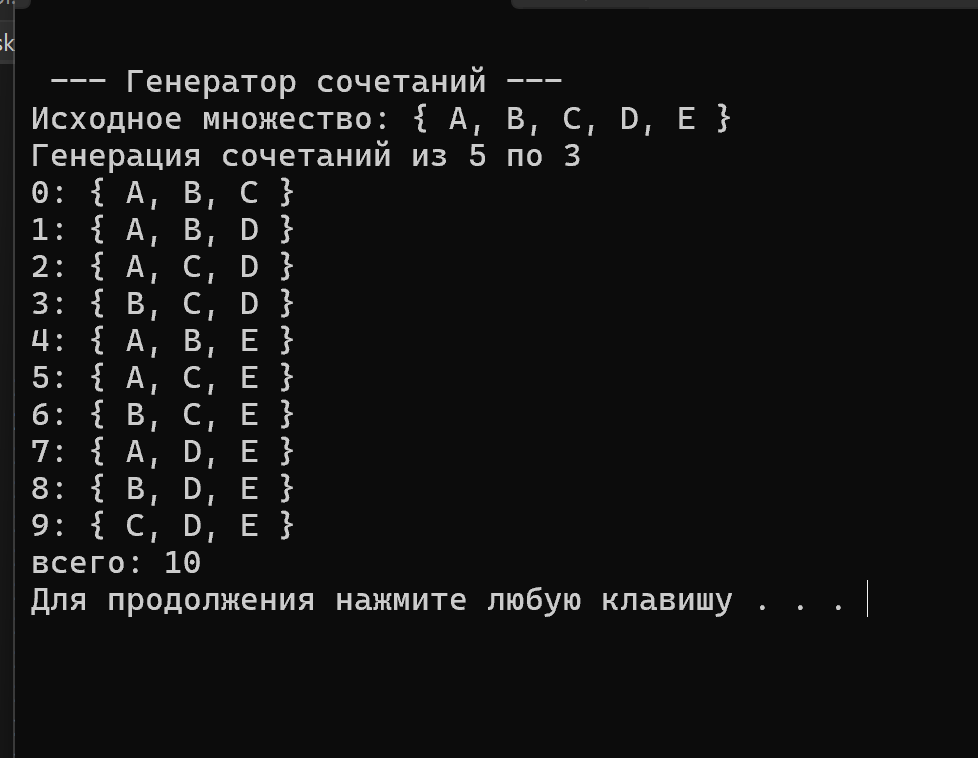


Рисунок 2- Результат работы программы

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

Был разобран генератор перестановок из методического пособия и представлен код реализации заголовочного файла Combi.h представлен в листинге 3.1.

Листинг 3.1 – Код файла Combi.h

|  |
| --- |
| #pragma once  namespace combi3  {  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 (кол-во эл-ов исх. мн-ва)  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  };  }; |

В листинге 3.2 представлен шаблон структуры генератора перестановок, а в листинге 3.3 реализация генератора.

Листинг 3.2 – Код файла Combi.cpp, который представляет собой шаблон

|  |
| --- |
| #include "Combi.h"  #include <algorithm>  #define NINF ((short)0x8000)  namespace combi3  {  permutation::permutation(short n)  {  this->n = n;  this->sset = new short[n];  this->dart = new bool[n];  this->reset();  };  void permutation::reset()  {  this->getfirst();  };  \_\_int64 permutation::getfirst()  {  this->np = 0;  for (int i = 0; i < this->n; i++)  {  this->sset[i] = i; this->dart[i] = L;  };  return (this->n > 0) ? this->np : -1;  };  \_\_int64 permutation::getnext() //  {  \_\_int64 rc = -1;  short maxm = NINF, idx = -1;  for (int i = 0; i < this->n; i++)  {  if (i > 0 &&  this->dart[i] == L &&  this->sset[i] > this->sset[i - 1] &&  maxm < this->sset[i]) maxm = this->sset[idx = i];  if (i < (this->n - 1) &&  this->dart[i] == R &&  this->sset[i] > this->sset[i + 1] &&  maxm < this->sset[i]) maxm = this->sset[idx = i];  };  if (idx >= 0)  {  std::swap(this->sset[idx],  this->sset[idx + (this->dart[idx] == L ? -1 : 1)]);  std::swap(this->dart[idx],  this->dart[idx + (this->dart[idx] == L ? -1 : 1)]);  for (int i = 0; i < this->n; i++)  if (this->sset[i] > maxm) this->dart[i] = !this->dart[i];  rc = ++this->np;  }  return rc;  };  short permutation::ntx(short i) { return this->sset[i]; };  unsigned \_\_int64 fact(unsigned \_\_int64 x) { return (x == 0) ? 1 : (x \* fact(x - 1)); };  unsigned \_\_int64 permutation::count() const { return fact(this->n); };  } |

Листинг 3.3 – Код файла Main3.cpp

|  |
| --- |
| #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 ";  combi3::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;  } |

На рисунке 3 представлен вывод в консоль генератора перестановок

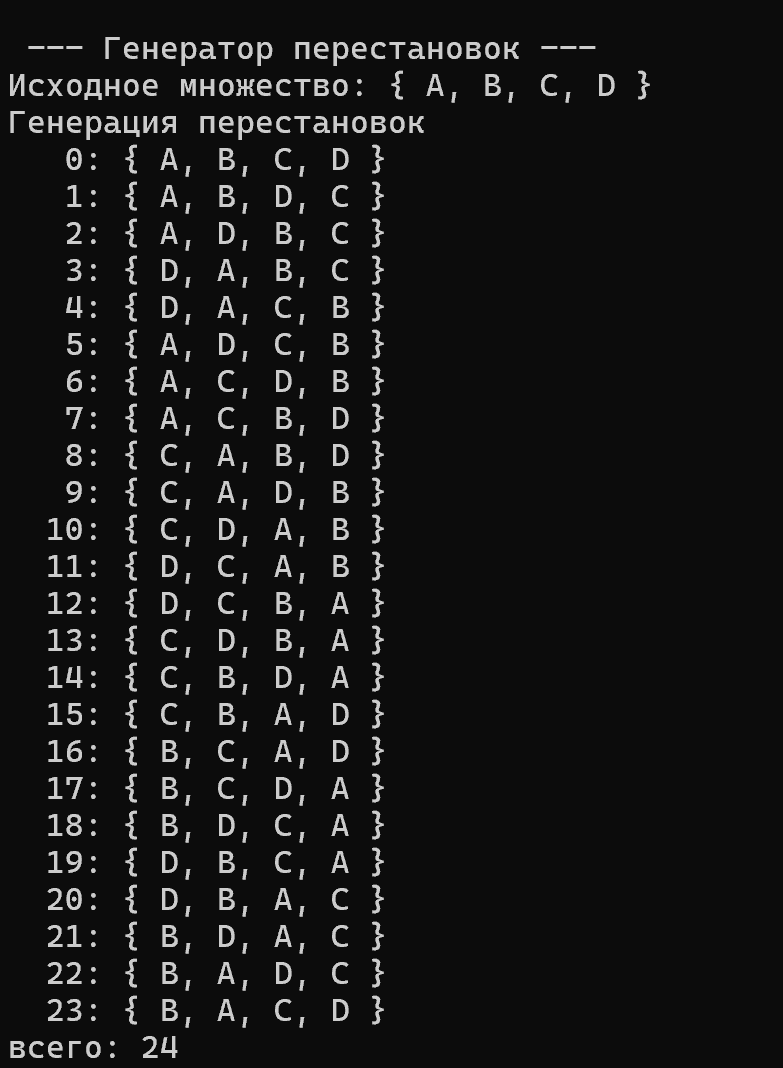


Рисунок 3- Результат работы программы

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

Был разобран генератор размещений из методического пособия и представлен код реализации заголовочного файла Combi.h представлен в листинге 4.1.

Листинг 4.1 – Код файла Combi.h

|  |
| --- |
| // Combi4.h  #pragma once  namespace combi4  {  struct xcombination // генератор сочетаний (эвристика)  {  short n, // кол-во 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(); // сформ. первый массив индексов  short getnext(); // сформ. следующий массив индексов  short ntx(short i); // получить i-й эл-т массива индексов  unsigned \_\_int64 nc; // номер сочетания 0,..., count()-1  unsigned \_\_int64 count() const; // вычислить кол-во 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 (кол-во эл-ов исх. мн-ва)  void reset(); // сбросить генератор, начать сначала  \_\_int64 getfirst(); // сформ. первый массив индексов  \_\_int64 getnext(); // сформ. 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  };  } |

В листинге 4.2 представлен шаблон структуры генератора размещений, а в листинге 4.3 реализация генератора.

Листинг 4.2 – Код файла Combi.cpp

|  |
| --- |
| #include "Combi.h"  #include <algorithm>  #define NINF ((short)0x8000)  namespace combi4  {  unsigned \_\_int64 fact(unsigned \_\_int64 x) { return(x == 0) ? 1 : (x \* fact(x - 1)); };  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;  };  short permutation::ntx(short i) { return this->sset[i]; };  unsigned \_\_int64 permutation::count() const { return fact(this->n); };  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 accomodation::count() const {  return (this->n >= this->m) ? fact(this->n) / fact(this->n - this->m) : 0;  };  } |

Листинг 4.3 – Код файла Main.cpp

|  |
| --- |
| #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;  combi4::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;  } |

На рисунке 4 представлен вывод в консоль генератора размещений.

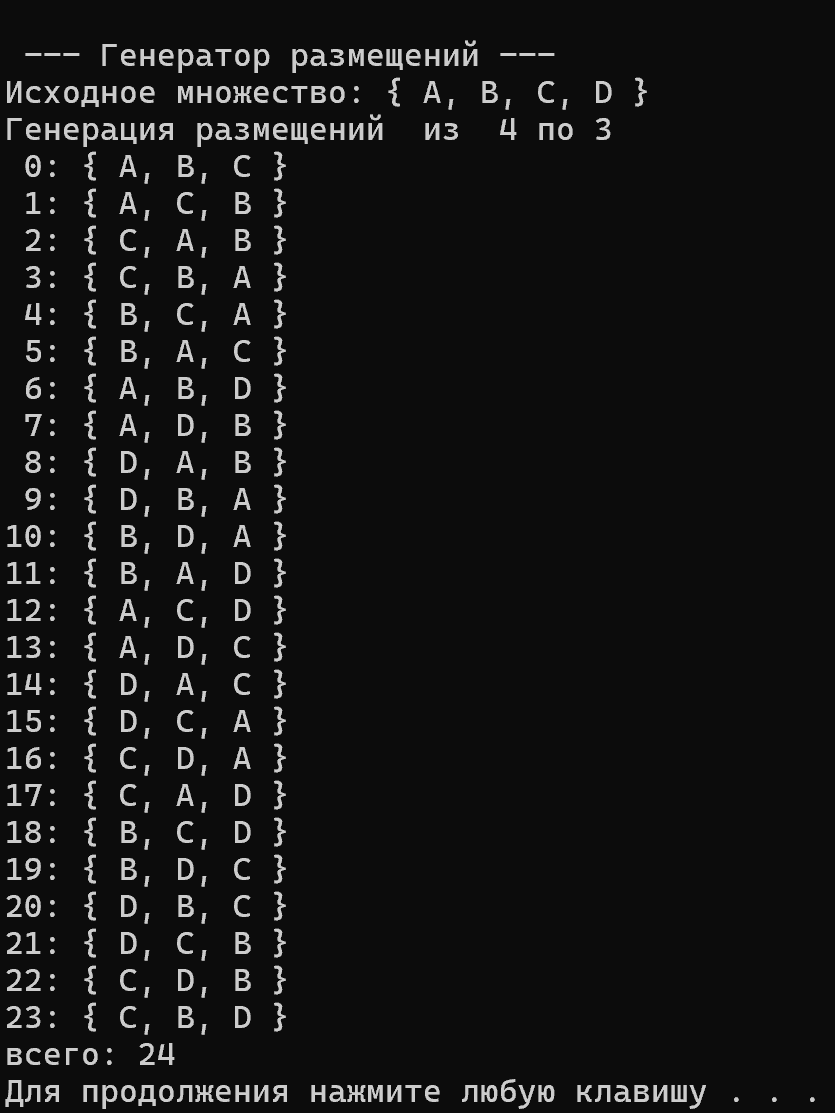


Рисунок 4 - Результат работы программы

**Задание 5.** Решить в соответствии с вариантом задачу и результат занести в отчет (Вариант распределяется по списку).

3, 7, 11, 15) 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.);

Реализация данной задачи представлена в листингах

Листинг 5.1 - Код файла Boat.h

|  |
| --- |
| #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  ); |

Листинг 5.2 - Код файла Boat.cpp

|  |
| --- |
| #include "Boat.h"  #include "Perestanovka.h"  #include "Sochetania.h"  namespace boatfnc  {  int calcv(sochetan::xcombination s, const int v[])  {  int rc = 0;  for (int i = 0; i < s.m; i++) rc += v[s.ntx(i)];  return rc;  };  int calcc(sochetan::xcombination s, const int c[])  {  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)  {  for (int i = 0; i < m; i++) r1[i] = r2[i];  };  }  int boat(  int V,  short m,  short n,  const int v[],  const int c[],  short r[]  )  {  sochetan::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;  }; |

Листинг 5.3 - Код файла Main\_boat.cpp

|  |
| --- |
| #include <iostream>  #include <iomanip>  #include "Boat.h"  #define NN 25  #define MM 5  int main()  {  setlocale(LC\_ALL, "rus");  int V = 1500,  v[] = { 100, 200, 300, 400, 500, 600, 700, 800, 900, 150, 250, 350, 450, 550, 650, 750, 850, 120, 220, 320, 420, 520, 620, 720, 820 },  c[NN] = { 10, 15, 20, 25, 30, 35, 40, 45, 50, 55, 60, 65, 70, 75, 80, 85, 90, 95, 100, 105, 110, 115, 120, 125, 150 };  short r[MM];  int cc = boat(  V, // [in] Ограничение по весу  MM, // [in] Количество мест для контейнеров  NN, // [in] Число контейнеров  v, // [in] Вес каждого контейнера  c, // [in] Масса содержимого каждого контейнера  r // [out] Результат: номера выбранных контейнеров  );  std::cout << std::endl << "- Задача о загрузке контейнеров на судне";  std::cout << std::endl << "- Число возможных контейнеров : " << NN;  std::cout << std::endl << "- Количество мест для контейнеров: " << MM;  std::cout << std::endl << "- Ограничение по 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 << "- Масса содержимого : ";  for (int i = 0; i < NN; i++) std::cout << std::setw(3) << c[i] << " ";  std::cout << std::endl << "- Выбранные контейнеры (0,1,...,m-1): ";  for (int i = 0; i < MM; i++) std::cout << r[i] << " ";  std::cout << std::endl << "- Масса содержимого выбранных контейнеров: " << 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;  } |

Результат выполнения программы представлен на рисунке 5.1

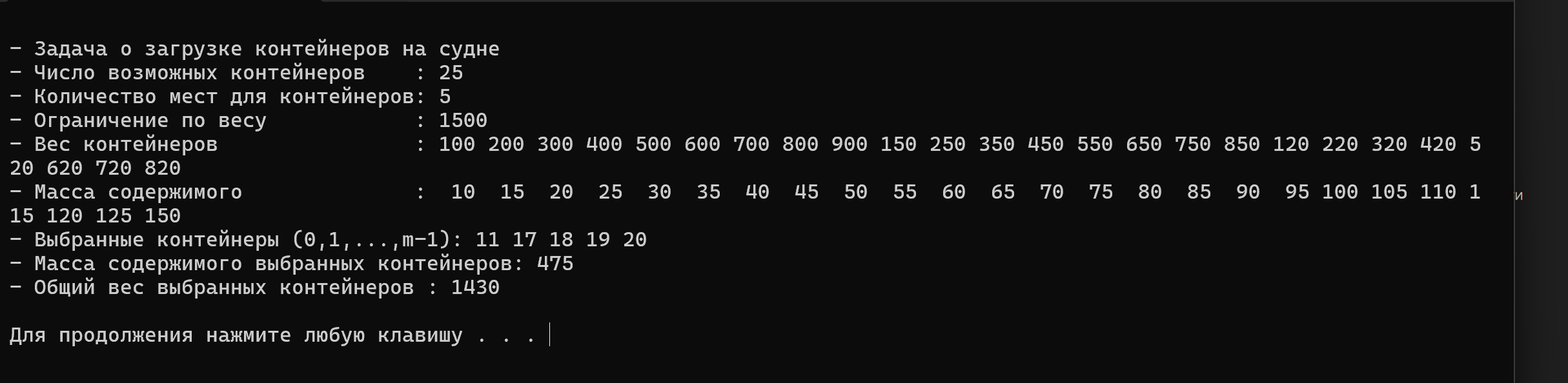


Рисунок 5.1 – результат работы программы

**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

Реализация задачи представлена в листнге 6.1

Листинг 6.1 - Код файла Main\_boat.cpp

|  |
| --- |
| #include <iostream>  #include <iomanip>  #include "Boat.h"  #include <time.h>  #define NN (sizeof(v)/sizeof(int))  #define MM 6  #define SPACE(n) std::setw(n)<<" "  int main()  {  setlocale(LC\_ALL, "rus");  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 };  int c[NN] = { 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 };  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 = 25; 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;  system("pause");  return 0;  } |

Результат выполнения программы представлен на рисунке 6.1.

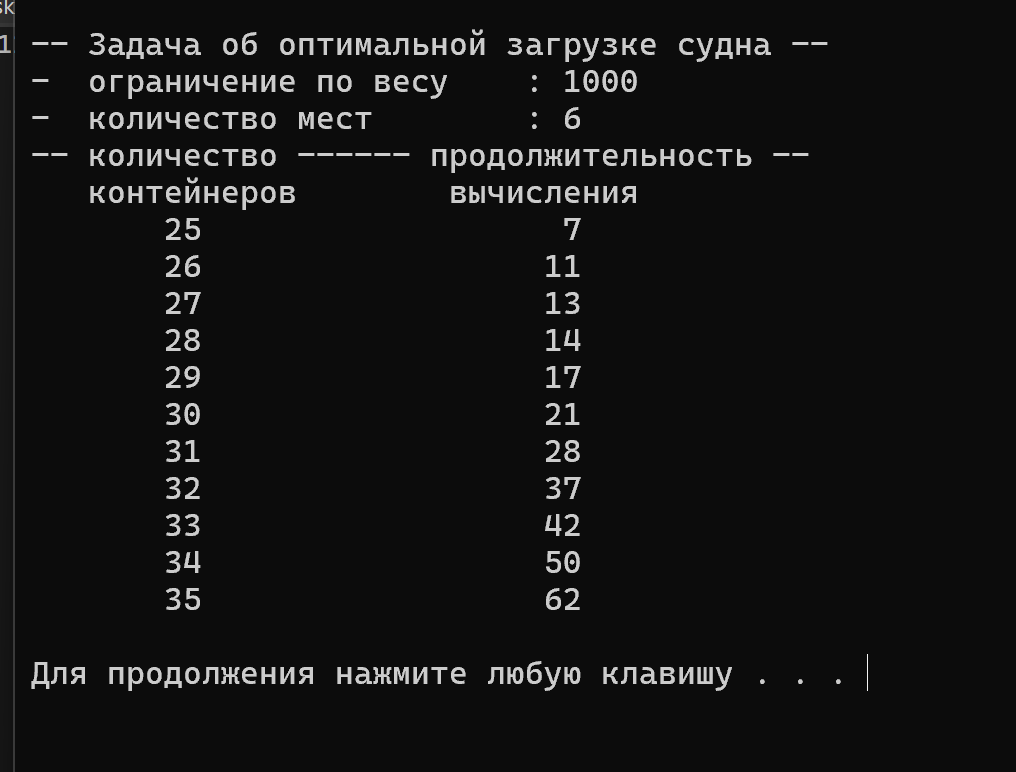


Рисунок 6.1 – результат выполнения программы

На рисунке 6.2 представлен график сделанный исходя из полученных данных.

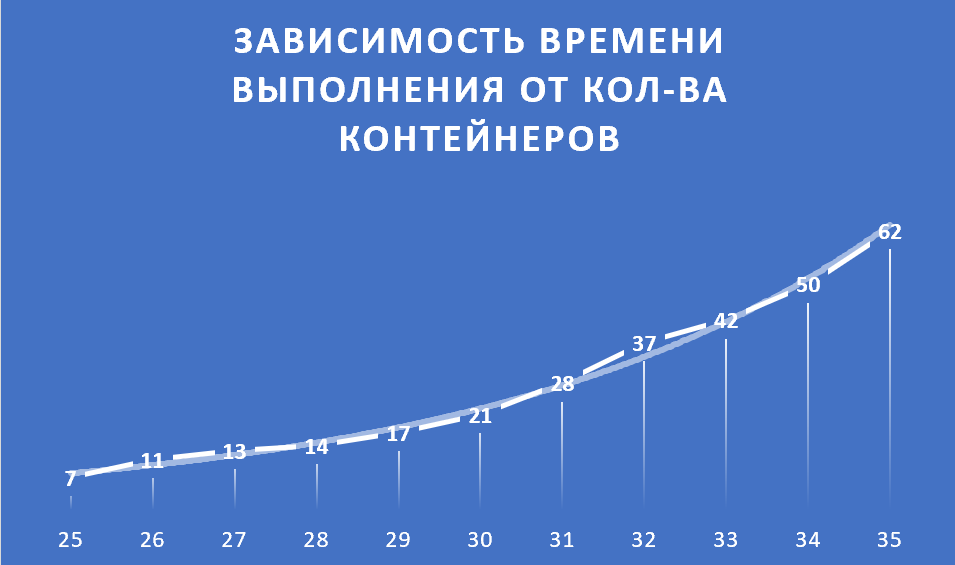


Рисунок 6.2 – график

Вывод: исходя из полученных данных и графика, можно заметить, что скорость выполнения программы плавно возрастает при добавлении количества контейнеров(экспонинцеальная зависимость ) .

**Вывод:** в лабораторной работе №2 были приобретены навыки разработки генераторов подмножеств, перестановок, сочетаний и размещений на С++; так же научились применять разработанные генераторы для решения задач о рюкзаке (упрощенную, коммивояжера, об оптимальной загрузке судна и об оптимальной загрузке судна с центровкой.

Для каждой задачи был предложен алгоритм решения, основанный на генерации соответствующих комбинаций, и выполнена их реализация на языке C++. Для исследования зависимости времени выполнения от размерности задачи был проведен эксперимент с генерацией случайных данных различной размерности и измерением времени выполнения алгоритмов.

**Лабораторная работа №3. Метод ветвей и границ. Задача коммивояжера и методы ее 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** |  | 6 | 24 |  | 3 |
| **2** | 3 |  | 18 | 65 | 81 |
| **3** | 5 | 9 |  | 86 | 52 |
| **4** | 20 | 55 | 12 |  | 9 |
| **5** | 90 | 69 | 52 | 16 |  |

где *n* – номер варианта или номер по журналу;

n = 3;

**Условие задачи:**

Дано **5 городов** и **матрица расстояний** между ними.  
Матрица содержит **INF** (бесконечность) на диагонали, так как нельзя поехать из города в него же.  
Цель — найти маршрут минимальной длины, который:

1. Начинается в одном из городов.
2. Проходит **через каждый город ровно один раз**.
3. Завершается в исходном городе.
4. Количество городов **N = 5**.

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

**Ход решения:**

Имеем 5 городов, построим матрицу расстояний между городами: 1)

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

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

2)

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
|  | 1 | 2 | 3 | 4 | 5 |
| 1 | INF | 6 | 24 | INF | 3 | 3 |
| 2 | 3 | INF | 18 | 65 | 81 | 3 |
| 3 | 5 | 9 | INF | 86 | 52 | 5 |
| 4 | 20 | 55 | 12 | INF | 9 | 9 |
| 5 | 90 | 69 | 52 | 16 | INF | 16 |
| **36** |

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

3)

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
|  | 1 | 2 | 3 | 4 | 5 |
| 1 | INF | 3 | 21 | INF | 0 | 3 |
| 2 | 0 | INF | 15 | 62 | 78 | 3 |
| 3 | 0 | 4 | INF | 81 | 47 | 5 |
| 4 | 11 | 46 | 3 | INF | 0 | 9 |
| 5 | 74 | 53 | 36 | 0 | INF | 16 |
| **36** |

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

4)

|  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- |
|  | 1 | 2 | | | 3 | 4 | | 5 |
| 1 | INF | 3 | | | 21 | INF | | 0 |
| 2 | 0 | INF | | | 15 | 62 | | 78 |
| 3 | 0 | 4 | | | INF | 81 | | 47 |
| 4 | 11 | 46 | | | 3 | INF | | 0 |
| 5 | 74 | 53 | | | 36 | 0 | | INF |
| 0 | | 3 | 3 | | | 0 | 0 | | 6 |

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

5)

|  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- |
|  | 1 | 2 | | | 3 | 4 | | 5 |
| 1 | INF | 0 | | | 18 | INF | | 0 |
| 2 | 0 | INF | | | 12 | 62 | | 78 |
| 3 | 0 | 1 | | | INF | 81 | | 47 |
| 4 | 11 | 43 | | | 0 | INF | | 0 |
| 5 | 74 | 50 | | | 33 | 0 | | INF |
| 0 | | 3 | 3 | | | 0 | 0 | | 6 |

6)НИЖНЯЯ ВЕРШИНА ГРАНИЦЫ ДЛИНЫ МАРШРУТА

**H=36+6=42.**

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

7)

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| С | 1 | 2 | 3 | 4 | 5 |
| 1 | INF | 0(1) | 18 | INF | 0(0) |
| 2 | 0(12) | INF | 12 | 62 | 78 |
| 3 | 0(1) | 1 | INF | 81 | 47 |
| 4 | 11 | 43 | 0(12) | INF | 0(0) |
| 5 | 74 | 50 | 33 | 0(95) | INF |

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.

8)

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| С | 1 | 2 | 3 | 4 | 5 |
| 1 | INF | 0(1) | 18 | INF | 0(0) |
| 2 | 0(12) | INF | 12 | 62 | 78 |
| 3 | 0(1) | 1 | INF | 81 | 47 |
| 4 | 11 | 43 | 0(12) | INF | 0(0) |
| 5 | 74 | 50 | 33 | 0(95) | INF |

**H = 42 + 62 = 104**

**9)**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| С | 1 | 2 | 3 | 5 |
| 1 | INF | 0 | 18 | 0 |
| 2 | 0 | INF | 12 | 78 |
| 3 | 0 | 1 | INF | 47 |
| 4 | 11 | 43 | 0 | 0 |

**H = H + ∑di+ ∑dj = 42**

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

10)

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| С | 1 | 2 | 3 | 5 |
| 1 | INF | 0(1) | 18 | 0(0) |
| 2 | 0(12) | INF | 12 | 78 |
| 3 | 0(1) | 1 | INF | 47 |
| 4 | 11 | 43 | 0(12) | 0(0) |

Следовательно

11)

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| С | 1 | 2 | 3 | 5 |
| 1 | INF | 0(1) | 18 | 0(0) |
| 2 | 0(12) | INF | 12 | 78 |
| 3 | 0(1) | 1 | INF | 47 |
| 4 | 11 | 43 | 0(12) | 0(0) |

**H = H + ∑di+ ∑dj = 42+12 = 54**

12)

|  |  |  |  |
| --- | --- | --- | --- |
| С | 1 | 2 | 5 |
| 1 | INF | 0 | 0 |
| 2 | 0 | INF | 78 |
| 3 | 0 | 1 | 47 |

**H = H + ∑di+ ∑dj = 42 + 0 = 42**

Ту строку и тот столбец, где образовалось два знака «INF», полностью вычеркиваем.

13)

|  |  |  |  |
| --- | --- | --- | --- |
| С | 1 | 2 | 5 |
| 1 | INF | 0(1) | 0(78) |
| 2 | 0 | INF | 78 |
| 3 | 0 | 1 | 47 |

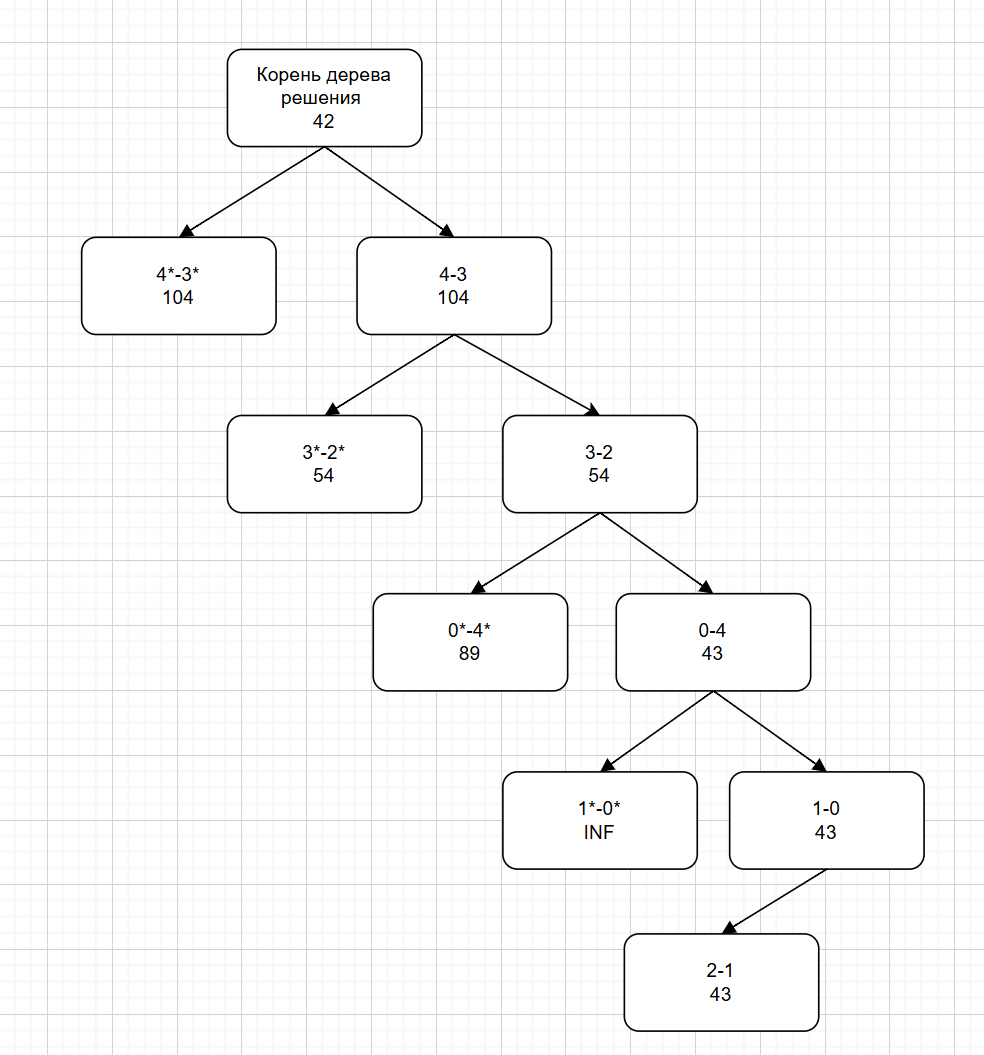
**H = 42 +47 = 89**

14)

|  |  |  |
| --- | --- | --- |
| С | 1 | 2 |
| 2 | 0(inf) | INF |
| 3 | 0(1) | 1 |

**H = 42 + 1 = 43**

Т.к. минимальное значение по столбцу 1, прибавляем к нашей нижней границе 42. φ=42+1=43.

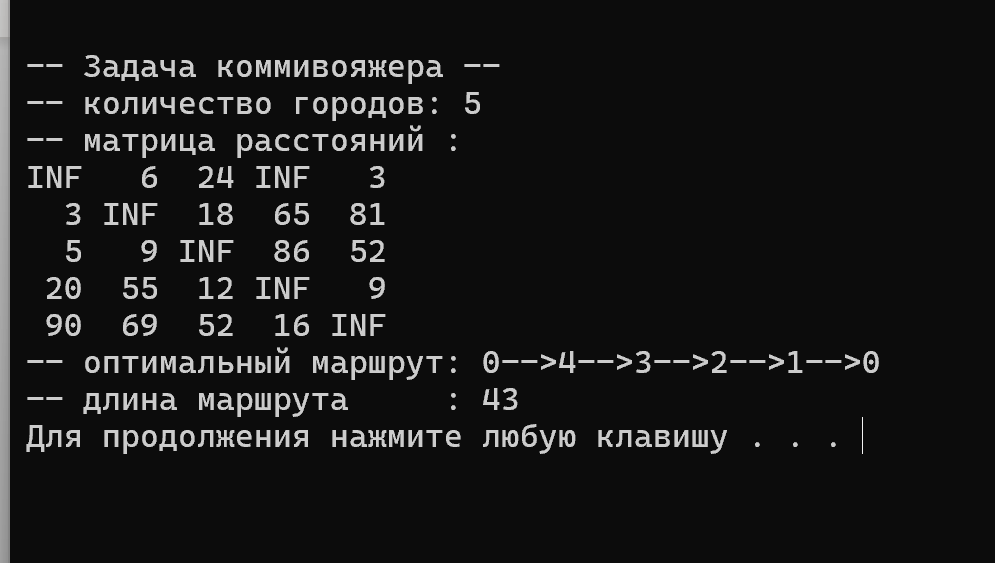




**Решение** :(0,4), (4,3), (3,2), (2,1), (1,0),

**Длина оптимального маршрута: φ=43**

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



Вывод. В ходе лабораторной работы были освоены общие принципы решения задач методом ветвей и границ, решена задача коммивояжера данным методом.

# **Лабораторная работа №4. Динамическое программирование**

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

**Задание 1.**

На языке С++ сгенерировать случайным образом строку букв латинского алфавита *S1* длиной 300 символов и *S2*длиной 200. Листинг кода представлен в листинге 1.1. Результат выполнения программы представлен на рисунке 1.1.

|  |
| --- |
| #include <iostream>  #include <windows.h>  #include <string>  #include <ctime>  using namespace std;  string generate(int countSymb) {  string str;  char symb;  for (int i = 0; i < countSymb; i++) {  //symb = 'a' + rand() % 26; // Генерируем случайную букву от 'a' до 'z'  symb = (rand() % 2 == 0) ? ('A' + rand() % 26) : ('a' + rand() % 26);  str += symb;  }  return str;  }  int main() {  SetConsoleCP(1251);  SetConsoleOutputCP(1251);  srand(time(NULL));  string s1 = generate(300);  string s2 = generate(200);  cout << "S1(300): " << s1 << endl;  cout << "S2(200): " << s2 << endl;  return 0;  } |

Листинг 1.1 – Генерация строк

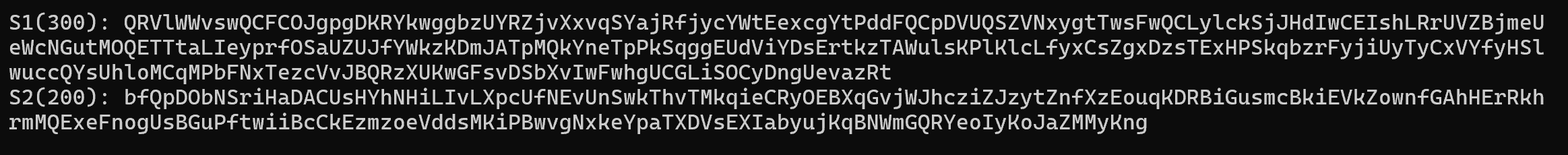


Рисунок 1.1 – Результат работы программы

**Задание 2.**

Вычислить двумя способами (рекурсивно и с помощью динамического программирования)  – Левенштейна для , где - длина строки ,  - строка, состоящая из первых  символов строки . (копии экрана и код вставить в отчет). Листинг кода представлен в листинге 2.1,2.2,2.3. Результат выполнения программы представлен на рисунке 2.1

|  |
| --- |
| // - Levenshtein.cpp  #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 2.1 – файл Levenshtein.cpp

|  |
| --- |
| #include <algorithm>  #include <iostream>  #include <ctime>  #include <iomanip>  #include <cstring>  #include "Levenshtein.h"  #include <Windows.h>  using namespace std;  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;  }  int main()  {  SetConsoleCP(1251);  SetConsoleOutputCP(1251);  const int threeHundred = 300;  const int twoHundred = 200;  char\* s1 = GenerateRandomString(threeHundred);  cout << "S1: " << endl;  for (int i = 0; i < threeHundred; i++) {  if (i % 50 == 0)  {  cout << "\n";  }  cout << s1[i];  }  cout << endl << endl;  srand(time(NULL) + 1);  char\* s2 = GenerateRandomString(twoHundred);  cout << "S2: " << endl;  for (int i = 0; i < twoHundred; i++) {  if (i % 50 == 0)  {  cout << "\n";  }  cout << s2[i];  }  cout << endl << endl;  clock\_t t1 = 0, t2 = 0, t3 = 0, t4 = 0;  int lx = strlen(s1);  int ly = strlen(s2);  int s1\_size[]{ threeHundred / 25, threeHundred / 20, threeHundred / 15, threeHundred / 10, threeHundred / 5, threeHundred / 2, threeHundred };  int s2\_size[]{ twoHundred / 25, twoHundred / 20, twoHundred / 15, twoHundred / 10, twoHundred / 5, twoHundred / 2, twoHundred };  cout << "\n\n-- расстояние Левенштейна -----";  cout << "\n\n--длина --- рекурсия -- дин.програм. ---\n";  for (int i = 0; i < min(sizeof(s1\_size) / sizeof(s1\_size[0]), sizeof(s2\_size) / sizeof(s2\_size[0])); i++)  {  t1 = clock();  levenshtein\_r(s1\_size[i], s1, s2\_size[i], s2);  t2 = clock();  t3 = clock();  levenshtein(s1\_size[i], s1, s2\_size[i], s2);  t4 = clock();  cout << right << setw(2) << s1\_size[i] << "/" << setw(2) << s2\_size[i]  << " " << left << setw(10) << (t2 - t1)  << " " << setw(10) << (t4 - t3) << endl;  }  system("pause");  return 0;  } |

Листинг 2.2 – файл Main.cpp

|  |
| --- |
| #pragma once  // - Levenshtein.h  // -- дистанции Левенштeйна (динамическое программирование)  int levenshtein(  int lx, // длина слова x  const char x[], // слово длиной lx  int ly, // длина слова y  const char y[] // слово y  );  // -- дистанции Левенштeйна (рекурсия)  int levenshtein\_r(  int lx, // длина строки x  const char x[], // строка длиной lx  int ly, // длина строки y  const char y[] // строка y  ); |

Листинг 2.3 – файл Levenshtein.h

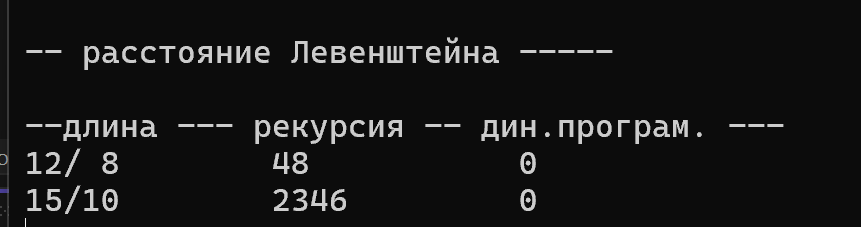


Рисунок 2.1 – Результат работы программы

**Задание 3.**

Выполнить сравнительный анализ времени затраченного на вычисление дистанции Левенштейна для двух методов решения. Построить графики зависимости времени вычисления от . На рисунке 3.1 представлены графики зависимости времени вычисления от *k*.

На графике, который изображён на рисунке 3.1, можно заметить, что использование динамического алгоритма во много раз эффективнее по затраченному времени, нежели рекурсивное выполнение.

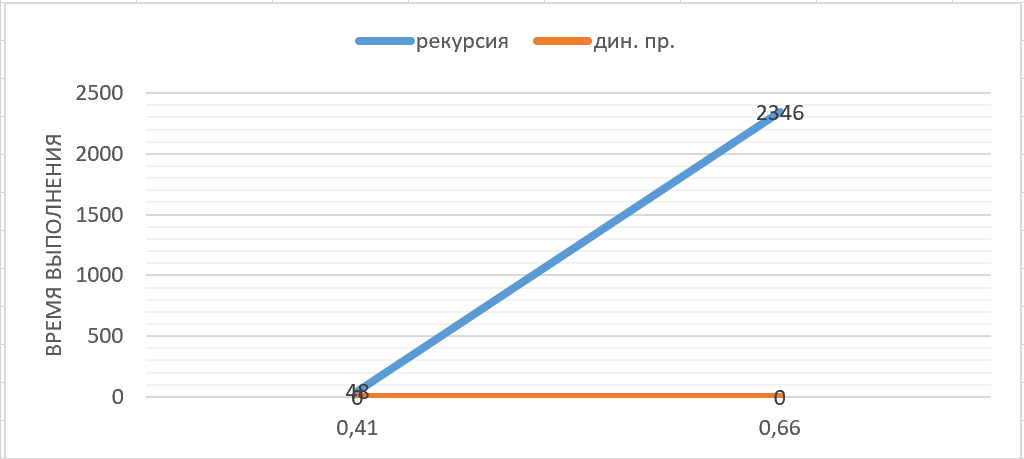


Рисунок 3.1 – График зависимости выполнения

**Задание 4.**

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





= 5.

= 4.



= 4.

= 3.



= 3.

= 2.



= 2.

= 1.



= 3.

= 2.



= 2.

= 1.

L(“Л”, “Г”) = 0

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

Таким образом самая короткая дистанция Левенштейна составляет 3 шага. Результат программы представлен на рисунке 4.1.

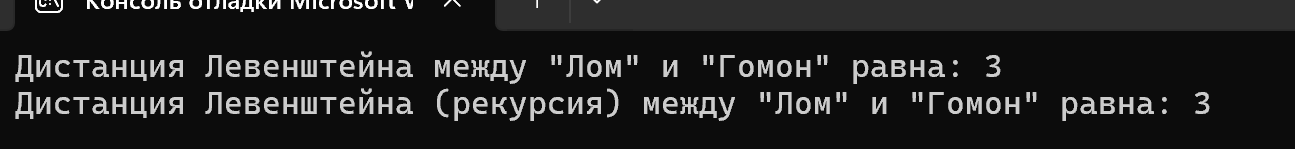


Рисунок 4.1 – результат выполнения программы

**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 .

Листинги кода представлены ниже:

|  |
| --- |
| #include <algorithm>  #include "LCS.h"  #include <cstring>  #include "LCS.h"  #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 };  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  }  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 |

Листинг 5.1 - LCS.cpp

|  |
| --- |
| #include <iostream>  #include "LCS.h"  int main()  {  setlocale(LC\_ALL, "rus");  clock\_t t1 = 0, t2 = 0, t3 = 0, t4 = 0;  t1 = clock();  char X[] = "PVTWNHO", Y[] = "RPTWYK";  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, // длина последовательности X  "PVTWNHO", // последовательность X  sizeof(Y) - 1, // длина последовательности Y  "RPTWYK" // последовательность Y  );  std::cout << std::endl << "-- длина LCS: " << s << std::endl;  t2 = clock();  /// <summary>    /// </summary>  t3 = clock();  char z[100] = "";  char x[] = "PVTWNHO",  y[] = "RPTWYK";  int l = lcsd(x, y, z);  std::cout << std::endl  << "-- наибольшая общая подпоследовательость - LCS(динамическое "  << "программирование)" << std::endl;  t4 = clock();  std::cout << std::endl << "последовательость X: " << x;  std::cout << std::endl << "последовательость Y: " << y;  std::cout << std::endl << " LCS: " << z;  std::cout << std::endl << " длина LCS: " << l;  std::cout << std::endl;  std::cout << '\n' << "Время выполнения рекурсивно: " << t2 - t1;  std::cout << '\n' << "Время выполнения динамичически: " << t4 - t3 << '\n' << std::endl;  system("pause");  return 0;  } |

Листинг 5.2 – main.cpp

|  |
| --- |
| #pragma once  int lcs(  int lenx, // длина последовательности X  const char x[], // последовательность X  int leny, // длина последовательности Y  const char y[] // последовательность Y  );  int lcsd(  const char x[], // последовательность X  const char y[], // последовательность Y  char z[] // наибольшая общая подпоследовательность  ); |

Листинг 5.3 – LCS.h

Выполнив сравнительный анализ времени затраченного на решение задачи о наибольшей общей подпоследовательности для двух методов решения, можно заметить, что динамический алгоритм работает быстрее, однако, в данном ситуации оба метода затрачивают для выполнения минимально времени. Результаты представлены на рисунке 5.1:

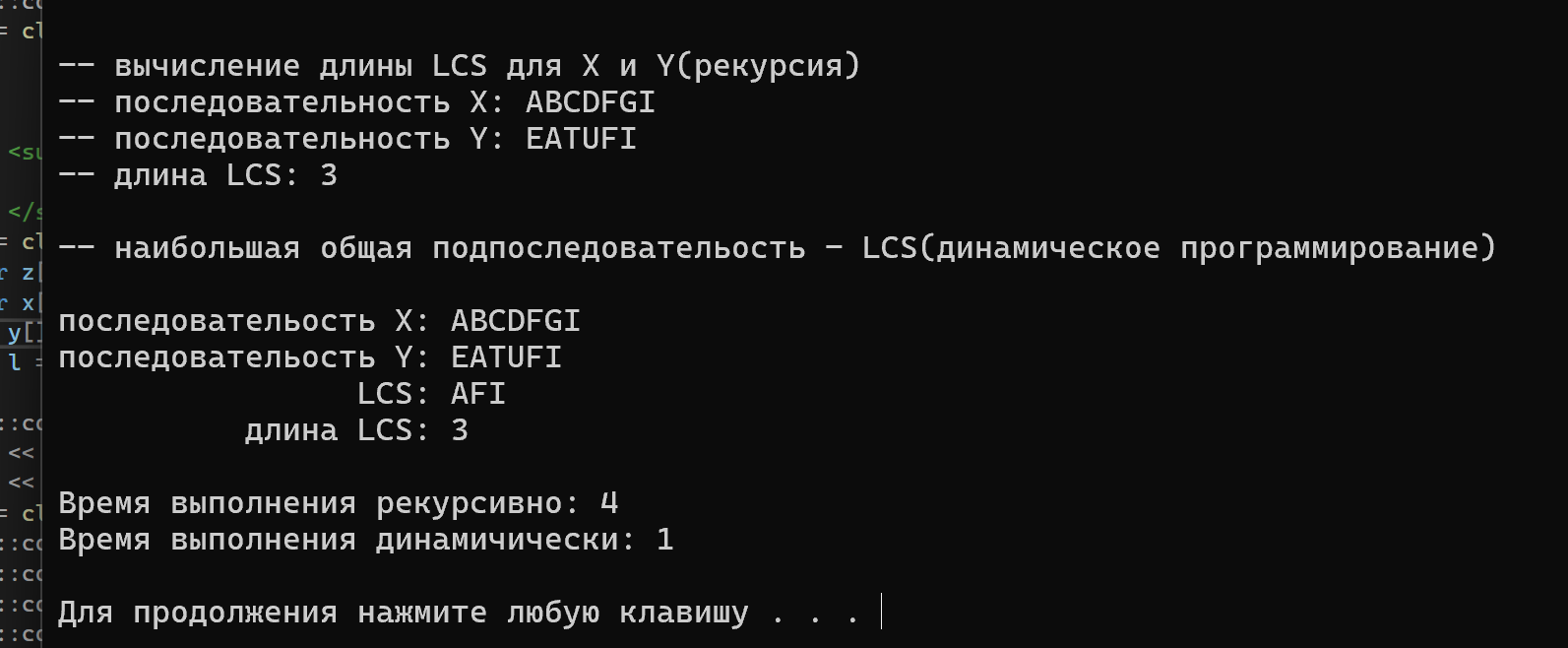


Рисунок 5.1 – выполнение программы

На рисунке 5.2 представлен график зависимости от суммарной длинны двух подстрок и необходимом времени выполнения при помощи динамического алгоритма:

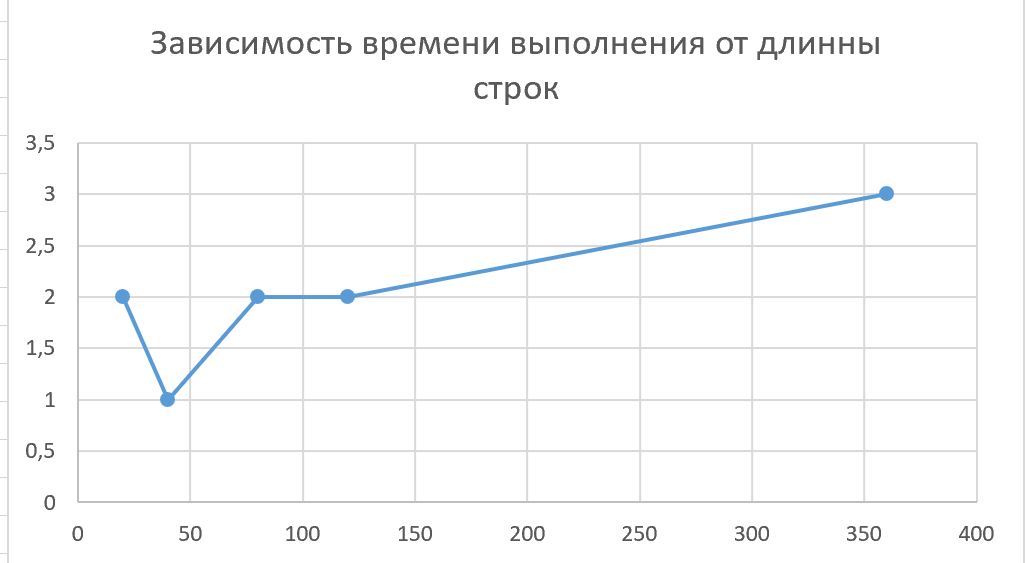


Рисунок 5.2 – график

Видно, что с увеличением длины входных последовательностей время выполнения имеет тенденцию к росту. На небольших длинах (до 50 символов) наблюдаются колебания, что может быть связано с особенностями работы процессора, кеширования или флуктуациями в системе. После 100 символов рост времени выполнения становится более плавным и линейным, что соответствует предсказуемому поведению метода динамического программирования.

**Алгоритм прохождения:**

* Первая строка и первый столбец всегда 0 (базовый случай).
* Если символы совпадают, то берем диагональное значение +1.
* Если не совпадают, берем максимум из верхней или левой ячейки.

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
|  |  | E | A | T | U | F | I |
|  | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| A | 0 | 0 | 1 | 1 | 1 | 1 | 1 |
| B | 0 | 0 | 1 | 1 | 1 | 1 | 1 |
| C | 0 | 0 | 1 | 1 | 1 | 1 | 1 |
| D | 0 | 0 | 1 | 1 | 1 | 1 | 1 |
| F | 0 | 0 | 1 | 1 | 1 | 2 | 2 |
| G | 0 | 0 | 1 | 1 | 1 | 2 | 2 |
| I | 0 | 0 | 1 | 1 | 1 | 2 | 3 |

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
|  |  | E | A | T | U | F | I |
|  |  |  |  |  |  |  |  |
| A |  |  |  |  |  |  |  |
| B |  |  |  |  |  |  |  |
| C |  |  |  |  |  |  |  |
| D |  |  |  |  |  |  |  |
| F |  |  |  |  |  |  |  |
| G |  |  |  |  |  |  |  |
| I |  |  |  |  |  |  |  |

**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*). Вариант 3.

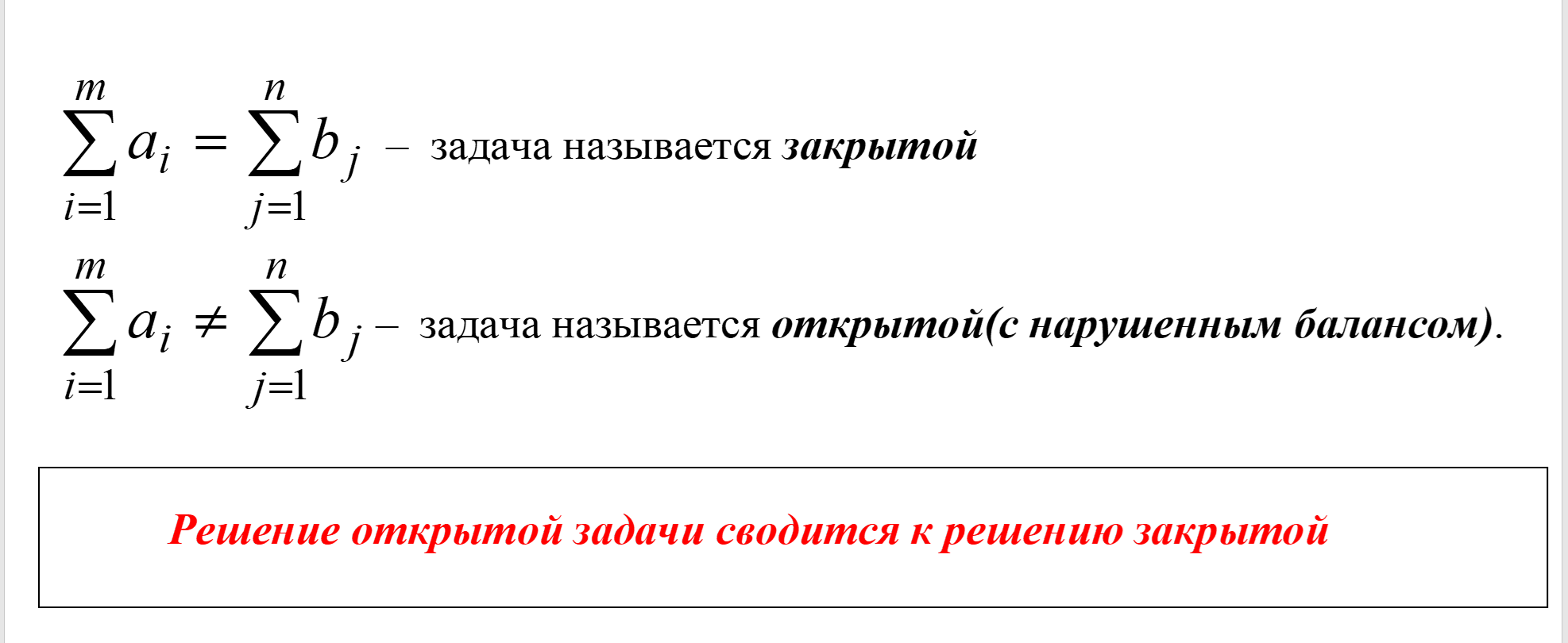
|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| ПОТРЕБИТЕЛИ  ПОСТАВЩИКИ | 1 | 2 | 3 | 4 | 5 | 6 | ЗАПАСЫ |
| 1 | **15** | **5** | **9** | **6** | **14** | **4** | **171** |
| 2 | **13** | **3** | **11** | **8** | **10** | **16** | **116** |
| 3 | **4** | **8** | **14** |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

**Ход решения:**

* Проверить, открытая задача или закрытая;
* Составить опорный план;
* Применить метод потенциалов;

Для разрешимости транспортной задачи необходимо, чтобы суммарные запасы продукции у поставщиков равнялись суммарной потребности потребителей.

Проверим необходимое и достаточное условие разрешимости задачи.  
∑a = 171+116+153+162+103=705  
∑b =146+110+134+196+98+166=850



Как видно, суммарная потребность груза в пунктах назначения превышает запасы груза на базах. Следовательно, модель исходной транспортной задачи является открытой. Чтобы получить закрытую модель, введем дополнительную (фиктивную) базу с запасом груза, равным 145 (704—850). Тарифы перевозки единицы груза из базы ко всем потребителям полагаем равны нулю.  
Занесем исходные данные в распределительную таблицу.

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| ПОТРЕБИТЕЛИ | b1 | b2 | b3 | b4 | B5 | b6 | ЗАПАСЫ |
| ПОСТАВЩИКИ |
| a 1 | **15** | **5** | **9** | **6** | **14** | **4** | **171** |
| a 2 | **13** | **3** | **11** | **8** | **10** | **16** | **116** |
| a 3 | **4** | **8** | **14** | **11** | **5** | **14** | **153** |
| a 4 | **7** | **13** | **13** | **6** | **16** | **5** | **162** |
| a 5 | **6** | **14** | **12** | **3** | **13** | **7** | **103** |
| a 6 | **0** | **0** | **0** | **0** | **0** | **0** | **145** |
| ПОТРЕБНОСТИ | **146** | **110** | **134** | **196** | **98** | **166** |  |

Теперь выполняется условие.

Используем 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 ai, или bj.  
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.

Искомый элемент равен c22=3. Для этого элемента запасы равны 116, потребности 110. Поскольку минимальным является 110, то вычитаем его.  
x22 = min(116,110) = 110.

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| ПОТРЕБИТЕЛИ | b1 | b2 | b3 | b4 | B5 | b6 | ЗАПАСЫ |
| ПОСТАВЩИКИ |
| a 1 | **15** | **x** | **9** | **6** | **14** | **4** | **171** |
| a 2 | **13** | **3** | **11** | **8** | **10** | **16** | **116-110=6** |
| a 3 | **4** | **x** | **14** | **11** | **5** | **14** | **153** |
| a 4 | **7** | **x** | **13** | **6** | **16** | **5** | **162** |
| a 5 | **6** | **x** | **12** | **3** | **13** | **7** | **103** |
| a 6 | **0** | **x** | **0** | **0** | **0** | **0** | **145** |
| ПОТРЕБНОСТИ | **146** | **110-110=0** | **134** | **196** | **98** | **166** |  |

Искомый элемент равен c54=3. Для этого элемента запасы равны 103, потребности 196. Поскольку минимальным является 103, то вычитаем его.  
x54 = min(103,196) = 103.

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| ПОТРЕБИТЕЛИ | b1 | b2 | b3 | b4 | B5 | b6 | ЗАПАСЫ |
| ПОСТАВЩИКИ |
| a 1 | **15** | **x** | **9** | **6** | **14** | **4** | **171** |
| a 2 | **13** | **3|110** | **11** | **8** | **10** | **16** | **6** |
| a 3 | **4** | **x** | **14** | **11** | **5** | **14** | **153** |
| a 4 | **7** | **x** | **13** | **6** | **16** | **5** | **162** |
| a 5 | **x** | **x** | **x** | **3** | **x** | **x** | **103-103=0** |
| a 6 | **0** | **x** | **0** | **0** | **0** | **0** | **145** |
| ПОТРЕБНОСТИ | **146** | **0** | **134** | **196-103=93** | **98** | **166** |  |

Искомый элемент равен c16=4. Для этого элемента запасы равны 171, потребности 166. Поскольку минимальным является 166, то вычитаем его.  
x16 = min(171,166) = 166.

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| ПОТРЕБИТЕЛИ | b1 | b2 | b3 | b4 | B5 | b6 | ЗАПАСЫ |
| ПОСТАВЩИКИ |
| a 1 | **15** | **x** | **9** | **6** | **14** | **4** | **171-166=5** |
| a 2 | **13** | **3|110** | **11** | **8** | **10** | **x** | **6** |
| a 3 | **4** | **x** | **14** | **11** | **5** | **x** | **153** |
| a 4 | **7** | **x** | **13** | **6** | **16** | **x** | **162** |
| a 5 | **x** | **x** | **x** | **3|103** | **x** | **x** | **0** |
| a 6 | **0** | **x** | **0** | **0** | **0** | **x** | **145** |
| ПОТРЕБНОСТИ | **146** | **0** | **134** | **93** | **98** | **166-166=0** |  |

Искомый элемент равен c31=4. Для этого элемента запасы равны 153, потребности 146. Поскольку минимальным является 146, то вычитаем его.  
x31 = min(153,146) = 146.

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| ПОТРЕБИТЕЛИ | b1 | b2 | b3 | b4 | B5 | b6 | ЗАПАСЫ |
| ПОСТАВЩИКИ |
| a 1 | **x** | **x** | **9** | **6** | **14** | **4|166** | **5** |
| a 2 | **x** | **3|110** | **11** | **8** | **10** | **x** | **6** |
| a 3 | **4** | **x** | **14** | **11** | **5** | **x** | **153-146=7** |
| a 4 | **x** | **x** | **13** | **6** | **16** | **x** | **162** |
| a 5 | **x** | **x** | **x** | **3|103** | **x** | **x** | **0** |
| a 6 | **x** | **x** | **0** | **0** | **0** | **x** | **145** |
| ПОТРЕБНОСТИ | **146-146=0** | **0** | **134** | **93** | **98** | **0** |  |

Искомый элемент равен c35=5. Для этого элемента запасы равны 7, потребности 98. Поскольку минимальным является 7, то вычитаем его.  
x35 = min(7,98) = 7.

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| ПОТРЕБИТЕЛИ | b1 | b2 | b3 | b4 | B5 | b6 | ЗАПАСЫ |
| ПОСТАВЩИКИ |
| a 1 | **x** | **x** | **9** | **6** | **14** | **4|166** | **5** |
| a 2 | **x** | **3|110** | **11** | **8** | **10** | **x** | **6** |
| a 3 | **4|146** | **x** | **x** | **x** | **5** | **x** | **7-7=0** |
| a 4 | **x** | **x** | **13** | **6** | **16** | **x** | **162** |
| a 5 | **x** | **x** | **x** | **3|103** | **x** | **x** | **0** |
| a 6 | **x** | **x** | **0** | **0** | **0** | **x** | **147** |
| ПОТРЕБНОСТИ | **0** | **0** | **134** | **93** | **98-7=91** | **0** |  |

7) Искомый элемент равен c14=6. Для этого элемента запасы равны 5, потребности 93. Поскольку минимальным является 5, то вычитаем его.  
x14 = min(5,93) = 5.

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| ПОТРЕБИТЕЛИ | b1 | b2 | b3 | b4 | B5 | b6 | ЗАПАСЫ |
| ПОСТАВЩИКИ |
| a 1 | **x** | **x** | **x** | **6** | **x** | **4|166** | **5-5=0** |
| a 2 | **x** | **3|110** | **11** | **8** | **10** | **x** | **6** |
| a 3 | **4|146** | **x** | **x** | **x** | **5|7** | **x** | **0** |
| a 4 | **x** | **x** | **13** | **6** | **16** | **x** | **162** |
| a 5 | **x** | **x** | **x** | **3|103** | **x** | **x** | **0** |
| a 6 | **x** | **x** | **0** | **0** | **0** | **x** | **145** |
| ПОТРЕБНОСТИ | **0** | **0** | **134** | **93-5=88** | **91** | **0** |  |

8) Искомый элемент равен c44=6. Для этого элемента запасы равны 162, потребности 88. Поскольку минимальным является 88, то вычитаем его.  
x44 = min(162,88) = 88.

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| ПОТРЕБИТЕЛИ | b1 | b2 | b3 | b4 | B5 | b6 | ЗАПАСЫ |
| ПОСТАВЩИКИ |
| a 1 | **x** | **x** | **x** | **6|5** | **x** | **4|166** | **0** |
| a 2 | **x** | **3|110** | **11** | **x** | **10** | **x** | **6** |
| a 3 | **4|146** | **x** | **x** | **x** | **5|7** | **x** | **0** |
| a 4 | **x** | **x** | **13** | **6** | **16** | **x** | **162-88=74** |
| a 5 | **x** | **x** | **x** | **3|103** | **x** | **x** | **0** |
| a 6 | **x** | **x** | **0** | **0** | **0** | **x** | **145** |
| ПОТРЕБНОСТИ | **0** | **0** | **134** | **88-88=0** | **91** | **0** |  |

9) Искомый элемент равен c25=10. Для этого элемента запасы равны 6, потребности 91. Поскольку минимальным является 6, то вычитаем его.  
x25 = min(6,91) = 6.

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| ПОТРЕБИТЕЛИ | b1 | b2 | b3 | b4 | B5 | b6 | ЗАПАСЫ |
| ПОСТАВЩИКИ |
| a 1 | **x** | **x** | **x** | **6|5** | **x** | **4|166** | **0** |
| a 2 | **x** | **3|110** | **x** | **x** | **10** | **x** | **6-6=0** |
| a 3 | **4|146** | **x** | **x** | **x** | **5|7** | **x** | **0** |
| a 4 | **x** | **x** | **13** | **6|88** | **16** | **x** | **74** |
| a 5 | **x** | **x** | **x** | **3|103** | **x** | **x** | **0** |
| a 6 | **x** | **x** | **0** | **0** | **0** | **x** | **145** |
| ПОТРЕБНОСТИ | **0** | **0** | **134** | **0** | **91-6=85** | **0** |  |

10) Искомый элемент равен c43=13. Для этого элемента запасы равны 74, потребности 134. Поскольку минимальным является 74, то вычитаем его.  
x43 = min(74,134) = 74.

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| ПОТРЕБИТЕЛИ | b1 | b2 | b3 | b4 | B5 | b6 | ЗАПАСЫ |
| ПОСТАВЩИКИ |
| a 1 | **x** | **x** | **x** | **6|5** | **x** | **4|166** | **0** |
| a 2 | **x** | **3|110** | **x** | **x** | **10|6** | **x** | **0** |
| a 3 | **4|146** | **x** | **x** | **x** | **5|7** | **x** | **0** |
| a 4 | **x** | **x** | **13** | **6|88** | **x** | **x** | **74-74=0** |
| a 5 | **x** | **x** | **x** | **3|103** | **x** | **x** | **0** |
| a 6 | **x** | **x** | **0** | **0** | **0** | **x** | **145** |
| ПОТРЕБНОСТИ | **0** | **0** | **134-74=60** | **0** | **85** | **0** |  |

11) Искомый элемент равен c63=0. Для этого элемента запасы равны 145, потребности 60. Поскольку минимальным является 60, то вычитаем его.  
x63 = min(145,60) = 60.

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| ПОТРЕБИТЕЛИ | b1 | b2 | b3 | b4 | B5 | b6 | ЗАПАСЫ |
| ПОСТАВЩИКИ |
| a 1 | **x** | **x** | **x** | **6|5** | **x** | **4|166** | **0** |
| a 2 | **x** | **3|110** | **x** | **x** | **10|6** | **x** | **0** |
| a 3 | **4|146** | **x** | **x** | **x** | **5|7** | **x** | **0** |
| a 4 | **x** | **x** | **13|74** | **6|88** | **x** | **x** | **0** |
| a 5 | **x** | **x** | **x** | **3|103** | **x** | **x** | **0** |
| a 6 | **x** | **x** | **0** | **0** | **0** | **x** | **145-60=85** |
| ПОТРЕБНОСТИ | **0** | **0** | **60-60=0** | **0** | **85** | **0** |  |

12) Искомый элемент равен c65=0. Для этого элемента запасы равны 85, потребности 85. Поскольку минимальным является 85, то вычитаем его.  
x65 = min(85,85) = 94.

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| ПОТРЕБИТЕЛИ | b1 | b2 | b3 | b4 | B5 | b6 | ЗАПАСЫ |
| ПОСТАВЩИКИ |
| a 1 | **x** | **x** | **x** | **6|5** | **x** | **4|166** | **0** |
| a 2 | **x** | **3|110** | **x** | **x** | **10|6** | **x** | **0** |
| a 3 | **4|146** | **x** | **x** | **x** | **5|7** | **x** | **0** |
| a 4 | **x** | **x** | **13|74** | **6|88** | **x** | **x** | **0** |
| a 5 | **x** | **x** | **x** | **3|103** | **x** | **x** | **0** |
| a 6 | **x** | **x** | **0|60** | **0** | **0|85** | **x** | **85-85=0** |
| ПОТРЕБНОСТИ | **0** | **0** | **0** | **0** | **85-85=0** | **0** |  |

13)

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| ПОТРЕБИТЕЛИ | b1 | b2 | b3 | b4 | B5 | b6 | ЗАПАСЫ |
| ПОСТАВЩИКИ |
| a 1 | **15** | **5** | **9** | **6[5]** | **14** | **4[166]** | **171** |
| a 2 | **13** | **3[110]** | **11** | **8** | **10[6]** | **16** | **116** |
| a 3 | **4[146]** | **8** | **14** | **11** | **5[7]** | **14** | **153** |
| a 4 | **7** | **13** | **13[74]** | **6[88]** | **16** | **5** | **162** |
| a 5 | **6** | **14** | **12** | **3[103]** | **13** | **7** | **103** |
| a 6 | **0** | **0** | **0[60]** | **0** | **0[85]** | **0** | **145** |
| ПОТРЕБНОСТИ | **146** | **110** | **134** | **196** | **98** | **166** |  |

В результате получен первый опорный план, который является допустимым, так как все грузы из баз вывезены, потребность потребителей удовлетворена, а план соответствует системе ограничений транспортной задачи.  
Расчёт итоговой стоимости перевозки:  
F(x) = 6\*5+4\*166+3\*110+10\*6+4\*146+5\*7+13\*74+6\*88+3\*103+0\*60+0\*85 = 3193

**Улучшение опорного плана *(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 Хij 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 10. Присваиваем одному из них произвольное u1=0

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 > 0

(1,1) ∆11 = 0 + 12 - 15 = -3

(1,5) ∆15 = 0 + 13 - 14 = -1

(2,1) ∆21 = -3 + 12 - 13 = -4

(2,3) ∆23 = -3 + 13 - 11 = -1

(2,4) ∆24 = -3 + 6 - 8 = -5

(2,6) ∆26 = -3 + 4 - 16 = -15

(3,2) ∆32 = -8 + 6 - 8 = -10

(3,3) ∆33 = -8 + 13 - 14 = -9

(3,4) ∆34 = -8 + 6 - 11 = -13

(3,6) ∆36 = -8 + 4 - 14 = -18

(4,2) ∆42 = 0 + 6 - 13 = -7

(4,5) ∆45 = 0 + 13 - 16 = -3

(4,6) ∆46 = 0 + 4 - 5 = -1

(5,2) ∆52 = -3 + 6 - 14 = -11

(5,3) ∆53 = -3 + 13 - 12 = -2

(5,5) ∆55 = -3 + 13 - 13 = -3

(5,6) ∆56 = -3 + 4 - 7 = -6

(6,1) ∆61 = -13 + 12 = -1

(6,2) ∆62 = -13 + 6 = -7

(6,4) ∆64 = -13 + 6 = -7

(6,6) ∆66 = -13 + 4 = -9

(1;2): 0 + 6 > 5; ∆12= 0 + 6 - 5 = 1 > 0  
(1;3): 0 + 13 > 9; ∆13= 0 + 13 - 9 = 4 > 0  
(4;1): 0 + 12 > 7; ∆41= 0 + 12 - 7 = 5 > 0  
(5;1): -3 + 12 > 6; ∆51= -3 + 12 - 6 = 3 > 0  
max(1,4,5,3) = 5

Выбираем клетку по максимальной оценке свободной клетки (4;1): 7

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 «-», «+», «-». Цикл необходим для корректного перераспределения груза между клетками при улучшении опорного плана методом потенциалов.

15) Начинаем с «+» в клетке (a4, b1).

Двигаемся по циклу:

(a4, b1) → «+» (добавляем поставки)

(a3, b1) → «−» (убираем поставки)

(a3, b5) → «+» (добавляем поставки)

(a6, b5) → «−» (убираем поставки)

(a6, b3) → «+» (добавляем поставки)

(a4, b3) → «−» (убираем поставки)

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| ПОТРЕБИТЕЛИ | b1 | b2 | b3 | b4 | B5 | b6 | ЗАПАСЫ |
| ПОСТАВЩИКИ |
| a 1 | **15** | **5** | **9** | **6[5]** | **14** | **4[166]** | **171** |
| a 2 | **13** | **3[110]** | **11** | **8** | **10[6]** | **16** | **116** |
| a 3 | **4[146][-]** | **8** | **14** | **11** | **5[7][+]** | **14** | **153** |
| a 4 | **7[+]** | **13** | **13[74][-]** | **6[88]** | **16** | **5** | **162** |
| a 5 | **6** | **14** | **12** | **3[103]** | **13** | **7** | **103** |
| a 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

16)

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| ПОТРЕБИТЕЛИ | b1 | b2 | b3 | b4 | B5 | b6 | ЗАПАСЫ |
| ПОСТАВЩИКИ |
| a 1 | **15** | **5** | **9** | **6[5]** | **14** | **4[166]** | **171** |
| a 2 | **13** | **3[110]** | **11** | **8** | **10[6]** | **16** | **116** |
| a 3 | **4[146-74=72]** | **8** | **14** | **11** | **5[7+74=81]** | **14** | **153** |
| a 4 | **7[0+74=74]** | **13** | **13[74-74=0]** | **6[88]** | **16** | **5** | **162** |
| a 5 | **6** | **14** | **12** | **3[103]** | **13** | **7** | **103** |
| a 6 | **0** | **0** | **0[60+74=134]** | **0** | **0[85-74=11]** | **0** | **145** |
| ПОТРЕБНОСТИ | **146** | **110** | **134** | **196** | **98** | **166** |  |

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

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 = 5  
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

17)

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
|  | v1=7 | v2=5 | 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[0]** | **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 ≤ 0.

(1,1) ∆11 = 0 + 7 - 15 = -8

(1,2) ∆12 = 0 + 5 - 5 = 0

(1,3) ∆13 = 0 + 8 - 9 = -1

(1,5 )∆15 = 0 + 8 - 14 = -6

(2,1) ∆21 = -2 + 7 - 13 = -8

(2,3) ∆23 = -2 + 8 - 11 = -5

(2,4) ∆24 = -2 + 6 - 8 = -4

(2,6) ∆26 = -2 + 4 - 16 = -14

(3,2) ∆32 = -3 + 10 - 17 = -10

(3,3) ∆33 = -3 + 17 - 23 = -9

(3,4) ∆34 = -3 + 15 - 20 = -8

(3,6) ∆36 = -3 + 13 - 23 = -13

(4,2) ∆42 = 0 + 5 - 13 = -8

(4,5) ∆45 = 0 + 8 - 16 = -8

(4,6) ∆46 = 0 + 4 - 5 = -1

(5,1) ∆51 = -3 + 7 - 6 = -2

(5,2) ∆52 = -3 + 5 - 14 = -12

(5,3) ∆53 = -3 + 8- 12 = -7

(5,5) ∆55 = -3 + 8 - 13 = -8

(5,6) ∆55 = -3 + 4 - 7 = -6

(6,1) ∆61 = -8 + 7 = -1

(6,2) ∆62 = -8 + 5 = -3

(6,4) ∆64 = -8 + 6 = -2

Минимальные затраты составят: F(x) = 6\*5+4\*166+3\*110+10\*6+4\*72+5\*81+7\*74+13\*0+6\*88+3\*103+0\*134+0\*11=3132  
**Анализ оптимального плана**.  
От 1-го поставщика необходимо груз направить к 4-у потребителю (5 ед.), к 6-у потребителю (166 ед.)  
От 2-го поставщика необходимо груз направить к 2-у потребителю (110 ед.), к 5-у потребителю (6 ед.)  
От 3-го поставщика необходимо груз направить к 1-у потребителю (72 ед.), к 5-у потребителю (81 ед.)  
От 4-го поставщика необходимо груз направить к 1-у потребителю (74 ед.), к 4-у потребителю (88 ед.)  
От 5-го поставщика необходимо весь груз направить к 4-у потребителю.

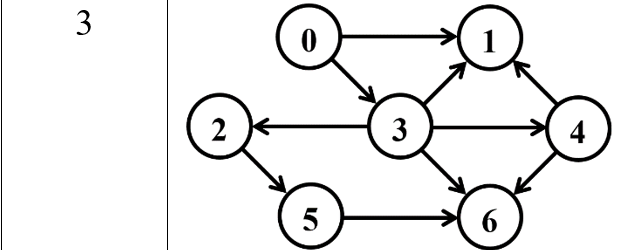
Минимальные затраты составят 3132 денежных единиц.

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



**Матрица смежности** — это способ представления графа в виде двумерной матрицы. Каждый элемент матрицы отображает наличие или отсутствие рёбер между вершинами графа.

**Матрица смежности:**

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| **Вершина** | **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 |



G = (V, E), V = {0, 1, 2, 3, 4, 5, 6}, E = { <0, 1>0, <0, 3>1, <3, 1>2, <3, 2>3, <3, 4>4, <3, 6>5, <2, 5>6, <5, 6>7, <1, 4>8, <4, 6>9}



7

9

8

5

4

3

2

6

0

1

если ребро исходит от вершины, то 1, если входит, то -1, 0 – если ничего

**Матрица инцидентности:**

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

Список смежных вершин - это список всех вершин, с которыми заданная вершина в графе имеет ребра. Другими словами, это список вершин, которые непосредственно соединены с данной вершиной ребрами.

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.

**Алгоритм поиска в ширину (англ. breadth-first search, BFS)** позволяет найти кратчайшие пути из одной вершины невзвешенного графа до всех остальных вершин

Белые вершины не посещены

Серый вершины в очереди

Голубые вершины посещаются

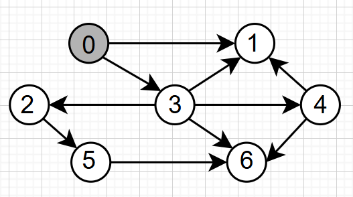
Оранжевый вершина является текущей вершиной

**Поиск в ширину:**

1. Посещенные вершины : {}

Очередь:{}

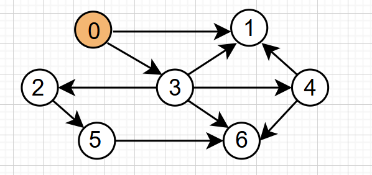
Текущая вершина: -



1. Посещенные вершины : {}

Очередь:{0}

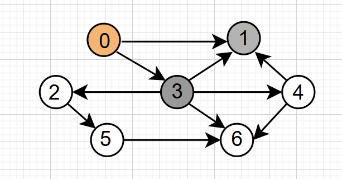
Текущая вершина: -



1. Посещенные вершины : {0}

Очередь:{}

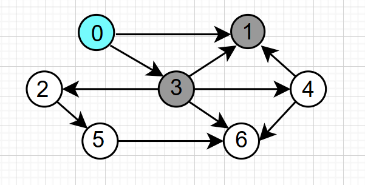
Текущая вершина: 0



1. Посещенные вершины : {0}

Очередь:{1,3}

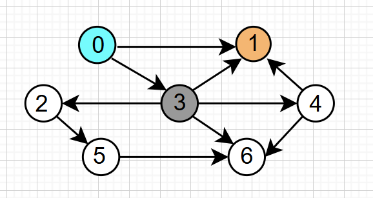
Текущая вершина: 0



1. Посещенные вершины : {0}

Очередь:{1,3}

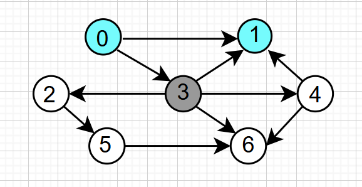
Текущая вершина: -



1. Посещенные вершины : {0,1}

Очередь:{3}

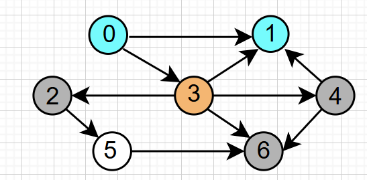
Текущая вершина: 1



1. Посещенные вершины : {0,1}

Очередь:{3}

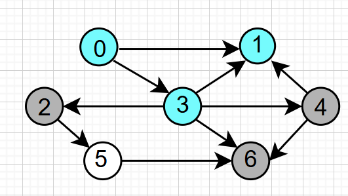
Текущая вершина: -



1. Посещенные вершины : {0,1,3}

Очередь:{2,4,6}

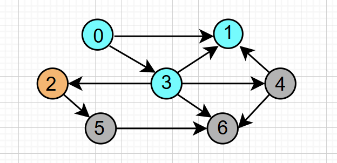
Текущая вершина: 3



1. Посещенные вершины : {0,1,3}

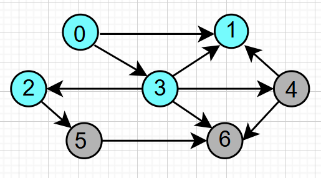
Очередь:{2,4,6}

Текущая вершина: -

1. Посещенные вершины : {0,1,3,2}

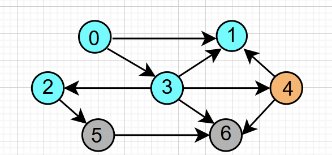
Очередь:{4,6,5}

Текущая вершина: 2

1. Посещенные вершины : {0,1,3,2}

Очередь:{4,6,5}

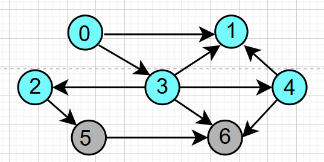
Текущая вершина: -



1. Посещенные вершины : {0,1,3,2,4}

Очередь:{6,5}

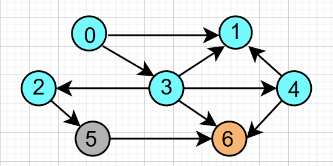
Текущая вершина: 4



1. Посещенные вершины : {0,1,3,2,4}

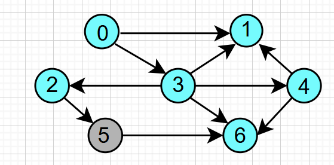
Очередь:{6,5}

Текущая вершина: -

1. Посещенные вершины : {0,1,3,2,4,6}

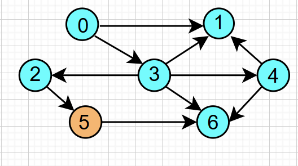
Очередь:{5}

Текущая вершина: 6

1. Посещенные вершины : {0,1,3,2,4,6}

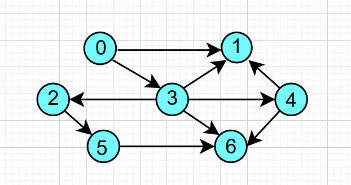
Очередь:{5}

Текущая вершина: -

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

Очередь:{}

Текущая вершина: 5

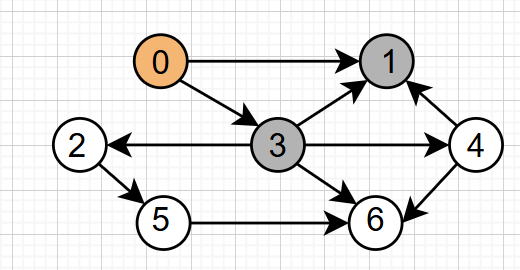
1. Посещенные вершины : {0,1,3,2,4,6,5}

Очередь:{}

Текущая вершина: -

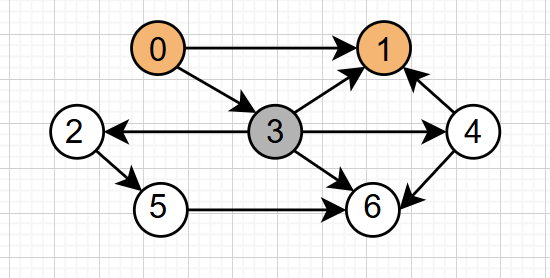
Порядок обхода : 0,1,3,2,4,6,5

**Поиск в глубину**

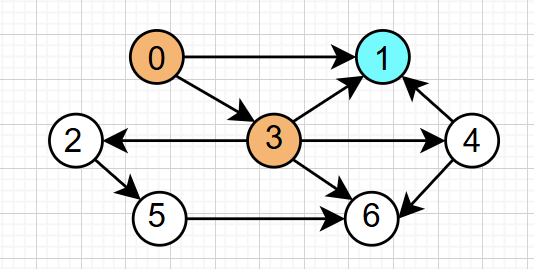
1. ****

Начинаем с первой доступной вершины – в данном случае это вершина 0. Помещаем 0 в стек и помечаем её как посещённую).

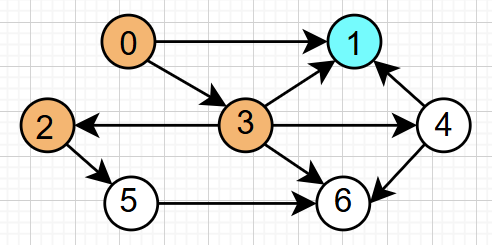
стек: {0}

1. 

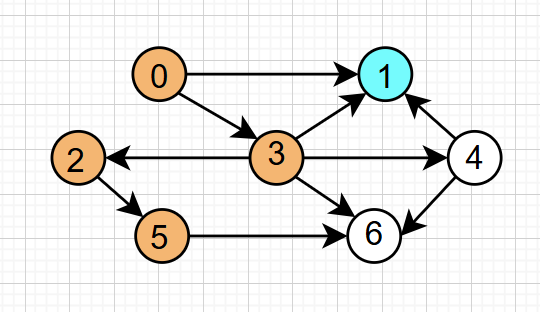
Спуск к вершине 1. Смотрим верхний элемент стека – сейчас это 0. У вершины 0 имеются смежные вершины 1 и 3. Берём первую из них по порядку, вершину 1, которая ещё не посещена. Помещаем 1 в стек (стек: {0, 1}). Вершина 1 не имеет непосещённых соседей (у 1 нет исходящих рёбер), поэтому извлекаем 1 из стека и выводим 1 – это первый элемент в результирующей последовательности вывода.

3) 

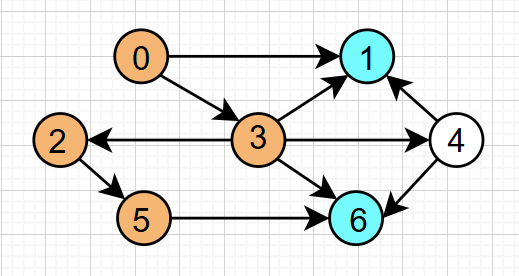
Переход к вершине 3. Теперь верхним элементом стека снова стала 0 (после удаления 1). У вершины 0 больше нет непосещённых соседей, кроме уже рассмотренной 1, поэтому переходим к следующему смежному узлу 0 – вершине 3. Помещаем 3 в стек и отмечаем как посещённую (стек: {0, 3}).

4) 

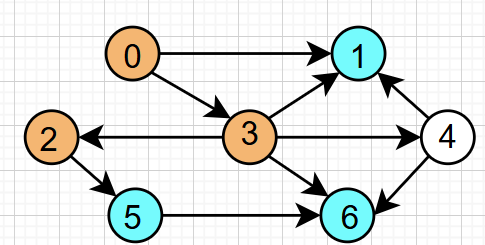
Спуск к вершине 2. Рассматриваем верхний элемент стека – вершину 3. Её смежные вершины (в заданном порядке) – 1, 2, 4, 6. Сосед 1 уже был посещён ранее, поэтому пропускаем его и берём следующий непосещённый сосед, вершину 2. Помещаем 2 в стек (стек: {0, 3, 2}).

5) 

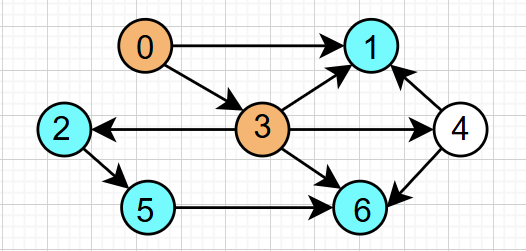
Спуск к вершине 5. Верхний элемент стека теперь 2. У вершины 2 единственный смежный узел – 5, который ещё не посещён. Помещаем вершину 5 в стек (стек: {0, 3, 2, 5}).

6) 

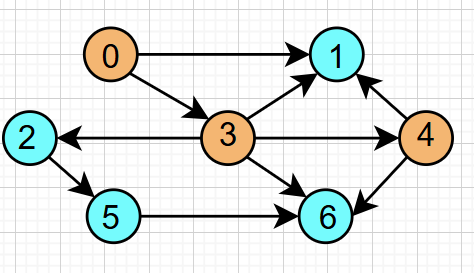
Спуск к вершине 6. Верхушка стека – вершина 5. Её смежная вершина 6 ещё не посещена. Помещаем 6 в стек (стек: {0, 3, 2, 5, 6}). Вершина 6 не имеет исходящих рёбер (смежных вершин нет), поэтому извлекаем 6 из стека и выводим 6 – это следующий элемент в результирующей последовательности.

7) 

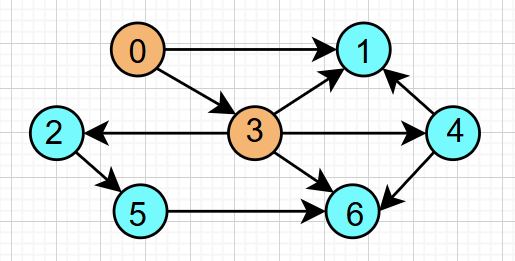
Возврат к вершине 5. После удаления 6 верхним элементом стека снова стала 5. У вершины 5 больше нет непосещённых соседей (единственный сосед 6 уже обработан), поэтому извлекаем 5 и выводим 5.

8) 

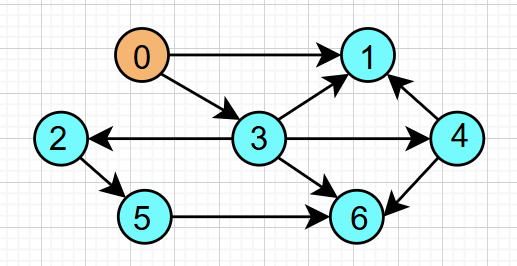
Возврат к вершине 2. Теперь верхушка стека – вершина 2. У 2 не осталось непосещённых соседей (она уже обработала 5), поэтому извлекаем 2 и выводим 2.

9) 

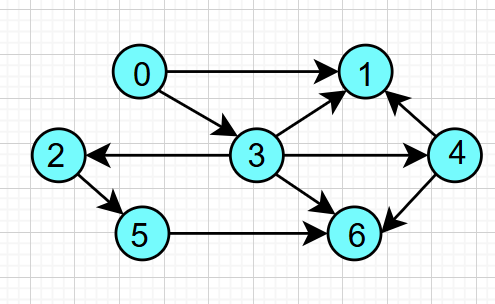
Возврат к вершине 3 и переход к 4. Верхним элементом стека снова стала 3. Мы уже посетили соседей 1 и 2 для вершины 3; следующий смежный узел по порядку – вершина 4 (непосещённая). Помещаем 4 в стек (стек: {0, 3, 4}).

10) 

Вершина 4 не имеет новых непосещённых соседей (её смежные 1 и 6 уже были посещены ранее), поэтому сразу извлекаем 4 и выводим 4.

11) 

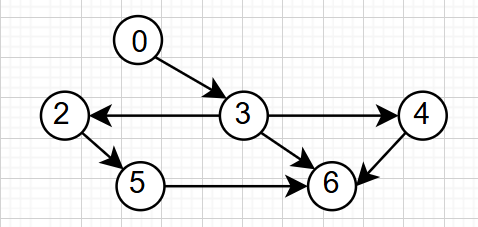
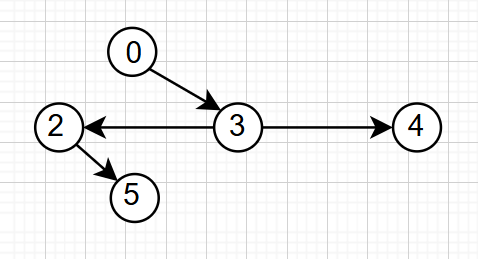
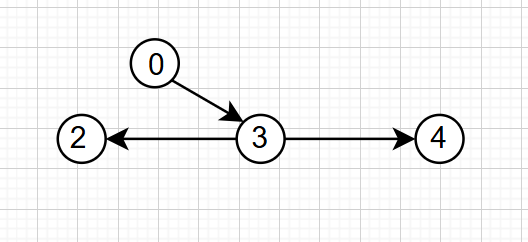
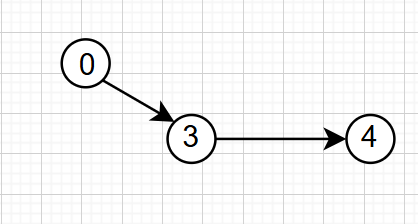
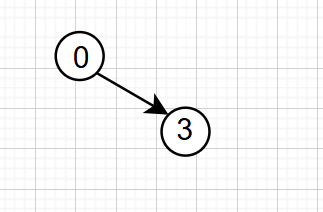
Возврат к вершине 3. После удаления 4 верхним элементом стека снова является 3. У вершины 3 больше нет непосещённых соседей (оставшийся сосед 6 уже был посещён). Поэтому извлекаем 3 из стека и выводим 3.

12) 

Завершение обхода на вершине 0. Верхушка стека вновь стала 0. У вершины 0 не осталось непосещённых соседей (оба её соседа, 1 и 3, уже были обработаны), поэтому извлекаем 0 и выводим 0.

Итоговая последовательность вывода получается: 1 6 5 2 4 3 0

**Топологическая сортирвка**

1. ** 1**
2. ** 6**
3. ** 5**
4. ** 2**
5. **4**

Результат: 1 6 5 2 4 3 0

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

|  |
| --- |
| #include "BFS.h"  void BFS::init(const graph::AList& al, int s)  {  this->al = &al;  this->c = new Color[this->al->n\_vertex];  this->d = new int[this->al->n\_vertex];  this->p = new int[this->al->n\_vertex];  for (int i = 0; i < this->al->n\_vertex; i++)  {  this->c[i] = WHITE;  this->d[i] = INF;  this->p[i] = NIL;  };  this->c[s] = GRAY;  this->q.push(s);  };  BFS::BFS(const graph::AList& al, int s) { this->init(al, s); };  BFS::BFS(const graph::AMatrix& am, int s)  {  this->init(\*(new graph::AList(am)), s);  };  int BFS::get()  {  int rc = NIL, v = NIL;  if (!this->q.empty())  {  rc = this->q.front();  for (int j = 0; j < this->al->size(rc); j++)  if (this->c[v = this->al->get(rc, j)] == WHITE)  {  this->c[v] = GRAY;  this->d[v] = this->d[rc] + 1;  this->p[v] = rc;  this->q.push(v);  };  this->q.pop();  this->c[rc] = BLACK;  };  return rc;  } |

Листинг 3.1 — файл BFS.cpp

|  |
| --- |
| #pragma once  #include "Graph.h"  #include <queue>  struct BFS  {  const static int INF = 0x7fffffff;  const static int NIL = -1;  enum Color { WHITE, GRAY, BLACK }; //  const graph::AList\* al; // исходный граф  Color\* c; // цвет вершины  int\* d; // расстояние до вершины  int\* p; // предшествующая вершина  std::queue<int> q; // очередь  BFS(const graph::AList& al, int s);  BFS(const graph::AMatrix& am, int s);  void init(const graph::AList& al, int s);  int get(); // получить следующую вершину  }; |

Листинг 3.2 — файл BFS.h

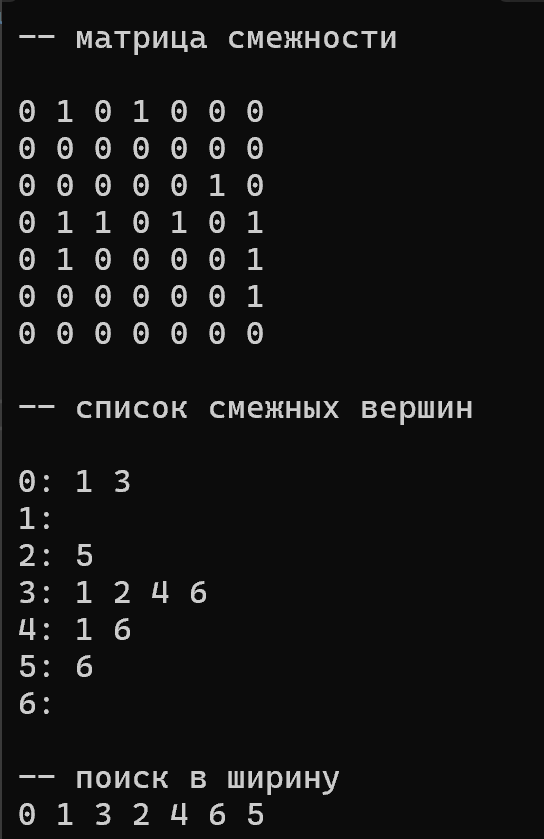


Рисунок 3.1 – результат программы

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

|  |
| --- |
| #include "DFS.h"  #define NINF 0x80000000  #define INF 0x7fffffff  void DFS::init(const graph::AList& al)  {  this->al = &al;  this->c = new Color[this->al->n\_vertex];  this->d = new int[this->al->n\_vertex];  this->f = new int[this->al->n\_vertex];  this->p = new int[this->al->n\_vertex];  this->t = 0;  for (int i = 0; i < this->al->n\_vertex; i++)  {  this->c[i] = WHITE;  this->d[i] = this->f[i] = 0;  this->p[i] = NIL;  };  for (int i = 0; i < this->al->n\_vertex; i++)  if (this->c[i] == WHITE)  {  this->visit(i);  this->topological\_sort.push\_back(i);  }  };  DFS::DFS(const graph::AList& al) { this->init(al); };  DFS::DFS(const graph::AMatrix& am)  {  this->init(\*(new graph::AList(am)));  };  void DFS::visit(int u)  {  int v = NIL;  this->c[u] = GRAY;  this->d[u] = ++(this->t);  for (int j = 0; j < this->al->size(u); j++)  if (this->c[v = this->al->get(u, j)] == WHITE)  {  this->p[v] = u;  this->visit(v);  this->topological\_sort.push\_back(v);  }  this->c[u] = BLACK;  this->f[u] = ++(this->t);  };  int DFS::get(int i)  {  int j = 0, min1 = INF, min2 = NINF, ntx = NIL;  for (int j = 0; j <= i; j++)  {  for (int k = 0; k < this->al->n\_vertex; k++)  if (this->f[k] < min1 && this->f[k] > min2)  {  min1 = this->f[k]; ntx = k;  };  min2 = min1; min1 = INF;  };  return ntx;  }; |

Листинг 4.1 — файл DFS.cpp

|  |
| --- |
| #pragma once  #include "Graph.h"  #include <vector>  struct DFS  {  const static int NIL = -1;  enum Color { WHITE, GRAY, BLACK }; //  const graph::AList\* al; // исходный граф  Color\* c; // цвет вершины  int\* d; // время обнаружения  int\* f; // время завершения обработки  int\* p; // предшествующая вершина  int t; // текущее время  DFS(const graph::AList& al);  DFS(const graph::AMatrix& am);  std::vector <int> topological\_sort; //результат топологической сортировки  void visit(int v);  void init(const graph::AList& al);  int get(int i); // получить вершину  }; |

Листинг 4.2 — файл DFS.h

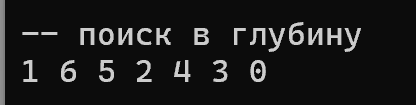


Рисунок 4.1 – результат программы

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

Реализация кода представлена в предыдущем задании, а результат выполнения – на рисунке 5.1.

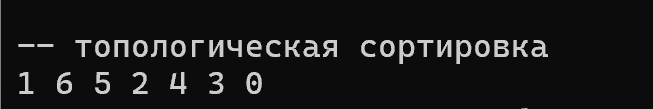


Рисунок 5.1 - результат программы

***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;

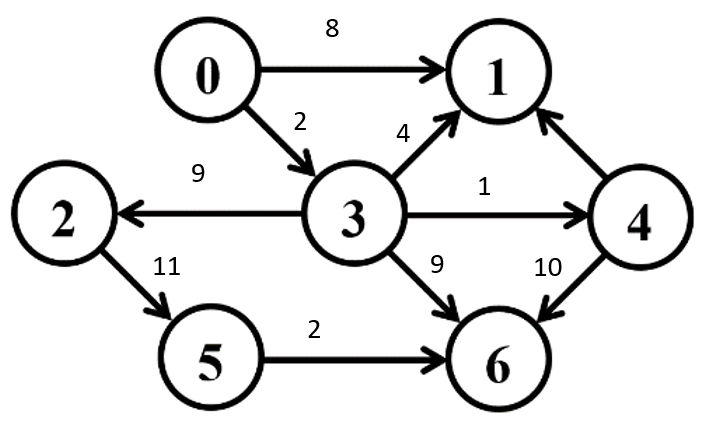
W(e4,3)=4; W(e3,4)=1;

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

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

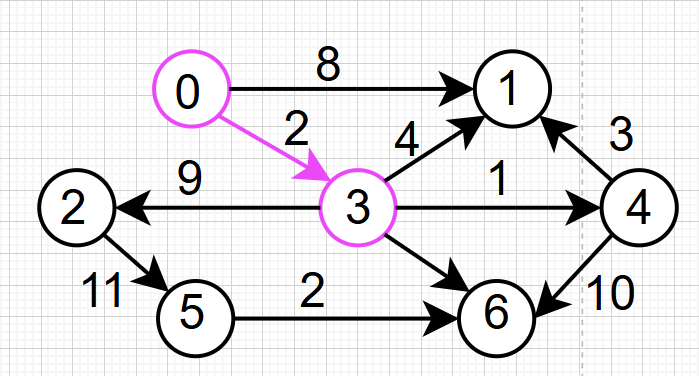
W(e5,3)=3; W(e3,5)=6;

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



Алгоритм Прима строит минимальное остовное дерево, начиная с произвольной вершины и постепенно расширяя дерево, выбирая минимальные рёбра, которые соединяют дерево с вершинами, ещё не входящими в дерево.

1. Выбираем вершину 0



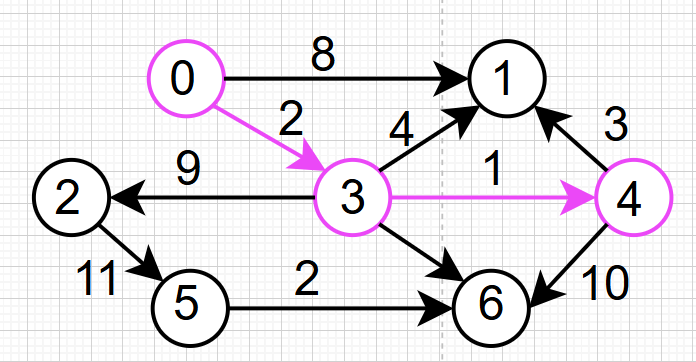
Начинаем с вершины 0. Из вершины 0 есть рёбра:

- 0→1 с весом 8

- 0→3 с весом 2

Выбираем минимальное ребро: 0→3 с весом 2.

**Посещённые вершины**: {0, 3}

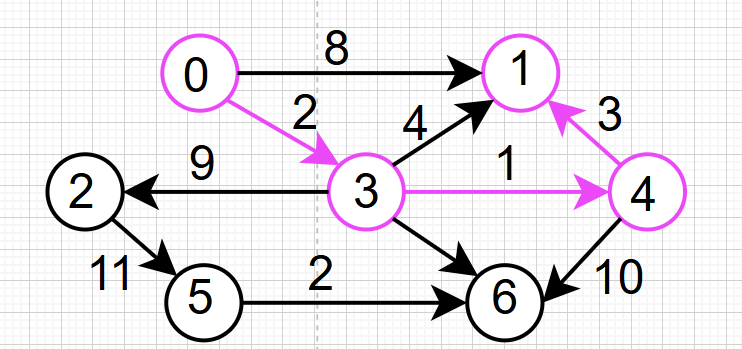
2) 

**Обработка вершины 3**: Из вершины 3 есть рёбра:

* 3→1 с весом 4
* 3→2 с весом 9
* 3→4 весом 1
* 3→6 с весом 9

Выбираем минимальное ребро: 3→4 с весом 1.

**Посещённые вершины**: {0, 3, 4}

**3)** ****

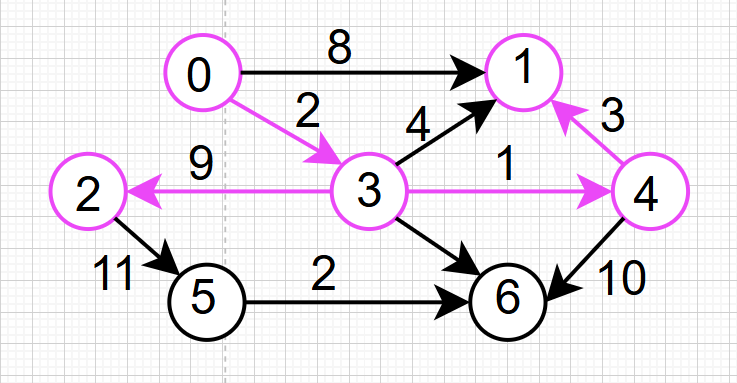
**Обработка вершины 4**: Из вершины 444 есть рёбра:

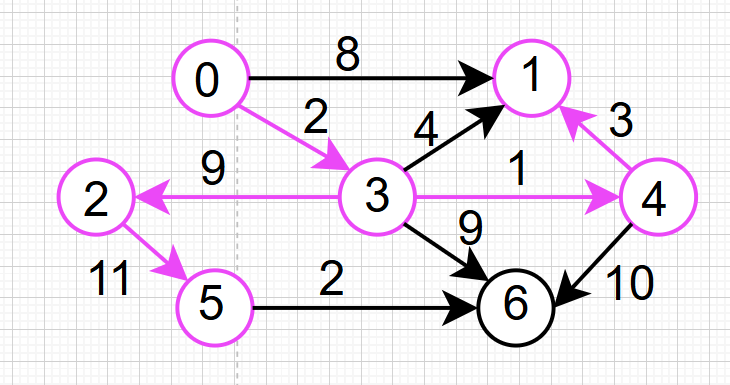
* 4→1 с весом 3
* 4→6 с весом 10

Выбираем минимальное ребро: 4→1 с весом 3.

**Посещённые вершины**: {0, 3, 4, 1}

4) Из вершины 1 больше нет исходящих рёбер (она не соединена с другими вершинами), поэтому переходим к следующему шагу.

5) 

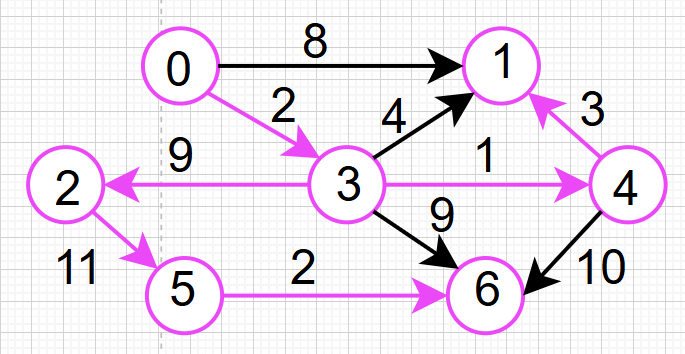
6) 

Из вершины 2 есть рёбра:

* 2→5 с весом 11

Выбираем минимальное ребро: 2→5 с весом 11.

**Посещённые вершины**: {0, 3, 4, 1, 2, 5}

**7) **

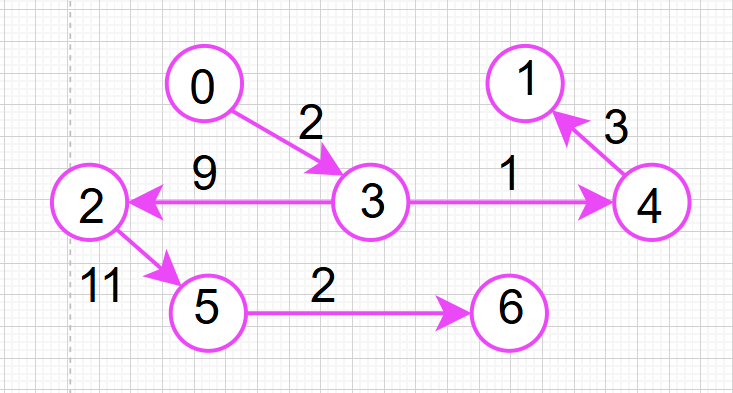
Из вершины 5 есть рёбра:

* 5→6 с весом 2

Выбираем минимальное ребро: 5→6 с весом 2.

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

**Результат** :

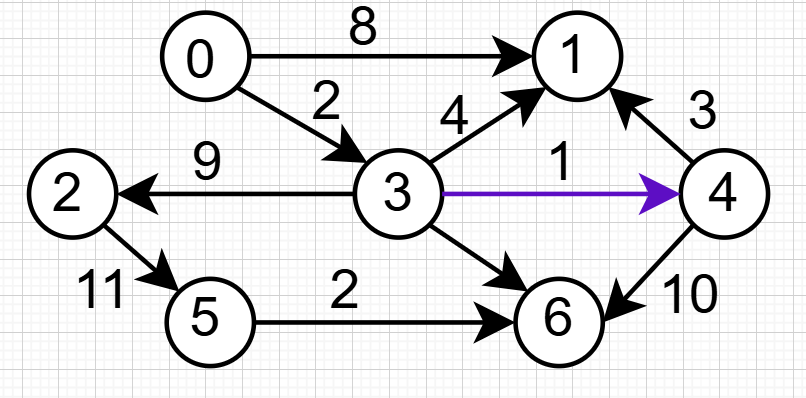


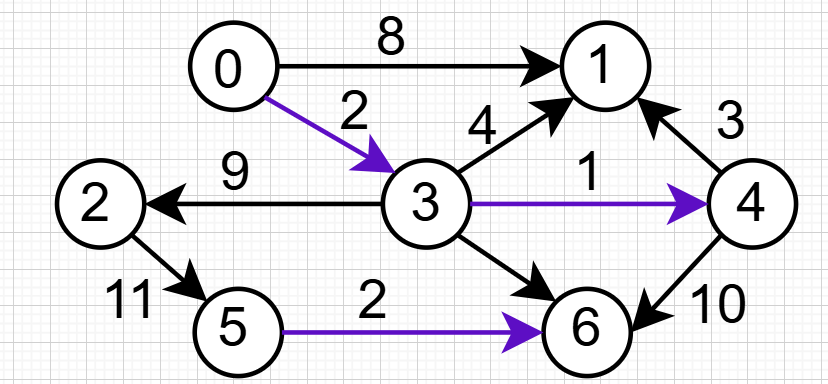
Вес минимального остовного дерева: (0→3),(3→4),(4→1),(3→2),(2→5) ,(5→6) = 2+1+3+9+11+2=28

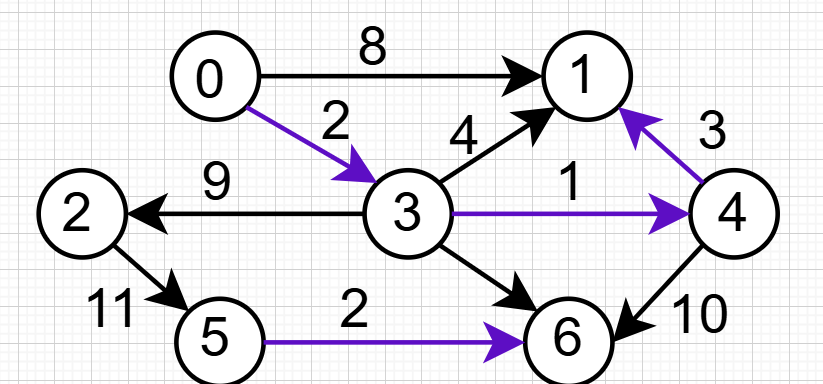
***Задание 7.*** По графу, соответствующему варианту составить минимальное остовное дерево по алгоритму Крускала. Шаги построения отразить в отчете.

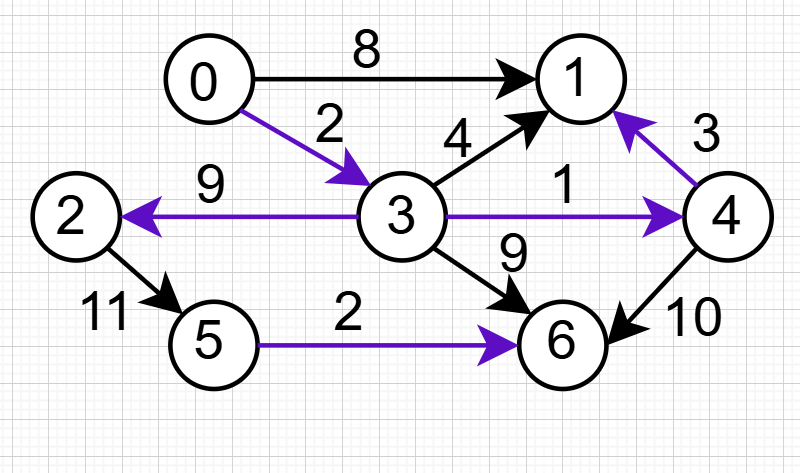
Алгоритм **Крцскала** — это жадный алгоритм для нахождения минимального остовного дерева в графе. Он строит остовное дерево, начиная с рёбер минимального веса и добавляя их в дерево, если они не образуют цикл.

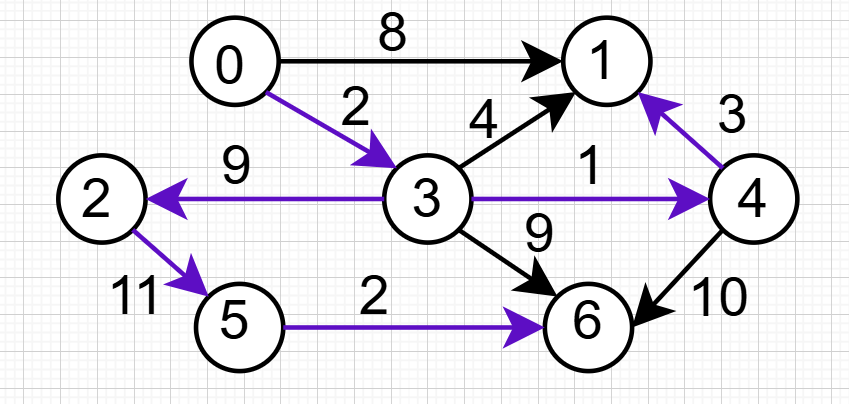
Из всех ребер выбираем ребро с минимальной стоимостью (весом) и включаем это ребро в остовное дерево. Повторяем так далее, однако смотрим на то, чтобы следующее ребро соединяло вершины из разных компонент (чтобы не было циклов)

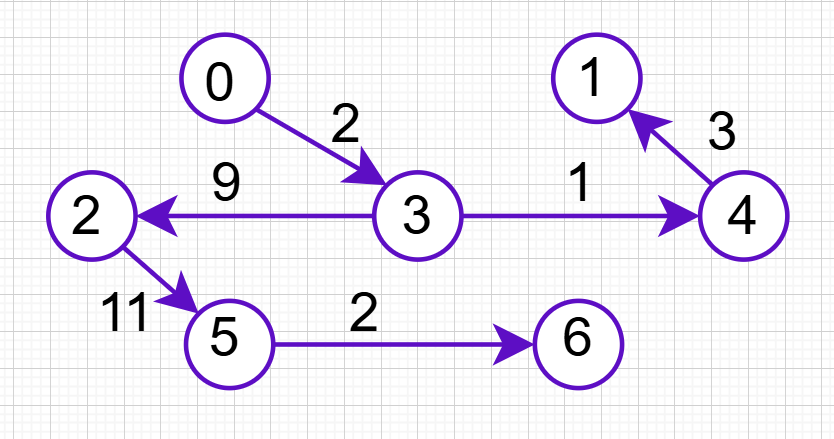
1) 

2)-3) 

4) 

5) 

6) 



2+1+3+9+11+2=28

**Вывод:** была освоена сущность и программная реализация: способов представления графов; алгоритмов поиска в ширину и глубину; алгоритма топологической сортировки графов. Был 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 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.

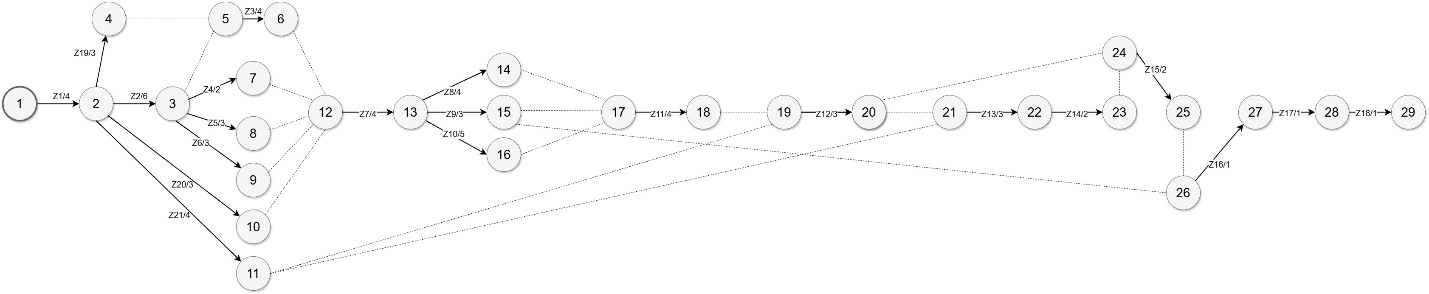
**Задание 2. Календарное планирование.**

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

|  |  |  |  |
| --- | --- | --- | --- |
| Код  операции | Наименование операции | Предшествующие операции | t |
| I. АНАЛИЗ | | | |
| Z1 | Системный анализ |  | 4 |
| Z2 | Анализ требований | Z1 | 6 |
| II. ПРОЕКТИРОВАНИЕ­­ | | | |
| Z3 | Проектирование базы данных | Z2, Z19 | 4 |
| Z4 | Проектирование UI/UX | Z2 | 2 |
| Z5 | Проектирование мер защиты и безопасности | Z2 | 3 |
| Z6 | Проектирование моделей взаимодействия (диаграммы, схемы) | Z2 | 3 |
| III. Разработка | | | |
| Z7 | Программирование функционала аутентификации(пароли) | Z3,Z4,Z5,Z6,Z20 | 4 |
| Z8 | Программирование функционала перевода средств | Z7 | 4 |
| Z9 | Разработка интерфейса для просмотра баланса и транзакций | Z7 | 3 |
| Z10 | Реализация функций безопасности | Z7 | 5 |
| Z11 | Подключение платежных систем и API(возможность принимать платежи через интернет) | Z8,Z9,Z10 | 4 |
| IV. ТЕСТИРОВАНИЕ | | | |
| Z12 | Функциональное тестирование | Z11, Z21 | 3 |
| Z13 | Структурное тестирование | Z12, Z21 | 3 |
| Z14 | Отладка и устранение ошибок | Z13 | 2 |
| V. ВНЕДРЕНИЕ | | | |
| Z15 | Разработка документации | Z12,Z14 | 2 |
| Z16 | Обучение пользователей | Z9,Z15 | 1 |
| Z17 | Финальные испытания и проверка | Z16 | 1 |
| Z18 | Завершение проекта и передача в эксплуатацию | Z17 | 1 |
| VI. ДОПОЛНИТЕЛЬНЫЕ РАБОТЫ | | | |
| Z19 | Установка СУБД | Z1 | 3 |
| Z20 | Установка web-сервера | Z1 | 3 |
| Z21 | Подготовка полигона | Z1 | 4 |

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

Найдите критический путь в составленном вами сетевом графике и обоснуйте его нахождение. Критический путь может быть меньше, чем время, отведенное на выполнение всех задач. Выделите, какие операции принадлежат критическому пути.



Критическим путём считается самая длинная последовательность зависимых операций (с максимальной суммой длительностей), от которой зависит общее время проекта.(нахожим просто максим время выполенния у z)

Z1 (4) → Z2 (6) → Z3 (4) → Z7 (4) → Z10 (5) → Z11 (4) → Z12 (3) → Z13 (3) → Z14 (2) → Z15 (2) → Z16 (1) → Z17 (1) → Z18 (1)

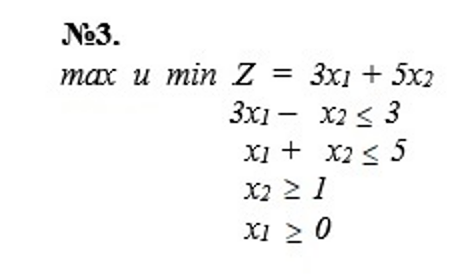
Сумма: 4 + 6 + 4 + 4 + 5 + 4 + 3 + 3 + 2 + 2 + 1 + 1 + 1 = 40 дней

**Вывод:** в ходе выполнения данной лабораторной работы, были изучены основные принципы и методы сетевого планирования, включая метод критического. Выбран был метод критического пути, так как выполнялись работы, которые имеют строго определенные продолжительности.

# **Лабораторная работа №8. Графический метод решения оптимизационных задач.**

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

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



Строим 3x-y ≤ 3

Чтобы построить график прямой 3x-y = 3 нам нужно найти ее точки пересечения с осями x и y.

Точка пересечения с осью x: мы берем y = 0,

3x - 0 = 3

x = 1.

Следовательно, (1; 0).

Точка пересечения с осью y: мы берем x = 0,

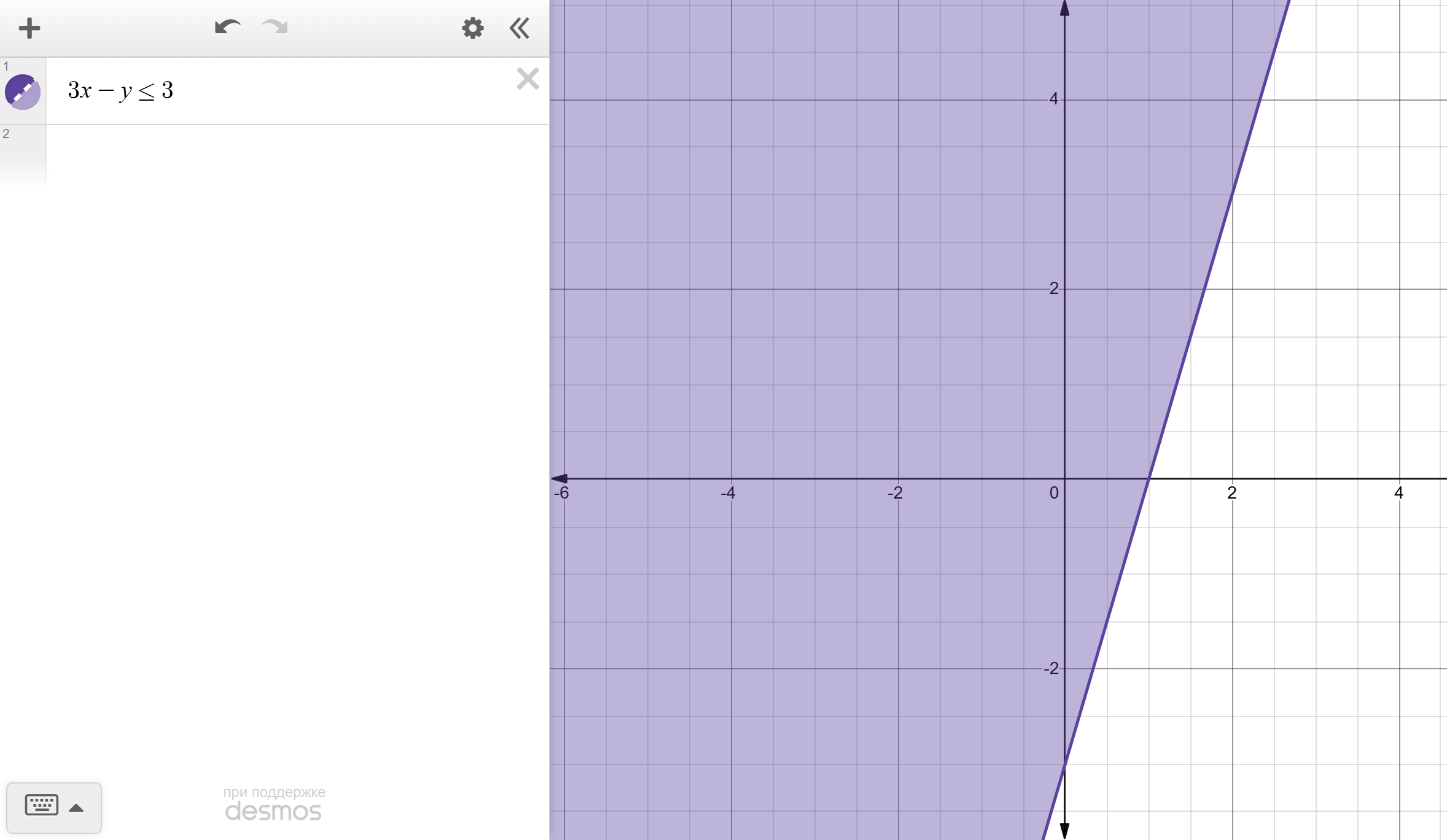
3\*0 – y =3

-y = 3

y=-3

Следовательно (0; -3)

Прямая 3x-y ≤ 3 будет проходить через точки (1; 0), (0; -3).



Строим x + y ≤ 5

Чтобы построить график прямой x + y = 5 нам нужно найти ее точки пересечения с осями X и Y.

Точка пересечения с осью X: мы берем y = 0,

x + 0 = 5

x = 5

Следовательно (5; 0)

Точка пересечения с осью Y: мы берем x = 0,

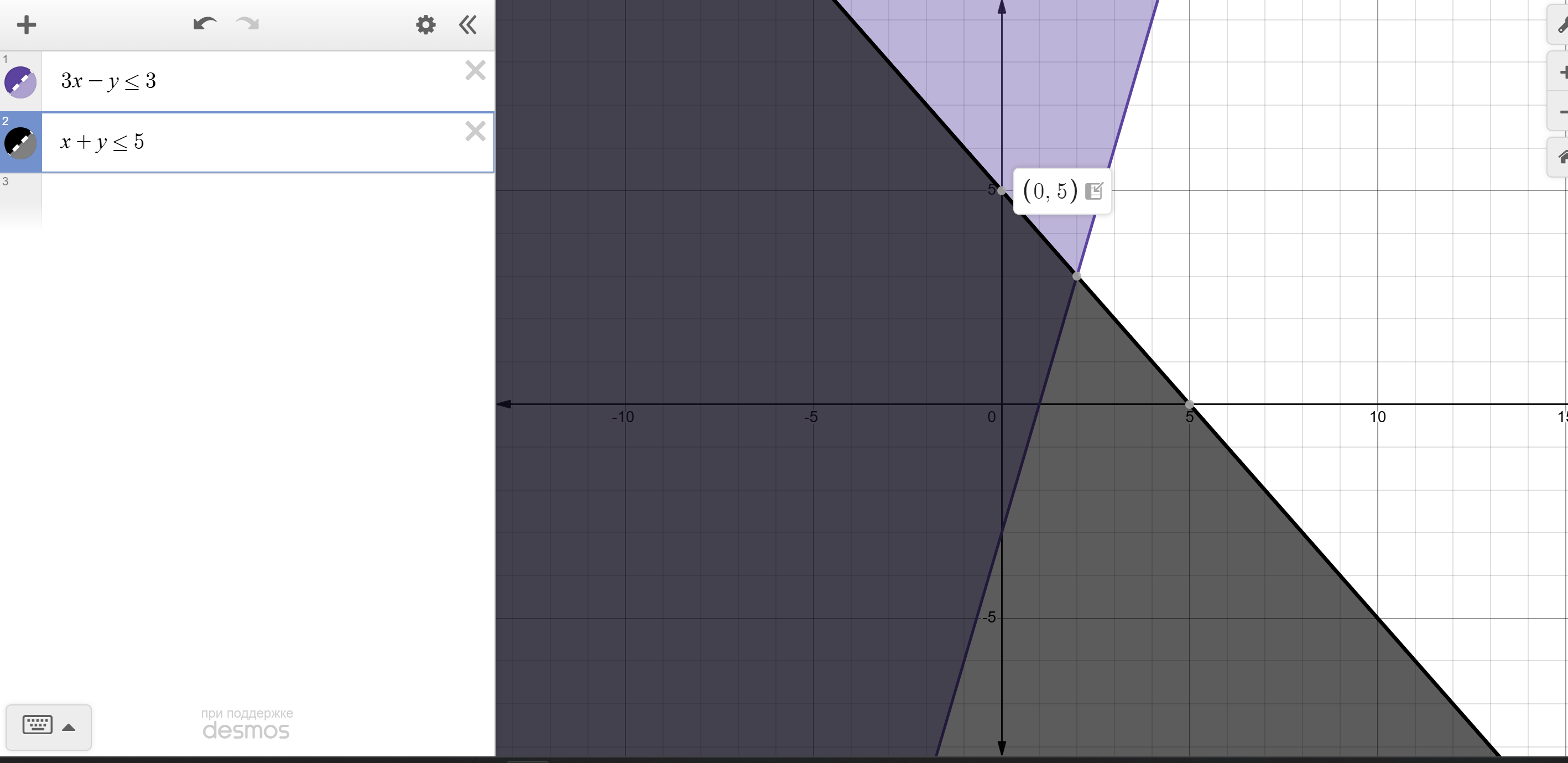
0 + y = 5

y = 5

y = 5

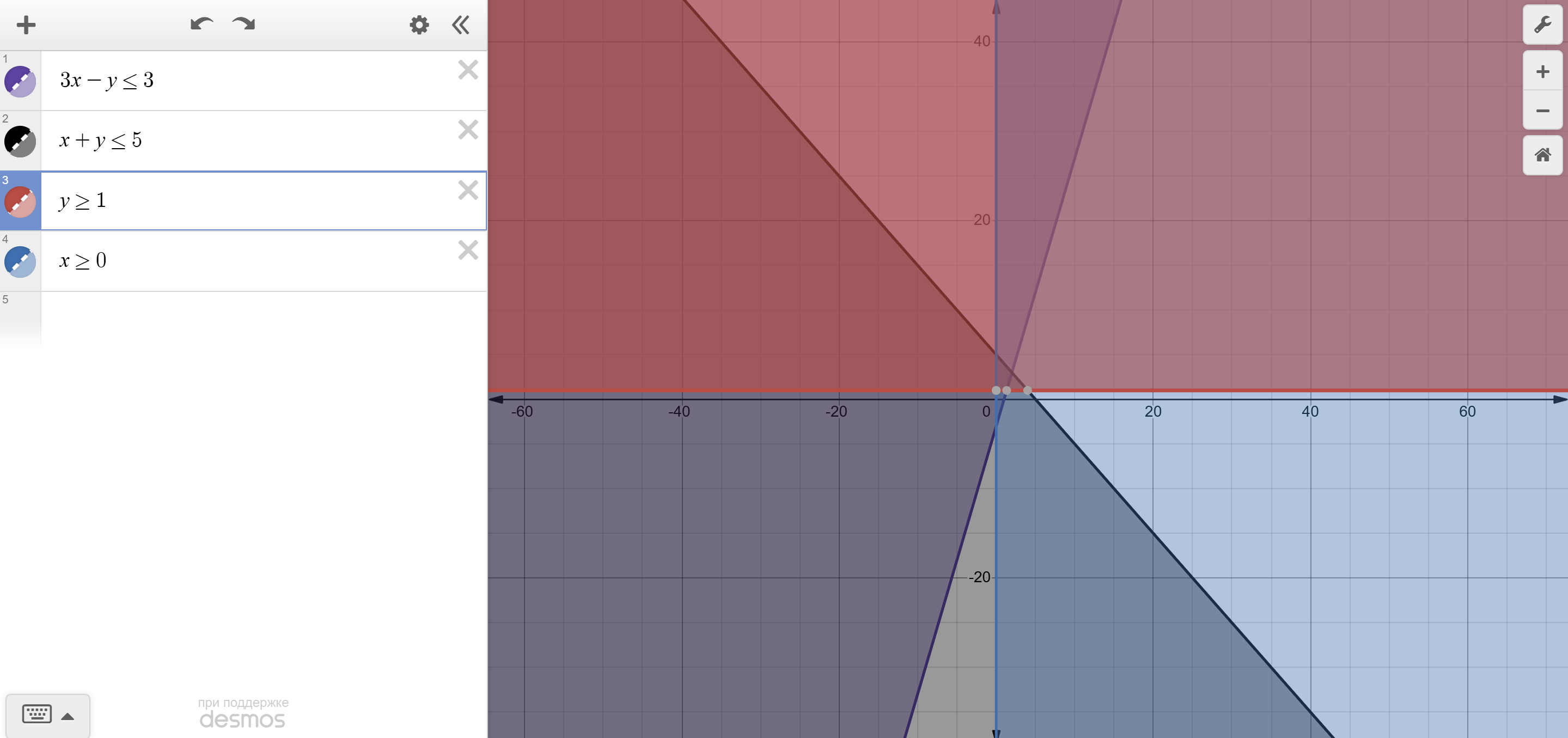
Следовательно (0; 5)

Прямая x + y = 5 будет проходить через точки (5; 0), (0; 5).



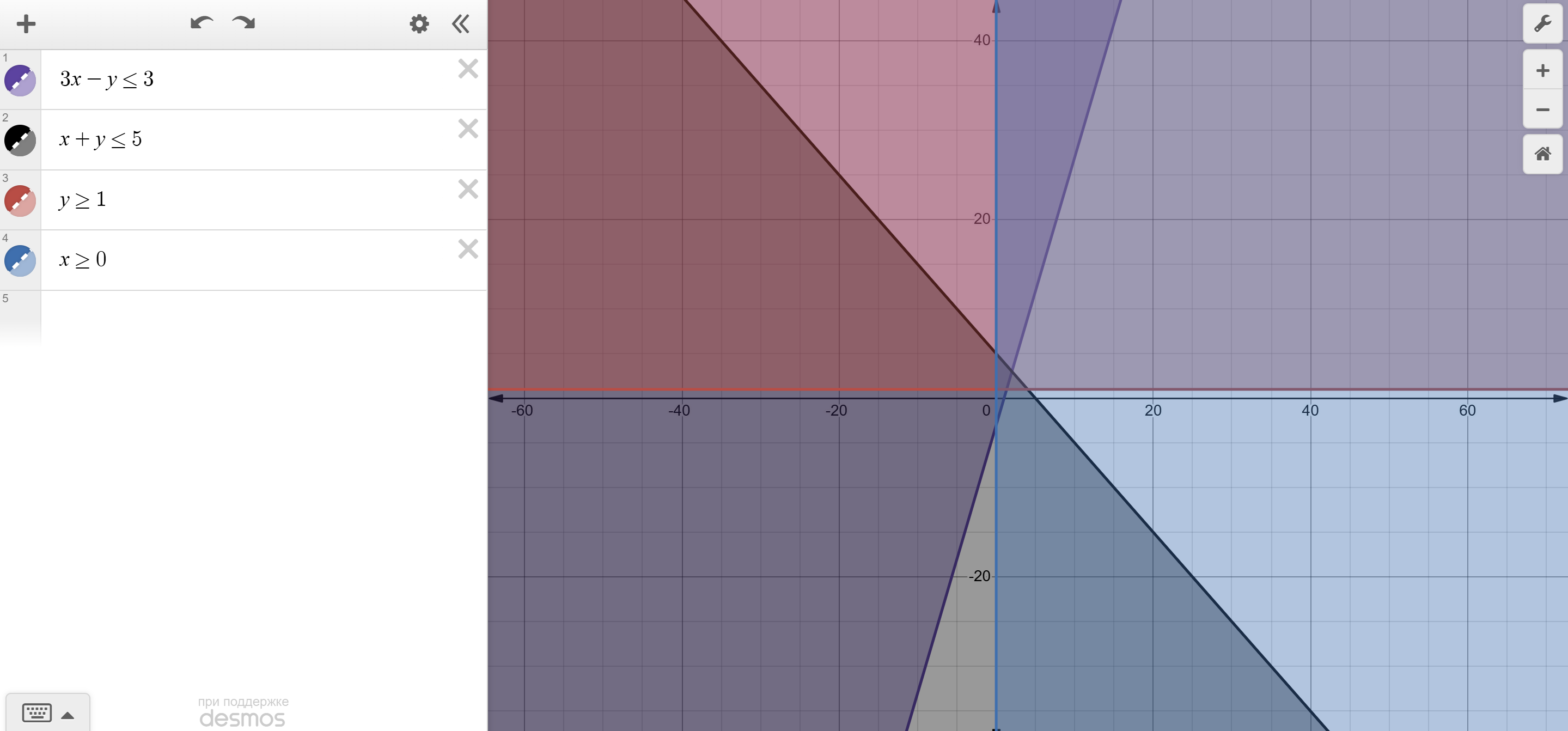
Строим y ≥ 0.

Это параллельная оси x прямая, которая пересекает ось y только в одной точке (0; 1).

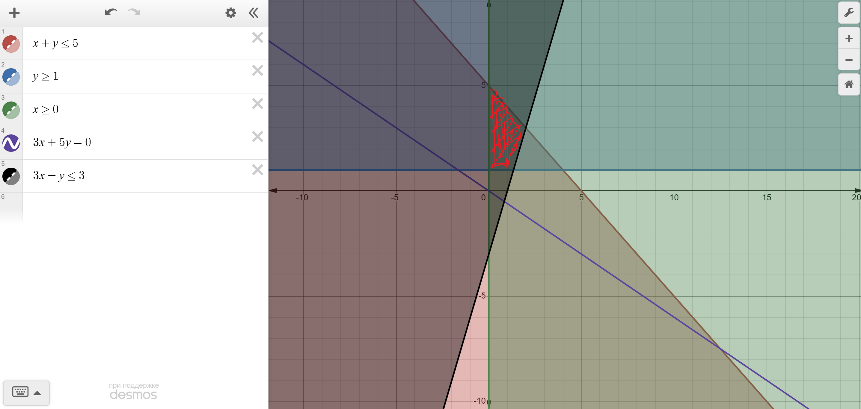


Строим x ≥ 1

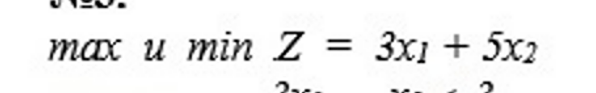
Это параллельная оси y прямая, которая лежит на ней и пересекает ось x только в одной точке (0; 0).



Пересечением полуплоскостей будет являться область, координаты точек которого удовлетворяют условию неравенствам системы ограничений задачи. Обозначим границы области многоугольника решений.



Тогда находим F (максимальную и минимальную). Напомним, что формула показана .



Минимальная точка (0;1)

Максимальная (0;5)

(x + y =5; x = 0); 0 + y = 5; y=5

min = 3\*0+5\*1 = 5

max =3\*0+5\*5 = 25