**Лабораторная работа №1. Вспомогательные функции**

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

**Ход работы**

## **1.1. Написание кода программы.**

Код файла lab1.cpp приведен на рисунке 1.1.

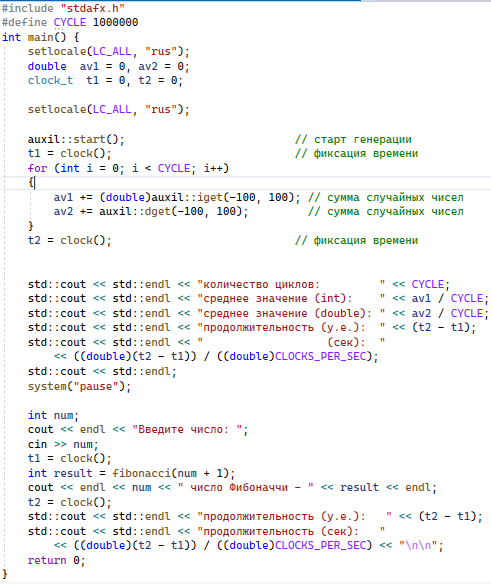


Рисунок 1.1 — Содержание файла main.cpp

Код файла auxil.h приведен на рисунке 1.2.

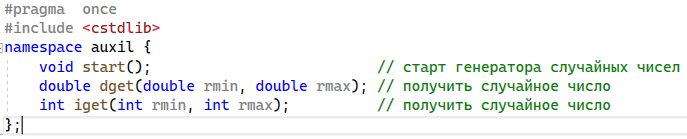


Рисунок 1.2 — Содержание файла auxil.h

Код файла auxil.cpp приведет в рисунке 1.3.

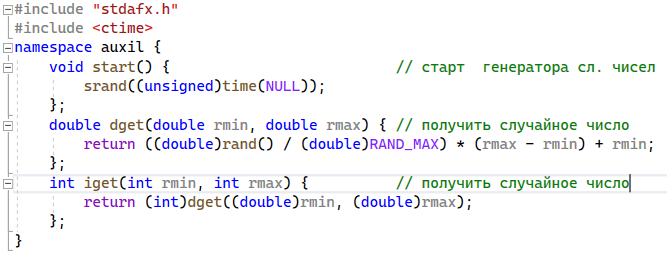


Рисунок 1.3 — Содержание файла auxil.cpp

Результат работы представлен на рисунке 1.4.

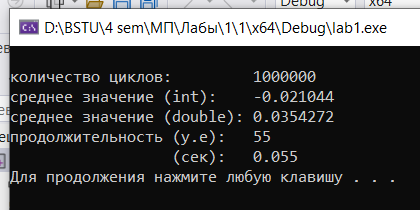


Рисунок 1.4 — Результат работы кода

## **1.2. Измерение скорости выполнения функции генерации случайных чисел.**

Результаты измерений и соответствующий график приведены на рисунке 1.8.

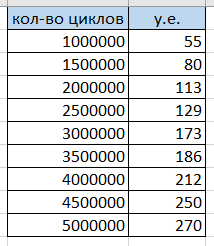


Рисунок 1.8 — Результаты измерений и их график

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

Также был проверен ещё один рекурсивный алгоритм — числа Фибоначчи.

## **1.3. Измерение скорости вычисления n числа.**

Код файла fibonacci.h приведен на рисунке 1.5.



Рисунок 1.5 — Содержание файла fibonacci.h

Код файла fibonacci.cpp приведет в рисунке 1.6.

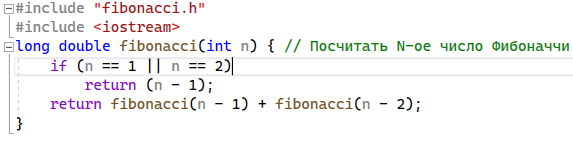


Рисунок 1.6 — Содержание файла fibonacci.cpp

Результат работы представлен на рисунке 1.7.

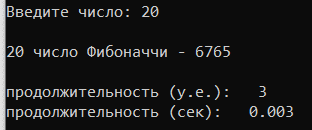


Рисунок 1.7 — Результат работы кода

Результаты измерений и соответствующий график приведены на рисунке 1.9.

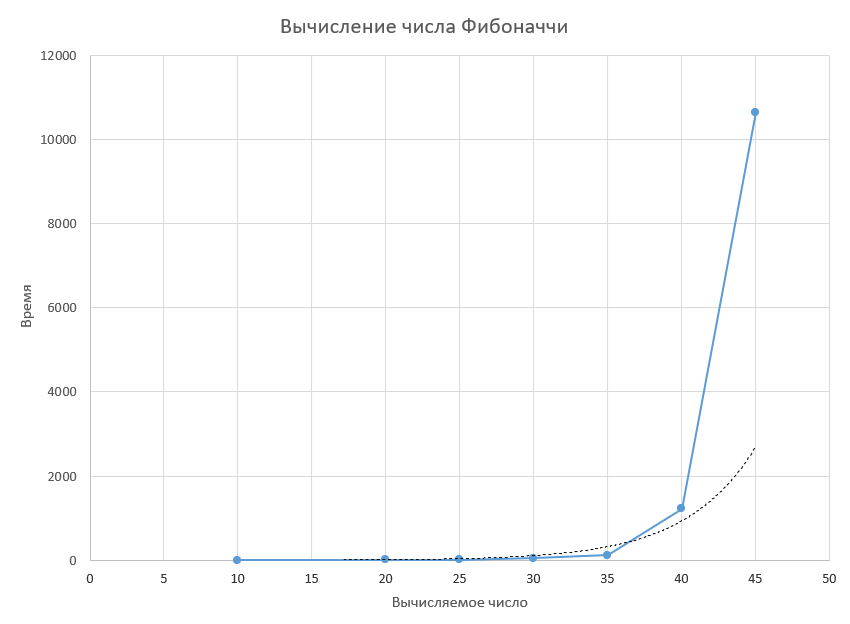


Рисунок 1.9 — Результаты измерений и их график

Вывод: сложность данного алгоритма является экспоненциальной. Это подтверждает и график.

**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 задачи об оптимальной загрузке судна с центровкой.

**Ход работы**

## **2.1. Написание кода генератора подмножеств заданного множества.**

Код файла combi.h приведен в листинге 2.1.

|  |
| --- |
| #pragma once  namespace combi {  struct subset { // генератор множества всех подмножеств  short n, // кол-во элементов исходного множества < 64  sn, // кол-во элементов текущего подмножества  \* sset; // массив индексов текущего подмножества  unsigned \_\_int64 mask; // битовая маска  subset(short n = 1); // конструктор (кол-во элементов исх. мн-ва)  short getfirst(); // сформировать массив индексов по бит. маске  short getnext(); // ++маска и сформировать массив индексов  short ntx(short i); // получить i-й элемент массива индексов  unsigned \_\_int64 count(); // вычислить общее количество подмножеств  void reset(); // сбросить генератор, начать сначала  };  }; |

Листинг 2.1 — Содержание файла combi.h

Код файла combi.cpp приведен в листинге 2.2.

|  |
| --- |
| #include "combi.h"  #include <algorithm>  #define NINF ((short)0x8000)  namespace combi {  subset::subset(short n) {  this->n = n;  this->sset = new short[n];  this->reset();  };  void subset::reset() {  this->sn = 0;  this->mask = 0;  };  short subset::getfirst() {  \_\_int64 buf = this->mask;  this->sn = 0;  for (short i = 0; i < n; i++) {  if (buf & 0x1)  this->sset[this->sn++] = i;  buf >>= 1;  }  return this->sn;  };  short subset::getnext() {  int rc = -1;  this->sn = 0;  if (++this->mask < this->count())  rc = getfirst();  return rc;  };  short subset::ntx(short i) {  return this->sset[i];  };  unsigned \_\_int64 subset::count() {  return (unsigned \_\_int64)(1 << this->n);  };  }; |

Листинг 2.2 — Содержание файла combi.cpp

Результат работы программы представлен на рисунке 2.1.

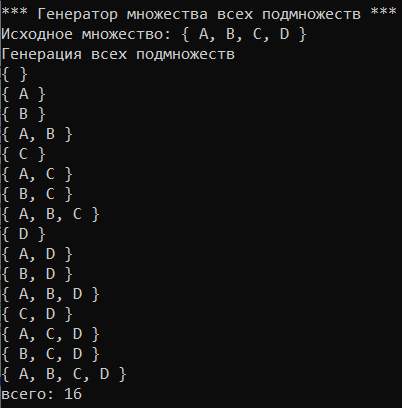


Рисунок 2.1 — Результат работы программы

## **2.2. Написание кода генератора сочетаний.**

Код файла combi.h приведен в листинге 2.3.

|  |
| --- |
| #pragma once  namespace combi {  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, // кол-во 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  };  }; |

Листинг 2.3 — Содержание файла Combi.h

Код файла combi.cpp приведен в листинге 2.4.

|  |
| --- |
| #include "combi.h"  #include <algorithm>  #define NINF ((short)0x8000)  namespace combi {  xcombination::xcombination(short n, short m) {  this->n = n;  this->m = m;  this->sset = new short[m + 2];  this->reset();  }  void xcombination::reset() { // сбросить генератор, начать сначала  this->nc = 0;  for (int i = 0; i < this->m; i++)  this->sset[i] = i;  this->sset[m] = this->n;  this->sset[m + 1] = 0;  };  short xcombination::getfirst() {  return (this->n >= this->m) ?  this->m : -1;  };  short xcombination::getnext() { // сформировать следующий массив индексов  short rc = getfirst();  if (rc > 0) {  short j;  for (j = 0; this->sset[j] + 1 == this->sset[j + 1]; ++j)  this->sset[j] = j;  if (j >= this->m)  rc = -1;  else {  this->sset[j]++;  this->nc++;  };  }  return rc;  };  short xcombination::ntx(short i) {  return this->sset[i];  };  unsigned \_\_int64 fact(unsigned \_\_int64 x) {  return(x == 0) ?  1 : (x \* fact(x - 1));  };  unsigned \_\_int64 xcombination::count() const {  return (this->n >= this->m) ?  fact(this->n) / (fact(this->n - this->m) \* fact(this->m)) : 0;  };  }; |

Листинг 2.4 — Содержание файла combi.cpp

Результат работы программы представлен на рисунке 2.2.

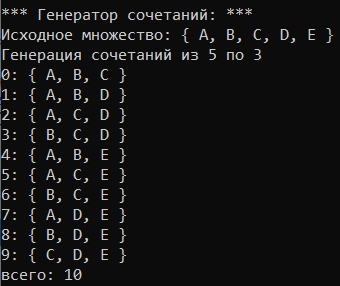


Рисунок 2.2 — Результат работы программы

## **2.3. Написание кода генератора перестановок.**

Код файла combi.h приведен в листинге 2.5.

|  |
| --- |
| #pragma once  namespace combi {  struct permutation { // генератор перестановок  const static bool L = true; // левая стрелка  const static bool R = false; // правая стрелка  short n, // количество элементов исходного множества  \* sset; // массив индексов текущей перестановки  bool\* dart; // массив стрелок (левых - L и правых - R)  permutation(short n = 1); // конструктор (количество элементов исходного множества)  void reset(); // сбросить генератор, начать сначала  \_\_int64 getfirst(); // сформировать первый массив индексов  \_\_int64 getnext(); // сформировать случайный массив индексов  short ntx(short i); // получить i-й элемент масива индексов  unsigned \_\_int64 np; // номер перествновки 0,... count() - 1  unsigned \_\_int64 count() const; // вычислить общее кол. перестановок  };  }; |

Листинг 2.5 — Содержание файла combi.h

Код файла combi.cpp приведен в листинге 2.6.

|  |
| --- |
| #include "combi.h"  #include <algorithm>  #define NINF ((short)0x8000)  namespace combi {  permutation::permutation(short n) {  this->n = n;  this->sset = new short[n];  this->dart = new bool[n];  this->reset();  };  void permutation::reset() {  this->getfirst();  };  \_\_int64 permutation::getfirst() {  this->np = 0;  for (int i = 0; i < this->n; i++) {  this->sset[i] = i;  this->dart[i] = L;  };  return (this->n > 0) ?  this->np : -1;  };  \_\_int64 permutation::getnext() { // сформировать следующий массив индексов  \_\_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); }; |

Листинг 2.6 — Содержание файла combi.cpp

Результат работы программы представлен на рисунке 2.3.

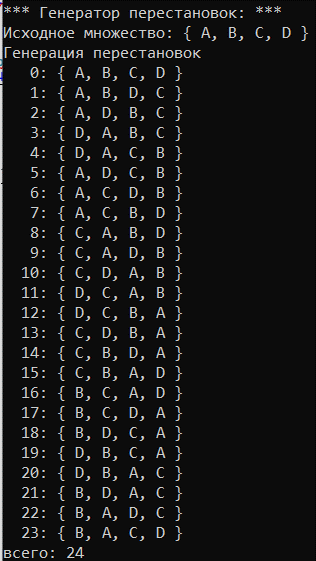


Рисунок 2.3 — Результат работы программы

## **2.4. Написание кода генератора размещений.**

Код файла combi.h приведен в листинге 2.7.

|  |
| --- |
| #pragma once  namespace combi {  struct accomodation { // генератор размещений  short n, // количество элементов исходного множества  m, // количество элементов в размещении  \* sset; // массив индексов текущего размещения  xcombination\* cgen; // указатель на генератор сочетаний  permutation\* pgen; // указатель на генератор перестановок  accomodation(short n = 1, short m = 1); // конструктор  void reset(); // сбросить генератор, начать сначала  short getfirst(); // сформировать первый массив индексов  short getnext(); // сформировать следующий массив индексов  short ntx(short i); // получить i-й элемент массива индексов  unsigned \_\_int64 na; // номер размещения 0, ..., count() - 1  unsigned \_\_int64 count() const; // общее количество размещений  };  }; |

Листинг 2.7 — Содержание файла combi.h

Код файла combi.cpp приведен в листинге 2.8.

|  |
| --- |
| #include "combi.h"  #include <algorithm>  #define NINF ((short)0x8000)  namespace combi {  accomodation::accomodation(short n, short m) {  this->n = n;  this->m = m;  this->cgen = new xcombination(n, m);  this->pgen = new permutation(m);  this->sset = new short[m];  this->reset();  }  void accomodation::reset() {  this->na = 0;  this->cgen->reset();  this->pgen->reset();  this->cgen->getfirst();  };  short accomodation::getfirst() {  short rc = (this->n >= this->m) ? this->m : -1;  if (rc > 0) {  for (int i = 0; i <= this->m; i++)  this->sset[i] = this->cgen->sset[this->pgen->ntx(i)];  };  return rc;  };  short accomodation::getnext() {  short rc;  this->na++;  if ((this->pgen->getnext()) > 0)  rc = this->getfirst();  else if ((rc = this->cgen->getnext()) > 0) {  this->pgen->reset();  rc = this->getfirst();  };  return rc;  };  short accomodation::ntx(short i) {  return this->sset[i];  };  unsigned \_\_int64 accomodation::count() const {  return (this->n >= this->m) ?  fact(this->n) / fact(this->n - this->m) : 0;  };  }; |

Листинг 2.8 — Содержание файла combi.cpp

Результат работы программы представлен на рисунке 2.4.

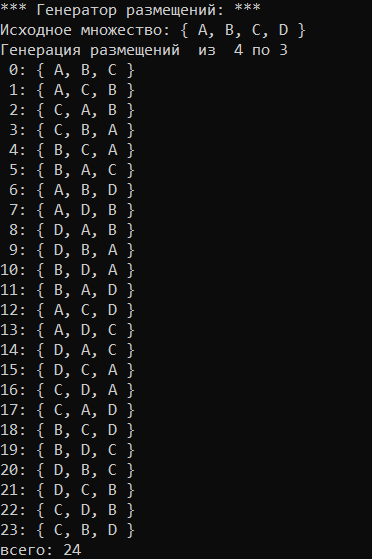


Рисунок 2.4 — Результат работы программы

## **2.5. Решение задачи об оптимальной загрузке судна.**

Схема решения задачи:



Код файла boat.h приведен в листинге 2.9.

|  |
| --- |
| int boat\_с(  short m, // [in] количество мест для контейнеров  int minv[], // [in] минимальный вес контейнера на каждом месте  int maxv[], // [in] максимальный вес контейнера на каждом месте  short n, // [in] всего контейнеров  const int v[], // [in] вес каждого контейнера  const int c[], // [in] доход от перевозки каждого контейнера  short r[] // [out] номера выбранных контейнеров  ); |

Листинг 2.9 — Содержание файла boat.h

Код файла boat.cpp приведен в листинге 2.10.

|  |
| --- |
| #include "boat.h"  #include "combi.h"  namespace boatfnc {  bool compv(combi::accomodation s, const int ming[], const int maxg[], const int v[]) {  int i = 0;  while (i < s.m && v[s.ntx(i)] <= maxg[i] && v[s.ntx(i)] >= ming[i])i++;  return (i == s.m);  };  int calcc(combi::accomodation 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\_с( // функция возвращает доход от перевозки контейнеров  short m, // [in] количество мест для контейнеров  int minv[], // [in] минимальный вес контейнера на каждом месте  int maxv[], // [in] максимальный вес контейнера каждом месте  short n, // [in] всего контейнеров  const int v[], // [in] вес каждого контейнера  const int c[], // [in] доход от перевозки каждого контейнера  short r[] // [out] номера выбранных контейнеров  ) {  combi::accomodation s(n, m);  int rc = 0, i = s.getfirst(), cc = 0;  while (i > 0) {  if (boatfnc::compv(s, minv, maxv, v))  if ((cc = boatfnc::calcc(s, c)) > rc) {  rc = cc;  boatfnc::copycomb(m, r, s.sset);  }  i = s.getnext();  };  return rc;  }; |

Листинг 2.10 — Содержание файла boat.cpp

Код файла lab2.cpp приведен в листинге 2.11.

|  |
| --- |
| #include <iostream>  #include <iomanip>  #include <tchar.h>  #include <time.h>  #include "combi.h"  #include "boat.h"  #include "auxil.h"  using namespace std;  #define N (sizeof(AA)/2)  #define M 3  #define NN 8  #define MM 5  #define SPACE(n) std::setw(n) << " "  int \_tmain(int argc, \_TCHAR\* argv[]) {  setlocale(LC\_ALL, "rus");  int v2[NN]; // вес  int c2[NN]; // доход  int minv2[NN]; // минимальный вес  int maxv2[NN]; // максимальный вес  short r2[MM];  auxil::start();  for (int i = 0; i < NN; i++) {  v2[i] = auxil::iget(100, 200);  c2[i] = auxil::iget(10, 100);  minv2[i] = auxil::iget(50, 120);  maxv2[i] = auxil::iget(150, 850);  }  int cc2 = boat\_с(  MM, // [in] количество мест для контейнеров  minv2, // [in] максимальный вес контейнера на каждом месте  maxv2, // [in] минимальный вес контейнера на каждом месте  NN, // [in] всего контейнеров  v2, // [in] вес каждого контейнера  c2, // [in] доход от перевозки каждого контейнера  r2 // [out] номера выбранных контейнеров  );  cout << endl << "\*\*\* Задача о размещении контейнеров на судне: \*\*\*";  cout << endl << "- общее количество контейнеров : " << NN;  cout << endl << "- количество мест для контейнеров : " << MM;  cout << endl << "- минимальный вес контейнера : ";  for (int i = 0; i < MM; i++)  cout << setw(3) << minv2[i] << " ";  cout << endl << "- максимальный вес контейнера : ";  for (int i = 0; i < MM; i++)  cout << setw(3) << maxv2[i] << " ";  cout << endl << "- вес контейнеров : ";  for (int i = 0; i < NN; i++)  cout << setw(3) << v2[i] << " ";  cout << endl << "- доход от перевозки :";  for (int i = 0; i < NN; i++)  cout << setw(3) << c2[i] << " ";  cout << endl << "- выбраны контейнеры (0,1,...,m-1) : ";  for (int i = 0; i < MM; i++)  cout << r2[i] << " ";  cout << endl << "- с весом : ";  for (int i = 0; i < MM; i++)  cout << v2[r2[i]] << " ";  cout << endl << "- с доходом : ";  for (int i = 0; i < MM; i++)  cout << c2[r2[i]] << " ";  cout << endl << "- доход от перевозки : " << cc2;  cout<< endl;  } |

Листинг 2.11 — Содержание файла lab2.cpp

Результат работы программы представлен на рисунке 2.5.

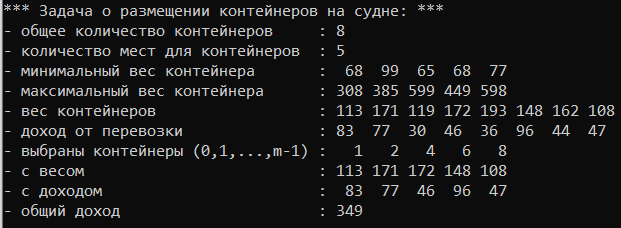


Рисунок 2.5 — Результат работы программы

## **2.6. Исследование зависимости времени вычисления необходимого для решения задачи от размерности задачи.**

Код файла lab2.cpp приведен в листинге 2.12.

|  |
| --- |
| #include <iostream>  #include <iomanip>  #include <tchar.h>  #include <time.h>  #include "combi.h"  #include "boat.h"  #include "auxil.h"  using namespace std;  #define N (sizeof(AA)/2)  #define M 3  #define NN 8  #define MM 5  #define SPACE(n) std::setw(n) << " "  int \_tmain(int argc, \_TCHAR\* argv[]) {  setlocale(LC\_ALL, "rus");  int v[NN + 1]; // вес  int c[NN + 1]; // доход  int minv[NN + 1]; // минимальный вес  int maxv[NN + 1]; // максимальный вес  short r[NN];  auxil::start();  for (int i = 0; i <= NN; i++) {  v[i] = auxil::iget(100, 200); c[i] = auxil::iget(10, 100);  minv[i] = auxil::iget(50, 120); maxv[i] = auxil::iget(150, 850);  }  cout << endl << "\*\*\* Задача о размещении контейнеров: \*\*\*";  cout << endl << "-- всего контейнеров: " << NN;  cout << endl << "-- количество ------ продолжительность -- ";  cout << endl << " мест вычисления ";  clock\_t t1, t2;  for (int i = 4; i <= NN; i++) {  t1 = clock();  boat\_с(  i, // [in] количество мест для контейнеров  minv, // [in] максимальный вес контейнера на каждом месте  maxv, // [in] максимальный вес контейнера на каждом месте  NN, // [in] всего контейнеров  v, // [in] вес каждого контейнера  c, // [in] доход от перевозки каждого контейнера  r // [out] номера выбранных контейнеров  );  t2 = clock();  cout << endl << SPACE(7) << setw(2) << i << SPACE(15) << setw(6) << (t2 - t1);  }  cout << endl;  system("pause");  return 0;  } |

Листинг 2.12 — Содержание файла lab2.cpp

Результат работы программы представлен на рисунке 2.6.

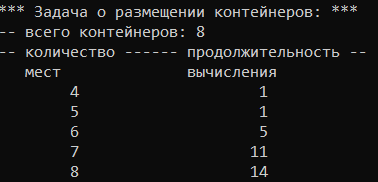


Рисунок 2.6 — Результат работы программы

Исследование зависимости представлено на рисунке 2.7.

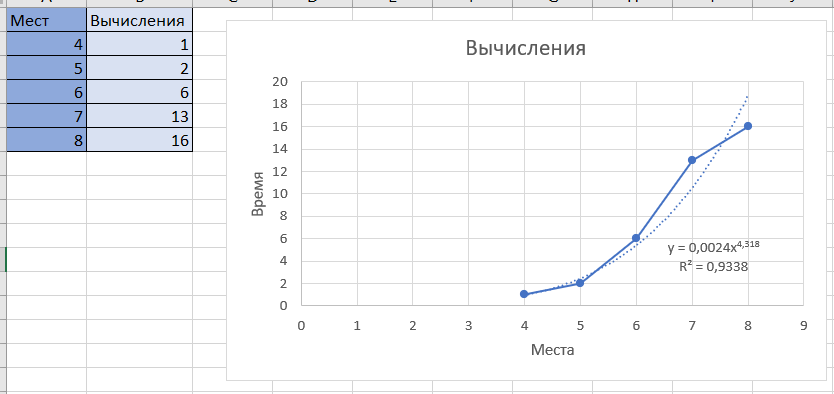


Рисунок 2.7 — Исследование зависимости

Вывод: скорость выполнения программы связана с количеством контейнеров степенной зависимостью.

**Лабораторная работа №3. Метод ветвей и границ. Задача коммивояжера и методы её решения.**

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

**Ход работы**

**Задание №1**

Условие задачи коммивояжера:

**Задание №2**

Вычитаем минимальное число из каждой строки. Приведенная по строкам матрица:

α = 8 + 8 + 10 + 24 + 21 = 71.

Вычитаем минимальное число из каждого столбца. Полностью приведенная матрица:

β = 8 + 8 = 16;

Нижняя граница длины кольцевого маршрута: φ = α + β = 71 + 16 = 87.

01,2 = 6; 01,5 = 0; 02,1 = 7; 03,1 = 6;

04,3 = 7; 04,5 = 0; 05,4 = 75.

Выбираем максимальное значение и получаем путь из города 5 в 4 и соответственно получим граф:

87

162

В случае, если мы идём по маршруту (5, 4), расстояние будет равно нижней границе кольцевого маршрута, то есть 87, а если не пойдём, то расстояние будет равно 87 + 75 = 162.

Так как меньшее расстояние 87, то мы идём из города 5 в город 4. Следовательно, для дальнейших вычислений вычёркиваем 5 строку и 4 столбец из матрицы и делаем обратный путь (4, 5) равным INF:

1 2 3 5

1

2

3

4

Полученная матрица уже является полностью приведённой, поэтому нижняя граница кольцевого маршрута останется неизменной и равной 87.

01,2 = 6, 01,5 = 47, 02,1 = 7,

03,1 = 6, 04,3 = 8.

Выбираем максимальное значение и получаем путь из города 1 в 5 и соответственно получим граф:

87

134

В случае если мы идём по маршруту (1, 5), то расстояние будет равно нижней границе кольцевого маршрута, то есть 87, а если не пойдём, то расстояние будет равно 87 + 47 = 134.

Так как меньшее расстояние 121, то мы идём из города 1 в город 5. Следовательно, для дальнейших вычислений вычёркиваем 1 строку и 5 столбец:

1 2 3

2

3

4

Данная матрица не является полностью приведённой, поэтому её надо привести по столбцам и соответственно она примет вид:

1 2 3

2

3

4

β = 6;

Следовательно, изменится нижняя граница кольцевого маршрута и соответственно: φ = 87 + 6 = 93.

02,1 = 0, 03,1 = 0, 03,2 = 0, 04,3 = 8.

Выбираем максимальное значение и получаем путь из города 4 в 3 и соответственно получим граф:

101

93

В случае если мы идём по маршруту (4, 3), то расстояние будет равно нижней границе кольцевого маршрута, то есть 93, а если не пойдём, то расстояние будет равно 93 + 8 = 101.

Так как меньшее расстояние 93, то мы идём из города 4 в город 3. Следовательно, для дальнейших вычислений вычёркиваем 4 строку и 3 столбец:

1 2

2

3

После анализа данной матрицы к нашему графу добавятся пути (4, 2) и (2, 3). Соответственно минимальное расстояние будет равно 127, и граф будет иметь следующий вид:

(5,4)(1,5)  
87+6

(5\*,4\*)  
87 + 75

R  
87

(5,4)  
87 + 0

(5,4)(1\*,5\*)  
87+47

(5,4)(1,5)(4\*,3\*)  
93+8

(5,4)(1,5)(4,3)  
93+0

Расставим переходы между городами в правильной последовательности и соответственно получим (1, 5), (5, 4), (4, 3), (3, 2), (2, 1). Итого длина маршрута будет равна 93.

Листинг кода для решения задачи на C++ представлен ниже.

|  |
| --- |
| #include "stdafx.h"  #include <iostream>  #include <iomanip>  #include "salesman.h"  #define N 5  using namespace std;  int main() {  setlocale(LC\_ALL, "rus");  int d[N][N] = { // 0 1 2 3 4  { INF, 16, 29, INF, 8 }, // 0  { 8, INF, 23, 60, 76 }, // 1  { 10, 24, INF, 86, 57 }, // 2  { 25, 50, 32, INF, 24 }, // 3  { 85, 74, 52, 21, INF } }; // 4  int r[N]; // результат  int s = salesman(  N, // [in] кол-во городов  (int\*)d, // [in] массив [n\*n] расстояний  r // [out] массив [n] маршрут 0 x x x x  );  cout << "\n-- Задача коммивояжера -- ";  cout << "\n-- количество городов: " << N;  cout << "\n-- матрица расстояний: ";  for (int i = 0; i < N; i++) {  cout << "\n";  for (int j = 0; j < N; j++)  if (d[i][j] != INF)  cout << setw(3) << d[i][j] << " ";  else  cout << setw(3) << "INF" << " ";  }  cout << "\n-- оптимальный маршрут: ";  for (int i = 0; i < N; i++)  cout << r[i] + 1 << "-->";  cout << 1;  cout << "\n-- длина маршрута : " << s << "\n";  system("pause");  return 0;  } |

Листинг 3.1 — Файл «lab3.cpp»

|  |
| --- |
| #pragma once  #include <stdio.h>  #include <tchar.h> |

Листинг 3.2 — Файл «stdafx.h»

|  |
| --- |
| #include "stdafx.h"  #include "salesman.h"  int sum(int x1, int x2) { // суммирование с учетом бесконечности  return (x1 == INF || x2 == INF) ? INF : (x1 + x2);  };  int\* firstpath(int n) { // формирование 1го маршрута 0,1,2,..., n-1, 0  int\* rc = new int[n + 1];  rc[n] = 0;  for (int i = 0; i < n; i++)  rc[i] = i;  return rc;  };  int\* source(int n) { // формирование исходного массива 1,2,..., n-1  int\* rc = new int[n - 1];  for (int i = 1; i < n; i++) rc[i - 1] = i;  return rc;  };  void copypath(int n, int\* r1, const int\* r2) { // копировать маршрут  for (int i = 0; i < n; i++)  r1[i] = r2[i];  };  int distance(int n, int\* r, const int\* d) { // длина маршрута  int rc = 0;  for (int i = 0; i < n - 1; i++)  rc = sum(rc, d[r[i] \* n + r[i + 1]]);  return sum(rc, d[r[n - 1] \* n + 0]); // последняя дуга (n-1,0)  };  void indx(int n, int\* r, const int\* s, const short\* ntx) {  for (int i = 1; i < n; i++)  r[i] = s[ntx[i - 1]];  }  int salesman(  int n, // [in] количество городов  const int\* d, // [in] массив [n\*n] рассто¤ний  int\* r // [out] массив [n] маршрут 0 x x x x  ) {  int\* s = source(n), \* b = firstpath(n), rc = INF, dist = 0; //формирует из n городов rc-служебная переменная  combi::permutation p(n - 1);  int k = p.getfirst();  while (k >= 0) { // цикл генерации перестановок  indx(n, b, s, p.sset); // новый маршрут  if ((dist = distance(n, b, d)) < rc) {  rc = dist;  copypath(n, r, b);  }  k = p.getnext();  };  return rc;  } |

Листинг 3.3 — Файл «salesman.cpp»

|  |
| --- |
| #pragma once  #define INF 0x7fffffff // бесконечность  #include "combi.h"  int salesman( // функция возвращает длину оптимального маршрута  int n, // [in] количество городов  const int\* d, // [in] массив [n\*n] расстояний  int\* r // [out] массив [n] маршрут 0 x x x x  ); |

Листинг 3.4 — Файл «salesman.h»

#include "stdafx.h"

#include "combi.h"

#include <algorithm>

#define NINF ((short)0x8000)

namespace combi {

permutation::permutation(short n) {

this->n = n;

this->sset = new short[n];

this->dart = new bool[n];

this->reset();

};

void permutation::reset() {

this->getfirst();

};

\_\_int64 permutation::getfirst() {

this->np = 0;

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

this->sset[i] = i;

this->dart[i] = L;

};

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

};

\_\_int64 permutation::getnext() {

\_\_int64 rc = -1;

short maxm = NINF, idx = -1;

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

if (i > 0 && this->dart[i] == L && this->sset[i] > this->sset[i - 1] && maxm < this->sset[i])

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

if (i < (this->n - 1) && this->dart[i] == R && this->sset[i] > this->sset[i + 1] && maxm < this->sset[i])

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

};

if (idx >= 0) {

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

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

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

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

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

if (this->sset[i] > maxm)

this->dart[i] = !this->dart[i];

rc = ++this->np;

}

return rc;

};

short permutation::ntx(short i) {

return this->sset[i];

};

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

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

}

Листинг 3.5 — Файл «combi.cpp»

#pragma once

namespace combi {

struct permutation { // генератор перестановок

const static bool L = true; // левая стрелка

const static bool R = false; // правая стрелка

short n, // кол-во элементов исх. мн-ва

\* sset; // массив индексов текущей перестановки

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

permutation(short n = 1); // конструктор (кол-во элементов 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

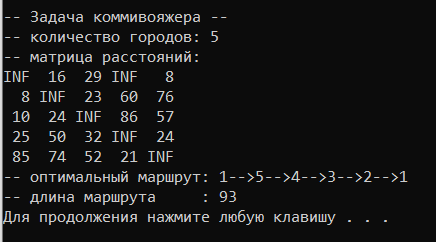
};

};

Листинг 3.6 — Файл «combi.h»

**Задание №3**

Проверка правильности решения задачи:



После проверки результата при помощи генератора перестановок из лабораторной работы 2 полученные ответы совпадали с исходным решением.

**Вывод**: мы освоили общие принципы решения задач методом ветвей и границ, решили задачу о коммивояжере данным методом, сравнили полученное решение задачи с комбинаторным методом перестановок.

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

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

**Ход работы**

**Задание №1.** На языке С++ сгенерировать случайным образом строку букв латинского алфавита *S1* длиной 300 символов и *S2* длиной 250.

Решение:

#include "stdafx.h"

#include <algorithm>

#include <iostream>

#include <ctime>

#include <iomanip>

#include "loewenstein.h"

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

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

setlocale(LC\_ALL, "rus");

srand(time(NULL));

char abc[25]; // наш алфавит

char s1[300];

char s2[250];

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

abc[n] = (char)i;

cout << "S1 = ";

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

s1[i] = abc[\_rand(0, 26)];

if (i % 50 == 0)

cout << endl;

cout << s1[i];

}

cout << endl << "S2 =";

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

s2[i] = abc[\_rand(0, 25)];

if (i % 50 == 0)

cout << endl;

cout << s2[i];

}

cout << endl;

clock\_t t1 = 0;

clock\_t t2 = 0;

clock\_t t3 = 0;

clock\_t t4 = 0;

int lx = sizeof(s1);

int ly = sizeof(s2);

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

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

cout << endl;

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

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

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

t1 = clock();

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

t2 = clock();

t3 = clock();

loewenstein(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 4.1 — Файл «main.cpp»

#pragma once

#include <SDKDDKVer.h>

#include <stdio.h>

#include <tchar.h>

using namespace std;

Листинг 4.2 — Файл «stdafx.h»

#include "stdafx.h"

#include <iomanip>

#include <algorithm>

#include "loewenstein.h"

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

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

}

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

int\*\* matr;

int w, left, top, left\_top;

matr = new int\* [lx];

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

matr[i] = new int[ly];

matr[0][0] = 0;

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

matr[i][0] = i;

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

matr[0][j] = j;

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

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

w = x[i - 1] == y[j - 1] ? 0 : 1;

top = matr[i - 1][j];

left = matr[i][j - 1];

left\_top = matr[i - 1][j - 1];

matr[i][j] = min(left\_top + w, min(top + 1, left + 1));

}

return matr[lx - 1][ly - 1];

}

int loewenstein\_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(

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

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

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

);

return rc;

};

Листинг 4.3 — Файл «loewenstein.cpp»

int loewenstein( // дистанции Левенштeйна (динамическое программирование)

int lx, // длина слова x

const char x[], // слово длиной lx

int ly, // длина слова y

const char y[] // слово y

);

int loewenstein\_r( // дистанции Левенштeйна (рекурсия)

int lx, // длина строки x

const char x[], // строка длиной lx

int ly, // длина строки y

const char y[] // строка y

);

Листинг 4.4 — Файл «loewenstein.h»

Пример генерации строк:

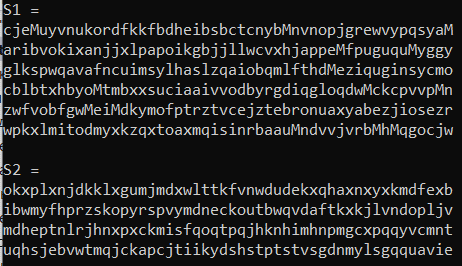


Рисунок 4.1 — Генерация строк

**Задание 2.** Вычислить двумя способами (рекурсивно и с помощью динамического программирования) дистанцию Левенштейна.

Ниже приведены варианты реализации нахождения дистанции Левенштейна при помощи динамического программирования и при помощи рекурсивного алгоритма.

Исходный код реализации через динамическое программирование:

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

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

}

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

int\*\* matr;

int w, left, top, left\_top;

matr = new int\* [lx];

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

matr[i] = new int[ly];

matr[0][0] = 0;

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

matr[i][0] = i;

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

matr[0][j] = j;

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

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

w = x[i - 1] == y[j - 1] ? 0 : 1;

top = matr[i - 1][j];

left = matr[i][j - 1];

left\_top = matr[i - 1][j - 1];

matr[i][j] = min(left\_top + w, min(top + 1, left + 1));

}

return matr[lx - 1][ly - 1];

}

Листинг 4.5 — Динамическое программирование

Пример реализации рекурсивным методом:

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

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

}

int loewenstein\_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(

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

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

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

);

return rc;

};

Листинг 4.6 — Рекурсия

На рисунке 4.2 представлены дистанции Левенштейна, вычисленные при помощи метода динамического программирования, а также рекурсивным алгоритмом.

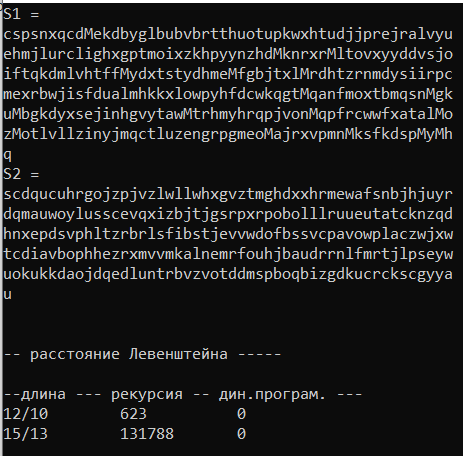


Рисунок 4.2 — Проверка работоспособности проекта

**Задание 3.** Выполнить сравнительный анализ времени, затраченного на вычисление дистанции Левенштейна для двух методов решения. Построить графики зависимости времени вычисления от *k*.

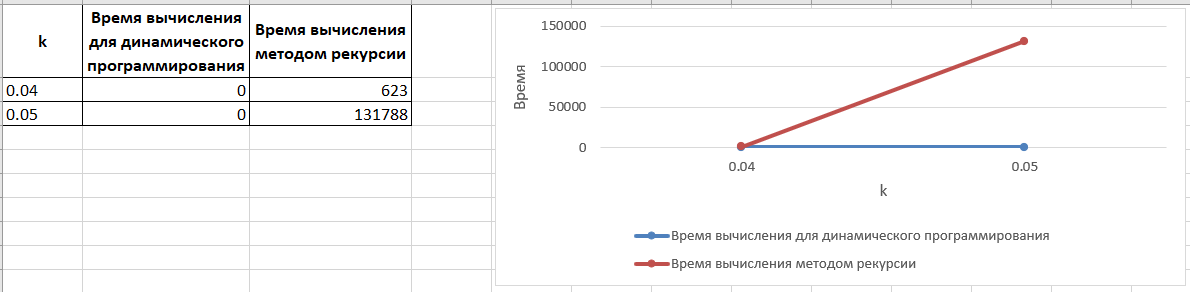
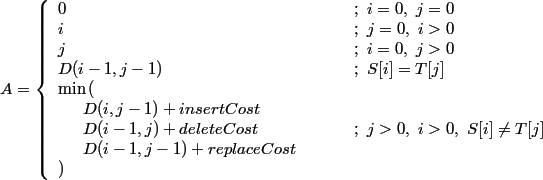


Рисунок 4.3 — Сравнительный анализ

Таким образом, выполнение с помощью динамического алгоритма вычисления производятся в разы быстрее, чем с помощью рекурсивного алгоритма.

**Задание 4.** Реализовать вручную пример вычисления дистанции Левенштейна при помощи рекурсивного алгоритма



**Решение:**

1. L(«Вар», «Баран») = min

2. L(«Ва», «Баран») = min

3. L(«Вар», «Бара») = min

4. L(«Ва», «Бара») = min

5. L(«В», «Баран») = min

L(«», «Баран») = 5,

L(«», «Бара») = 4

6. L(«В», «Бара») = min

L(«», «Бара») = 4,

L(«», «Бар») = 3

7. L(«Вар», «Бар») = min

8. L(«Ва», «Бар») = min

9. L(«Вар», «Ба») = min

10. L(«Вар», «Б») = min

L(«Вар», «») = 3,

L(«Ва», «») = 2,

11. L(«Ва», «Ба») = min

12. L(«В», «Ба») = min

L(«», «Ба») = 2,

L(«», «Б») = 1,

13. L(«Ва», «Б») = min

L(«Ва», «») = 2,

L(«В», «») = 1

14. L(«В», «Б») = min

L(«», «Б») = 1,

L(«В», «») = 1,

15. L(«В», «Б») = min (2, 2, 1) = 1

16. L(«Ва», «Б») = min (2, 3, 2) = 2

17. L(«В», «Ба») = min (3, 2, 2) = 2

18. L(«Ва», «Ба») = min (3, 3, 2) = 2

19. L(«Вар», «Б») = min (3, 4, 3) = 3

20. L(«Вар», «Ба») = min (2, 4, 3) = 2

21. L(«Ва», «Бар») = min (4, 2, 3) = 2

22. L(«Вар», «Бар») = min (3, 3, 1) = 1

23. L(«В», «Бара») = min (5, 4, 4) = 4

24. L(«В», «Баран») = min (6, 5, 5) = 5

25. L(«Ва», «Бара») = min (5, 3, 4) = 3

26. L(«Вар», «Бара») = min (4, 2, 3) = 2

27. L(«Ва», «Баран») = min (6, 4, 5) = 4

28. L(«Вар», «Баран») = min (5, 3, 3) = 3

Дистанция Левенштейна для слов «Вар» и «Баран»: 3.

**Задание 5.** Выполнить сравнительный анализ времени затраченного на решение задачи об оптимальной расстановке скобок при умножении нескольких матриц для двух методов решения (рекурсивное решение, динамическое программирование).

Дано: 10\*15, 15\*80, 80\*23, 23\*50, 50\*40, 40\*71.

Решение:

#include "stdafx.h"

#include <memory.h>

#include "multimatrix.h"

// расстановка скобок (рекурсия)

#define INFINITY 0x7fffffff

#define NINFINITY 0x80000000

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

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

int o = INFINITY;

int bo = INFINITY;

if (i < j) {

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

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

if (bo < o) {

o = bo;

OPTIMALM\_S(i, j) = k;

}

}

}

else o = 0;

return o;

#undef OPTIMALM\_S

};

// расстановка скобок (динамическое программирование)

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

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

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

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

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

OPTIMALM\_M(i, i) = 0;

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

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

j = i + l - 1;

OPTIMALM\_M(i, j) = INFINITY;

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

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

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

OPTIMALM\_M(i, j) = q;

OPTIMALM\_S(i, j) = k;

}

}

}

}

return OPTIMALM\_M(1, n);

#undef OPTIMALM\_M

#undef OPTIMALM\_S

};

Листинг 4.7 — Файл «multimatrix.cpp»

#pragma once

// расстановка скобок при умножении матриц

// функции возвращают минимальное количество операций умножения

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

int OptimalM( // рекурсия

int i, // [in] номер первой матрицы

int j, // [in] номер последней матрицы

int n, // [in] количество матриц

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

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

);

int OptimalMD( // динамическое программирование

int n, // [in] количество матриц

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

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

);

Листинг 4.7 — Файл «multimatrix.h»

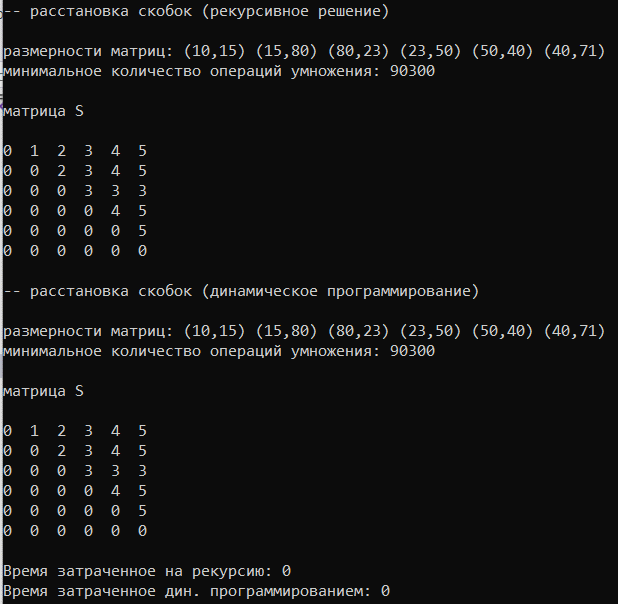


Рисунок 4.4 — Результат выполнения программы

Сравнительный анализ времени, затраченного на решение задачи с помощью рекурсивного и динамического программирования, представлен на рисунке 4.5.

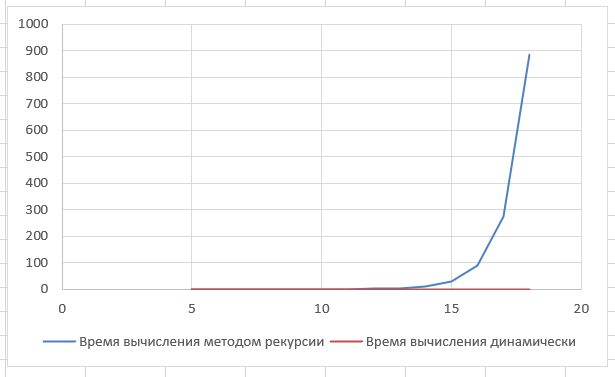


Рисунок 4.5 — Сравнительный анализ

Таким образом, при решении данной задачи более эффективным оказался метод решения с помощью динамического программирования.

**Принцип расстановки скобок по итоговой матрице:**

Скобки расставляются по принципу «сначала внешние – затем внутренние». Имеется 6 матриц, вот их размерность: А1=10\*15, А2=15\*80, А3=80\*23, А4=23\*50, А5 =50\*40, А6 =40\*71.

Матрица S:

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

Найдем элемент (1,6) в матрице S, он равен 5. Это означает, что точка разрыва между 1-ой и 6-ой матрицей находится после 5-ой матрицы. Что позволяет расставить скобки следующим образом: (A1\*A2\*A3\*A4\*A5)\*A6.

Далее берем элемент (1,5) и получаем, что он равен 4. Следовательно получаем: ((A1\*A2\*A3\*A4)\*A5)\*A6.

Далее берем элемент (1,4) и получаем, что он равен 3. Следовательно получаем: (((A1\*A2\*A3)\*A4)\*A5)\*A6.

И на последнем шаге мы возьмем элемент (1,3), который равен 2: ((((A1\*A2)\*A3)\*A4)\*A5)\*A6.

Это выражение и есть конечное.

Полученная расстановка скобок позволяет получить минимальное количество операций умножения, равное 90300.

**Лабораторная работа №5. Транспортная задача.**

**Цель работы:** приобретение навыков решения открытой транспортной задачи.

**Ход работы**

Стоимость доставки единицы груза из каждого пункта отправления в соответствующие пункты назначения задана матрицей тарифов:

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
|  | В1 | В2 | В3 | В4 | В5 | В6 | Запасы |
| А1 | 20 | 10 | 14 | 11 | 19 | 9 | 176 |
| А2 | 18 | 8 | 16 | 13 | 15 | 21 | 121 |
| А3 | 9 | 13 | 19 | 16 | 10 | 19 | 158 |
| А4 | 12 | 18 | 18 | 11 | 21 | 10 | 167 |
| А5 | 11 | 19 | 17 | 8 | 18 | 12 | 108 |
| Потребности | 151 | 115 | 139 | 201 | 103 | 171 |  |

**Решение:**

Проверим необходимое и достаточное условие разрешимости задачи.

∑a = 176 + 121 + 158 + 167 + 108 = 730

∑b = 151 + 115 + 139 + 201 + 103 + 171 = 880

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 150 (880 - 730). Тарифы перевозки единицы груза из базы ко всем потребителям полагаем равны нулю.

Занесем исходные данные в распределительную таблицу.

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
|  | В1 | В2 | В3 | В4 | В5 | В6 | Запасы |
| А1 | 20 | 10 | 14 | 11 | 19 | 9 | 176 |
| А2 | 18 | 8 | 16 | 13 | 15 | 21 | 121 |
| А3 | 9 | 13 | 19 | 16 | 10 | 19 | 158 |
| А4 | 12 | 18 | 18 | 11 | 21 | 10 | 167 |
| А5 | 11 | 19 | 17 | 8 | 18 | 12 | 108 |
| А6 | 0 | 0 | 0 | 0 | 0 | 0 | 150 |
| Потребности | 151 | 115 | 139 | 201 | 103 | 171 |  |

**Этап 1. Поиск первого опорного плана:**

1. Используя метод наименьшей стоимости, построим первый опорный план транспортной задачи. 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 c2,2 = 8. Для этого элемента запасы равны 121, потребности 115. Поскольку минимальным является 115, то вычитаем его:

x2,2 = min(121, 115) = 115.

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
|  | В1 | В2 | В3 | В4 | В5 | В6 | Запасы |
| А1 | 20 | Х | 14 | 11 | 19 | 9 | 176 |
| А2 | 18 | **8 | 115** | 16 | 13 | 15 | 21 | **6** |
| А3 | 9 | Х | 19 | 16 | 10 | 19 | 158 |
| А4 | 12 | Х | 18 | 11 | 21 | 10 | 167 |
| А5 | 11 | Х | 17 | 8 | 18 | 12 | 108 |
| А6 | 0 | Х | 0 | 0 | 0 | 0 | 150 |
| Потребности | 151 | **0** | 139 | 201 | 103 | 171 |  |

Искомый элемент равен c5,4 = 8. Для этого элемента запасы равны 108, потребности 201. Поскольку минимальным является 108, то вычитаем его:

x5,4 = min(108, 201) = 108.

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
|  | В1 | В2 | В3 | В4 | В5 | В6 | Запасы |
| А1 | 20 | Х | 14 | 11 | 19 | 9 | 176 |
| А2 | 18 | **8 | 115** | 16 | 13 | 15 | 21 | **6** |
| А3 | 9 | Х | 19 | 16 | 10 | 19 | 158 |
| А4 | 12 | Х | 18 | 11 | 21 | 10 | 167 |
| А5 | Х | Х | Х | **8 | 108** | Х | Х | **0** |
| А6 | 0 | Х | 0 | 0 | 0 | 0 | 150 |
| Потребности | 151 | **0** | 139 | **93** | 103 | 171 |  |

Искомый элемент равен c3,1 = 9. Для этого элемента запасы равны 158, потребности 151. Поскольку минимальным является 151, то вычитаем его:

x3,1 = min(158, 151) = 151.

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
|  | В1 | В2 | В3 | В4 | В5 | В6 | Запасы |
| А1 | Х | Х | 14 | 11 | 19 | 9 | 176 |
| А2 | Х | **8 | 115** | 16 | 13 | 15 | 21 | **6** |
| А3 | **9 | 151** | Х | 19 | 16 | 10 | 19 | **7** |
| А4 | Х | Х | 18 | 11 | 21 | 10 | 167 |
| А5 | Х | Х | Х | **8 | 108** | Х | Х | **0** |
| А6 | Х | Х | 0 | 0 | 0 | 0 | 150 |
| Потребности | **0** | **0** | 139 | **93** | 103 | 171 |  |

Искомый элемент равен c1,6 = 9. Для этого элемента запасы равны 176, потребности 171. Поскольку минимальным является 171, то вычитаем его:

x1,6 = min(176, 171) = 171.

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
|  | В1 | В2 | В3 | В4 | В5 | В6 | Запасы |
| А1 | Х | Х | 14 | 11 | 19 | **9 | 171** | **5** |
| А2 | Х | **8 | 115** | 16 | 13 | 15 | Х | **6** |
| А3 | **9 | 151** | Х | 19 | 16 | 10 | Х | **7** |
| А4 | Х | Х | 18 | 11 | 21 | Х | 167 |
| А5 | Х | Х | Х | **8 | 108** | Х | Х | **0** |
| А6 | Х | Х | 0 | 0 | 0 | Х | 150 |
| Потребности | **0** | **0** | 139 | **93** | 103 | **0** |  |

Искомый элемент равен c3,5 = 10. Для этого элемента запасы равны 7, потребности 103. Поскольку минимальным является 7, то вычитаем его:

x3,5 = min(7, 103) = 7.

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
|  | В1 | В2 | В3 | В4 | В5 | В6 | Запасы |
| А1 | Х | Х | 14 | 11 | 19 | **9 | 171** | **5** |
| А2 | Х | **8 | 115** | 16 | 13 | 15 | Х | **6** |
| А3 | **9 | 151** | Х | Х | Х | **10 | 7** | Х | **0** |
| А4 | Х | Х | 18 | 11 | 21 | Х | 167 |
| А5 | Х | Х | Х | **8 | 108** | Х | Х | **0** |
| А6 | Х | Х | 0 | 0 | 0 | Х | 150 |
| Потребности | **0** | **0** | 139 | **93** | **96** | **0** |  |

Искомый элемент равен c1,4 = 11. Для этого элемента запасы равны 5, потребности 5. Поскольку минимальным является 5, то вычитаем его:

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

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
|  | В1 | В2 | В3 | В4 | В5 | В6 | Запасы |
| А1 | Х | Х | X | **11 | 5** | X | **9 | 171** | **0** |
| А2 | Х | **8 | 115** | 16 | 13 | 15 | Х | **6** |
| А3 | **9 | 151** | Х | Х | Х | **10 | 7** | Х | **0** |
| А4 | Х | Х | 18 | 11 | 21 | Х | 167 |
| А5 | Х | Х | Х | **8 | 108** | Х | Х | **0** |
| А6 | Х | Х | 0 | 0 | 0 | Х | 150 |
| Потребности | **0** | **0** | 139 | **88** | **96** | **0** |  |

Искомый элемент равен c4,4 = 11. Для этого элемента запасы равны 167, потребности 88. Поскольку минимальным является 88, то вычитаем его:

X4,4 = min(167, 88) = 88.

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
|  | В1 | В2 | В3 | В4 | В5 | В6 | Запасы |
| А1 | Х | Х | X | **11 | 5** | X | **9 | 171** | **0** |
| А2 | Х | **8 | 115** | 16 | X | 15 | Х | **6** |
| А3 | **9 | 151** | Х | Х | Х | **10 | 7** | Х | **0** |
| А4 | Х | Х | 18 | **11 | 88** | 21 | Х | **79** |
| А5 | Х | Х | Х | **8 | 108** | Х | Х | **0** |
| А6 | Х | Х | 0 | X | 0 | Х | 150 |
| Потребности | **0** | **0** | 139 | **0** | **96** | **0** |  |

Искомый элемент равен c2,5 = 15. Для этого элемента запасы равны 6, потребности 103. Поскольку минимальным является 6, то вычитаем его:

X2,5 = min(6, 103) = 6.

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
|  | В1 | В2 | В3 | В4 | В5 | В6 | Запасы |
| А1 | Х | Х | X | **11 | 5** | X | **9 | 171** | **0** |
| А2 | Х | **8 | 115** | X | X | **15 | 6** | Х | **0** |
| А3 | **9 | 151** | Х | Х | Х | **10 | 7** | Х | **0** |
| А4 | Х | Х | 18 | **11 | 88** | 21 | Х | **79** |
| А5 | Х | Х | Х | **8 | 108** | Х | Х | **0** |
| А6 | Х | Х | 0 | X | 0 | Х | 150 |
| Потребности | **0** | **0** | 139 | **0** | **90** | **0** |  |

Искомый элемент равен c4,3 = 18. Для этого элемента запасы равны 79, потребности 139. Поскольку минимальным является 79, то вычитаем его:

X4,3 = min(5, 5) = 5.

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
|  | В1 | В2 | В3 | В4 | В5 | В6 | Запасы |
| А1 | Х | Х | X | **11 | 5** | X | **9 | 171** | **0** |
| А2 | Х | **8 | 115** | X | X | **15 | 6** | Х | **0** |
| А3 | **9 | 151** | Х | Х | Х | **10 | 7** | Х | **0** |
| А4 | Х | Х | **18 | 79** | **11 | 88** | X | Х | **0** |
| А5 | Х | Х | Х | **8 | 108** | Х | Х | **0** |
| А6 | Х | Х | 0 | X | 0 | Х | 150 |
| Потребности | **0** | **0** | **60** | **0** | **90** | **0** |  |

Искомый элемент равен c6,3 = 0. Для этого элемента запасы равны 150, потребности 60. Поскольку минимальным является 60, то вычитаем его:

X6,3 = min(150, 60) = 60.

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
|  | В1 | В2 | В3 | В4 | В5 | В6 | Запасы |
| А1 | Х | Х | X | **11 | 5** | X | **9 | 171** | **0** |
| А2 | Х | **8 | 115** | X | X | **15 | 6** | Х | **0** |
| А3 | **9 | 151** | Х | Х | Х | **10 | 7** | Х | **0** |
| А4 | Х | Х | **18 | 79** | **11 | 88** | X | Х | **0** |
| А5 | Х | Х | Х | **8 | 108** | Х | Х | **0** |
| А6 | Х | Х | **0 | 60** | X | 0 | Х | **90** |
| Потребности | **0** | **0** | **0** | **0** | **90** | **0** |  |

Искомый элемент равен c6,5 = 0. Для этого элемента запасы равны 90, потребности 90. Поскольку минимальным является 90, то вычитаем его:

X6,5 = min(90, 90) = 90.

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
|  | В1 | В2 | В3 | В4 | В5 | В6 | Запасы |
| А1 | Х | Х | X | **11 | 5** | X | **9 | 171** | **0** |
| А2 | Х | **8 | 115** | X | X | **15 | 6** | Х | **0** |
| А3 | **9 | 151** | Х | Х | Х | **10 | 7** | Х | **0** |
| А4 | Х | Х | **18 | 79** | **11 | 88** | X | Х | **0** |
| А5 | Х | Х | Х | **8 | 108** | Х | Х | **0** |
| А6 | Х | Х | **0 | 60** | X | **0 | 90** | Х | **0** |
| Потребности | **0** | **0** | **0** | **0** | **0** | **0** |  |

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.

2. Подсчитаем число занятых клеток таблицы, их 11, а должно быть:

m + n - 1 = 11.

Следовательно, опорный план является невырожденным.

Значение целевой функции для этого опорного плана равно:

F(x) = 11\*5 + 9\*171 + 8\*115 + 15\*6 + 9\*151 + 10\*7 + 18\*79 + 11\*88 + 8\*108 + 0\*60 + 0\*90 = 7287.

**Этап 2. Метод потенциалов:**

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

u1 + v4 = 11; 0 + v4 = 11; v4 = 11

u1 + v6 = 9; 0 + v6 = 9; v6 = 9

u4 + v4 = 11; 11 + u4 = 11; u4 = 0

u4 + v3 = 18; 0 + v3 = 18; v3 = 18

u6 + v3 = 0; 18 + u6 = 0; u6 = -18

u6 + v5 = 0; -18 + v5 = 0; v5 = 18

u2 + v5 = 15; 18 + u2 = 15; u2 = -3

u2 + v2 = 8; -3 + v2 = 8; v2 = 11

u5 + v4 = 8; 11 + u5 = 8; u5 = -3

u3 + v5 = 10; 18 + u3 = 10; u3 = -8

u3 + v1 = 9; -8 + v1 = 9; v1 = 17

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
|  | В1 | В2 | В3 | В4 | В5 | В6 | U |
| А1 | 20 | 10 | 14 | **11 | 5** | 19 | **9 | 171** | u1 = 0 |
| А2 | 18 | **8 | 115** | 16 | 13 | **15 | 6** | 21 | u2 = -3 |
| А3 | **9 | 151** | 13 | 19 | 16 | **10 | 7** | 19 | u3 = -8 |
| А4 | 12 | 18 | **18 | 79** | **11 | 88** | 21 | 10 | u4 = 0 |
| А5 | 11 | 19 | 17 | **8 | 108** | 18 | 12 | u5 = -3 |
| А6 | 0 | 0 | **0 | 60** | 0 | **0 | 90** | 0 | u6 = -18 |
| V | v1 = 17 | v2 = 11 | v3 = 18 | v4 = 11 | v5 = 18 | v6 = 9 |  |

Опорный план не является оптимальным, так как существуют оценки свободных клеток, для которых ui + vj > ci,j:

(1; 1) ∆1,1 = 0 + 17 - 20 = -3 < 0

(1; 5) ∆1,5 = 0 + 18 - 19 = -1 < 0

(2; 1) ∆2,1 = -3 + 17 - 18 = -4 < 0

(2; 3) ∆2,3 = -3 + 18 - 16 = -1 < 0

(2; 4) ∆2,4 = -3 + 11 - 13 = -5 < 0

(2; 6) ∆2,6 = -3 + 9 - 21 = -15 < 0

(3; 2) ∆3,2 = -8 + 11 - 13 = -10 < 0

(3; 3) ∆3,3 = -8 + 18 - 19 = -9 < 0

(3; 4) ∆3,4 = -8 + 11 - 16 = -13 < 0

(3; 6) ∆3,6 = -8 + 9 - 19 = -18 < 0

(4; 2) ∆4,2 = 0 + 11 - 18 = -7 < 0

(4; 5) ∆4,5 = 0 + 18 - 21 = -3 < 0

(4; 6) ∆4,6 = 0 + 9 - 10 = -1 < 0

(5; 2) ∆5,2 = -3 + 11 - 19 = -11 < 0

(5; 3) ∆5,3 = -3 + 18 - 17 = -2 < 0

(5; 5) ∆5,5 = -3 + 18 - 18 = -3 < 0

(5; 6) ∆5,6 = -3 + 9 - 12 = -6 < 0

(6; 1) ∆6,1 = -18 + 17 = -1 < 0

(6; 2) ∆6,2 = -18 + 11 = -7 < 0

(6; 4) ∆6,4 = -18 + 11 = -7 < 0

(6; 5) ∆6,5 = -18 + 9 = -9 < 0

(1; 2): ∆1,2 = 0 + 11 - 10 = 1 > 0

(1; 3): ∆1,3 = 0 + 18 - 14 = 4 > 0

(4; 1): ∆4,1 = 0 + 17 - 12 = 5 > 0

(5; 1): ∆5,1 = -3 + 17 - 11 = 3 > 0

max(1, 4, 5, 3) = 5

Выбираем максимальную оценку свободной клетки (4;1): 12. Для этого в перспективную клетку (4;1) поставим знак «+», а в остальных вершинах многоугольника чередующиеся знаки «-», «+», «-», «+», «-».

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
|  | В1 | В2 | В3 | В4 | В5 | В6 | Запасы |
| А1 | 20 | 10 | 14 | **11 | 5** | 19 | **9 | 171** | 0 |
| А2 | 18 | **8 | 115** | 16 | 13 | **15 | 6** | 21 | 0 |
| А3 | **9 | 151 [-]** | 13 | 19 | 16 | **10 | 7 [+]** | 19 | 0 |
| А4 | 12 [+] | 18 | **18 | 79 [-]** | **11 | 88** | 21 | 10 | 0 |
| А5 | 11 | 19 | 17 | **8 | 108** | 18 | 12 | 0 |
| А6 | 0 | 0 | **0 | 60 [+]** | 0 | **0 | 90 [-]** | 0 | 0 |
| Потребности | 0 | 0 | 0 | 0 | 0 | 0 |  |

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

Из грузов хi,j, стоящих в минусовых клетках, выбираем наименьшее, т.е. прибавляем 79 к объемам грузов, стоящих в плюсовых и вычитаем 79 из xi,j, стоящих в минусовых клетках. В результате получим новый опорный план.

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
|  | В1 | В2 | В3 | В4 | В5 | В6 | Запасы |
| А1 | 20 | 10 | 14 | **11 | 5** | 19 | **9 | 171** | 0 |
| А2 | 18 | **8 | 115** | 16 | 13 | **15 | 6** | 21 | 0 |
| А3 | **9 | 72** | 13 | 19 | 16 | **10 | 86** | 19 | 0 |
| А4 | 12 | 79 | 18 | **18** | **11 | 88** | 21 | 10 | 0 |
| А5 | 11 | 19 | 17 | **8 | 108** | 18 | 12 | 0 |
| А6 | 0 | 0 | **0 | 139** | 0 | **0 | 11** | 0 | 0 |
| Потребности | 0 | 0 | 0 | 0 | 0 | 0 |  |

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

u1 + v4 = 11; 0 + v4 = 11; v4 = 11

u1 + v6 = 9; 0 + v6 = 9; v6 = 9

u4 + v4 = 11; 11 + u4 = 11; u4 = 0

u5 + v4 = 8; 11 + u5 = 8; u5 = -3

u4 + v1 = 12; 0 + v1 = 12; v1 = 12

u3 + v1 = 9; 12 + u3 = 9; u3 = -3

u3 + v5 = 10; -3 + v5 = 10; v5 = 13

u2 + v5 = 15; 13 + u2 = 15; u2 = 2

u2 + v2 = 8; 2 + v2 = 8; v2 = 6

u6 + v5 = 0; 13 + u6 = 0; u6 = -13

u6 + v3 = 0; -13 + v3 = 0; v3 = 13

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
|  | В1 | В2 | В3 | В4 | В5 | В6 | U |
| А1 | 20 | 10 | 14 | **11 | 5** | 19 | **9 | 171** | u1 = 0 |
| А2 | 18 | **8 | 115** | 16 | 13 | **15 | 6** | 21 | u2 = 2 |
| А3 | **9 | 72** | 13 | 19 | 16 | **10 | 86** | 19 | u3 = -3 |
| А4 | 12 | 79 | 18 | **18** | **11 | 88** | 21 | 10 | u4 = 0 |
| А5 | 11 | 19 | 17 | **8 | 108** | 18 | 12 | u5 = -3 |
| А6 | 0 | 0 | **0 | 139** | 0 | **0 | 11** | 0 | u6 = -13 |
| V | v1 = 12 | v2 = 6 | v3 = 13 | v4 = 11 | v5 = 13 | v6 = 9 |  |

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

(1; 1) ∆1,1 = 0 + 12 - 20 = -8

(1; 2) ∆1,2 = 0 + 6 - 10 = -4

(1; 3) ∆1,3 = 0 + 13 - 14 = -1

(1; 5) ∆1,5 = 0 + 13 - 19 = -6

(2; 1) ∆2,1 = 2 + 12 - 18 = -4

(2; 3) ∆2,3 = 2 + 13 - 16 = -1

(2; 4) ∆2,4 = 2 + 11 - 13 = 0

(2; 6) ∆2,6 = 2 + 9 - 21 = -10

(3; 2) ∆3,2 = -3 + 6 - 13 = -10

(3; 3) ∆3,3 = -3 + 13 - 19 = -9

(3; 4) ∆3,4 = -3 + 11 - 16 = -8

(3; 6) ∆3,6 = -3 + 9 - 19 = -13

(4; 2) ∆4,2 = 0 + 6 - 18 = -12

(4; 3) ∆4,3 = 0 + 13 - 18 = -5

(4; 5) ∆4,5 = 0 + 13 - 21 = -8

(4; 6) ∆4,6 = 0 + 9 - 10 = -1

(5; 1) ∆5,1 = -3 + 12 - 11 = -2

(5; 2) ∆5,2 = -3 + 6 - 19 = -16

(5; 3) ∆5,3 = -3 + 13 - 17 = -7

(5; 5) ∆5,5 = -3 + 13 - 18 = -8

(5; 6) ∆5,6 = -3 + 9 - 12 = -6

(6; 1) ∆6,1 = -13 + 12 = -1

(6; 2) ∆6,2 = -13 + 6 = -7

(6; 4) ∆6,4 = -13 + 11 = -2

(6; 6) ∆6,6 = -13 + 9 = -4

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

**Анализ оптимального плана:**

Из 1-го склада необходимо доставить 5 ед. товара 4-му потребителю и 171 ед. товара 6-му потребителю.

Из 2-го склада необходимо доставить 115 ед. товара 2-му потребителю и 6 ед. товара 5-му потребителю.

Из 3-го склада необходимо доставить 72 ед. товара 1-му потребителю и 86 ед. товара 5-му потребителю.

Из 4-го склада необходимо доставить 79 ед. товара 1-му потребителю и 88 ед. товара 4-му потребителю.

Из 5-го склада необходимо доставить 108 ед. товара 4-му потребителю.

Потребность 3-го потребителя остается неудовлетворенной на 60 ед.

Потребность 5-го потребителя остается неудовлетворенной на 90 ед.

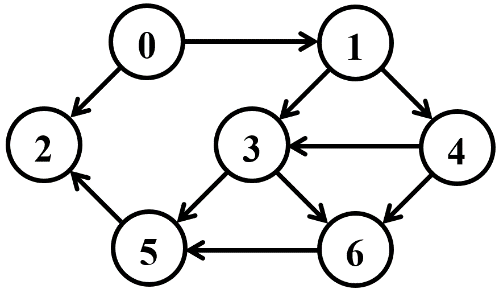
Минимальные затраты составят 6892 денежных единицы.

**Вывод:** в ходе выполнения данной лабораторной работы были приобретены навыки решения открытой транспортной задачи. Для решения задачи был применен метод наименьшей стоимости и метод потенциалов, которые позволяют определить оптимальный план перевозок при минимальных затратах. Были проведены расчеты с использованием данных методов, что позволило получить оптимальный план перевозок.

**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:

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

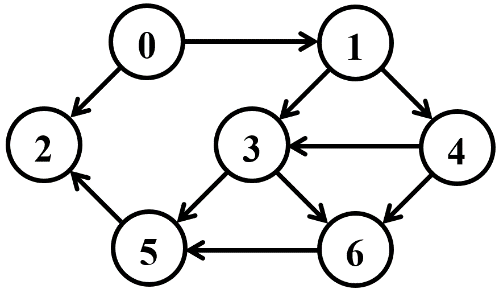
Список смежных вершин:

0 → { 1, 2 } 1 → { 3, 4 } 2 → { - }

3 → { 5, 6 } 4 → { 3 } 5 → { 2 }

6 → { 5 }

Матрица инцидентности:



9

8

7

6

4

5

3

2

1

0

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

**Задание 2.** Осуществить алгоритмы поиска в ширину и глубину, а также алгоритма топологической сортировки аналогично примерам, рассмотренным на лекциях.

**Обход в ширину:**

Текущее состояние алгоритма хранится в следующих структурах памяти:

Q: очередь вершин,

C: массив окраски вершин,

D: массив расстояний,

P: массив предшествующих вершин.

*Шаг 1:*

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

*Шаг 2:*

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

*Шаг 3:*

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

*Шаг 4:*

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

*Шаг 5:*

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

*Шаг 6:*

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

*Шаг 7:*

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

*Шаг 8:*

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

*В результате получили такое BFS-дерево:*

2

1

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

5

4

3

Обход в ширину: 0, 1, 2, 3, 4, 5, 6

7

6

**Обход в глубину:**

Текущее состояние алгоритма хранится в следующих структурах памяти:

C – массив окраски вершин,

D – время окраски вершин в серый цвет,

P – массив предшествующих вершин,

F – время окраски в чёрный цвет,

t – номер шага алгоритма.

*Шаг 1:*

Стек: 0

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

*Шаг 2:*

Стек: 0, 1

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

*Шаг 3:*

Стек: 0, 1, 3

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

*Шаг 4:*

Стек: 0, 1, 3, 5

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

*Шаг 5:*

Стек: 0, 1, 3, 5, 2

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| t | 5 |  |  |  |  |  |  |
| C | G | G | G | G | W | G | W |
| D | 1 | 2 | 5 | 3 | I | 4 | I |
| P | N | 0 | 5 | 1 | N | 3 | N |
| F | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

*Шаг 6:*

Стек: 0, 1, 3, 5

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| t | 6 |  |  |  |  |  |  |
| C | G | G | B | G | W | G | W |
| D | 1 | 2 | 5 | 3 | I | 4 | I |
| P | N | 0 | 5 | 1 | N | 3 | N |
| F | 0 | 0 | 6 | 0 | 0 | 0 | 0 |

*Шаг 7:*

Стек: 0, 1, 3

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| t | 7 |  |  |  |  |  |  |
| C | G | G | B | G | W | B | W |
| D | 1 | 2 | 5 | 3 | I | 4 | I |
| P | N | 0 | 5 | 1 | N | 3 | N |
| F | 0 | 0 | 6 | 0 | 0 | 7 | 0 |

*Шаг 8:*

Стек: 0, 1, 3, 6

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| t | 8 |  |  |  |  |  |  |
| C | G | G | B | G | W | B | G |
| D | 1 | 2 | 5 | 3 | I | 4 | 8 |
| P | N | 0 | 5 | 1 | N | 3 | 3 |
| F | 0 | 0 | 6 | 0 | 0 | 7 | 0 |

*Шаг 9:*

Стек: 0, 1, 3

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| t | 9 |  |  |  |  |  |  |
| C | G | G | B | G | W | B | B |
| D | 1 | 2 | 5 | 3 | I | 4 | 8 |
| P | N | 0 | 5 | 1 | N | 3 | 3 |
| F | 0 | 0 | 6 | 0 | 0 | 7 | 9 |

*Шаг 10:*

Стек: 0, 1

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| t | 10 |  |  |  |  |  |  |
| C | G | G | B | B | W | B | B |
| D | 1 | 2 | 5 | 3 | I | 4 | 8 |
| P | N | 0 | 5 | 1 | N | 3 | 3 |
| F | 0 | 0 | 6 | 10 | 0 | 7 | 9 |

*Шаг 11:*

Стек: 0, 1, 4

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| t | 11 |  |  |  |  |  |  |
| C | G | G | B | B | G | B | B |
| D | 1 | 2 | 5 | 3 | 11 | 4 | 8 |
| P | N | 0 | 5 | 1 | 1 | 3 | 3 |
| F | 0 | 0 | 6 | 10 | 0 | 7 | 9 |

*Шаг 12:*

Стек: 0, 1

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| t | 12 |  |  |  |  |  |  |
| C | G | G | B | B | B | B | B |
| D | 1 | 2 | 5 | 3 | 11 | 4 | 8 |
| P | N | 0 | 5 | 1 | 1 | 3 | 3 |
| F | 0 | 0 | 6 | 10 | 12 | 7 | 9 |

*Шаг 13:*

Стек: 0

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| t | 13 |  |  |  |  |  |  |
| C | G | B | B | B | B | B | B |
| D | 1 | 2 | 5 | 3 | 11 | 4 | 8 |
| P | N | 0 | 5 | 1 | 1 | 3 | 3 |
| F | 0 | 13 | 6 | 10 | 12 | 7 | 9 |

*Шаг 14:*

Стек:

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| t | 14 |  |  |  |  |  |  |
| C | B | B | B | B | B | B | B |
| D | 1 | 2 | 5 | 3 | 11 | 4 | 8 |
| P | N | 0 | 5 | 1 | 1 | 3 | 3 |
| F | 14 | 13 | 6 | 10 | 12 | 7 | 9 |

*В результате получили такое DFS-дерево:*

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| P | N | 0 | 5 | 1 | 1 | 3 | 3 |

Обход в глубину: 0, 1, 3, 5, 2, 4, 6

**Топологическая сортировка:**

*Шаг 1: Шаг 2:*

2/

1/

1/

*Шаг 3: Шаг 4:*

2/

2/

1/

1/

3/

3/

4/

*Шаг 5: Шаг 6:*

2/

2/

1/

1/

5/

5/6

3/

3/

4/

4/

*Шаг 7: Шаг 8:*

2/

2/

1/

1/

5/6

5/6

3/

3/

8/

4/7

4/7

*Шаг 9: Шаг 10:*

2/

2/

1/

1/

5/6

3/10

5/6

3/

8/9

8/9

4/7

4/7

*Шаг 11: Шаг 12:*

2/

2/

1/

1/

5/6

11/12

11/

3/10

3/10

5/6

8/9

8/9

4/7

4/7

*Шаг 13: Шаг 14:*

2/13

1/14

2/13

1/

5/6

11/12

11/12

3/10

3/10

5/6

8/9

8/9

4/7

4/7

Формируем очередь по мере окрашивания вершин (2-5-6-3-4-1-0):

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

**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.

#include <iostream>

#include "Graph.h"

#include "BFS.h"

#include "DFS.h"

int main() {

int m[7][7] = {

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

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

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

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

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

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

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

};

setlocale(LC\_ALL, "rus");

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

std::cout << std::endl;

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

for (int i = 0; i < g1.n\_vertex; i++) {

std::cout << std::endl;

for (int j = 0; j < g1.n\_vertex; j++)

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

};

std::cout << std::endl;

graph::AList g2(g1);

std::cout << std::endl;

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

for (int i = 0; i < g1.n\_vertex; i++) {

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

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

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

}

std::cout << std::endl;

graph::AMatrix g3(g1);

std::cout << std::endl;

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

for (int i = 0; i < g3.n\_vertex; i++) {

std::cout << std::endl;

for (int j = 0; j < g3.n\_vertex; j++)

std::cout << g3.get(i, j) << " ";

};

std::cout << std::endl;

graph::AList g4(7, (int\*)m);

std::cout << std::endl;

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

for (int i = 0; i < g4.n\_vertex; i++) {

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

for (int j = 0; j < g4.size(i); j++)

std::cout << g4.get(i, j) << " ";

}

std::cout << std::endl;

BFS b1(g2, 0);

std::cout << std::endl;

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

int k1;

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

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

std::cout << std::endl;

DFS b2(g2);

std::cout << std::endl;

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

for (int i = 0; i < g2.n\_vertex; i++)

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

std::cout << std::endl;

int m3[7][7] = {

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

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

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

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

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

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

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

};

graph::AList g5(7, (int\*)m3);

DFS b3(g5);

std::cout << std::endl;

std::cout << std::endl << "-- поиск в глубину (ориентированный граф) " << std::endl;

for (int i = 0; i < g5.n\_vertex; i++)

std::cout << b3.get(i) << " ";

std::cout << std::endl;

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

for (std::vector <int>::iterator i(b3.topological\_sort.begin());

i != b3.topological\_sort.end(); ++i)

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

std::cout << std::endl;

system("pause");

return 0;

}

Листинг 6.1 — Файл «lab6.cpp»

#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 6.2 — Файл «BFS.cpp»

#pragma once

#include "Graph.h"

#include <queue>

struct BFS { // поиск в ширину (связный граф)

const static int INF = 0x7fffffff;

const static int NIL = -1;

enum Color { WHITE, GRAY, BLACK };

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

Color\* c; // цвет вершины

int\* d; // расстояние до вершины

int\* p; // предшествующая вершина

std::queue<int> q; // очередь

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

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

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

int get(); // получить следующую вершину

};

Листинг 6.3 — Файл «BFS.h»

#include "Graph.h"

namespace graph {

AMatrix::AMatrix(int n) {

this->n\_vertex = n;

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

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

mr[i] = 0;

};

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

this->n\_vertex = n;

this->mr = mr;

};

AMatrix::AMatrix(const AMatrix& am) {

this->n\_vertex = am.n\_vertex;

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

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

for (int j = 0; j < this->n\_vertex; j++)

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

};

AMatrix::AMatrix(const AList& al) {

this->n\_vertex = al.n\_vertex;

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

for (int k = 0; k < this->n\_vertex \* this->n\_vertex; k++)mr[k] = 0;

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

for (int j = 0; j < al.size(i); j++) this->set(i, al.get(i, j), 1);

};

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

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

};

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

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

};

void AList::create(int n) { this->mr = new std::list<int>[this->n\_vertex = n]; };

AList::AList(int n) { create(n); }

AList::AList(const AMatrix& am) {

this->create(am.n\_vertex);

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

for (int j = 0; j < this->n\_vertex; j++)

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

this->add(i, j);

};

AList::AList(const AList& al) {

this->create(al.n\_vertex);

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

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

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

};

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

this->create(n);

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

for (int j = 0; j < this->n\_vertex; j++)

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

this->add(i, j);

};

void AList::add(int i, int j) { this->mr[i].push\_back(j); };

int AList::size(int i) const { return (int)this->mr[i].size(); };

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

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

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

rc++;

return (int)\*rc;

};

};

Листинг 6.4 — Файл «Graph.cpp»

#pragma once

#include <list>

#include "Graph.h"

namespace graph {

struct AList;

struct AMatrix { // матрица смежности

int n\_vertex; // количество вершин

int\* mr; // матрица

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

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

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

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

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

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

};

struct AList { // списки смежности

int n\_vertex; // количество вершин

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

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

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

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

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

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

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

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

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

};

};

Листинг 6.5 — Файл «Graph.h»

Результат выполнения программы представлен на рисунке 6.1:

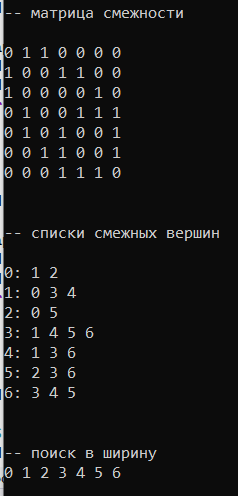


Рисунок 6.1 — Результат

**Задание 4-5.** Разработать функцию DFS обхода вершин графа, используя метод поиска глубину. Доработайте функцию DFS, для выполнения топологической сортировки графа. Продемонстрировать работу функций.

#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++) { // iая статистика

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 6.6 — Файл «DFS.cpp»

#pragma once

#include "Graph.h"

#include <vector>

struct DFS { // depth-first search поиск в глубину

const static int NIL = -1;

enum Color { WHITE, GRAY, BLACK };

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

Color\* c; // цвет вершины

int\* d; // время обнаружения

int\* f; // время завершения обработки

int\* p; // предшествующая вершина

int t; // текущее время

DFS(const graph::AList& al);

DFS(const graph::AMatrix& am);

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

void visit(int v);

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

int get(int i); // получить вершину

};

Листинг 6.7 — Файл «DFS.h»

Результат выполнения программы представлен на рисунке 6.2:

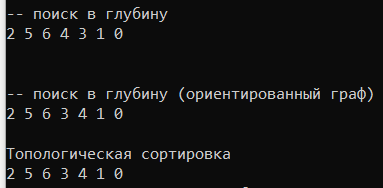


Рисунок 6.2 — Результат

**Задание 6.** По графу, соответствующему варианту составить минимальное остовное дерево по алгоритму Прима.

W(e0,1) = 8; W(e0,2) = 1;

W(e1,3) = 11; W(e1,4) = 5;

W(e5,2) = 10; W(e4,3) = 4;

W(e4,6) = 10; W(e6,5) = 6;

W(e3,5) = 6; W(e3,6) = 9;

8

5

11

1

4

9

6

6

10

10

*Шаг 1:*

Предположим, что в множество U пока только входит вершина 0, нужно найти ребро с минимальной стоимостью, которое соединяет вершину 0 с вершиной, которая в множество U не входит.

U = { 0 }, T = { }

Вершина 0 соединена с вершинами 1 и 2. Веса их ребер равно 8 и 1 соответственно. Из этих ребер минимальной стоимостью обладает ребро 0-2 (с весом 1). Следовательно, вершину 2 включаем в множество U, а ребро 0-2 — в множество T.

1

*Шаг 2:*

U = { 0, 2 }, T = { 0-2 }

Вершины 0 и 2 соединены с вершинами 1 и 5. Веса их ребер равно 8 и 10 соответственно. Из этих ребер минимальной стоимостью обладает ребро 0-1 (с весом 8). Следовательно, вершину 1 включаем в множество U, а ребро 0-1 — в множество T.

8

1

*Шаг 3:*

U = { 0, 2, 1 }, T = { 0-2, 0-1 }

Вершины 0, 2 и 1 соединены с вершинами 5, 3, 4. Веса их ребер равно 10, 11 и 5 соответственно. Из этих ребер минимальной стоимостью обладает ребро 1-4 (с весом 5). Следовательно, вершину 4 включаем в множество U, а ребро 1-4 — в множество T.

8

5

1

*Шаг 4:*

U = { 0, 2, 1, 4 }, T = { 0-2, 0-1, 1-4 }

Вершины 0, 2, 1 и 4 соединены с вершинами 5, 3, 6. Веса их ребер равно 10, 11 (1-3), 4 (4-3) и 10 соответственно. Из этих ребер минимальной стоимостью обладает ребро 4-3 (с весом 4). Следовательно, вершину 3 включаем в множество U, а ребро 4-3 — в множество T.

8

5

1

4

*Шаг 5:*

U = { 0, 2, 1, 4, 3 }, T = { 0-2, 0-1, 1-4, 4-3 }

Вершины 0, 2, 1, 4 и 3 соединены с вершинами 5 и 6. Веса их ребер равно 10 (2-5), 6 (3-5), 9 (3-6) и 10 (4-6) соответственно. Из этих ребер минимальной стоимостью обладает ребро 3-5 (с весом 6). Следовательно, вершину 5 включаем в множество U, а ребро 3-5 — в множество T.

8

5

1

4

6

*Шаг 6:*

U = { 0, 2, 1, 4, 3, 5 }, T = { 0-2, 0-1, 1-4, 4-3, 3-5 }

Вершины 0, 2, 1, 4, 3 и 5 соединены с вершиной 6. Веса их ребер — 10 (4-6) и 6 (5-6) соответственно. Из этих ребер минимальной стоимостью обладает ребро 5-6 (с весом 6). Следовательно, вершину 6 включаем в множество U, а ребро 5-6 — в множество T.

8

5

1

4

6

6

*Вес минимального остовного дерева: 30.*

**Задание 7.** По графу, соответствующему варианту составить минимальное остовное дерево по алгоритму Крускала.

8

5

11

1

4

9

6

6

10

10

*Шаг 1:*

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.

1

*Шаг 2:*

1

4

*Шаг 3:*

5

1

4

*Шаг 4:*

5

1

4

6

*Шаг 5:*

5

1

4

6

6

*Шаг 6:*

8

5

1

4

6

6

*Вес минимального остовного дерева: 30.*

**Лабораторная работа №7. Сетевые модели.**

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

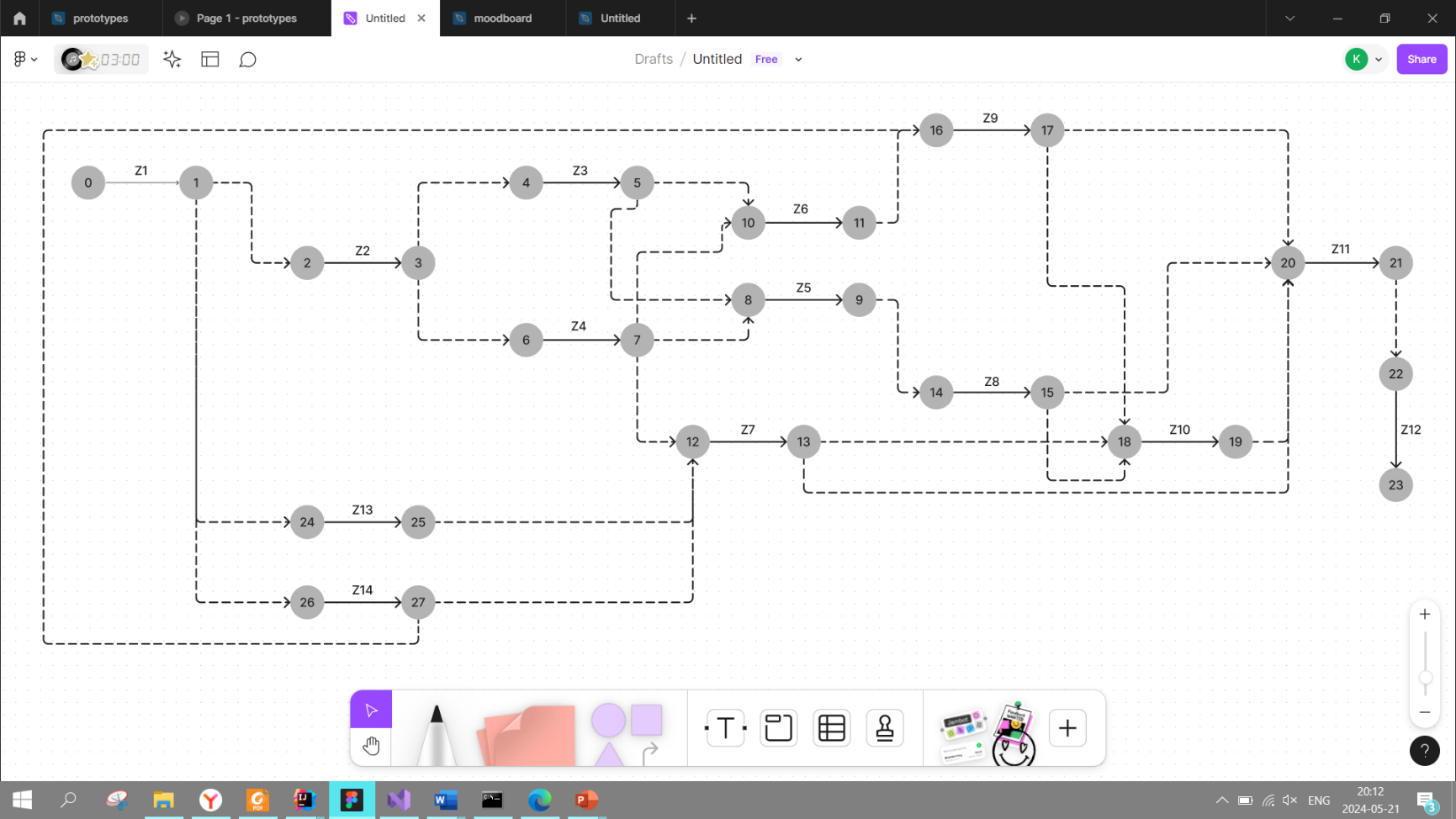
**Задание 1-2.** Подумайте и выделите в проекте, согласно вашему варианту не менее 4 этапов работ. Также разбейте полученные этапы на задачи, их количество в совокупности по этапам должно быть не менее 12.

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

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **Начальное событие** | **Код операции** | **Наименование операции** | **Предшествующие операции** | **t** | **Конечное событие** |
| I. Исследование и планирование | | | | | |
| 0 | **Z1** | Исследование рынка |  | 2 | 1 |
| 2 | **Z2** | Создание концепции игры | **Z1** | 3 | 3 |
| 4 | **Z3** | Планирование проекта и распределение ресурсов | **Z2** | 3 | 5 |
| II. Проектирование | | | | | |
| 6 | **Z4** | Проектирование интерфейса и UX | **Z2** | 4 | 7 |
| 8 | **Z5** | Проектирование игровых механик | **Z3, Z4** | 4 | 9 |
| 10 | **Z6** | Проектирование графики и звуков | **Z3, Z4** | 4 | 11 |
| III. Разработка и кодирование | | | | | |
| 12 | **Z7** | Разработка игровых механик | **Z4, Z14, Z13** | 6 | 13 |
| 14 | **Z8** | Разработка интерфейса | **Z5** | 5 | 15 |
| 16 | **Z9** | Создание графики и звуков | **Z6, Z14** | 6 | 17 |
| IV. Тестирование и отладка | | | | | |
| 18 | **Z10** | Тестирование игры | **Z7, Z8, Z9** | 5 | 19 |
| 20 | **Z11** | Отладка и устранение ошибок | **Z7, Z8, Z9, Z10** | 3 | 21 |
| 22 | **Z12** | Подготовка к выпуску и оптимизация | **Z11** | 2 | 23 |
| V. Дополнительные работы | | | | | |
| 24 | **Z13** | Установка СУБД | **Z1** | 1 | 25 |
| 26 | **Z14** | Установка инструментария | **Z1** | 1 | 27 |

**Задание 3.** Согласно составленному перечню задач и распределённому времени составьте сетевой график вашего проекта. Помните о правилах составления графика и вводите фиктивные операции и операции ожидания если это необходимо.

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.



|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **Код операции** | **Наименование операции** | **Предшествующие операции** | **t** | **Ранние сроки: начало** | **Ранние сроки: конец** |
| I. Исследование и планирование | | | | | |
| **Z1** | Исследование рынка |  | 2 | 0 | 2 |
| **Z2** | Создание концепции игры | **Z1** | 3 | 2 | 5 |
| **Z3** | Планирование проекта и распределение ресурсов | **Z2** | 3 | 5 | 8 |
| II. Проектирование | | | | | |
| **Z4** | Проектирование интерфейса и UX | **Z2** | 4 | 5 | 9 |
| **Z5** | Проектирование игровых механик | **Z3, Z4** | 4 | 9 | 13 |
| **Z6** | Проектирование графики и звуков | **Z3, Z4** | 4 | 9 | 13 |
| III. Разработка и кодирование | | | | | |
| **Z7** | Разработка игровых механик | **Z4, Z14, Z13** | 6 | 9 | 15 |
| **Z8** | Разработка интерфейса | **Z5** | 5 | 9 | 14 |
| **Z9** | Создание графики и звуков | **Z6, Z14** | 6 | 9 | 15 |
| IV. Тестирование и отладка | | | | | |
| **Z10** | Тестирование игры | **Z7, Z8, Z9** | 5 | 15 | 20 |
| **Z11** | Отладка и устранение ошибок | **Z7, Z8, Z9, Z10** | 3 | 20 | 23 |
| **Z12** | Подготовка к выпуску и оптимизация | **Z11** | 2 | 23 | 25 |
| V. Дополнительные работы | | | | | |
| **Z13** | Установка СУБД | **Z1** | 1 | 2 | 3 |
| **Z14** | Установка инструментария | **Z1** | 1 | 2 | 3 |

Критический путь в данной сети задач будет следующим:

Z1 → Z2 → Z4 → Z7 → Z10 → Z11 → Z12

Сложив все эти значения, получим общее время выполнения всех задач по критическому пути: 2 + 3 + 4 + 6 + 5 + 3 + 2 = 25 дней. Это минимальное время, необходимое для завершения всего проекта, при условии, что все задачи по критическому пути выполняются последовательно и без задержек.

**Задание 4.** Предложите варианты оптимизации вашего проекта с привлечением денежных средств или человеческого ресурса.

1. Найм дополнительных специалистов для ускорения создания игровых ресурсов (3D-моделей, анимации, звуковых эффектов и музыки) и сокращения времени на этапе разработки игрового движка. Это позволит распределить рабочую нагрузку и сократить время выполнения некоторых задач.
2. Использование готовых решений и инструментов, таких как Unity или Unreal Engine, для уменьшения времени разработки игрового движка и программирования геймплея и интерфейса. Это позволит сэкономить время на разработке собственного движка и сосредоточиться на создании уникального контента и геймплея.
3. Проведение параллельного тестирования и отладки игры на разных этапах разработки, чтобы выявлять и устранять проблемы на ранних стадиях и сократить время, затрачиваемое на финальное тестирование и отладку.
4. Улучшение коммуникации и координации между командами разработчиков, что позволит эффективнее организовывать работу, предотвращать дублирование усилий и сократить время выполнения задач.
5. Организация обучения и тренингов для сотрудников, чтобы повысить их профессиональные навыки и ускорить выполнение задач.
6. Применение гибких методологий разработки, таких как Agile или Scrum, для более быстрой адаптации к изменениям в проекте и улучшения совместной работы команды.

Каждый из этих подходов требует дополнительных вложений — времени, денег или и того, и другого. Однако они могут помочь сократить общее время выполнения проекта и улучшить качество конечного продукта.

**Вывод**: в процессе выполнения лабораторной работы мы изучили основные принципы и методы сетевого планирования, в том числе метод критического пути и метод оценки и обзора программ. Мы также создали сетевую диаграмму проекта, определили критический путь и его стоимость. Применение этих методов позволяет эффективно управлять проектами, планировать и контролировать время и ресурсы, необходимые для выполнения задач, а также способно повысить эффективность управления проектами.

**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 x + 4y ≥ 1.

Чтобы построить график прямой x + 4y = 1 нам нужно найти ее точки пересечения с осями X и Y.

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

x + 0 = 1

x = 1.

Следовательно, (1; 0).

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

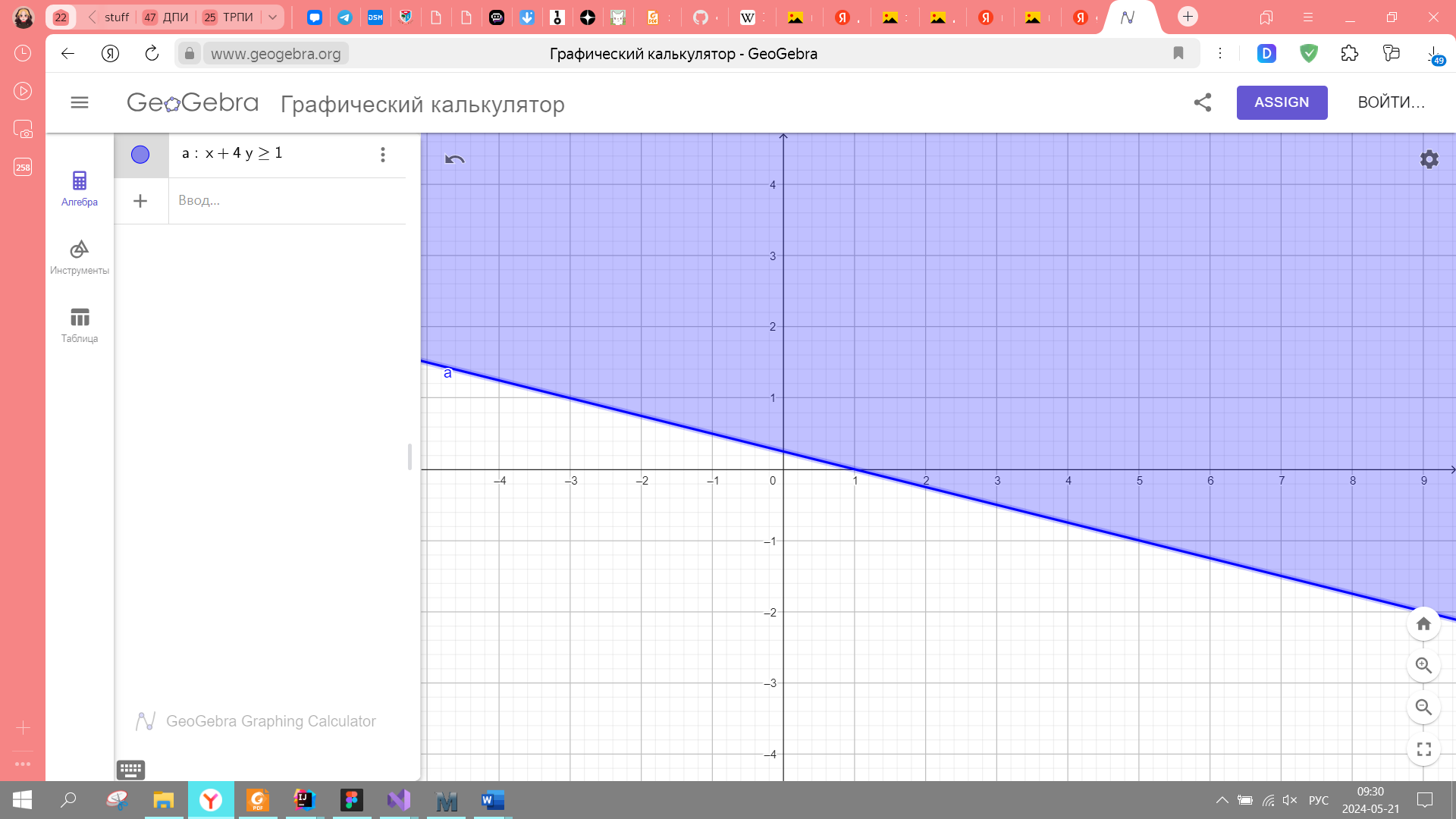
0 + 4y = 1

y = 1 / 4

y = 0,25

Следовательно (0; 0,25)

Прямая x + 4y ≥ 1 будет проходить через точки (1; 0), (0; 0,25).



Строим x + 2y ≥ 4

Чтобы построить график прямой x + 2y = 4 нам нужно найти ее точки пересечения с осями X и Y.

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

x + 0 = 4

x = 4

Следовательно (4; 0)

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

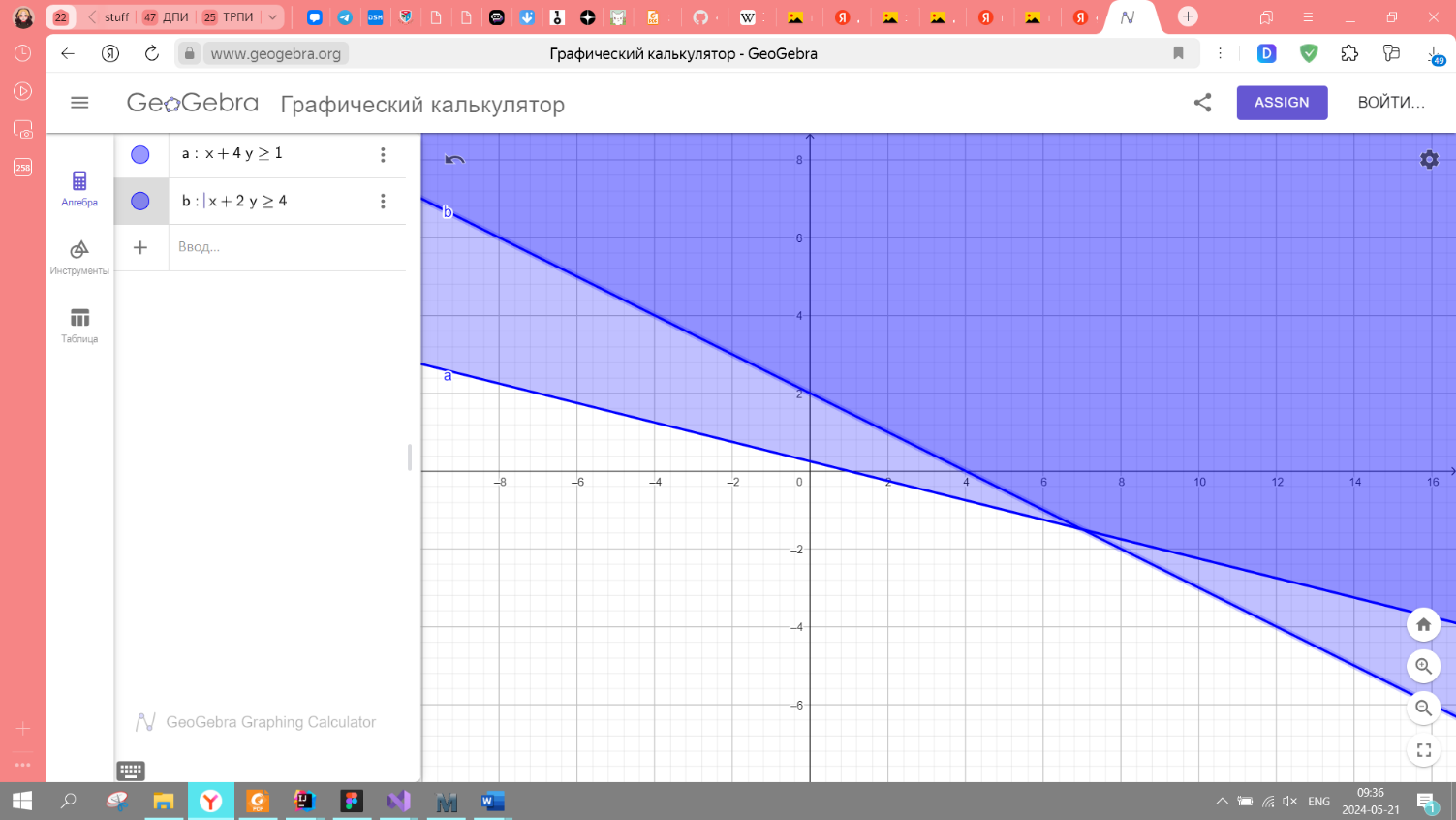
0 + 2y = 4

y = 4/2

y = 2

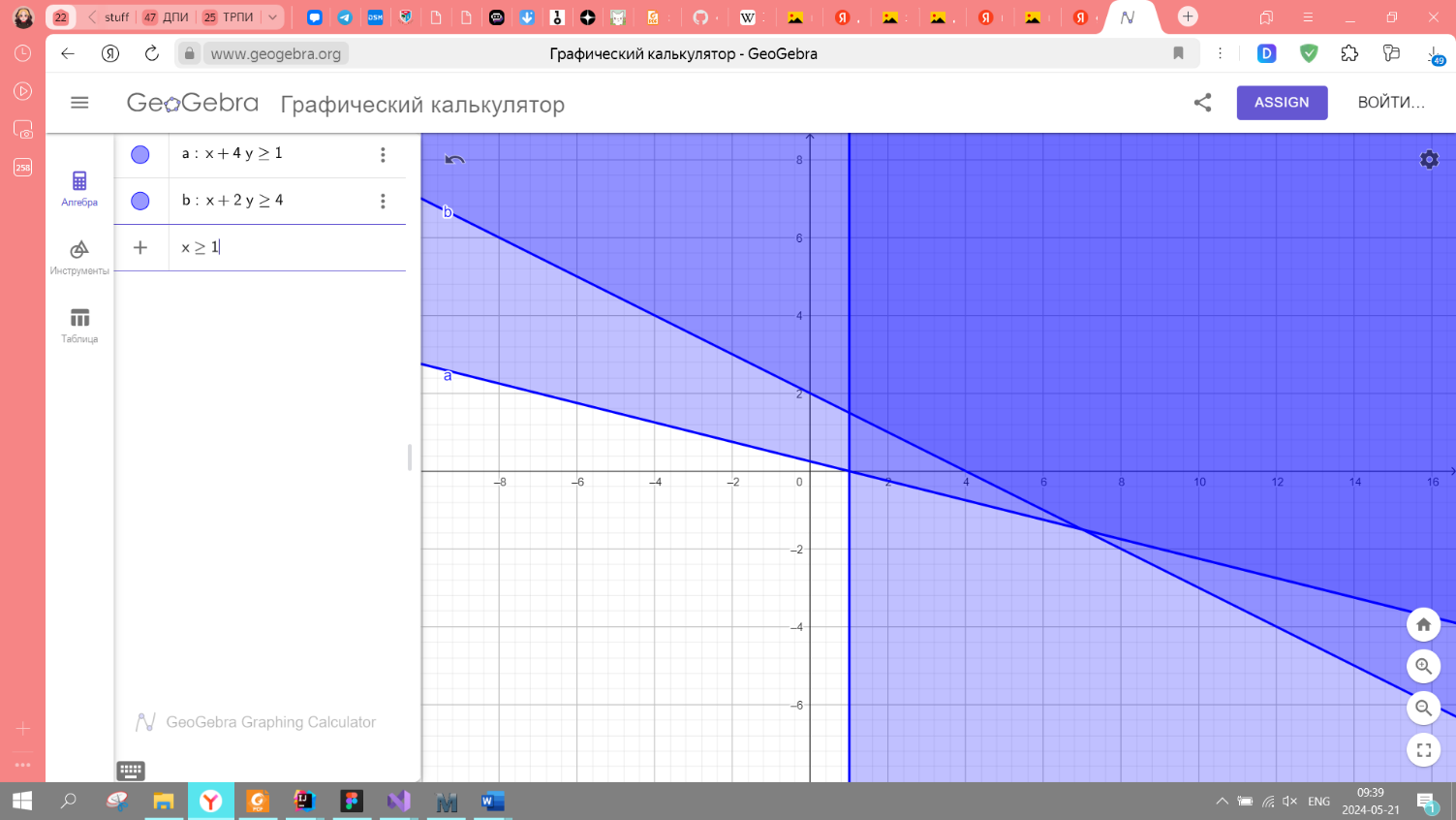
Следовательно (0; 2)

Прямая x + 2y = 4 будет проходить через точки (4; 0), (0; 2).



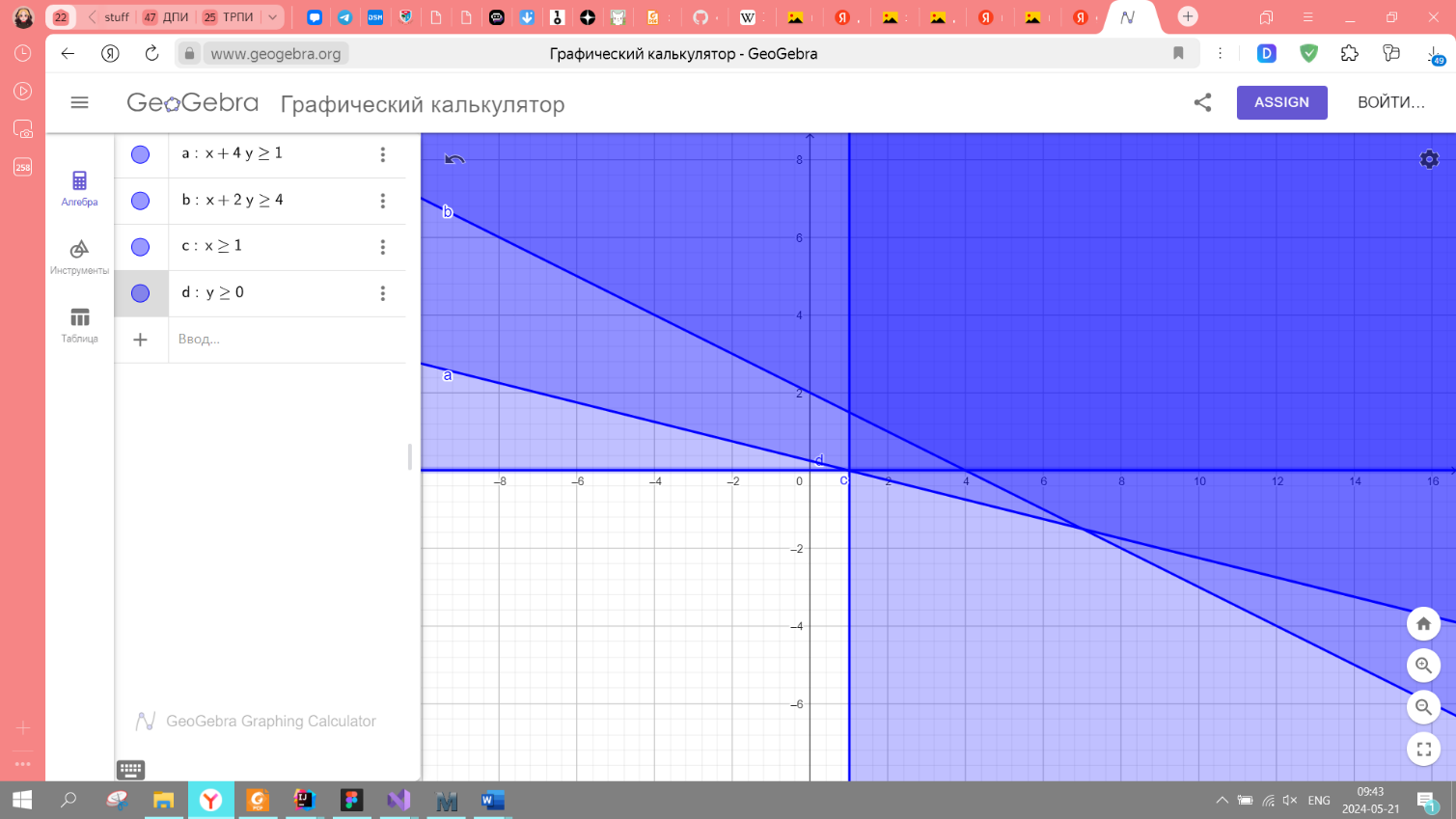
Строим x ≥ 1.

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

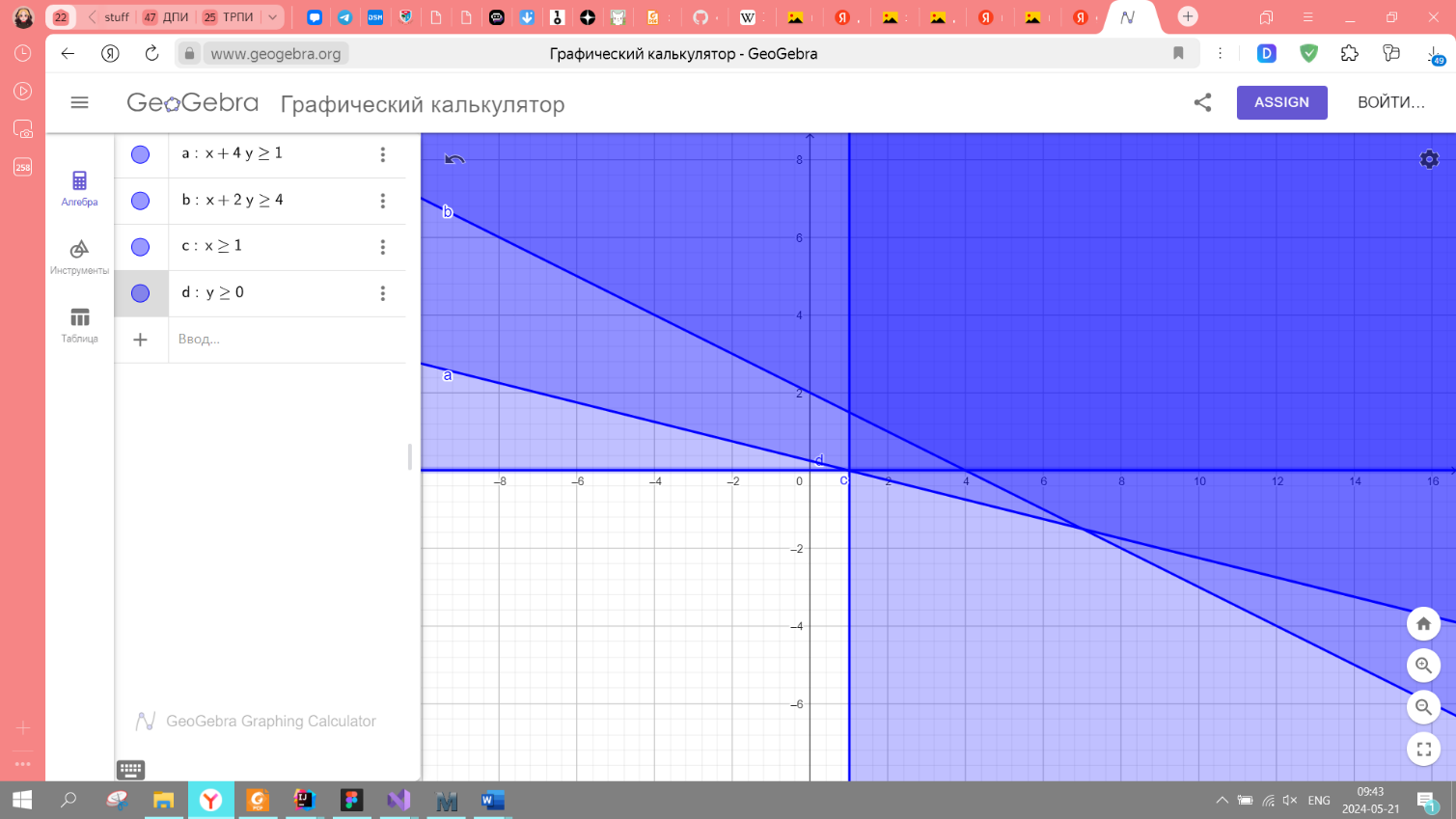


Строим y ≥ 0

Это параллельная оси X прямая, которая лежит на ней и пересекает ось Y 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.



B

A

Точка А: это пересечение прямых x + 2y = 4 и y = 0.

Точка A: (4, 0)

Так как x > 1, точка A попадает под ограничение x1.

Если координаты точки A подставить в неравенство x + 4y 1, то получится:

=> .

Следовательно, точка A попадает под ограничение .

Точка B: это пересечение прямых

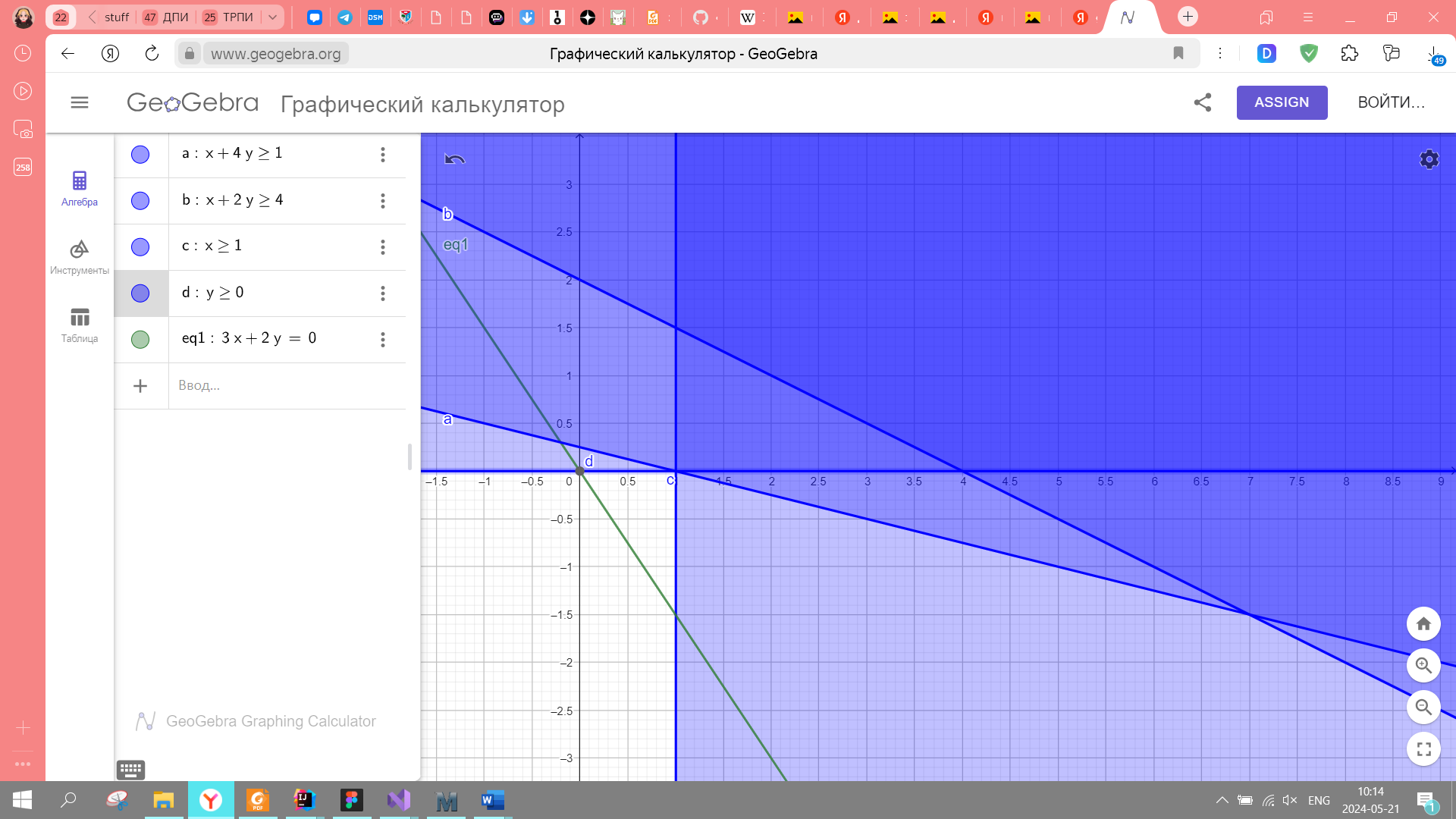
Точка B: (1; 1,5)

Так как y > 0, точка B попадает под ограничение y0.

Если координаты точки B подставить в неравенство x + 4y 1, то получится:

=> , следовательно, точка B попадает под ограничение .

Cтроим прямую 3x + 2y = 0.



A

B

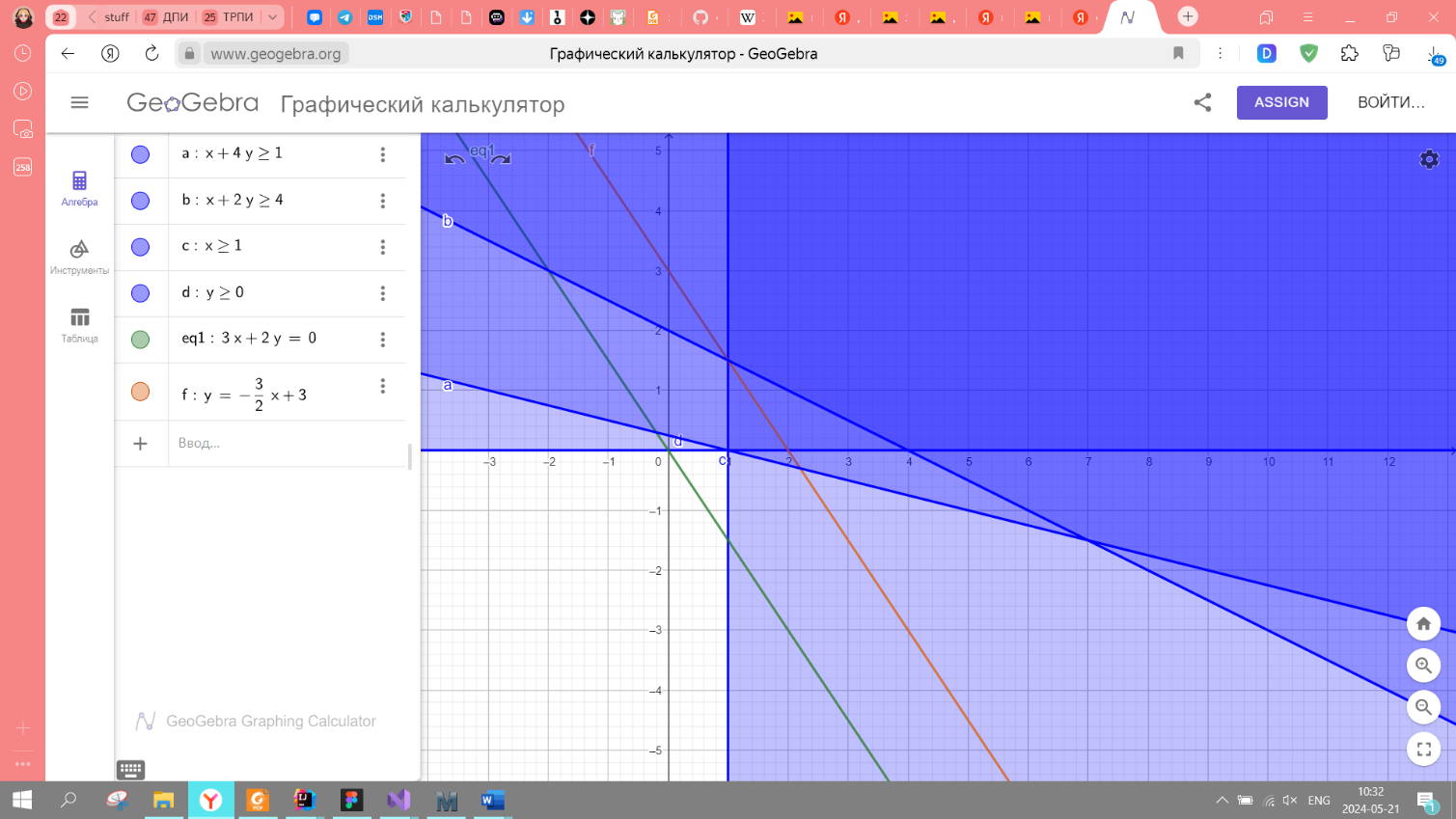
Наше min будет равно значению коэффициента b пересечения целевой функции с точкой B, так как в этой точке наша прямая в первый раз соприкасается с областью решений перемещаясь вверх по оси Y.

*Уравнением прямой будет*

*Bmin* будет равно коэффициенту b прямой при прохождении точки B: .

*Bmin = b = y + =*

*Bmin = 3*



Bmin

A

B

**Вывод:** в ходе выполнения лабораторной работы был освоен графический метод решения задач.