Учреждение образования

«БЕЛОРУССКИЙ ГОСУДАРСТВЕННЫЙ ТЕХНОЛОГИЧЕСКИЙ УНИВЕРСИТЕТ»

Дисциплина «Математическое программирование»

Отчёт по лабораторным работ

Студент: Лемешевский В.О

ФИТ 2 курс 2 группа

.

Минск 2024

**Содержание**

[Лабораторная работа 1. Вспомогательные функции 3](#_Toc166283757)

[Лабораторная работа 2. Комбинаторные алгоритмы решения оптимизационных задач 10](#_Toc166283758)

[Лабораторная работа 3. Метод ветвей и границ. Задача коммивояжера и методы её решения. 26](#_Toc166283759)

[Лабораторная работа 4 33](#_Toc166283760)

[Лабораторная работа 5. ТРАНСПОРТНАЯ ЗАДАЧА 41](#_Toc166283761)

[Лабораторная работа 6. АЛГОРИТМЫ НА ГРАФАХ 51](#_Toc166283762)

[Лабораторная работа 7. Сетевые модели 71](#_Toc166283763)

[Лабораторная работа 8. Графический метод решения оптимизационных задач 75](#_Toc166283764)

# Лабораторная работа 1. Вспомогательные функции

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

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

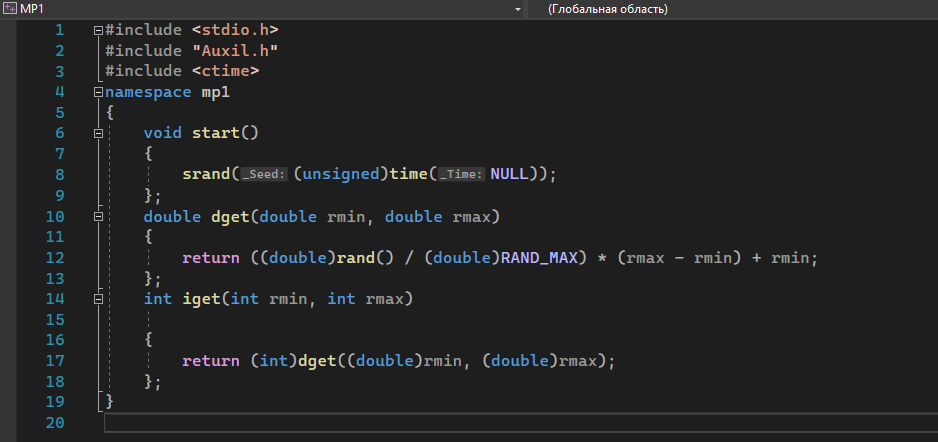


Рисунок 1.1 – Файл Auxio.cpp

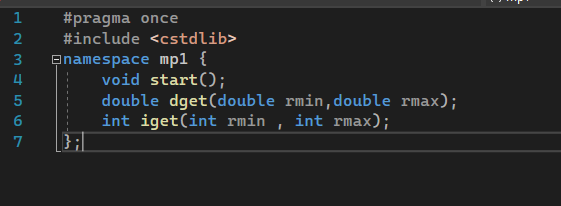


Рисунок 1.1 – Файл Auxio.cpp

***Задание 2***

1. Реализовать пример 2. Представлено на рисунке 1.3.

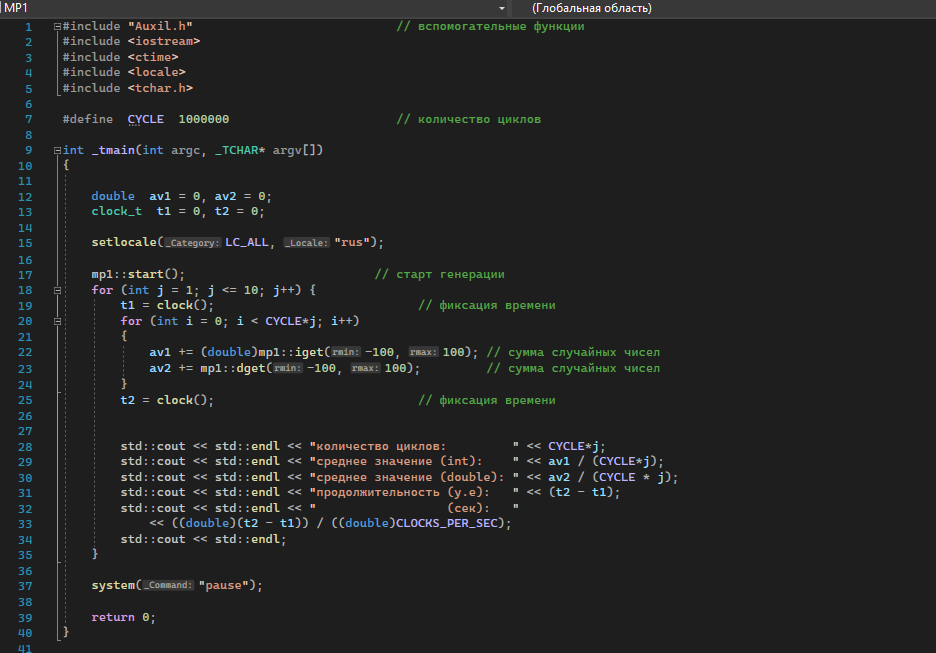


Рисунок 1.3 – Файл Lab1.cpp

1. Для проверки работоспособности разработанных функций и приобретения навыков замера продолжительности процесса вычисления реализуйте программу, приведенную в примере 2. Представлено на рисуноке 1.4

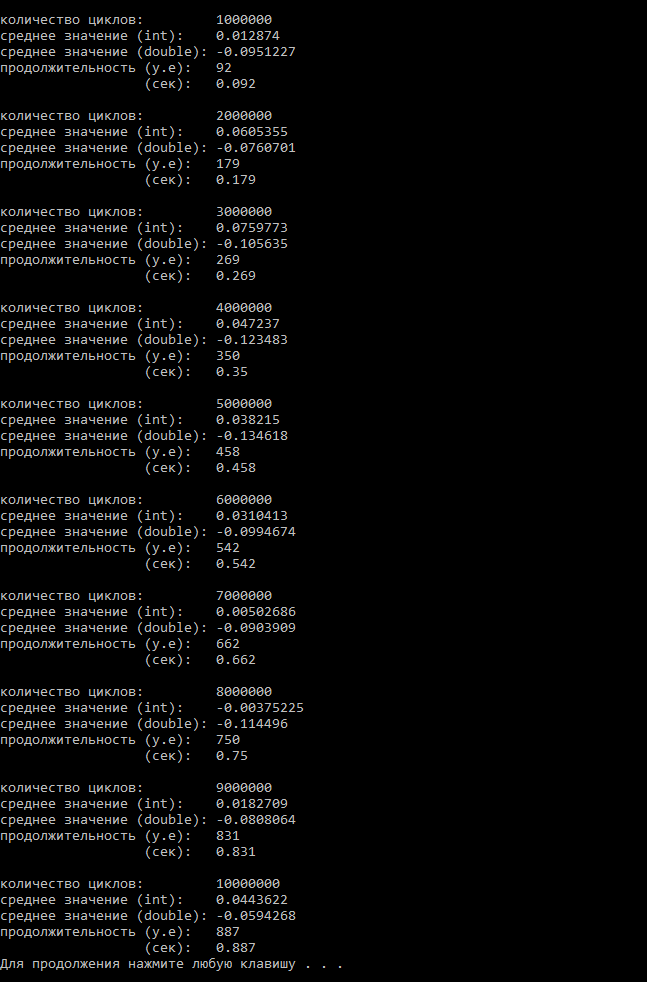


Рисунок 1.4 – Результат замера 10-ти циклов

**Задание 3**

Проведите необходимые эксперименты и постройте график зависимости (Excel) продолжительности процесса вычисления от количества циклов в примере 2. Проанализируйте характер зависимости. Результаты измерений и соответствующий график линейной зависимости приведены на рисунке 1.5.

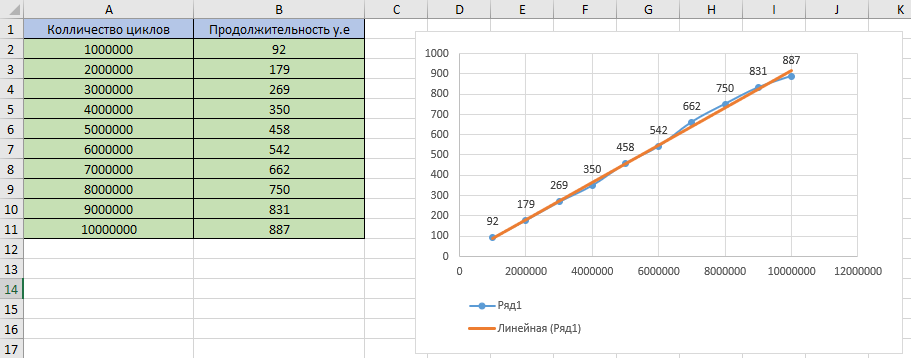


Рисунок 1.5 – Данные и график в Excel

Проведите исследование любого другого рекурсивного алгоритма, например, вычисления факториала или генератора чисел Фибоначчи (прим. – например вычислите каким будет 100-е, 200-е, 300-е и т.д число), и включите в отчет график.

Для примера был взят алгоритм вычисления чисел Фибоначчи , а именно какая будет продолжительность вычисления от 35го до 45го числа. Рабочий вариант кода представлен на рисунке 1.6.

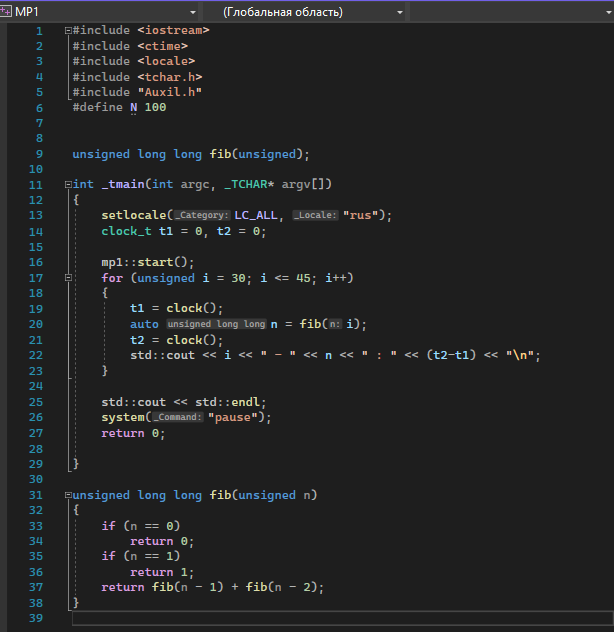


Рисунок 1.6 – Алгоритм вычисления

На рисунке 1.7 представлен результат выполнения программы.

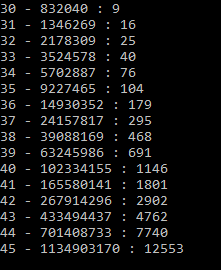


Рисунок 1.7 – Результат выполнения

На рисунке 1.8 и 1.9 представлен график и таблица значений и времени потраченного на вычисления чисел от 30 до 45.

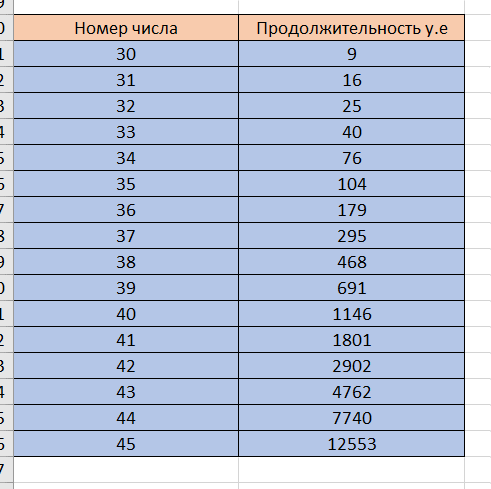


Рисунок 1.8 – Таблица чисел

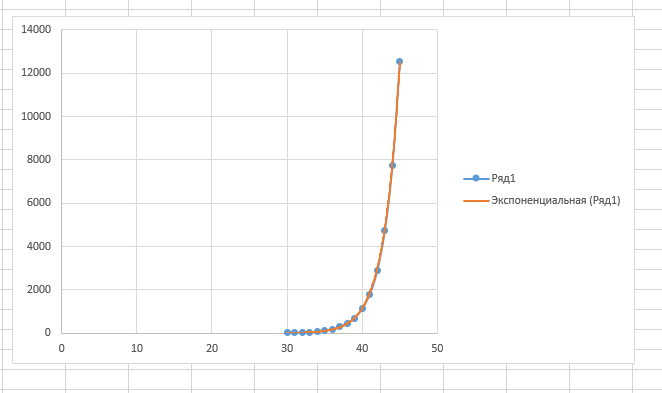


Рисунок 1.9 – График затрат

**Вывод:**

Продолжительность выполнения программы линейно зависит от количества итерация и циклов выполнения, что видно на графике рисунка 1.4.

В результате по графику определено, что чем больше число Фибоначчи, тем больше времени нужно для его подсчета в экспоненциальной зависимости.

# 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.

**ЗАДАНИЕ ДЛЯ ВЫПОЛНЕНИЯ:**

Для выполнения лабораторной работы был разработан функционал консольного меню, код представлен на рисунке 1.



Рисунок 1 – Реализация меню с помощью switch/case.

Так же на рисунке 2 представлен внешний вид консольного меню.

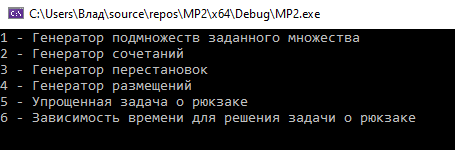


Рисунок 2 – Формат меню.

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

Заголовочный файл , со всеми методами представлен на рисунке 1.1.

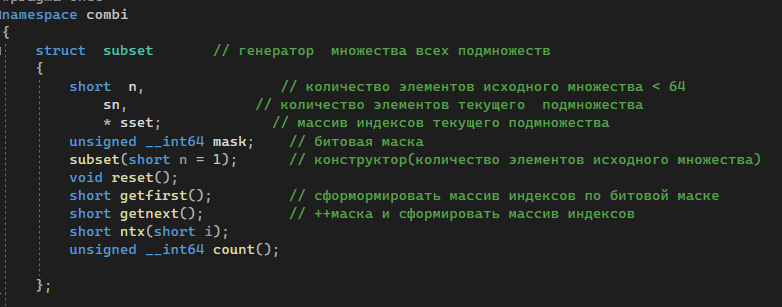


Рисунок 1.1 – Код заголовочного файла для генератора множества всех подмножеств.

На рисунке 1.2 представлена реализация методов заголовочного файла.

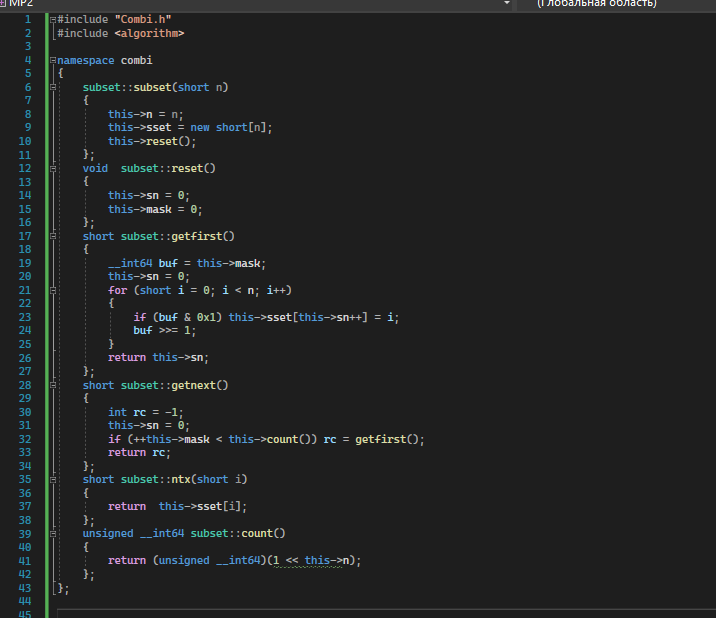


Рисунок 1.2 – Реализация генератора

Пример применения и его функционал реализован в файле MP2.cpp.

Код представлен на рисунке 1.3.

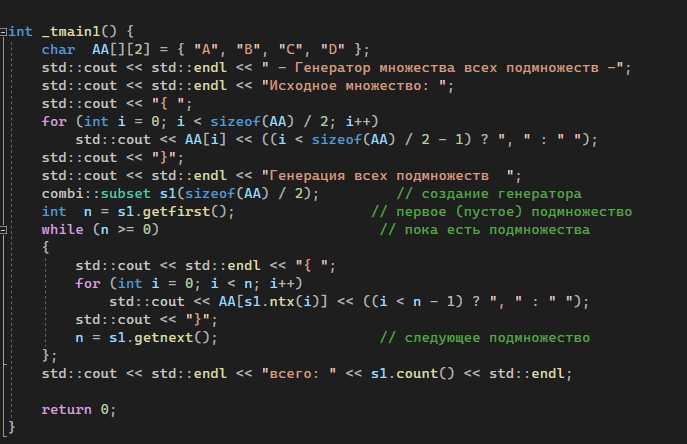


Рисунок 1.3 – Пример применения

Результат выполнения представлен на рисунке 1.4.

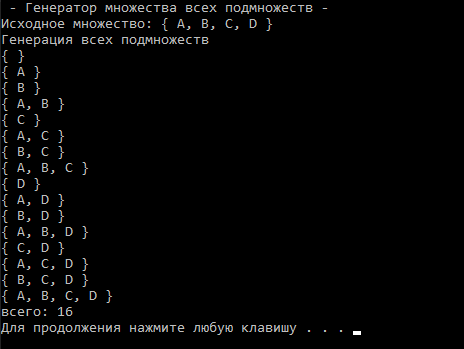


Рисунок 1.4 – Результат генератора множества всех подмножеств.

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

Заголовочный файл , со всеми методами представлен на рисунке 2.1.

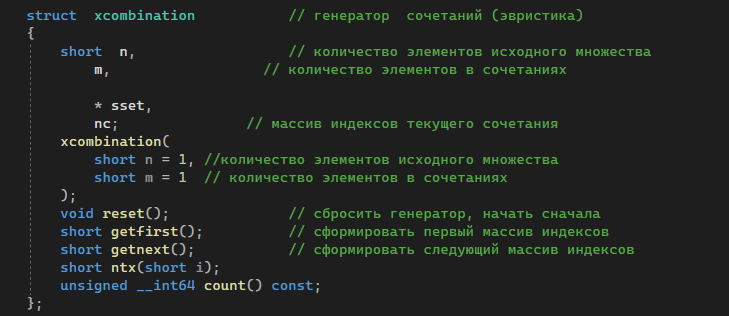


Рисунок 2.1 – Код заголовочного файла для генератора сочетаний

На рисунке 2.2 представлена реализация методов заголовочного файла.

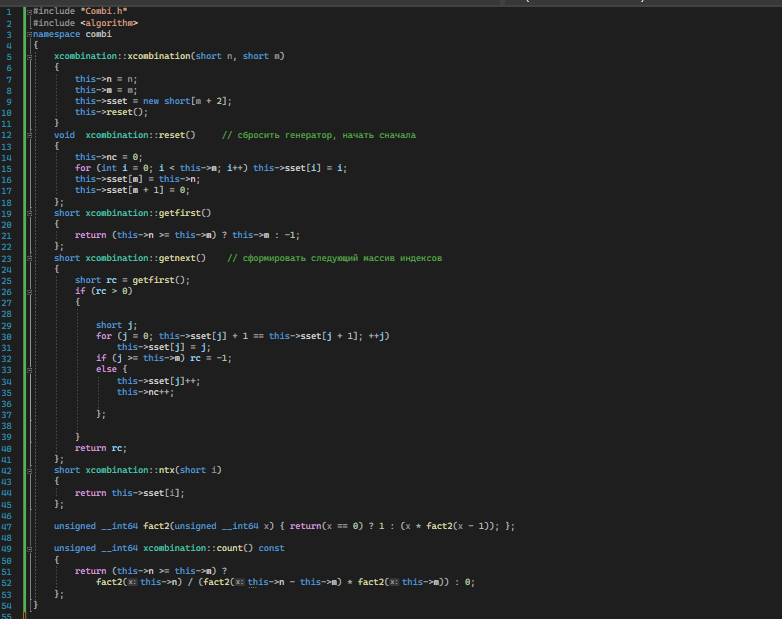


Рисунок 2.2 – Реализация генератора

Пример применения и его функционал реализован в файле MP2.cpp.

Код представлен на рисунке 2.3.

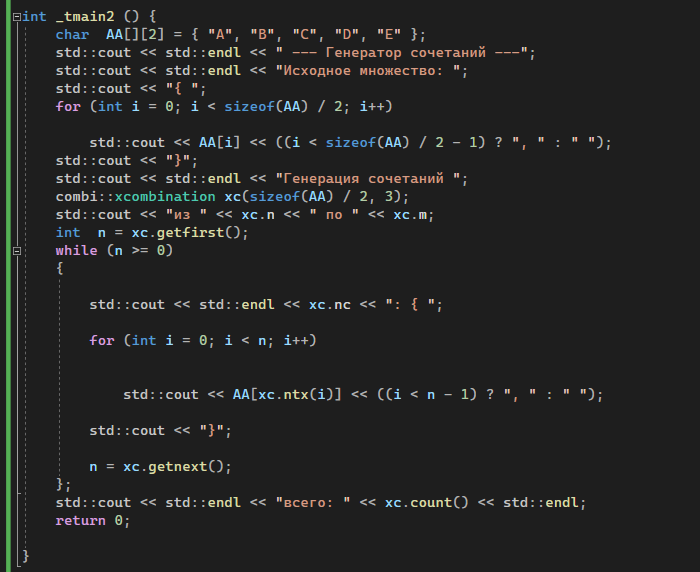


Рисунок 2.3 – Пример применения

Результат выполнения представлен на рисунке 2.4.

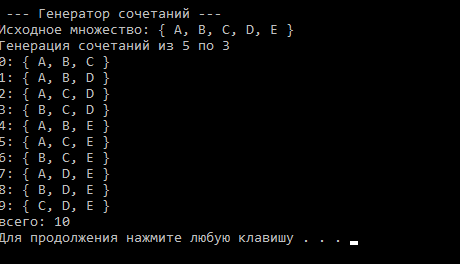


Рисунок 2.4 – Результат генератора сочетаний

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

Заголовочный файл , со всеми методами представлен на рисунке 3.1.

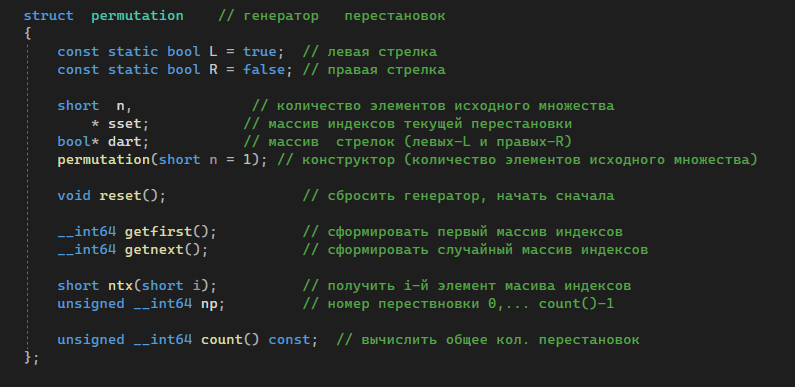


Рисунок 3.1 – Код заголовочного файла для генератора сочетаний

На рисунке 3.2 представлена реализация методов заголовочного файла.

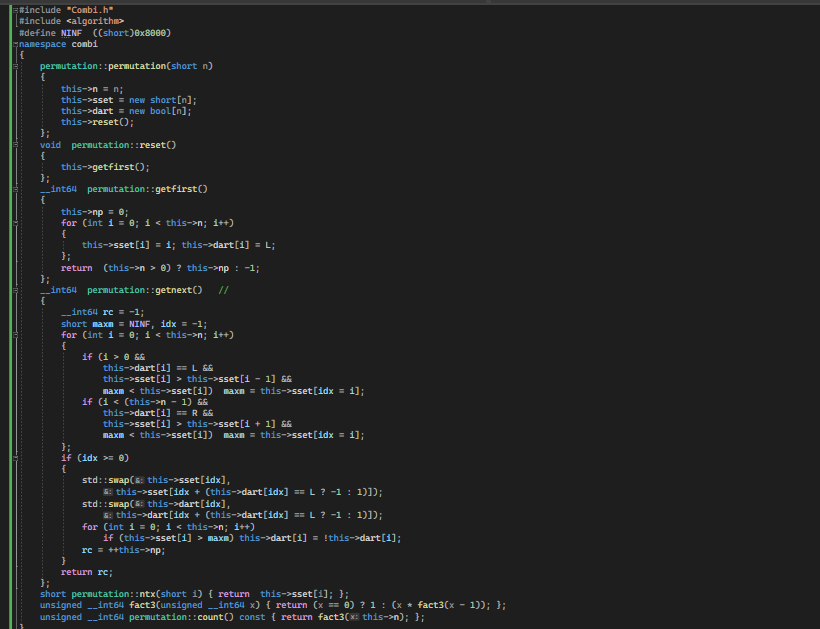


Рисунок 3.2 – Реализация генератора перестановок

Пример применения и его функционал реализован в файле MP2.cpp.

Код представлен на рисунке 3.3.

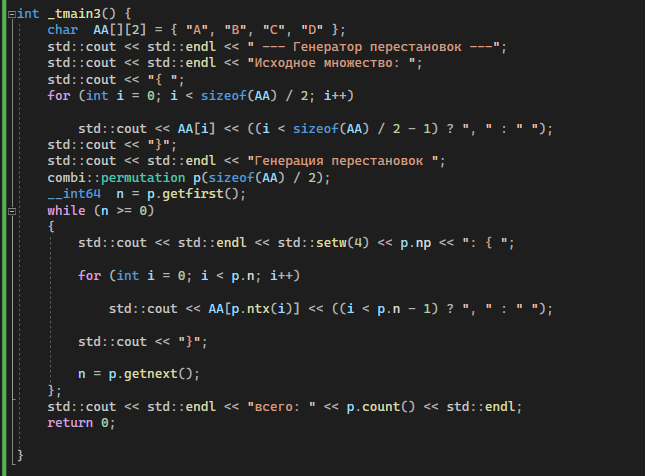


Рисунок 3.3 – Пример применения

Результат выполнения представлен на рисунке 3.4.

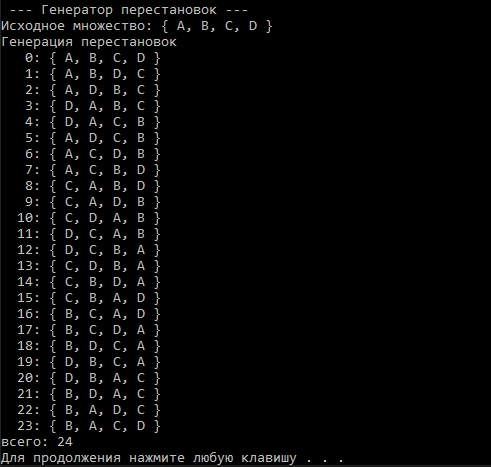


Рисунок 3.4 – Результат генератора перестановок.

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

Заголовочный файл , со всеми методами представлен на рисунке 4.1.

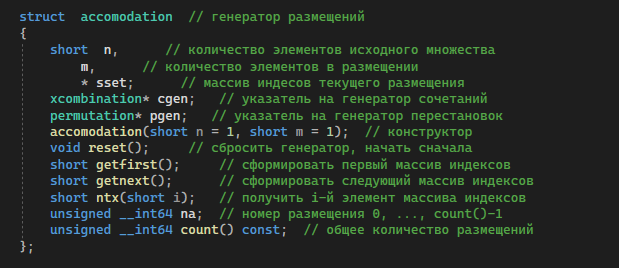


Рисунок 3.1 – Код заголовочного файла для генератора размещений

На рисунке 4.2 представлена реализация методов заголовочного файла.

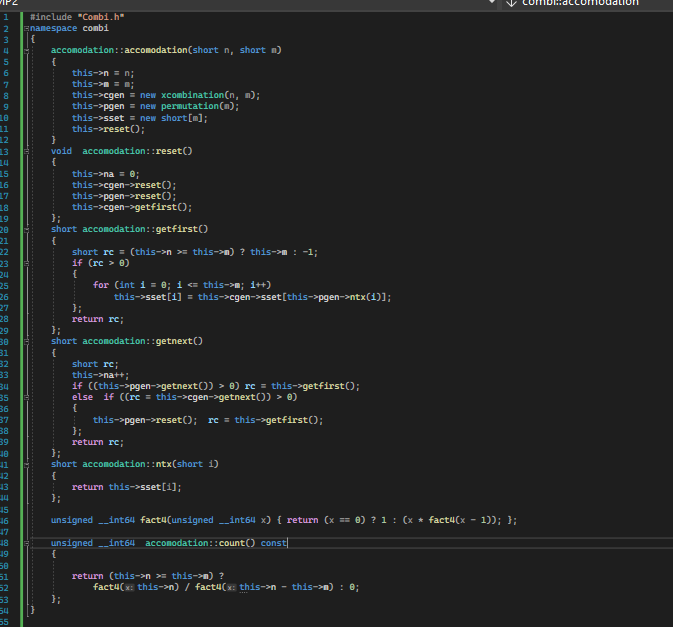


Рисунок 4.2 – Реализация генератора размещений

Пример применения и его функционал реализован в файле MP2.cpp.

Код представлен на рисунке 4.3.

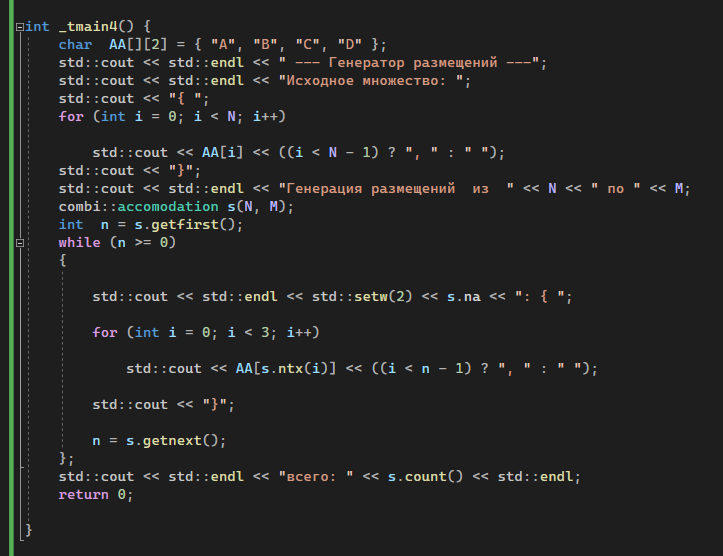


Рисунок 4.3 – Пример применения

Результат выполнения представлен на рисунке 4.4.

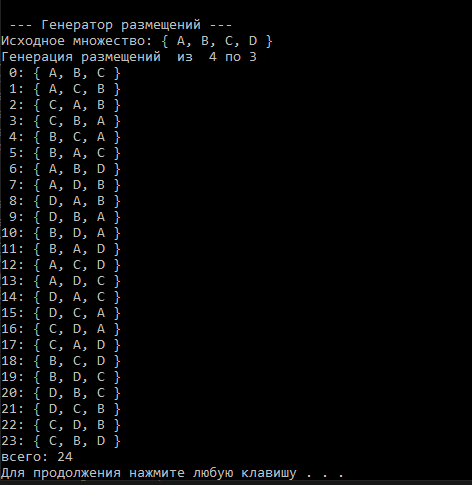


Рисунок 4.4 – Результат генератора размещений.

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

Упрощенную о рюкзаке (веса предметов и их стоимость сгенерировать случайным образом: вместимость рюкзака 300 кг, веса предметов 10 – 300 кг, стоимость предметов 5 – 55 у.е.; количество предметов – 18 шт.);

Для генерации случайных чисел использовался код из Лабораторной работы №1.

На рис. 5.1 изображена схема решения задачи о рюкзаке с применением генератора множества всех подмножеств.. Задача имеет следующие исходные данные:

V= 300 – вместимость (объем) рюкзака;

n = 18 – количество предметов;

Берутся случайным образом – вектор объемов предметов;

Берутся случайным образом – вектор стоимостей предметов.



Рисунок 5.1 – Схема решения задачи

Заголовочный файл с шаблоном реализации представлен на рисунке 5.2.

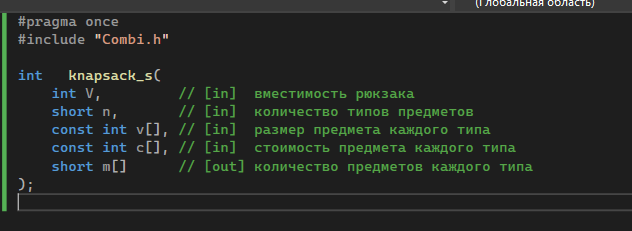


Рисунок 5.2 – файл Knapsack.h

Реализация заголовочного файла и вся логика находится в файле Knapsack.cpp. Общий вид представлен на рисунке 5.3.

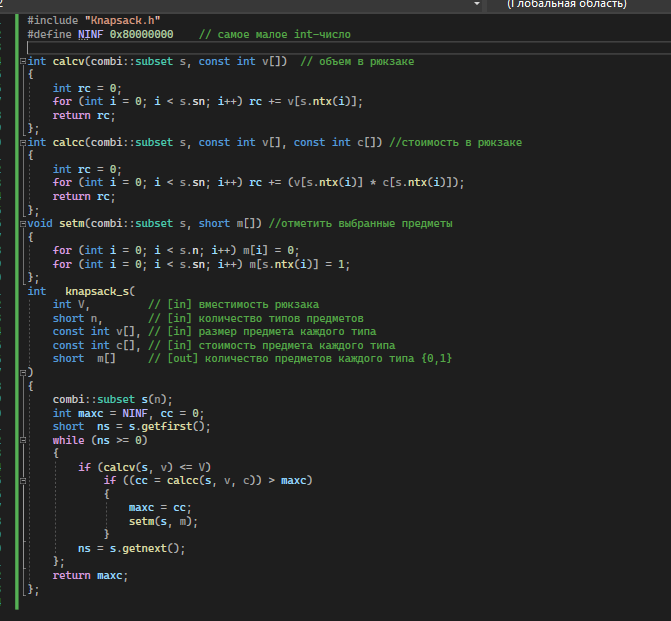


Рисунок 5.3 – Реализация задачи

Пример применения данной реализации представлен на рисунке 5.4.

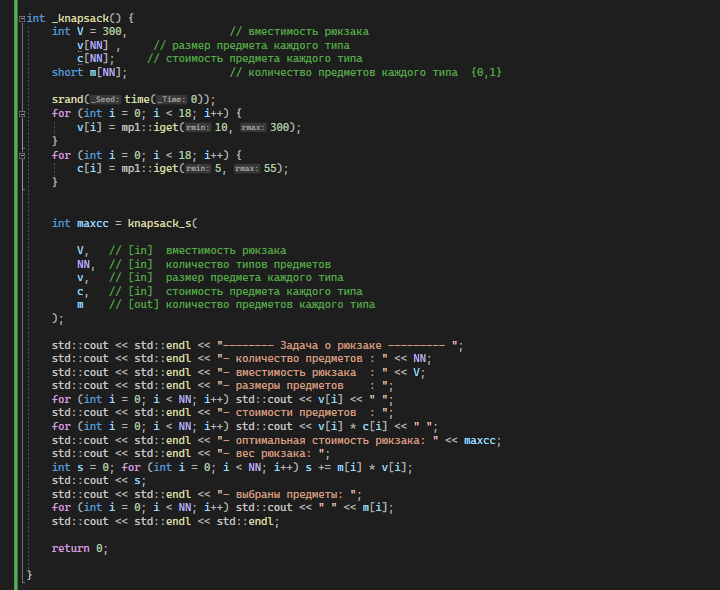


Рисунок 5.4 – Пример реализации задачи упрощенной о рюкзаке

Результат выполнения задачи представлен на рисунке 5.5.

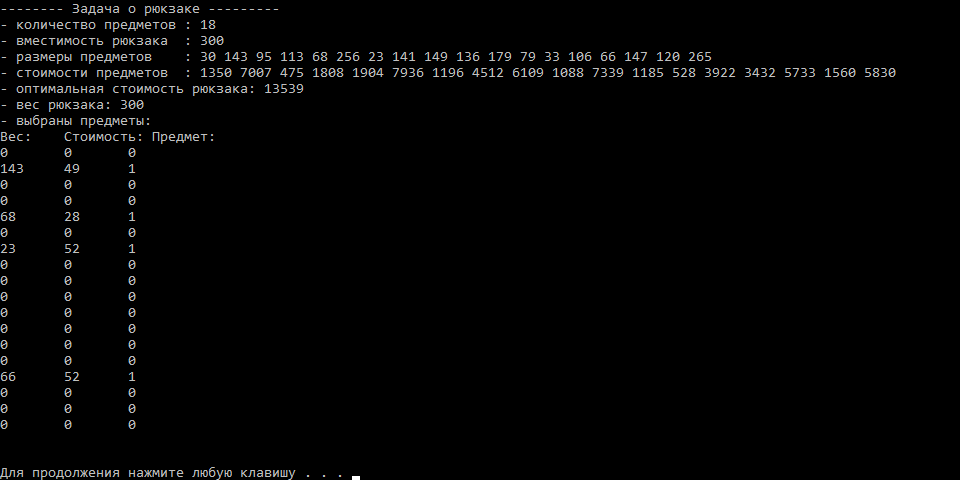


Рисунок 5.5 – Результат выполнения

**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:

(2, 6, 10, 14) упрощенную о рюкзаке (количество предметов 12 – 20 шт.);

Для решения этой задачи использовался тот же заголовочный файл и его реализация. Для определения времени использовался метод clock().

Пример выполнения программы и код представлен на рисунке 6.1.

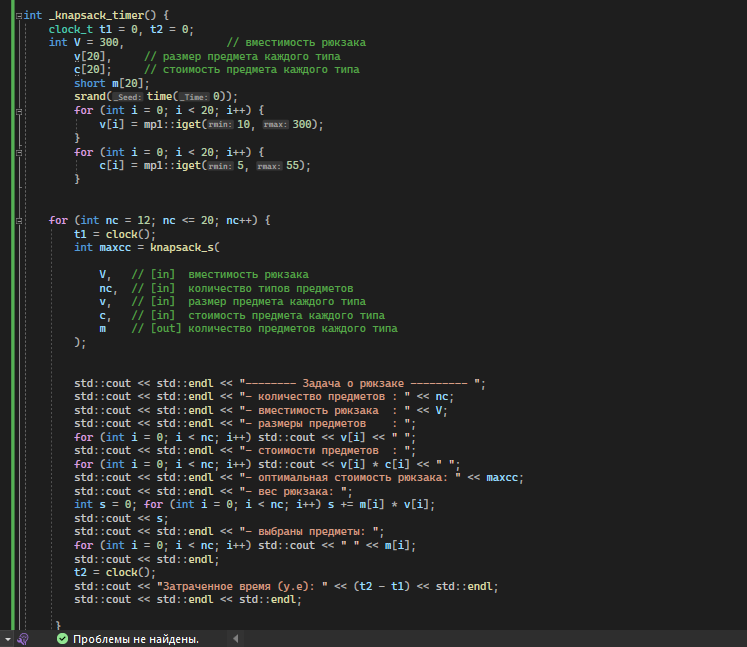
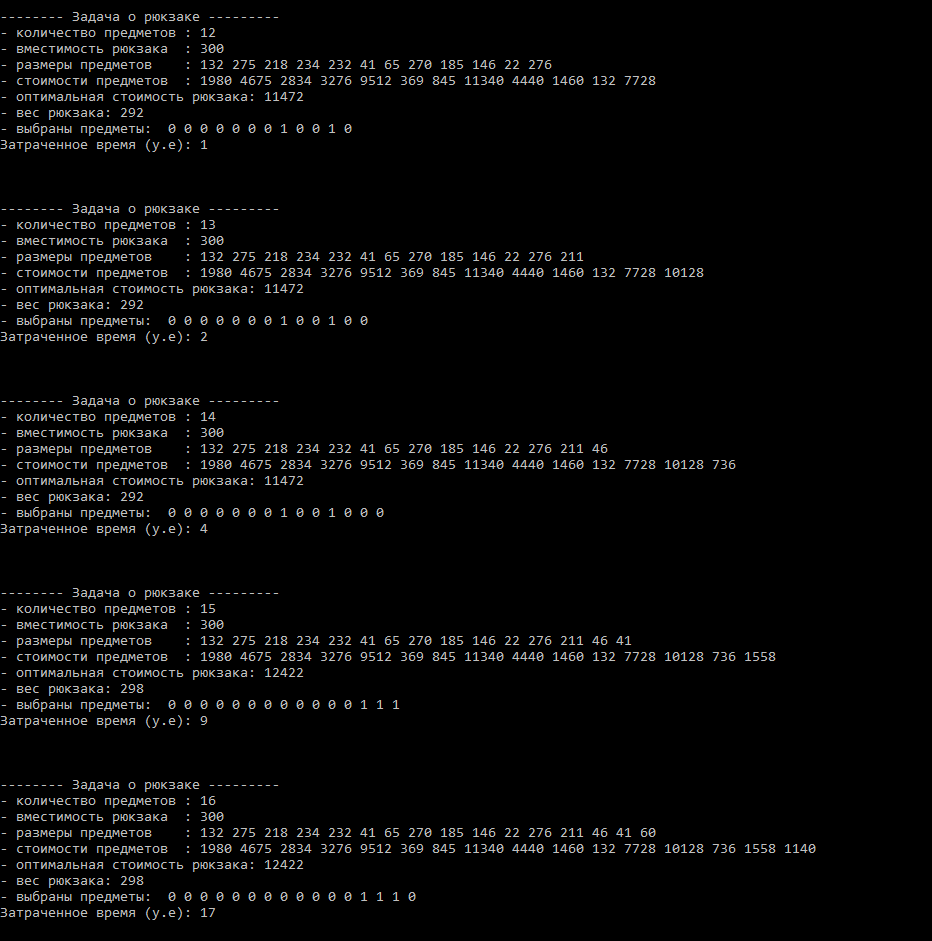


Рисунок 6.1 – Пример выполнения задачи о рюкзаке с

12-20 предметами

Результат на рисунке 6.2.



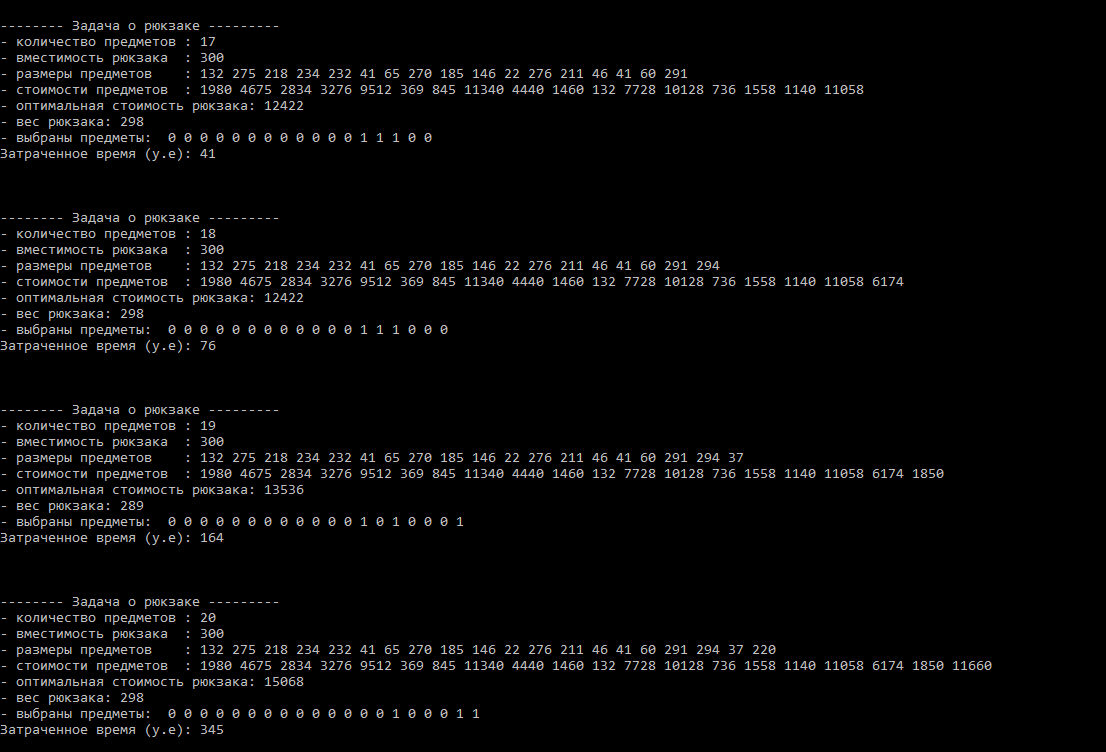


Рисунок 6.2 – Результат генерации

Результат выполнения так же был представлен в таблице и графике Excel. Таблица и график представлены на рисунка 6.3 и 6.4.

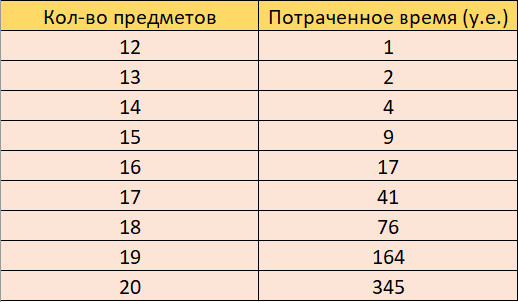


Рисунок 6.3 – Таблица зависимости

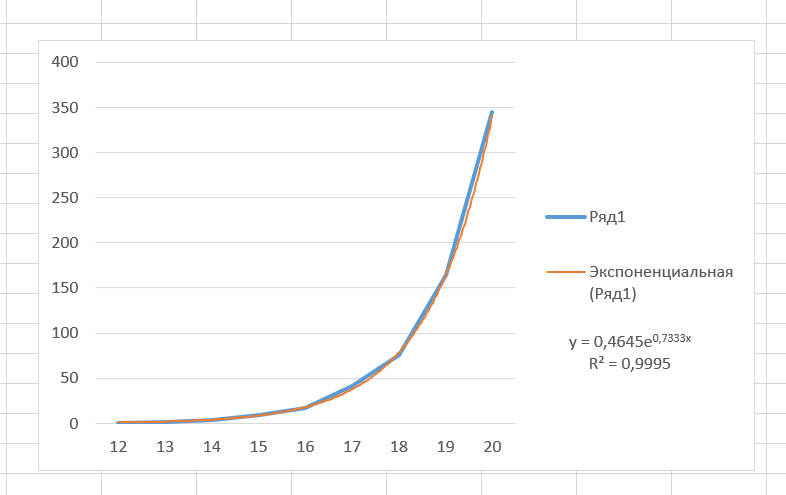


Рисунок 6.4 – График зависимости

**Вывод:**

В ходе выполнения работы были приобретены навыки разработки генераторов подмножеств, перестановок, сочетаний и размещений на С++; научились применять разработанные генераторы для решения задач о рюкзаке.

В результате эксперимента с задачей о рюкзаке и определения зависимости времени выполнения задачи от количества предметов было определено что график представляет экспоненциальную зависимости.

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

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

**Задание 1.** Сформулировать условие задачи коммивояжера с параметром. Для этого:

* принять элементы матрицы расстояний равными:

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| * **Город** | **1** | **2** | **3** | **4** | **5** |
| **1** |  | 20 | 31 |  | 10 |
| **2** | 10 |  | 25 | 58 | 74 |
| **3** | 12 | 30 |  | 86 | 59 |
| **4** | 27 | 48 | 40 |  | 30 |
| **5** | 83 | 76 | 52 | 23 |  |

где *n* – номер варианта или номер по журналу (10);

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

Задачу следует решить с использованием метода ветвей и границ.

Находим минимальное значение в каждой строке и выписываем его в отдельный столбец.

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
|  | 20 | 31 |  | 10 | 10 |
| 10 |  | 25 | 58 | 74 | 10 |
| 12 | 30 |  | 86 | 59 | 12 |
| 27 | 48 | 40 |  | 30 | 27 |
| 83 | 76 | 52 | 23 |  | 23 |

Производим приведение строк.

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
|  | 10 | 21 |  | 0 |
| 0 |  | 15 | 48 | 64 |
| 0 | 18 |  | 74 | 47 |
| 0 | 21 | 13 |  | 3 |
| 60 | 53 | 29 | 0 |  |

Такую же операцию проводим по столбцам, для чего в каждом столбце находим минимальный элемент:

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
|  | 10 | 21 |  | 0 |
| 0 |  | 15 | 48 | 64 |
| 0 | 18 |  | 74 | 47 |
| 0 | 21 | 13 |  | 3 |
| 60 | 53 | 29 | 0 |  |
| 0 | 10 | 13 | 0 | 0 |

Производим приведение столбцов.

Полностью приведенная матрица:

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
|  | 0 | 8 |  | 0 |
| 0 |  | 2 | 48 | 64 |
| 0 | 8 |  | 74 | 47 |
| 0 | 11 | 0 |  | 3 |
| 60 | 43 | 16 | 0 |  |

После вычитания минимальных элементов получаем полностью приведенную матрицу, где величины di и dj называются **константами приведения**. Сумма констант приведения определяет нижнюю границу

H = 10+10+12+27+23+0+10+13+0+0= 105;

H=105;

**Определяем ребро ветвления** и разобьем все множество маршрутов относительно этого ребра на два подмножества. С этой целью для всех клеток матрицы с нулевыми элементами заменяем поочередно нули на бесконечность и определяем для них сумму образовавшихся констант приведения, они приведены в скобках.

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
|  | 0(8+0) | 8 |  | 0(0+3) |
| 0(0+2) |  | 2 | 48 | 64 |
| 0(0+8) | 8 |  | 74 | 47 |
| 0(0+0) | 11 | 0(2+0) |  | 3 |
| 60 | 43 | 16 | 0(16+48) |  |

Получим:

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
|  | 0(8) | 8 |  | 0(3) |
| 0(2) |  | 2 | 48 | 64 |
| 0(8) | 8 |  | 74 | 47 |
| 0(0) | 11 | 0(2) |  | 3 |
| 60 | 43 | 16 | 0(64) |  |

Наибольшая сумма констант приведения равна (16+48) = 64 для ребра (5,4), то множество разбивается на два подмножества (5,4) и (5\*,4\*).

**Исключение ребра** (5,4) проводим путем замены элемента d54 = 0 на бесконечность, после чего осуществляем очередное приведение матрицы расстояний для образовавшегося подмножества (5\*,4\*), в результате получим приведенную матрицу. Нижняя граница гамильтоновых циклов этого подмножества: H(5\*,4\*) = 105 + 64=169.

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
|  | 0 | 8 |  | 0 | 0 |
| 0 |  | 2 | 48 | 64 | 0 |
| 0 | 8 |  | 74 | 47 | 0 |
| 0 | 11 | 0 |  | 3 | 0 |
| 60 | 43 | 16 |  |  | 16 |
| 0 | 0 | 0 | 48 | 0 | 64 |

**Включение ребра** (5,4) проводится путем исключения всех элементов 5-ой строки и 4-го столбца, в которой элемент d54 заменяем на бесконечность, для исключения образования не гамильтонова цикла.

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
|  | 0 | 8 |  | 0 |
| 0 |  | 2 | 48 | 64 |
| 0 | 8 |  | 74 | 47 |
| 0 | 11 | 0 |  |  |
| 60 | 43 | 16 |  |  |

В результате получим другую сокращенную матрицу (4 x 4), которая подлежит операции приведения. В итоге получаем приведенную матрицу. Сумма констант приведения сокращенной матрицы 0. Нижняя граница подмножества (5,4) равна: H(5,4) = 105 + 0 = 105 ≤ 169. Поскольку нижняя граница этого подмножества (5,4) меньше, чем подмножества (5\*,4\*), то ребро (5,4) включаем в маршрут с новой границей H = 105.

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
|  | 0 | 8 | 0 | 0 |
| 0 |  | 2 | 64 | 0 |
| 0 | 8 |  | 47 | 0 |
| 0 | 11 | 0 |  | 0 |
| 0 | 0 | 0 | 0 | 0 |

Полностью приведенная матрица:

|  |  |  |  |
| --- | --- | --- | --- |
|  | 0 | 8 | 0 |
| 0 |  | 2 | 64 |
| 0 | 8 |  | 47 |
| 0 | 11 | 0 |  |

**Определяем ребро ветвления**.

|  |  |  |  |
| --- | --- | --- | --- |
|  | 0(0+8) | 8 | 0(0+47) |
| 0(0+2) |  | 2 | 64 |
| 0(0+8) | 8 |  | 47 |
| 0(0+0) | 11 | 0(0+2) |  |

|  |  |  |  |
| --- | --- | --- | --- |
|  | 0(8) | 8 | 0(47) |
| 0(2) |  | 2 | 64 |
| 0(8) | 8 |  | 47 |
| 0(0) | 11 | 0(2) |  |

**Исключение ребра** (1,5): d15=∞.

H(1\*,5\*) = 105 + 47 = 152

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
|  | 0 | 8 |  | 0 |
| 0 |  | 2 | 64 | 0 |
| 0 | 8 |  | 47 | 0 |
| 0 | 11 | 0 |  | 0 |
| 0 | 0 | 0 | 47 | 47 |

**Включение ребра** (1,5): d15=∞.

|  |  |  |  |
| --- | --- | --- | --- |
|  | 0 | 8 |  |
| 0 |  | 2 | 64 |
| 0 | 8 |  | 47 |
| 0 | 11 | 0 |  |

∑di + ∑dj = 8.

H(1,5) = 105 + 8 = 113 ≤ 152.

Чтобы исключить подциклы, запретим следующий переход: (2,2).  
Ребро (1,5) включаем в маршрут с новой границей H=113.

|  |  |  |  |
| --- | --- | --- | --- |
| 0 |  | 2 | 0 |
| 0 | 8 |  | 0 |
| 0 | 11 | 0 | 0 |
| 0 | 8 | 0 | 8 |

Полностью приведенная матрица:

|  |  |  |
| --- | --- | --- |
| 0 |  | 2 |
| 0 | 0 |  |
| 0 | 3 | 0 |

**Определяем ребро ветвления**.

|  |  |  |
| --- | --- | --- |
| 0(2) |  | 2 |
| 0(0) | 0(3) |  |
| 0(3) | 3 | 0(2) |

1) Не включаем 3-2

2) Включаем 3-2

**Исключение ребра** (3,2): d32=∞.

H(3\*,2\*) = 113 + 3 = 116

|  |  |  |  |
| --- | --- | --- | --- |
| 0 |  | 2 | 0 |
| 0 |  |  | 0 |
| 0 | 3 | 0 | 0 |
| 0 | 3 | 0 | 3 |

**Включение ребра** (3,2): d32=∞.

|  |  |  |
| --- | --- | --- |
| 0 |  | 2 |
| 0 |  |  |
| 0 | 3 | 0 |

∑di +∑dj =0.

H(4,3) = 113 + 0 = 113 ≤ 116.

Ребро (3,2) включаем в маршрут с новой границей H=113.

|  |  |  |
| --- | --- | --- |
| 0 | 2 | 0 |
| 0 | 0 | 0 |
| 0 | 0 | 0 |

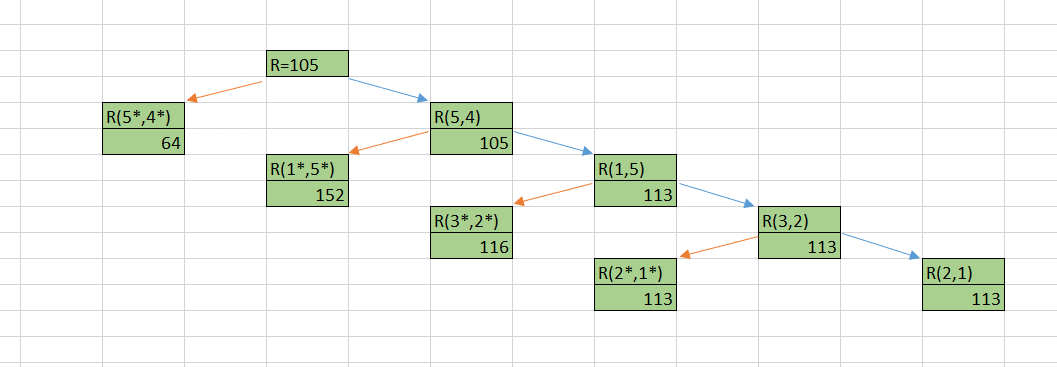
В соответствии с этой матрицей включаем в гамильтонов маршрут ребра (4,3) и(2,1).

|  |  |
| --- | --- |
| 0 | 2 |
| 0 | 0 |

Длина маршрута равна F(∞) = 113

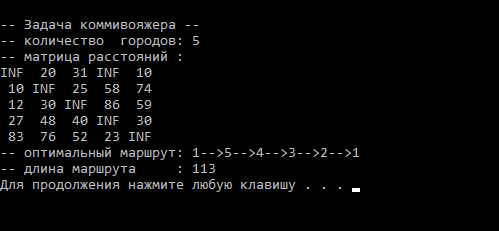


Граф пути выглядит следующим образом:



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

Результат выполнения программы:



**Вывод:** Освоили общие принципы решения задач методом ветвей и границ, решили задачу о коммивояжере комбинаторным методом перестановок. Решение, которое мы получили при помощи генератора перестановок, совпадает с решением, полученным при использовании метода ветвей и границ.

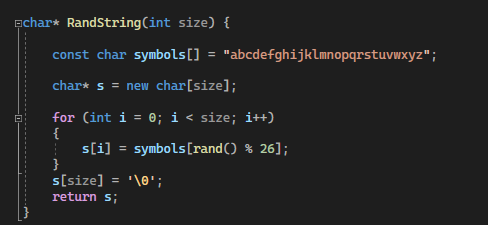
# Лабораторная работа 4

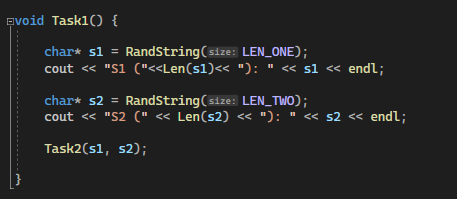
**ДИНАМИЧЕСКОЕ ПРОГРАММИРОВАНИЕ**

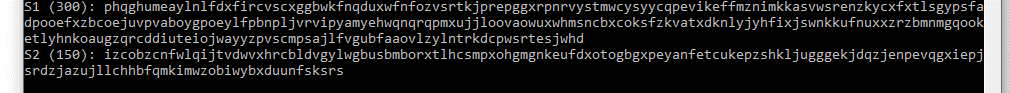
**ЦЕЛЬ РАБОТЫ:** освоить общие принципы решения задач методом динамического программирования, сравнить полученные решения задач с рекурсивным методом.

***Задание 1.***

На языке С++ сгенерировать случайным образом строку букв латинского алфавита S1 длиной 300 символов и S2 длиной150.

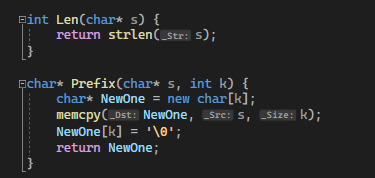


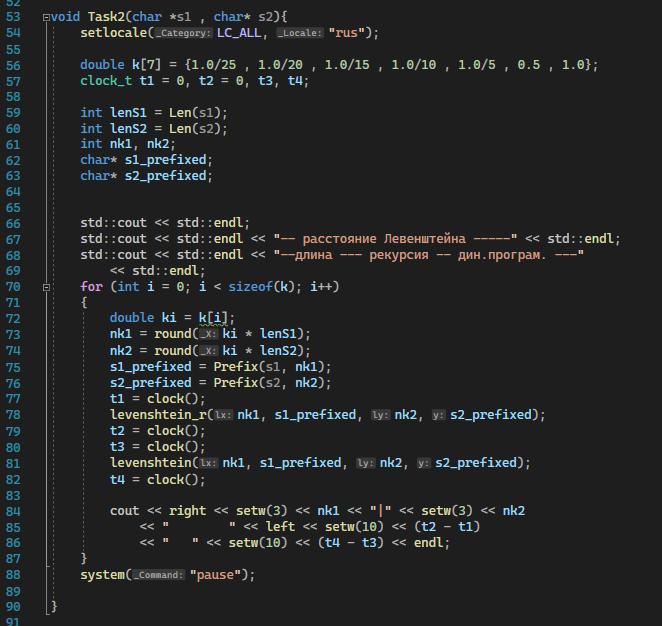


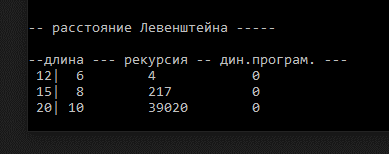


***Задание 2.***

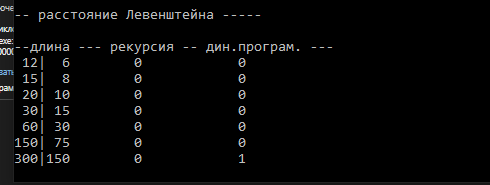
Вычислить двумя способами (рекурсивно и с помощью динамического программирования)  – дистанцию Левенштейна для , где - длина строки ,  - строка состоящая из первых  символов строки . (копии экрана и код вставить в отчет).





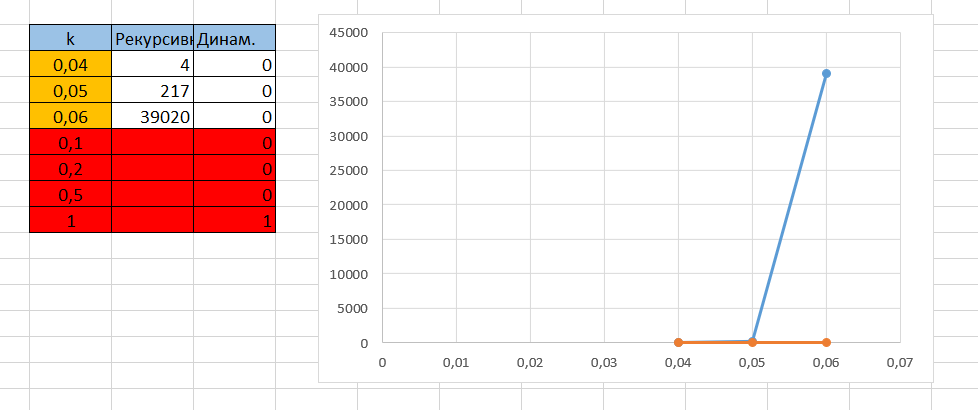


Если рассматривать только динамическое программирование:



***Задание 3.***

Выполнить сравнительный анализ времени затраченного на вычисление дистанции Левенштейна для двух методов решения. Построить графики зависимости времени вычисления от . (копии экрана и график вставить в отчет).



***Задание 4.***

Реализовать вручную пример вычисления дистанции Левенштейна при помощи рекурсивного алгоритма (в соответствии с вариантом) (каждый шаг алгоритма по примеру из лекции вставить в отчет).

|  |  |  |
| --- | --- | --- |
| 10 | Ель | Дрель |

* + - 1. L(“ель”, “дрель”) =
      2. L(“ел”, “дрель ”) =
      3. L(“ель”, “дрел ”) =
      4. L(“ел”, “дрел ”) =
      5. L(“е”, “дрель ”) =

L(“”, “дрель”) = 5

L(“”, “дрел”) = 4

* + - 1. L(“е”, “дрел ”) =

L(“”, “дрел”) = 4

L(“”, “дре”) = 3

* + - 1. L(“ель”, “дре ”) =
      2. L(“ел”, “дре”) =
      3. L(“ель”, “др”) =
      4. L(“ель”, “д”) =

L(“ель”, “”) = 3

L(“ел”, “”) = 2

* + - 1. L(“е”, “дре”) =

L(“”, “дре”) = 3

L(“”, “др”) = 2

* + - 1. L(“ел”, “др”) =

L(“е”, “д”) = 1

* + - 1. L(“е”, “др”) =

L(“”, “др”) = 2

L(“е”, “д”) = 1

L(“”, “д”) = 1

* + - 1. L(“ел”, “д”) =

L(“ел”, “”) = 2

L(“е”, “”) = 1

15. L(“е”, “д”) =

L(“е”, “”) = 1

L(“”, “д”) = 1

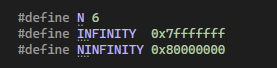
L(“”, “”) = 0

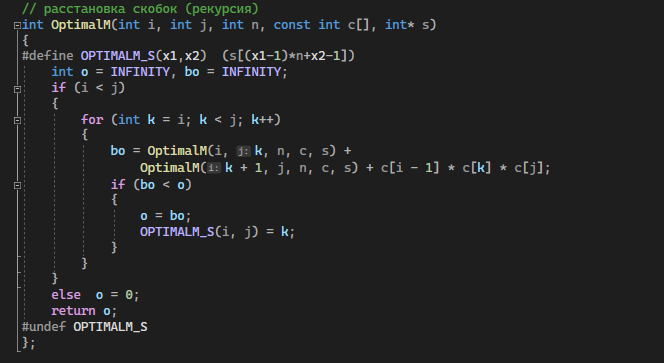
16. L(“е”, “д”) = min(2,2,1) = 1

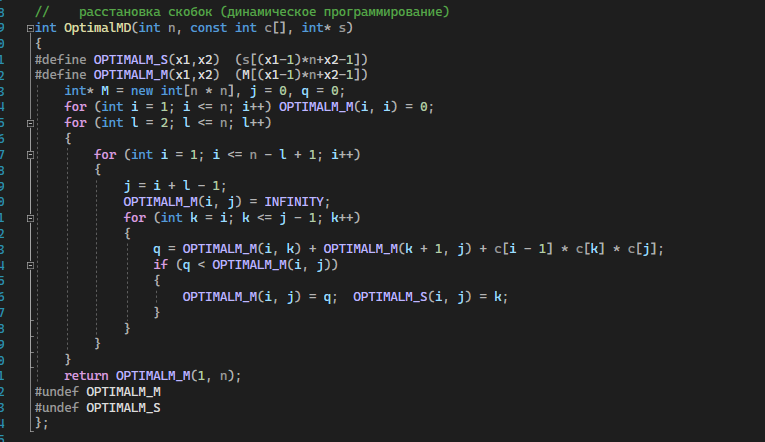
* + - 1. L(“ел”, “д”) = min(2, 3, 2) = 2
      2. L(“е”, “др”) = min(3, 2, 2) = 2
      3. L(“ел”, “др”) = min(3, 3, 2) = 2
      4. L(“е”, “дре”) = min(4, 3, 2) = 2
      5. L(“ель”, “д”) = min(3, 4, 3) = 3
      6. L(“ель”, “др”) = min(3, 4, 3) = 3
      7. L(“ел”, “дре”) = min(4, 3, 3) = 3
      8. L(“ель”, “дре”) = min(4, 4, 3) = 3
      9. L(“е”, “дрел”) = min(5, 4, 3) = 3
      10. L(“е”, “дрель”) = min(6, 4, 5) = 4
      11. L(“ел”, “дрел”) = min(4, 4, 2) = 2
      12. L(“ель”, “дрел”) = min(3, 4, 3) = 3
      13. L(“ел”, “дрель”) = min(5, 3, 4) = 3
      14. L(“ель”, “дрель”) = min(4, 4, 2) = 2

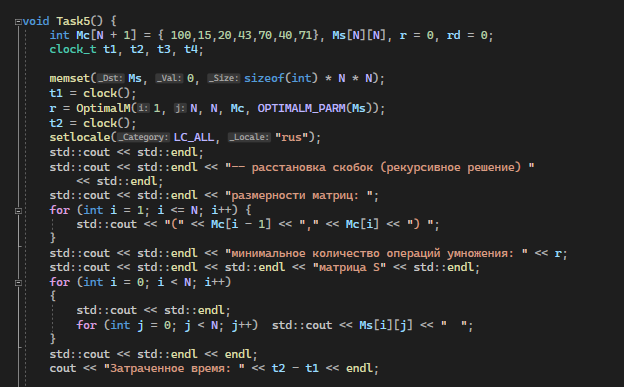
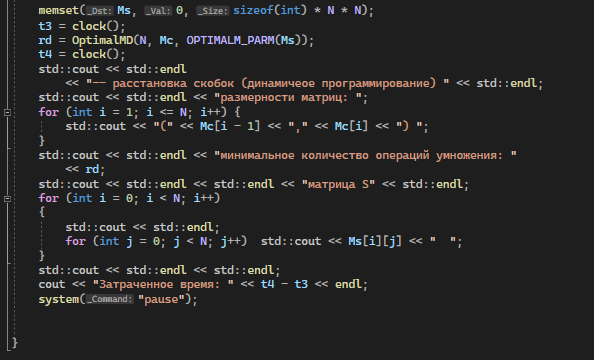
***Задание 5.***

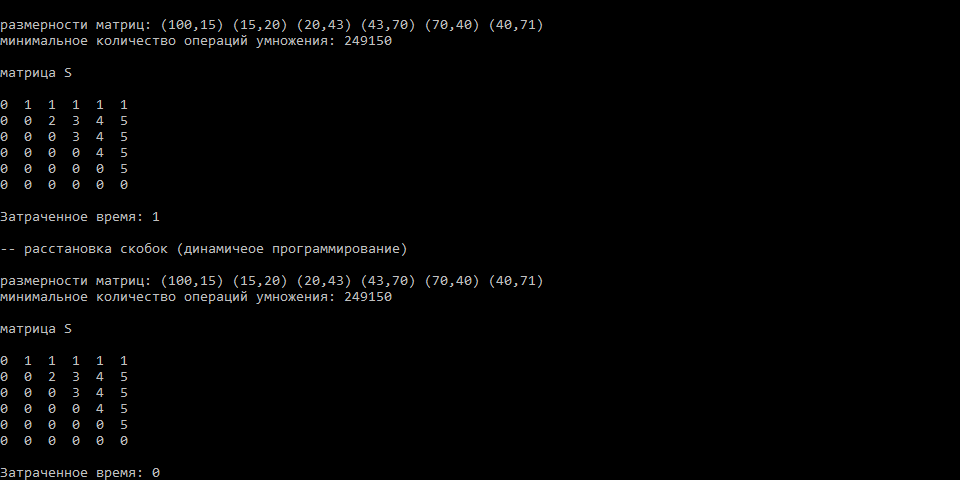
**Четные варианты**. Выполнить сравнительный анализ времени затраченного на решение задачи об оптимальной расстановке скобок при умножении нескольких матриц для двух методов решения (рекурсивное решение, динамическое программирование). Размерность матриц взять в соответствии с вариантом. Объяснить в отчете принцип расставления скобок по итоговой матрице + код + копии экрана.









Таким образом, для решения данной задачи с большим количеством матриц предпочтительнее использовать метод динамического программирования, хотя использование рекурсивного метода и является допустимым в некоторых случаях.

Объяснение принципа расстановки скобок.

(Индексы в матрице начинаются с 1)

Исходные матрицы: (100,15); (15,20); (20,43); (43,70); (70,40); (40,71).

Скобки расставляются по принципу «сначала внешние – затем внутренние». Элемент матрицы, который имеет индекс (1, 5) – это число 4, оно означает, что точка разрыва скобок находится между 1 и 5 матрицей после 4 матрицы, что позволяет расставить скобки следующим образом:

(A1 x A2 x A3 x A4) x A5 x A6

Точка разрыва между 2 и 6 матрицей находится в элементе матрицы под индексом (2, 6) – это число 4, что означает, что точка разрыва скобок находится между 2 и 6 матрицей после 4 матрицы, что позволяет расставить скобки следующим образом:

((((A1 x A2 ) x A3 )x A4))) x A5 ) x A6

(((((100,15) × (15,20)) × (20,43)) × (43,70)) × (70,40)) × (40,71)

**Вывод**: в ходе работы были освоены общие принципы решения задач методом динамического программирования, выполнено сравнение полученных решений задач с рекурсивным методом. Рекурсивный метод при малом количестве данных высчитывает результат довольно быстро, однако при большом количестве резко возрастает время ожидания. Однако рекурсивный метод более прост в понимании и реализации. Метод динамического программирования заключается в разбиении задач на подзадачи и работает всегда очень быстро и гораздо быстрее рекурсивного метода. Однако этот метод более сложен в понимании и реализации.

# Лабораторная работа 5. ТРАНСПОРТНАЯ ЗАДАЧА

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

**Задание для выполнения:**

**Задание.** Решить транспортную задачу. Имеется 5 поставщиков продукции и 6 потребителей. Величина запасов, потребностей и стоимость затрат на перевозку продукции взять в соответствии с вариантом (*N*). Оформить отчет.

Условие:

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| ПОТРЕБИТЕЛИ  ПОСТАВЩИКИ | 1 | 2 | 3 | 4 | 5 | 6 | ЗАПАСЫ |
| 1 | **22** | **12** | **16** | **13** | **21** | **11** | **178** |
| 2 | **20** | **10** | **18** | **15** | **18** | **23** | **123** |
| 3 | **11** | **15** | **21** | **18** | **12** | **21** | **160** |
| 4 | **14** | **20** | **21** | **13** | **24** | **13** | **169** |
| 5 | **13** | **21** | **19** | **10** | **20** | **14** | **110** |
| ПОТРЕБНОСТИ | **153** | **117** | **141** | **203** | **105** | **173** |  |

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

∑запасы = 178 + 123 + 160 + 169 + 110 = 740

∑потребности = 153 + 117 + 141 + 203 + 105 + 173 = 892

Так как запасы поставщиков меньше потребности потребителей, введем фиктивного поставщика 6, с запасом продукции равным 892-740=152. Стоимость доставки единицы продукции от фиктивного поставщика ко всем потребителям примем равной нулю.

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| Поставщик | Потребитель | | | | | | Запас |
| 1 | 2 | 3 | 4 | 5 | 6 |
| 1 | **22** | **12** | **16** | **13** | **21** | **11** | **178** |
| 2 | **20** | **10** | **18** | **15** | **18** | **23** | **123** |
| 3 | **11** | **15** | **21** | **18** | **12** | **21** | **160** |
| 4 | **14** | **20** | **21** | **13** | **24** | **13** | **169** |
| 5 | **13** | **21** | **19** | **10** | **20** | **14** | **110** |
| 6 | 0 | 0 | 0 | 0 | 0 | 0 | 152 |
| Потребность | **153** | **117** | **141** | **203** | **105** | **173** |  |

Теперь выполняется условие.

Этап I

*Метод наименьшей стоимости*

Суть метода заключается в том, что из всей таблицы стоимостей выбирают клетку с наименьшей стоимостью, для этой ячейки присваиваем 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 строку и столбец, если израсходованы запасы поставщика и удовлетворены потребности потребителя. Повторяем, пока все запасы не будут распределены, а потребности удовлетворены.

Искомый элемент равен c22=10

x22 = min(123,117) = 117.

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| Поставщик | Потребитель | | | | | | Запас |
| 1 | 2 | 3 | 4 | 5 | 6 |
| 1 | **22** | **12** | **16** | **13** | **21** | **11** | **178** |
| 2 | **20** | **10** | **18** | **15** | **18** | **23** | **123 – 117 = 6** |
| 3 | **11** | **15** | **21** | **18** | **12** | **21** | **160** |
| 4 | **14** | **20** | **21** | **13** | **24** | **13** | **169** |
| 5 | **13** | **21** | **19** | **10** | **20** | **14** | **110** |
| 6 | 0 | 0 | 0 | 0 | 0 | 0 | 152 |
| Потребность | **153** | **117-117 = 0** | **141** | **203** | **105** | **173** |  |

Искомый элемент равен c54=10.

x54 = min(110,203) = 110.

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| Поставщик | Потребитель | | | | | | Запас |
| 1 | 2 | 3 | 4 | 5 | 6 |
| 1 | **22** | **12** | **16** | **13** | **21** | **11** | **178** |
| 2 | **20** | **10** | **18** | **15** | **18** | **23** | **6** |
| 3 | **11** | **15** | **21** | **18** | **12** | **21** | **160** |
| 4 | **14** | **20** | **21** | **13** | **24** | **13** | **169** |
| 5 | **13** | **21** | **19** | **10** | **20** | **14** | **110 – 110 = 0** |
| 6 | 0 | 0 | 0 | 0 | 0 | 0 | 152 |
| Потребность | **153** | **0** | **141** | **203-110 = 93** | **105** | **173** |  |

Искомый элемент равен c16=11

x16 = min(178,173) = 173.

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| Поставщик | Потребитель | | | | | | Запас |
| 1 | 2 | 3 | 4 | 5 | 6 |
| 1 | **22** | **12** | **16** | **13** | **21** | **11** | **178-173= 5** |
| 2 | **20** | **10** | **18** | **15** | **18** | **23** | **6** |
| 3 | **11** | **15** | **21** | **18** | **12** | **21** | **160** |
| 4 | **14** | **20** | **21** | **13** | **24** | **13** | **169** |
| 5 | **13** | **21** | **19** | **10** | **20** | **14** | **0** |
| 6 | 0 | 0 | 0 | 0 | 0 | 0 | 152 |
| Потребность | **153** | **0** | **141** | **93** | **105** | **173 – 173 = 0** |  |

Искомый элемент равен c31=11.

x31 = min(160,153) = 153.

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| Поставщик | Потребитель | | | | | | Запас |
| 1 | 2 | 3 | 4 | 5 | 6 |
| 1 | **22** | **12** | **16** | **13** | **21** | **11** | **5** |
| 2 | **20** | **10** | **18** | **15** | **18** | **23** | **6** |
| 3 | **11** | **15** | **21** | **18** | **12** | **21** | **160 – 153 = 7** |
| 4 | **14** | **20** | **21** | **13** | **24** | **13** | **169** |
| 5 | **13** | **21** | **19** | **10** | **20** | **14** | **0** |
| 6 | 0 | 0 | 0 | 0 | 0 | 0 | 152 |
| Потребность | **153 – 153 = 0** | **0** | **141** | **93** | **105** | **0** |  |

Искомый элемент равен c35=12.

x35 = min(7,105) = 7.

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| Поставщик | Потребитель | | | | | | Запас |
| 1 | 2 | 3 | 4 | 5 | 6 |
| 1 | **22** | **12** | **16** | **13** | **21** | **11** | **5** |
| 2 | **20** | **10** | **18** | **15** | **18** | **23** | **6** |
| 3 | **11** | **15** | **21** | **18** | **12** | **21** | **7-7=0** |
| 4 | **14** | **20** | **21** | **14** | **24** | **13** | **169** |
| 5 | **13** | **21** | **19** | **10** | **20** | **14** | **0** |
| 6 | 0 | 0 | 0 | 0 | 0 | 0 | 152 |
| Потребность | **153 – 153 = 0** | **0** | **141** | **93** | **105-7=98** | **0** |  |

Искомый элемент равен c14=13.

x14 = min(5,93) = 5.

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| Поставщик | Потребитель | | | | | | Запас |
| 1 | 2 | 3 | 4 | 5 | 6 |
| 1 | **22** | **12** | **16** | **13** | **21** | **11** | **5-5=0** |
| 2 | **20** | **10** | **18** | **15** | **18** | **23** | **6** |
| 3 | **11** | **15** | **21** | **18** | **12** | **21** | **0** |
| 4 | **14** | **20** | **20** | **13** | **23** | **13** | **169** |
| 5 | **13** | **21** | **19** | **10** | **20** | **14** | **0** |
| 6 | 0 | 0 | 0 | 0 | 0 | 0 | 152 |
| Потребность | **153 – 153 = 0** | **0** | **141** | **93-5=88** | **98** | **0** |  |

Искомый элемент равен c44=13.

x44 = min(169,88) = 88.

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| Поставщик | Потребитель | | | | | | Запас |
| 1 | 2 | 3 | 4 | 5 | 6 |
| 1 | **22** | **12** | **16** | **13** | **21** | **11** | **0** |
| 2 | **20** | **10** | **18** | **15** | **18** | **23** | **6** |
| 3 | **11** | **15** | **21** | **18** | **12** | **21** | **0** |
| 4 | **14** | **20** | **20** | **13** | **23** | **13** | **169-88=81** |
| 5 | **13** | **21** | **19** | **10** | **20** | **14** | **0** |
| 6 | 0 | 0 | 0 | 0 | 0 | 0 | 152 |
| Потребность | **153 – 153 = 0** | **0** | **141** | **88-88=0** | **98** | **0** |  |

Искомый элемент равен c25=18.

x25 = min(6,98) = 6.

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| Поставщик | Потребитель | | | | | | Запас |
| 1 | 2 | 3 | 4 | 5 | 6 |
| 1 | **22** | **12** | **16** | **13** | **21** | **11** | **0** |
| 2 | **20** | **10** | **18** | **15** | **18** | **23** | **6-6=0** |
| 3 | **11** | **15** | **21** | **18** | **12** | **21** | **0** |
| 4 | **14** | **20** | **20** | **13** | **23** | **12** | **81** |
| 5 | **13** | **21** | **19** | **10** | **20** | **14** | **0** |
| 6 | 0 | 0 | 0 | 0 | 0 | 0 | 152 |
| Потребность | **0** | **0** | **141** | **0** | **98-6=92** | **0** |  |

Искомый элемент равен c43=20.

x43 = min(81,141) = 81.

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| Поставщик | Потребитель | | | | | | Запас |
| 1 | 2 | 3 | 4 | 5 | 6 |
| 1 | **22** | **12** | **16** | **13** | **21** | **11** | **0** |
| 2 | **20** | **10** | **18** | **15** | **18** | **23** | **0** |
| 3 | **11** | **15** | **21** | **18** | **12** | **21** | **0** |
| 4 | **14** | **20** | **20** | **13** | **23** | **12** | **81-81=0** |
| 5 | **13** | **21** | **19** | **10** | **20** | **14** | **0** |
| 6 | 0 | 0 | 0 | 0 | 0 | 0 | 152 |
| Потребность | **0** | **0** | **141-81=60** | **0** | **92** | **0** |  |

Искомый элемент равен c63=0.

x63 = min(152,60) = 60.

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| Поставщик | Потребитель | | | | | | Запас |
| 1 | 2 | 3 | 4 | 5 | 6 |
| 1 | **22** | **12** | **16** | **13** | **21** | **11** | **0** |
| 2 | **20** | **10** | **18** | **15** | **18** | **23** | **0** |
| 3 | **11** | **15** | **21** | **18** | **12** | **21** | **0** |
| 4 | **14** | **20** | **20** | **13** | **23** | **12** | **0** |
| 5 | **13** | **21** | **19** | **10** | **20** | **14** | **0** |
| 6 | 0 | 0 | 0 | 0 | 0 | 0 | 152-60=  92 |
| Потребность | **0** | **0** | **60-60=0** | **0** | **92** | **0** |  |

Искомый элемент равен c65=0.

x65 = min(92,92) = 92.

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| Поставщик | Потребитель | | | | | | Запас |
| 1 | 2 | 3 | 4 | 5 | 6 |
| 1 | **22** | **12** | **16** | **13** | **21** | **11** | **0** |
| 2 | **20** | **10** | **18** | **15** | **18** | **23** | **0** |
| 3 | **11** | **15** | **21** | **18** | **12** | **21** | **0** |
| 4 | **14** | **20** | **20** | **13** | **23** | **12** | **0** |
| 5 | **13** | **21** | **19** | **10** | **20** | **14** | **0** |
| 6 | 0 | 0 | 0 | 0 | 0 | 0 | 92-92=0 |
| Потребность | **0** | **0** | **0** | **0** | **92-92=0** | **0** |  |

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| Поставщик | Потребитель | | | | | | Запас |
| 1 | 2 | 3 | 4 | 5 | 6 |
| 1 | **22** | **12** | **16** | **13 | 5** | **21** | **11| 173** | **0** |
| 2 | **20** | **10 | 117** | **18** | **15** | **18 | 6** | **23** | **0** |
| 3 | **11 | 153** | **15** | **21** | **18** | **12 | 7** | **21** | **0** |
| 4 | **14** | **20** | **20 | 81** | **13 | 88** | **23** | **12** | **0** |
| 5 | **13** | **21** | **19** | **10 | 110** | **20** | **14** | **0** |
| 6 | **0** | **0** | **0 | 60** | **0** | **0 | 92** | **0** | **0** |
| Потребность | **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.

Подсчитаем число базисных переменных. Их должно быть m + n - 1 = 11

Значение целевой функции для этого опорного плана равно:

F(x) = 11 \* 153 + 10 \* 117 + 20 \* 81 + 0 \* 60 + 13 \* 5 + 13 \* 88 + 10 \* 110 + 18 \* 6 + 12 \* 7 + 0 \* 92 + 11 \* 173 = 8877

Этап II

*Метод потенциалов*

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

ui + vj = cij

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

u1 + v4 = 13; 0 + v4 = 13; v4 = 13

u1 + v6 = 11; 0 + v6 = 11; v6 = 11

u4 + v4 = 13; 13 + u4 = 13; u4 = 0

u4 + v3 = 20; 0 + v3 = 20; v3 = 20

u6 + v3 = 0; 20 + u6 = 0; u6 = -20

u6 + v5 = 0; -20 + v5 = 0; v5 = 20

u2 + v5 = 18; 20 + u2 = 18; u2 = -2

u2 + v2 = 10; -2 + v2 = 10; v2 = 12

u5 + v4 = 10; 13 + u5 = 10; u5 = -3

u3 + v5 = 12; 20 + u3 = 12; u3 = -8

u3 + v1 = 11; -8 + v1 = 11; v1 = 19

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| Поставщик | Потребитель | | | | | | U |
| b 1 | b 2 | b3 | b 4 | b 5 | b 6 |
| a 1 | **22** | **12** | **16** | **13 | 5** | **21** | **11| 173** | u1 = 0 |
| a 2 | **20** | **10 | 117** | **18** | **15** | **18 | 6** | **23** | u2 = -2 |
| a 3 | **11 | 153** | **15** | **21** | **18** | **12 | 7** | **21** | u3 = -8 |
| a 4 | **14** | **20** | **20 | 81** | **13 | 88** | **23** | **12** | u4 = 0 |
| a 5 | **13** | **21** | **19** | **10 | 110** | **20** | **14** | u5 =  -3 |
| a 6 | **0** | **0** | **0 | 60** | **0** | **0 | 92** | **0** | u6 = -20 |
| V | v1 = 19 | v2 = 12 | v3 = 20 | v4 = 13 | v5 = 20 | v6 = 11 |  |

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

(1;3): 16 = 0 + 20 – 16 = 4 > 0

(4;1): 14 = 0 + 19 – 14 =5 > 0

(5;1): 13 = -3 + 19 -13 = 3 > 0

max(4,5,3) = 5

Выбираем максимальную оценку свободной клетки (4;1): 14. Для этого в перспективную клетку (4;1) поставим знак «+», а в остальных вершинах многоугольника чередующиеся знаки «-», «+», «-».

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| Поставщик | Потребитель | | | | | | Запас |
| 1 | 2 | 3 | 4 | 5 | 6 |
| 1 | **22** | **12** | **16** | **13 | 5** | **21** | **11| 173** | 0 |
| 2 | **20** | **10 | 117** | **18** | **15** | **18 | 6** | **23** | 0 |
| 3 | **11 | 153[-]** | **15** | **21** | **18** | **12 | 7[+]** | **21** | 0 |
| 4 | **14[+]** | **20** | **20 | 81[-]** | **13 | 88** | **23** | **12** | 0 |
| 5 | **13** | **21** | **19** | **10 | 110** | **20** | **14** | 0 |
| 6 | **0** | **0** | **0 | 60[+]** | **0** | **0 | 92[-]** | **0** | 0 |
| Потребность | 0 | 0 | 0 | 0 | 0 | 0 |  |

Из грузов хij стоящих в минусовых клетках, выбираем наименьшее, т.е. прибавляем 81 к объемам грузов, стоящих в плюсовых и вычитаем 81 из xij, стоящих в минусовых клетках. В результате получим новый опорный план.

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| Поставщик | Потребитель | | | | | | Запас |
| 1 | 2 | 3 | 4 | 5 | 6 |
| 1 | **22** | **12** | **16** | **13 | 5** | **21** | **11| 173** | 0 |
| 2 | **20** | **10 | 117** | **18** | **15** | **18 | 6** | **23** | 0 |
| 3 | **11 |**  **72[-]** | **15** | **21** | **18** | **12 | 88[+]** | **21** | 0 |
| 4 | **14| 81[+]** | **20** | **20 |**  **0[-]** | **13 | 88** | **23** | **12** | 0 |
| 5 | **13** | **21** | **19** | **10 | 110** | **20** | **14** | 0 |
| 6 | **0** | **0** | **0 | 141[+]** | **0** | **0 | 11[-]** | **0** | 0 |
| Потребность | 0 | 0 | 0 | 0 | 0 | 0 |  |

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

u1 + v4 = 13; 0 + v4 = 13; v4 = 13

u1 + v6 = 11; 0 + v6 = 11; v6 = 11

u4 + v4 = 13; 13 + u4 = 13; u4 = 0

u5 + v4 = 10; 13 + u5 = 10; u5 = -2

u4 + v1 = 14; 0 + v1 = 14; v1 = 14

u3 + v1 = 11; 14 + u3 = 11; u3 = -3

u3 + v5 = 12; -3 + v5 = 12; v5 = 15

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

u2 + v2 = 10; 3 + v2 = 10; v2 = 7

u6 + v5 = 0; 15 + u6 = 0; u6 = -15

u6 + v3 = 0; -15 + v3 = 0; v3 = 15

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| Поставщик | Потребитель | | | | | | U |
| b 1 | b 2 | b3 | b 4 | b 5 | b 6 |
| a 1 | **22** | **12** | **16** | **13 | 5** | **21** | **11| 173** | u1 = 0 |
| a 2 | **20** | **10 | 117** | **18** | **15** | **18 | 6** | **23** | u2 = 3 |
| a 3 | **11 |**  **72[-]** | **15** | **21** | **18** | **12 | 88[+]** | **21** | u3 = -3 |
| a 4 | **14| 81[+]** | **20** | **20 |**  **0[-]** | **13 | 88** | **23** | **12** | u4 = 0 |
| a 5 | **13** | **21** | **19** | **10 | 110** | **20** | **14** | u5 = -2 |
| a 6 | **0** | **0** | **0 | 141[+]** | **0** | **0 | 11[-]** | **0** | u6 = -15 |
| V | v1 = 14 | v2 = 7 | v3 = 15 | v4 = 13 | v5 = 15 | v6 = 11 |  |

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

Минимальные затраты составят: F(x) = 13 \* 5 + 11 \* 173 + 10 \* 117 + 18 \* 6 + 11 \*72 +12 \* 88 + 14 \* 81 + 20 \* 0 + 13 \* 88 + 10 \*110 + 0 \* 141 +0 \* 11

= 9336

*Анализ оптимального плана.*

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| Поставщик | Потребитель | | | | | | Запас |
| 1 | 2 | 3 | 4 | 5 | 6 |
| 1 | **22** | **12** | **16** | **13 | 5** | **21** | **11| 173** | 0 |
| 2 | **20** | **10 | 117** | **18** | **15** | **18 | 6** | **23** | 0 |
| 3 | **11 | 153** | **15** | **21** | **18** | **12 | 7** | **21** | 0 |
| 4 | **14** | **20** | **20 | 81** | **13 | 88** | **23** | **12** | 0 |
| 5 | **13** | **21** | **19** | **10 | 110** | **20** | **14** | 0 |
| 6 | **0** | **0** | **0 | 60** | **0** | **0 | 92** | **0** | 0 |
| Потребность | 0 | 0 | 0 | 0 | 0 | 0 |  |

Из 1-го склада необходимо доставить 5 ед. товара 4-му потребителю и 173 ед. товара 6-му потребителю.

Из 2-го склада необходимо доставить 117 ед. товара 2-му потребителю и 6 ед. товара 5-му потребителю.

Из 3-го склада необходимо доставить 153 ед. товара 1-му потребителю и 7 ед. товара 5-му потребителю.

Из 4-го склада необходимо доставить 81 ед. товара 3-му потребителю и 88 ед. товара 4-му потребителю.

Из 5-го склада необходимо доставить 110 ед. товара 4-му потребителю.

Потребность 3-го потребителя остается неудовлетворенной на 60 ед.

Потребность 5-го потребителя остается неудовлетворенной на 92 ед.

*Небазисные переменные*

X11 = u1 + v1 – c11 = 0 + 14 – 22 = -8

X12 = u1 + v2 – c12 = 0 + 7 – 12 = -5

X13 = u1 + v3 – c13 = 0 + 15 – 16 = -1

X15 = u1 + v5 – c15 = 0 + 15 - 21 = -6

X21 = u2 + v1 – c21 = 3 + 14 – 20 = -3

X23 = u2 + v3 – c23 = 3 + 15 – 18 = 0

X24 = u2 + v4 – c24 = 3 + 13 – 15 = 1

X26 = u2 + v6 – c26 = 3 + 11 – 23 = 9

X32 = u3 + v2 – c32 = -3 + 7 – 15 = -11

X33 = u3 + v3 – c33 = -3 + 15 – 21 = -9

X34 = u3 + v4 – c34 = -3 + 13 – 18 = -8

X36 = u3 + v6 – c36 = -3 + 11 - 21 = -13

X42 = u4 + v2 – c42 = 0 + 7 – 20 = -13

X45 = u4 + v5 – c45 = 0 + 15 – 23 = -8

X46 = u4 + v6 – c46 = 0 + 11 – 12 = -1

X51 = u5 + v1 – c51 = -2 + 14 – 13 = -1

X52 = u5 + v2 – c52 = -2 + 7 – 21 = -16

X53 = u5 + v3 – c53 = -2 + 15 – 19 = -6

X55 = u5 + v5 – c55 = -2 + 15 – 20 = -7

X56 = u5 + v6 – c56 = -2 + 11 – 14 = -6

X61= u6 + v1 – c61 = -15 + 14 – 0 = -1

X62 = u6 + v2 – c62 = -15 + 7 – 0 = -8

X64 = u6 + v4 – c64 = -15 + 13 - 0 = -2

X66 = u6 + v6 – c66 = -15 + 11 – 0 = -4

**Вывод:**

Приобрели навыки решения открытой транспортной задачи, методом наименьшей стоимости и потенциалов, проанализировали оптимальный план и нашли небазисные переменные.

# Лабораторная работа 6. АЛГОРИТМЫ НА ГРАФАХ

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

**ЗАДАНИЕ ДЛЯ ВЫПОЛНЕНИЯ:**

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

**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 Краскала.

**Задание №1**

**Условие:** ориентированный граф **G** взять в соответствии с вариантом. Представить его в отчете в виде матрицы смежности, матрицы инцидентности, списка смежных вершин.



**Выполнение:**

Матрица смежности:

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

Матрица инцидентности:

|  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
|  | **(0; 1)** | **(0; 3)** | **(1; 4)** | **(3; 1)** | **(3; 2)** | **(3; 5)** | **(4; 6)** | **(5; 2)** | **(6; 5)** |
| **0** | 1 | 1 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| **1** | -1 | 0 | 1 | -1 | 0 | 0 | 0 | 0 | 0 |
| **2** | 0 | 0 | 0 | 0 | -1 | 0 | 0 | -1 | 0 |
| **3** | 0 | -1 | 0 | 1 | 1 | 1 | 0 | 0 | 0 |
| **4** | 0 | 0 | -1 | 0 | 0 | 0 | 1 | 0 | 0 |
| **5** | 0 | 0 | 0 | 0 | 0 | -1 | 0 | 1 | -1 |
| **6** | 0 | 0 | 0 | 0 | 0 | 0 | -1 | 0 | 1 |

Список смежных вершин:

S0 = {1, 3}; S1 = {4}; S2 = {∅}; S3 = {2, 5, 1}; S4 = {6}; S5 = {2}; S6 = {5}.

**Задание №2**

**Условие:** 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.



**Выполнение:**

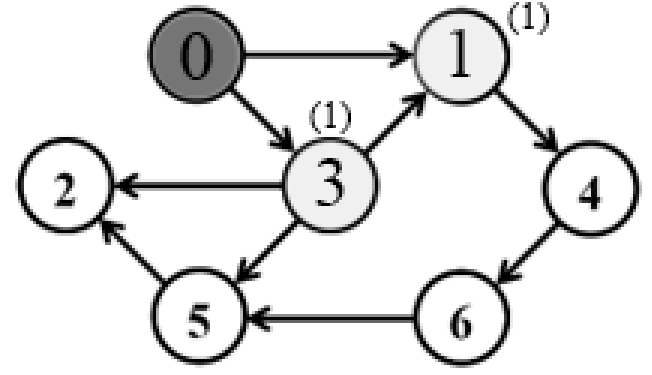
**Поиск в ширину**



1. Выбираем стартовую вершину 0. Добавляем ее в очередь.

**Очередь: [0]**

**Посещенная очередь: []**



1. Извлекаем из очереди вершину 0 и заносим в очередь посещения. У вершины 0 имеются смежные вершины – 1 и 3. Добавляем их в очередь.

**Очередь: [1, 3]**

**Посещенная очередь: [0]**

1. Извлекаем из очереди первую вершину (1) и добавляем ее в очередь посещения. У вершины 1 имеется смежная вершина 4, добавляем ее в очередь.

**Очередь: [3, 4]**

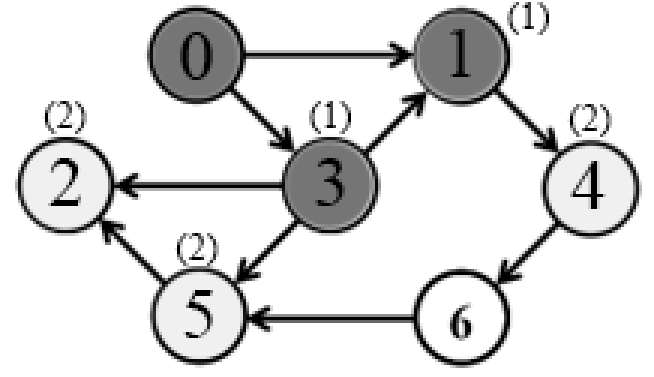
**Посещенная очередь: [0, 1]**

1. Извлекаем из очереди первую вершину (3) и добавляем ее в очередь посещения. У вершины 3 имеются смежные вершины 2 и 5, заносим их в очередь.

**Очередь: [4, 2, 5]**

**Посещенная очередь: [0, 1, 3]**

1. Извлекаем из очереди первую вершину (4) и добавляем ее в очередь посещения. У вершины 4 смежная вершина 6, добавляем ее в очередь.



**Очередь: [2, 5, 6]**

**Посещенная очередь: [0, 1, 3, 4]**

1. Извлекаем из очереди первую вершину (2) и добавляем ее в очередь посещения. У вершины 2 нет смежных вершин, продолжаем алгоритм.

**Очередь: [5, 6]**

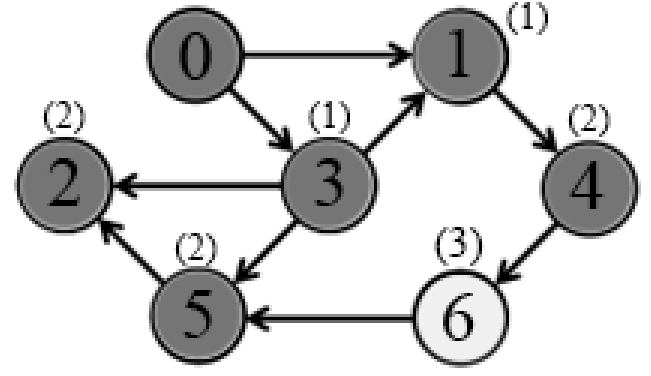
**Посещенная очередь: [0, 1, 3, 4, 2]**

1. Извлекаем из очереди первую вершину (5) и добавляем ее в очередь посещения. У вершины 5 имеется смежная вершина 2, но она уже в очереди, продолжаем алгоритм.

**Очередь: [6]**

**Посещенная очередь: [0, 1, 3, 4, 2, 5]**

1. Извлекаем из очереди первую вершину (6) и добавляем ее в очередь посещения. У вершины 5 имеются смежная вершина 2, однако она есть в очереди посещения, пропускаем ее.

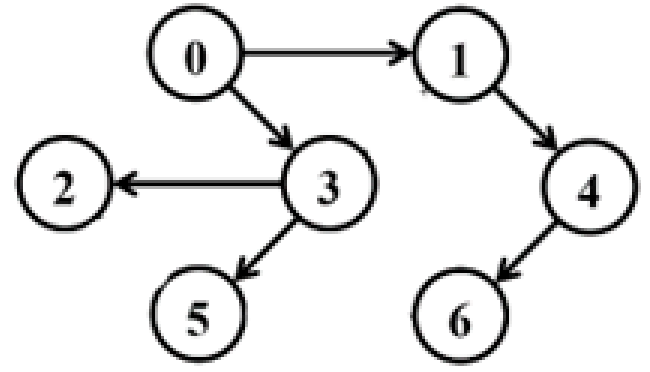


**Очередь: []**

**Посещенная очередь: [0, 1, 3, 4, 2, 5, 6]**

1. Все вершины пройдены. Конец алгоритма.

**Поиск в ширину: 0 -> 1 -> 3 -> 4 -> 2 -> 5 -> 6**



**Поиск в глубину**

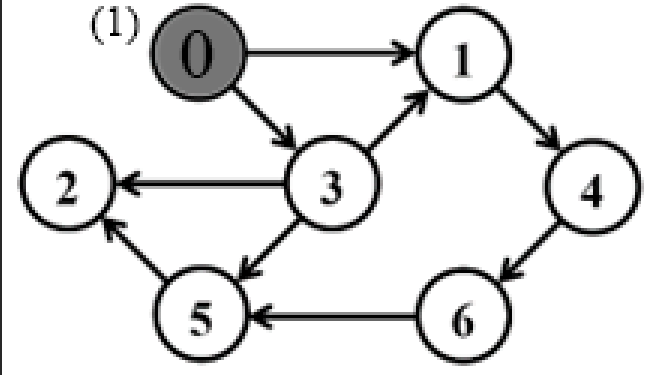


Создаем стек и помещаем в него начальную вершину 0. Помечаем вершину 0 как посещенную.

**Стек: [0]**

**Посещенные вершины: {0}**

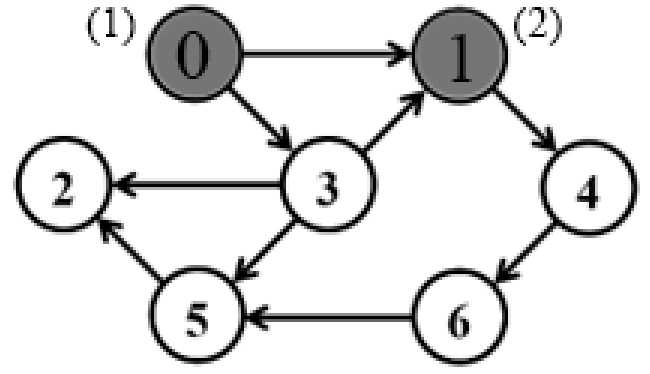
У вершины 0 имеется 2 смежных вершины - 1 и 3. Добавляем первую смежную вершину (1) в стек.



**Стек: [0, 1]**

**Посещенные вершины: {0, 1}**

У вершины 1 имеется смежная вершина 4, добавляем ее в стек, отмечаем 1 как посещенную.



**Стек: [0, 1, 4]**

**Посещенные вершины: {0, 1, 4}**

У вершины 4 смежная вершина 6, добавляем ее в стек, отмечаем 4 как посещенную.

**Стек: [0, 1, 4, 6]**

**Посещенные вершины: {0, 1, 4, 6}**

У вершины 6 смежная вершина 5, добавляем ее в стек, отмечаем 6 как посещенную.

**Стек: [0, 1, 4, 6, 5]**

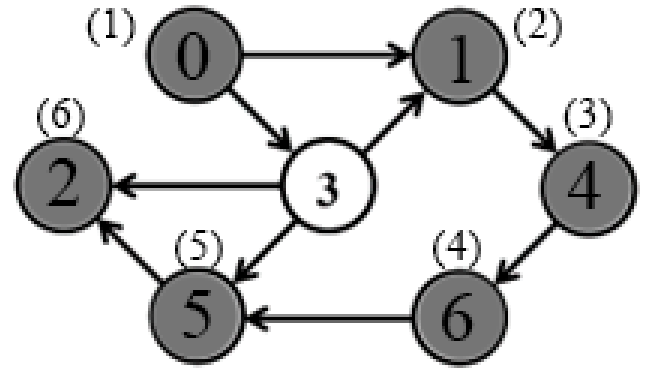
**Посещенные вершины: {0, 1, 4, 6, 5}**

У вершины 5 смежная вершина 2, добавляем ее в стек, отмечаем 6 как посещенную.

**Стек: [0, 1, 4, 6, 5, 2]**

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

У вершины 2 нет смежных вершин, отмечаем 2 как посещенную.



**Стек: [0, 1, 4, 6, 5]**

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

Убираем вершину 2 из стека.

**Стек: [0, 1, 4, 6]**

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

Убираем вершину 5 из стека.

**Стек: [0, 1, 4]**

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

Убираем вершину 6 из стека.

**Стек: [0, 1]**

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

Убираем вершину 4 из стека.

**Стек: [0]**

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

Убираем вершину 1 из стека.

**Стек: [0, 3]**

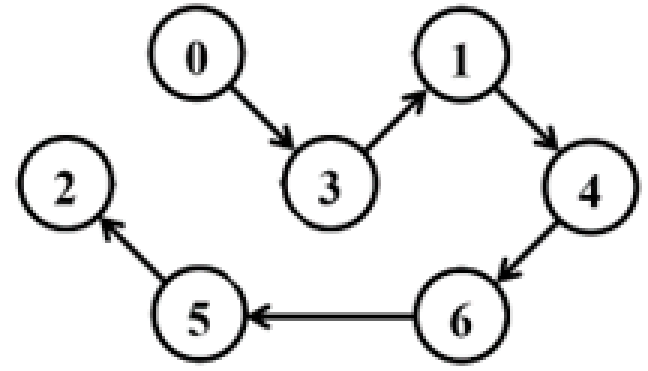
**Посещенные вершины: {0, 1, 4, 6, 5, 2, 3}**

1. У вершины 3 нет смежных вершин, отмечаем 3 как посещенную.

**Стек: [0]**

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

1. У вершины 0 нет не посещенных смежных вершин. Конец алгоритма.



**Поиск в глубину: 2 -> 5 -> 6 -> 4 -> 1 -> -> 3 -> 0**

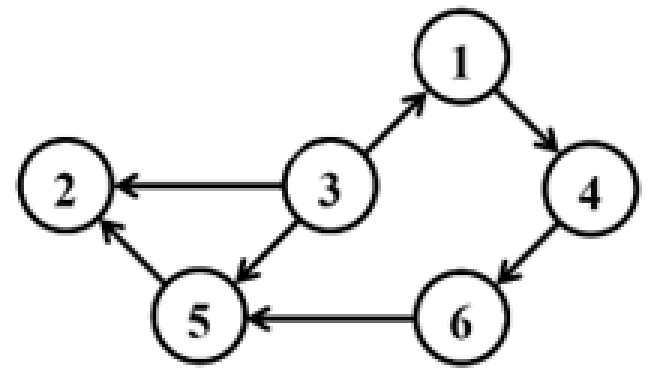
**Топологическая сортировка**

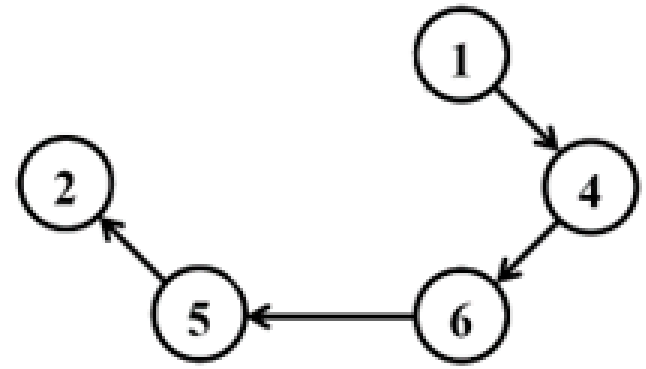


Топологическая сортировка будет выполнена с помощью метода Демукрона.

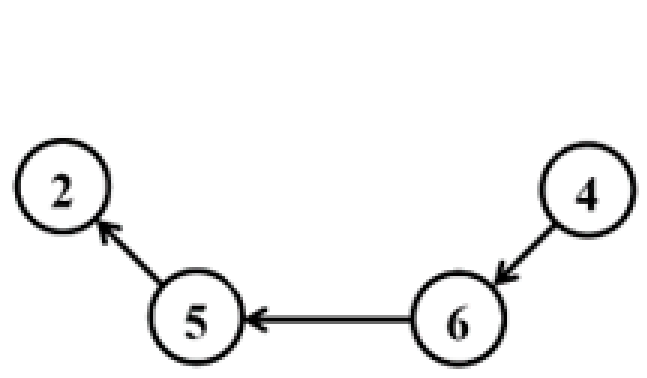
В исходном графе вершины, являющиеся истоками, будут вычеркиваться, а также дуги, связанные с ними. Каждому такому удалению будет соответствовать запись вычеркнутых вершин на новый уровень обновленного графа.

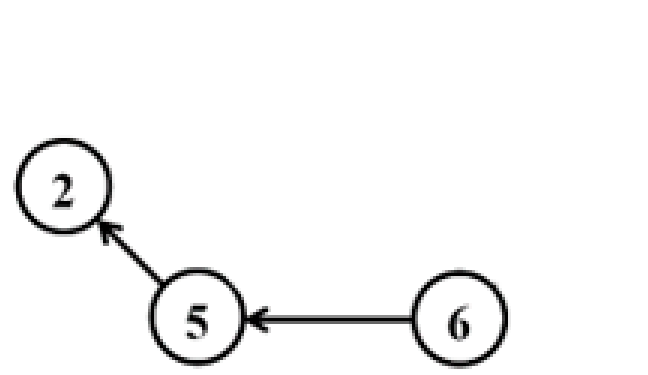
В исходном графе исток – нулевая вершина. Ее и вычеркиваем. Записываем эту вершину на нулевой уровень отсортированного графа.

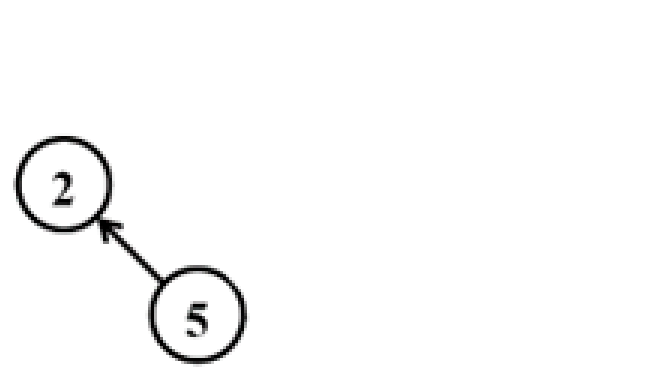


После этого убираем вершины без вхождений – [3]

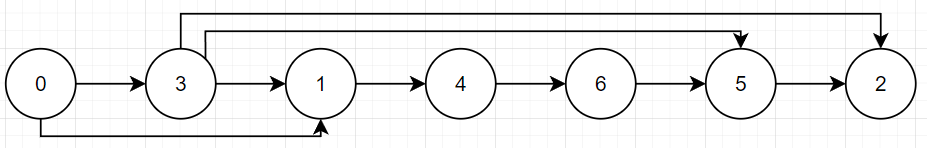
Проделываем все тоже самое до того момента пока не останется одна вершина

[2].







После исключения всех истоков получим отсортированный граф. В этом графе все вершины будут соединены точно так же, как и в исходном. 

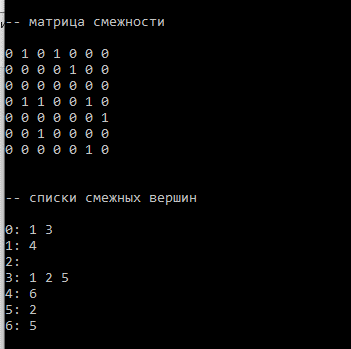
**Топологическая сортировка: 2 -> 5 -> 6 -> 4 -> 1 -> 3 -> 0**

**Задание №3**

**Условие:** 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.

**Выполнение:**





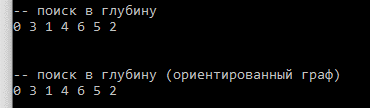


**Задание №4**

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

**Выполнение:**





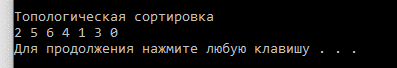
Конечный ответ отличается от моего, однако в данной программной реализации метод начинает действовать также как и я, в то время как я осуществлял поиск по принципу увеличения номера вершины и поэтому я пошел сначала в вершину 1.

**Задание №5**

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

**Выполнение:**





**Задание №6**

**Условие:** 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;

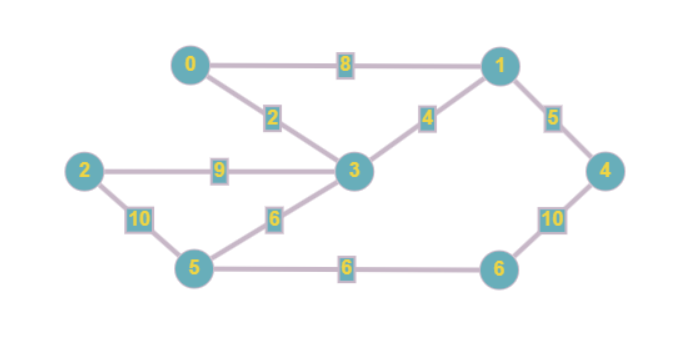
W(e4,3)=4; W(e3,4)=1;

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;

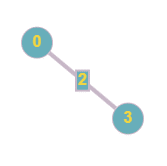


**Выполнение:**

1. Выбираем стартовую вершину, пусть она будет 0. Добавляем вершину 0 в дерево.

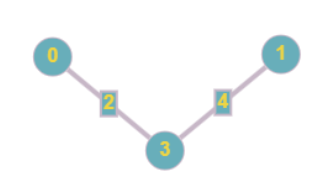
**Дерево: {0}**

1. Доступны два ребра: (0, 3) и (0, 1), выбираем ребро с наименьшим весом 2 (0, 3), добавляем вершину 3 в дерево.



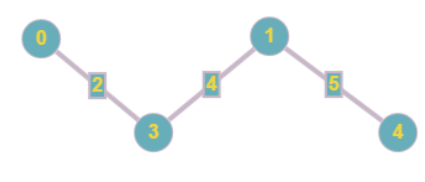
**Дерево: {0, 3}**

1. Из доступных ребер выбираем ребро с наименьшим весом 4 (3, 1), добавляем вершину 1 в дерево.



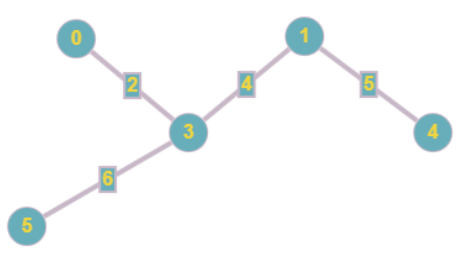
**Дерево: {0, 3, 1}**

1. Из доступных ребер выбираем ребро с наименьшим весом 5 (1, 4), добавляем вершину 4 в дерево.



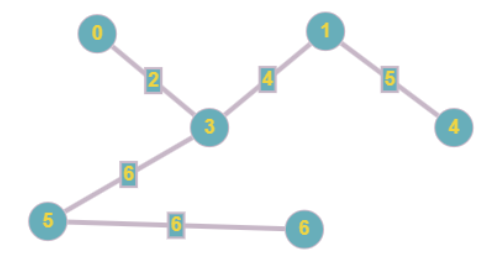
**Дерево: {0, 3, 1, 4}**

1. Из доступных ребер выбираем ребро с наименьшим весом 6 (3, 5), добавляем вершину 5 в дерево.



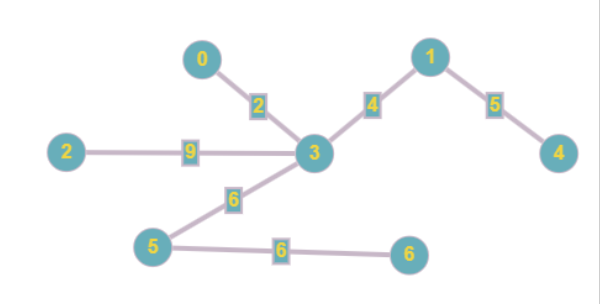
**Дерево: {0, 3, 1, 4, 5}**

1. Из доступных ребер выбираем ребро с наименьшим весом 6 (5, 6), добавляем вершину 6 в дерево.

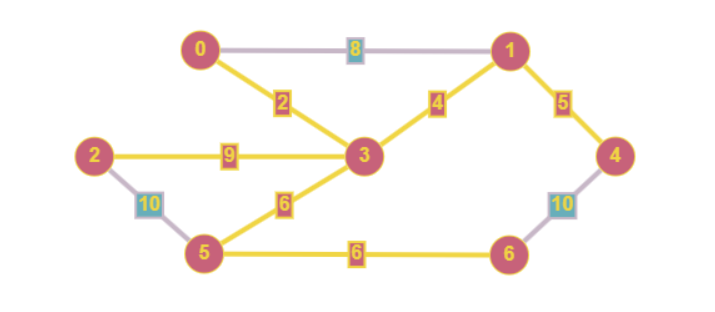


**Дерево: {0, 3, 1, 4, 5, 6}**

1. Из доступных ребер выбираем ребро с наименьшим весом 9 (3, 2), добавляем вершину 2 в дерево.



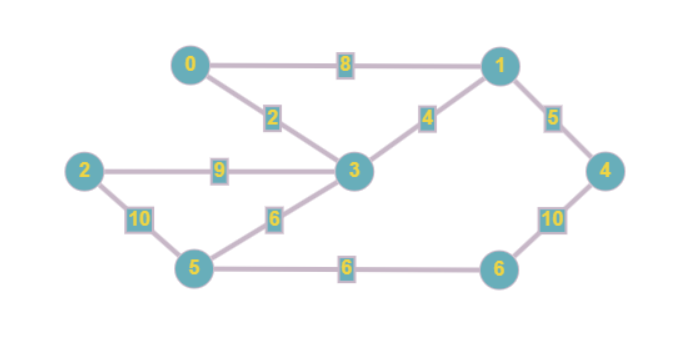
**Дерево: {0, 3, 1, 4, 5, 6, 2}**



Вес минимального остовного дерева равен 32.

**Задание №7**

**Условие:** По графу, соответствующему варианту составить минимальное остовное дерево по алгоритму Краскала. Шаги построения отразить в отчете.



**Выполнение:**

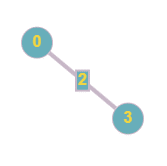
Отличие алгоритма Краскала от алгоритма Прима в том, что в алгоритме Прима мы проходимся по тому же графу и строим единое дерево, а также мы выбираем какую-то стартовую вершину и от нее ищем ребра с минимальными весами, связывая таким образом вершины. В алгоритме Краскала может получиться так(в большинстве случаев), что мы создаем несколько деревьев и объединяем их в одно по ходу алгоритма. Мы сортируем по неубыванию веса всех ребер и в зависимости от того, где располагается ребро со следующим минимальным весом, мы можем либо продолжать строить текущее дерево, либо строим новое (его корень лежит в вершине, к которой ведет ребро с текущим минимальным весом), а затем их соединяем.

|  |  |
| --- | --- |
| Неотсортированные веса | Отсортированные по неубыванию веса |
| 0 🡨🡪 1; W = 8 0 🡨🡪 3; W = 2 1 🡨🡪 3; W = 4 1 🡨🡪 4; W = 5 3 🡨🡪 2; W = 9 3 🡨🡪 5; W = 6 4 🡨🡪 6; W = 10 5 🡨🡪 2; W = 10 5 🡨🡪 6; W = 6 | 0 🡨🡪 3; W = 2 1 🡨🡪 3; W = 4 1 🡨🡪 4; W = 5 3 🡨🡪 5; W = 6 5 🡨🡪 6; W = 6 0 🡨🡪 1; W = 8 3 🡨🡪 2; W = 9 4 🡨🡪 6; W = 10 5 🡨🡪 2; W = 10 |

1. Выбираем вершину 0, так как к ней прилегает ребро с минимальным весом 2. Создаем дерево с корнем в вершине 0. Ребро ведет к вершине 3, записываем эту вершину как вершину первого дерева.

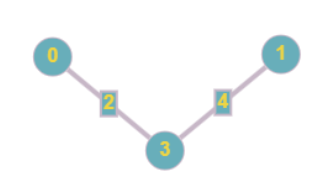
**Дерево 1: {0, 3}**

1. Выбираем следующее ребро с минимальным весом, это ребро с весом 4, которое идет между вершинами 1 и 3, в котором уже имеется дерево. Записываем эту вершину как вершину первого дерева.



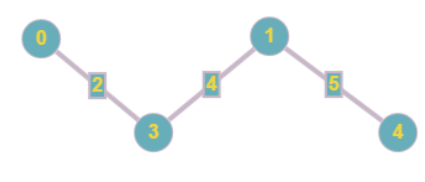
**Дерево 1: {0, 3, 1}**

1. Выбираем следующее ребро с минимальным весом, это ребро с весом 5, которое идет между вершинами 1 и 4, в котором уже имеется дерево. Записываем эту вершину как вершину первого дерева.



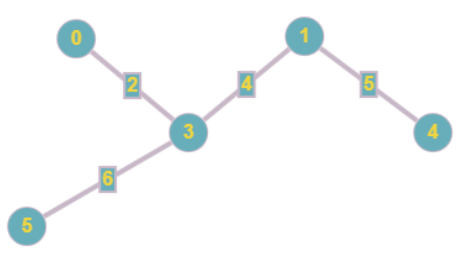
**Дерево 1: {0, 3, 1, 4}**

1. Выбираем следующее ребро с минимальным весом, это ребро с весом 6, которое идет между вершинами 3 и 5, вершина 3 уже привязана к первому дереву, поэтому добавляем к этому дереву вершину 5.



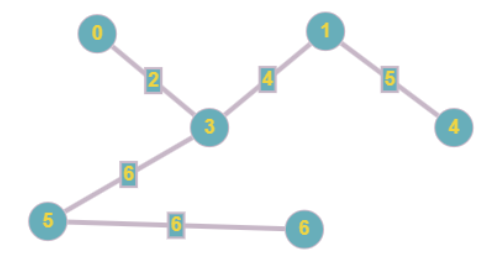
**Дерево 1: {0, 3, 1, 4, 5}**

1. Выбираем следующее ребро с минимальным весом, это ребро с весом 6, которое идет между вершинами 5 и 6, вершина 5 уже привязана к первому дереву, поэтому добавляем к этому дереву вершину 6.

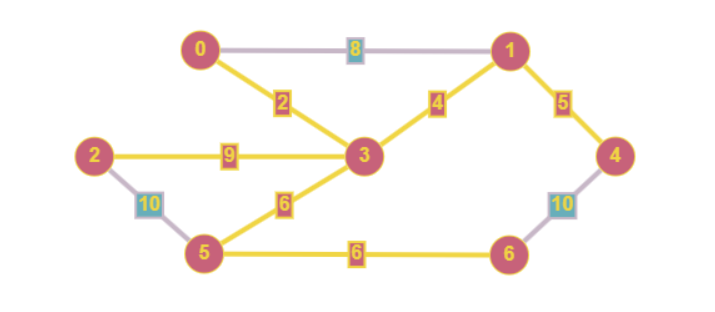


**Дерево 1: {0, 3, 1, 4, 5, 6}**

1. Пропускаем следующее ребро с минимальным весом, так как оно идет между вершинами 0 и 1, которые уже включены в первое дерево, и берем следующее ребро, которое имеет вес 9 и лежит между вершинами 3 и 2, вершина 3 уже находится в дереве, включаем вершину 2 в дерево.



**Дерево 1: {0, 3, 1, 4, 5, 6, 2}**



Вес минимального остовного дерева равен 32.

**Вывод:** Алгоритмы поиска в глубину и ширину применяются для обхода графов и поиска путей между вершинами. Они также могут использоваться для проверки наличия циклов в графах и для поиска компонент связности.

Алгоритм топологической сортировки применяется для упорядочивания вершин в ориентированном ациклическом графе таким образом, чтобы все дуги указывали от более ранней вершины к более поздней. Это может быть полезно для определения последовательности выполнения задач в проектах или для обнаружения зависимостей между задачами.

Алгоритм Прима и Краскала применяются для построения минимального остовного дерева в связном взвешенном графе. Они могут использоваться для оптимизации сетевых и транспортных систем, в телекоммуникационных сетях и в других областях, где требуется выбрать наиболее экономичный маршрут или связь между точками.

# 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 4, 10, 16 | «Создание облачного хранилища» | 60 дней |

**Задание №1 и №2. Тема «Создание облачного хранилища»**

|  |  |  |  |
| --- | --- | --- | --- |
| **Код операции** | **Наименование операции** | **Предшествующие операции** | **Дней** |
| 1. **АНАЛИЗ** | | | |
| Z1 | Изучение потребностей и требований пользователей |  | 3 |
| Z2 | Анализ функциональных и нефункциональных требований | Z1 | 2 |
| Z3 | Определение основных характеристик и возможностей системы хранения данных | Z2 | 5 |
| 1. **Проектирование архитектуры** | | | |
| Z4 | Выбор технологического стека и платформы для облачного хранилища | Z2, Z3 | 3 |
| Z5 | Проектирование системы хранения данных | Z2,Z3,Z4 | 7 |
| Z6 | Разработка архитектурных диаграмм и схемы взаимодействия компонентов | Z5 | 5 |
| 1. **Разработка и тестирование** | | | |
| Z7 | Создание объектов инфраструктуры | Z5,Z6 | 5 |
| Z8 | Реализация клиентской части облачного хранилища | Z7 | 10 |
| Z9 | Разработка серверной части | Z7,Z8 | 6 |
| Z10 | Написание тестов для проверки функциональности | Z7,Z8,Z9 | 6 |
| 1. **Развертывание и масштабирование** | | | |
| Z11 | Подготовка инфраструктуры для развертывания облачного хранилища. | Z7,Z8,Z9,Z10 | 2 |
| Z12 | Развертывание системы на выбранной платформе. | Z11 | 4 |
| Z13 | Настройка механизмов масштабирования и балансировки нагрузки | Z12 | 2 |

Таблица 1 – Структурное и календарное планирование

**Задание №3. Сетевой график, нахождение критического пути**

Результирующий граф представлен на рисунке 1.

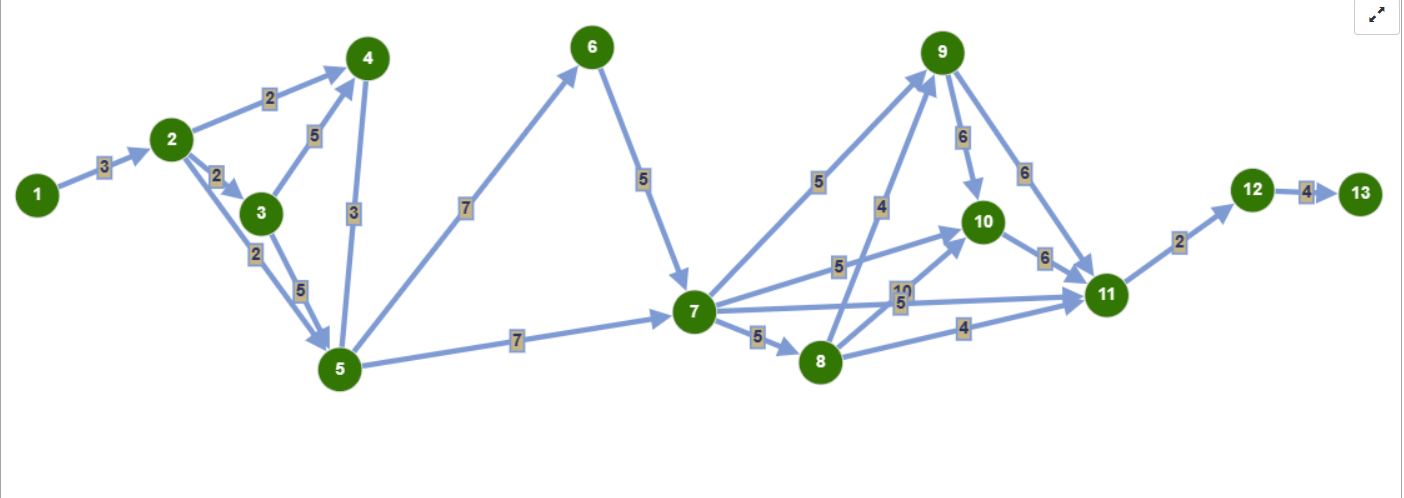


Рисунок 1 – Результирующий граф

Далее найдём критический путь для данного графа – наибольший путь при обходе. Критический путь: 1 – 2 – 3 – 4 – 5 – 6 – 7 – 8 – 10 – 11 – 12 –13. Сумма пути = 52 – Рисунок 2 и 3.

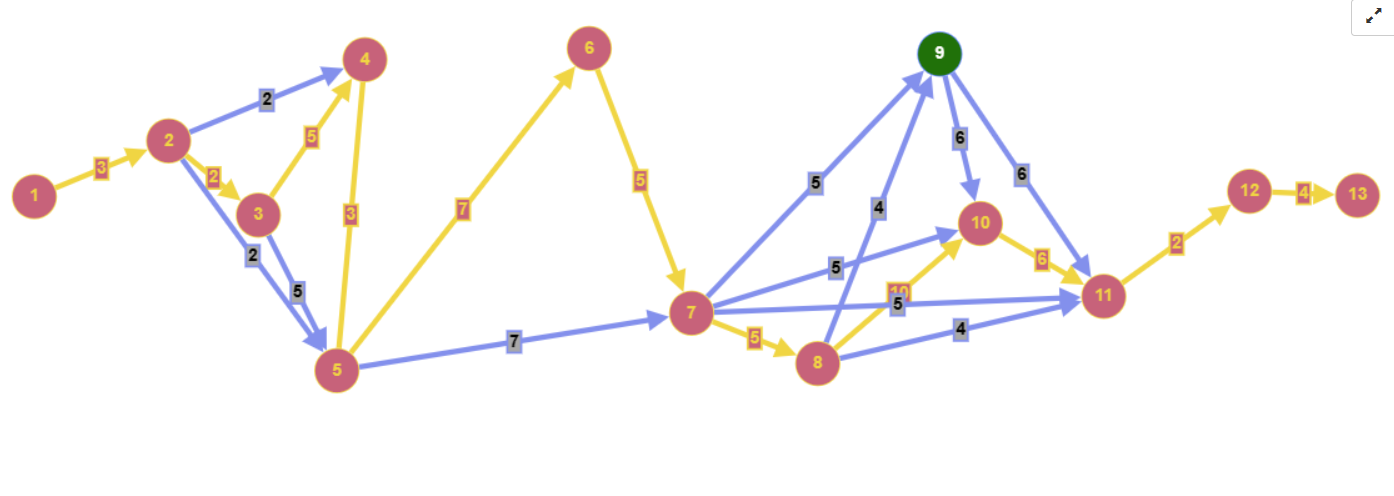


Рисунок 2 – Критический путь

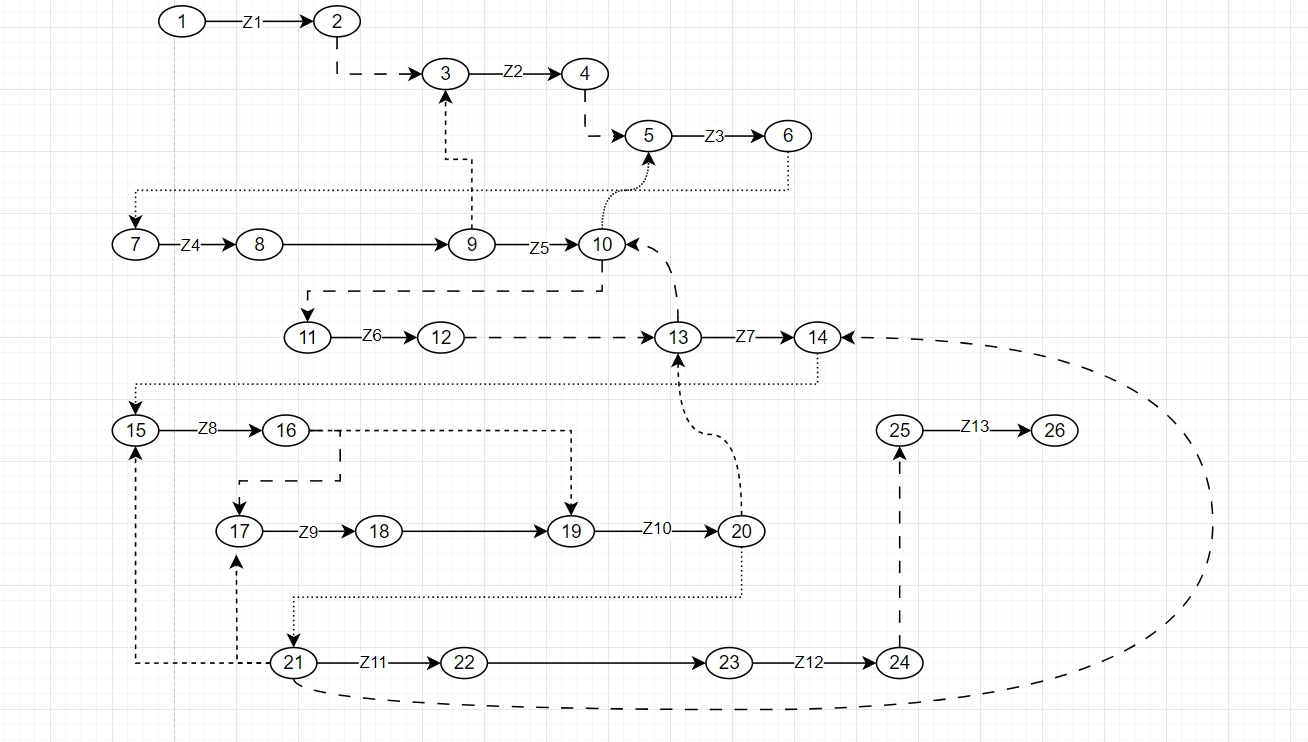


Рисунок 3 – Сетевой график

Это – минимальное время, необходимое для завершения всего проекта, при последовательном выполнении задач.

**Вывод**: В результате данной лабораторной работы были приобретены навыки сетевого планирования, построения сетевого графа, поиска критического пути а также структурное и календарное планирование создания облачного хранилища.

# Лабораторная работа 8. Графический метод решения оптимизационных задач

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

**Задание для выполнения:**

Задание рассчитано на повторение пройденного материала.

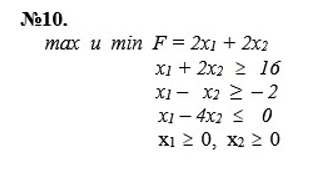


Рисунок 1 – Условие задачи

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

Полуплоскости:

1)

Если x = 0:

Если y = 0:

2)

Если x = 0:

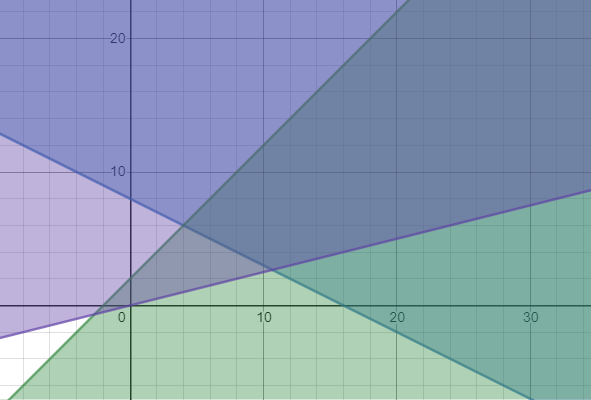
Если y = 0:

3)

Если x = 0:

Если y = 0:

Строим каждую прямую и определяем полуплоскости, заданные неравенствами. В результате получаем следующий график – рисунок 2.



Границам области допустимых решений будут точки пересечения 1 и 2 линии, а также 2 и 3.

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

Рассмотрим целевую функцию F = 2x1 + 2x2 = 0. Строим прямую.

Т.к. конец плоскости на графике будет не определен и из этого следует, что Max значение будет ∞

Max = ∞

Построим нормаль к целевой функции.

Будем двигать эту прямую параллельным образом. Поскольку нас интересует минимальное решение, поэтому двигаем прямую до первого касания обозначенной области – рисунок 3.

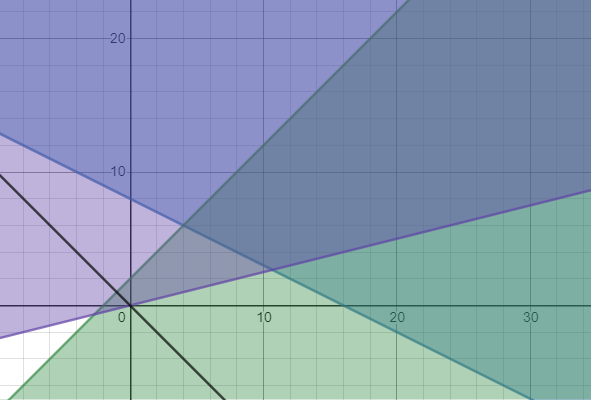


Рисунок 3 – Нахождение минимума

Поскольку нас интересует минимальное решение, поэтому двигаем прямую до первого касания обозначенной области. Прямая F(x) = const пересекает область в точке A, которая была получена пересечением первой и второй прямых.

Min = (2,3);

Следовательно, нужно решить систему уравнений, так как первая точка получена в результате пересечения двух прямых, и её координаты удовлетворяют уравнениям этих прямых:

Решив систему уравнений, получим: x1 =4, x2 =6. Подставив в целевую функцию, получим 2 \* 4+ 2 \* 6= 20.Следовательно, максимальное значение функции равно 20.

Min= 20.

Max = ∞

**Вывод**: В результате данной лабораторной работы было освоено решение задач графическим методом.