Учреждение образования

«БЕЛОРУССКИЙ ГОСУДАРСТВЕННЫЙ ТЕХНОЛОГИЧЕСКИЙ УНИВЕРСИТЕТ»

Дисциплина «Математическое программирование»

Отчёт по лабораторным работам

Студент: Турчинович Н. А.

ФИТ 2 курс 2 группа

.

Минск 2024

**Содержание**

[Лабораторная работа 1. Вспомогательные функции 3](#_Toc165902289)

[Лабораторная работа 2. Комбинаторные алгоритмы решения оптимизационных задач 8](#_Toc165902290)

[Лабораторная работа 3. Метод ветвей и границ 19](#_Toc165902291)

[Лабораторная работа 4. Динамическое программирование 24](#_Toc165902292)

[Лабораторная работа 5. Транспортная задача 32](#_Toc165902293)

[Лабораторная работа 6. Алгоритмы на графах 40](#_Toc165902294)

# 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:

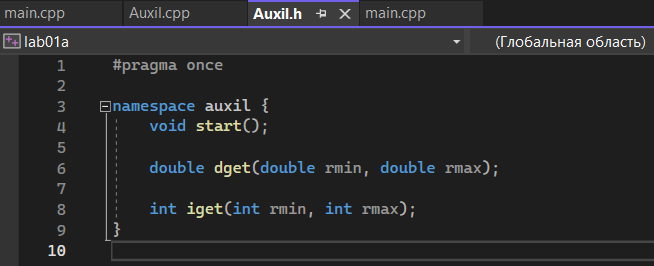


Рисунок 1.1 – Файл Auxio.h

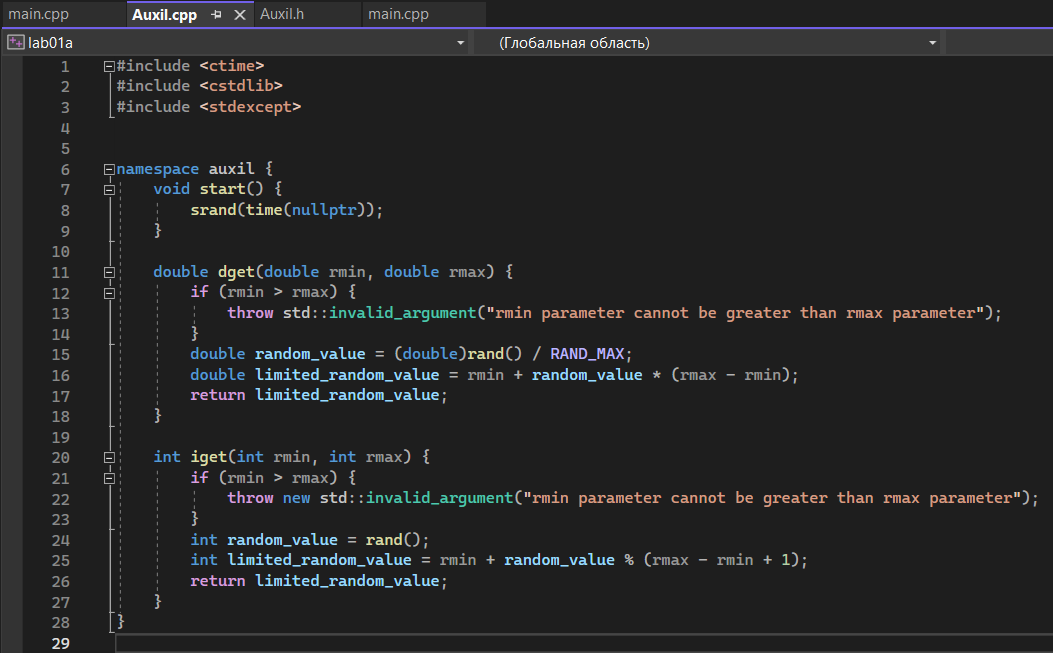


Рисунок 1.1 – Файл Auxio.cpp

***Задание 2***

1. Реализовать пример 2. Представлено на рисунке 1.3.

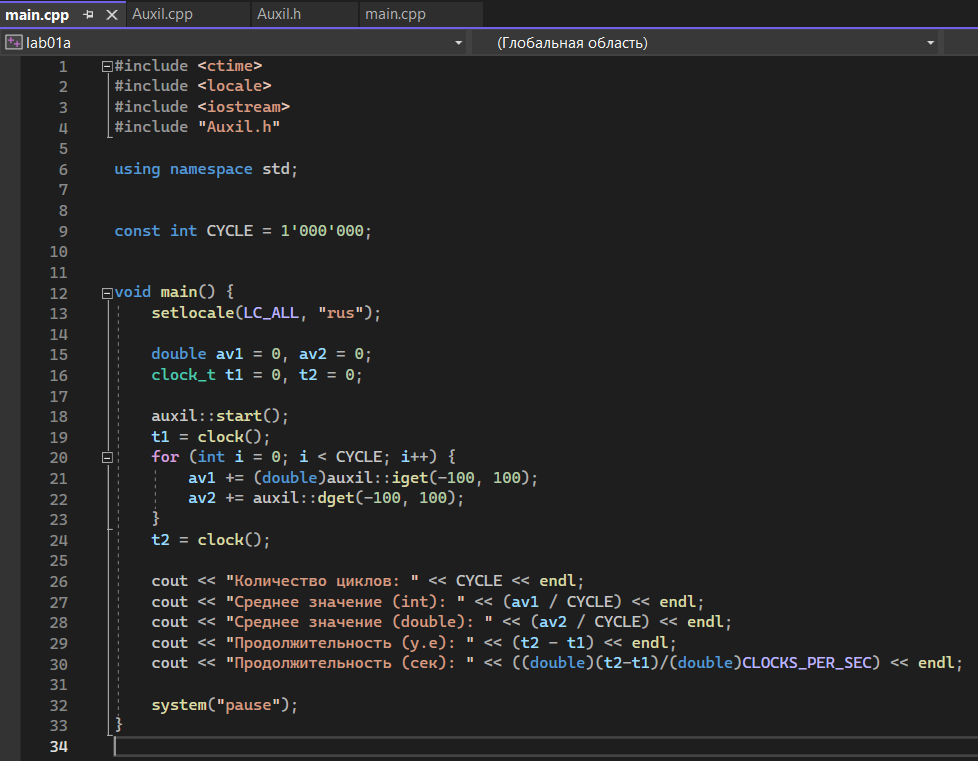


Рисунок 1.3 – Файл Lab1.cpp

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

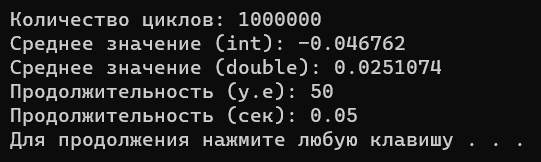


Рисунок 1.4 – Результат замера 1 000 000 циклов

**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.

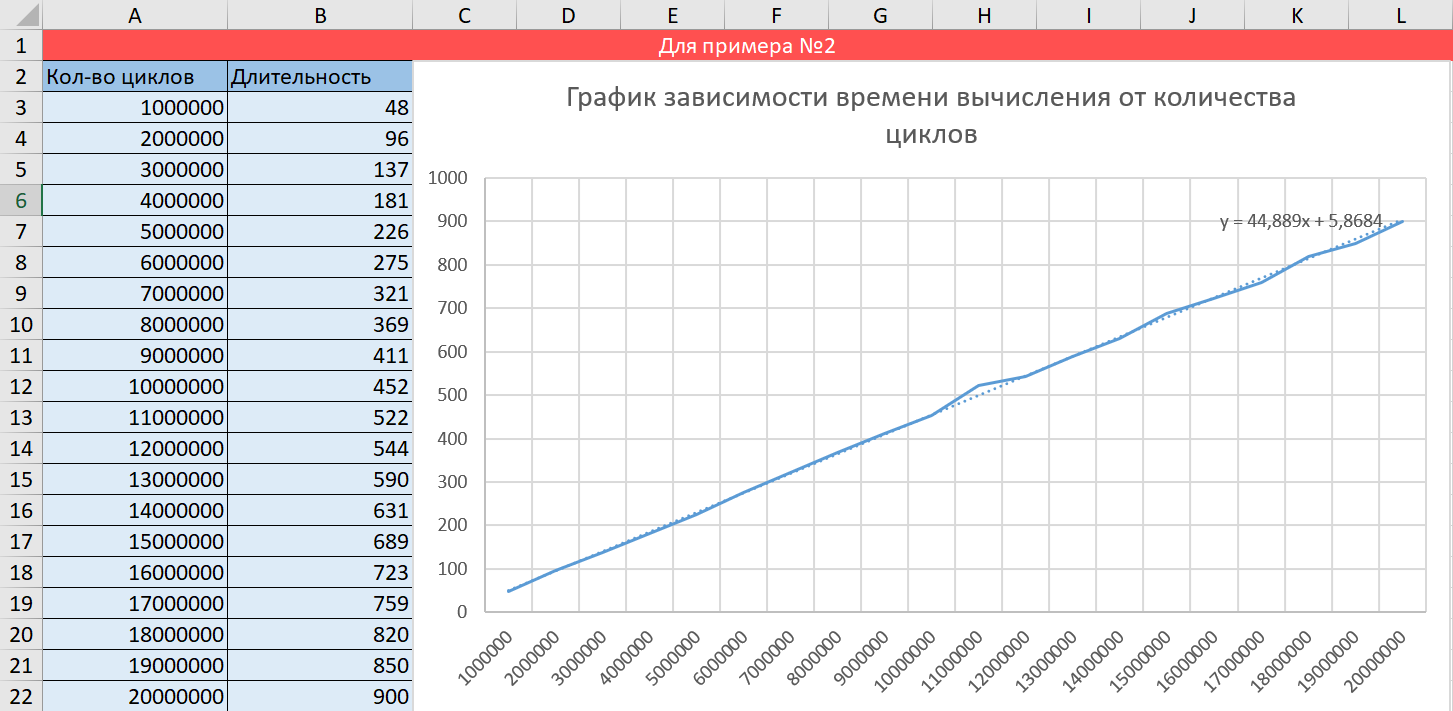


Рисунок 1.5 – Данные и график в Excel

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. Рабочий вариант кода представлен на рисунке 1.6.

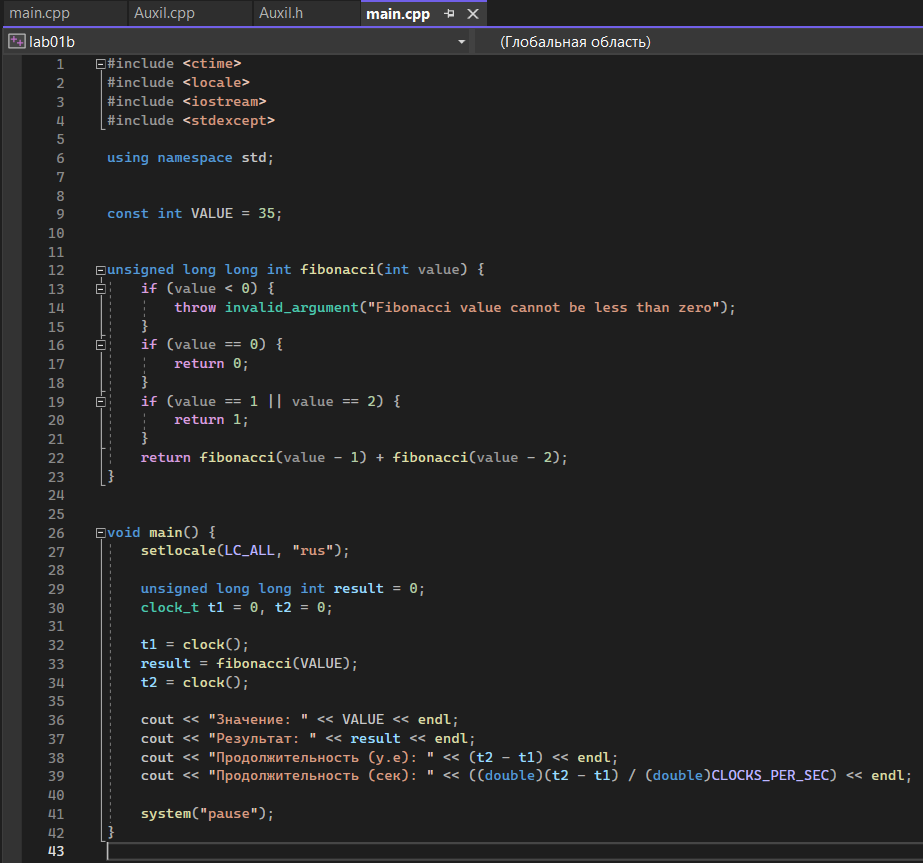


Рисунок 1.6 – Алгоритм вычисления

На рисунке 1.7 представлен результат выполнения программы.

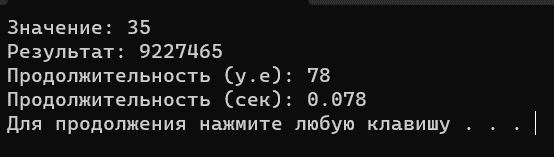


Рисунок 1.7 – Результат выполнения

На рисунке 1.8 представлен график с таблицей значений и времени, потраченного на вычисления чисел от 30 до 50.



Рисунок 1.8 – Таблица чисел

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

**Вывод:**

Продолжительность выполнения программы линейно зависит от количества итерация и циклов выполнения, что видно на графике рисунка 1.5.

Продолжительность подсчета чисел Фиббоначчи имеет экспоненциальную зависимость, что видно на графике рисунка 1.8.

# 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 1:** Разработать генератор подмножеств заданного множества. Генератор подмножеств заданного множества будет на рисунке 2.1.

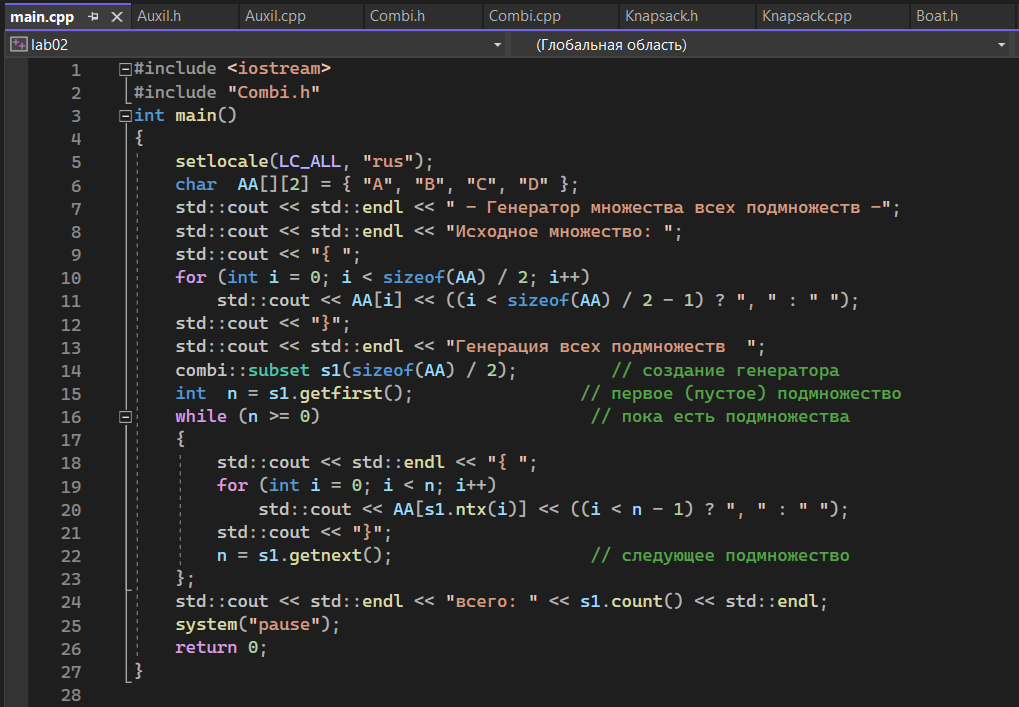


Рисунок 2.1 – Генератор подмножеств

Результат выполнения будет представлен на рисунке 2.2.

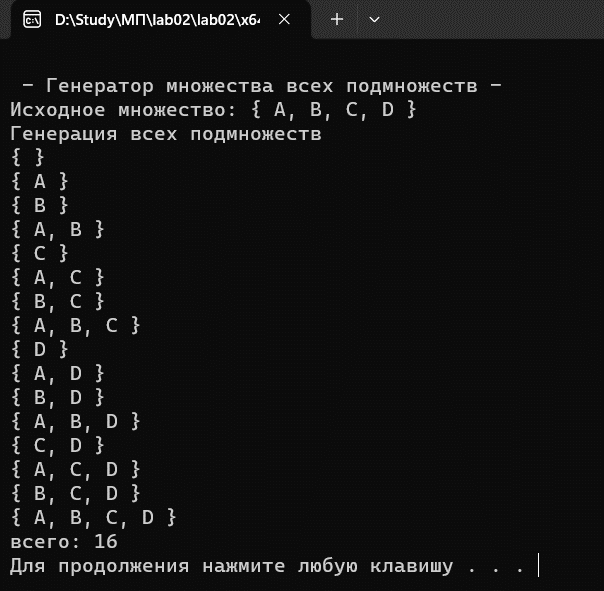


Рисунок 2.2 – Генератор подмножеств заданного множества

Алгоритм будет представлен на рисунке 2.3.



Рисунок 2.3 – Генератор подмножеств заданного множества

**Задание 2:** Разработать генератор сочетаний. Код будет представлен на рисунке 2.4.

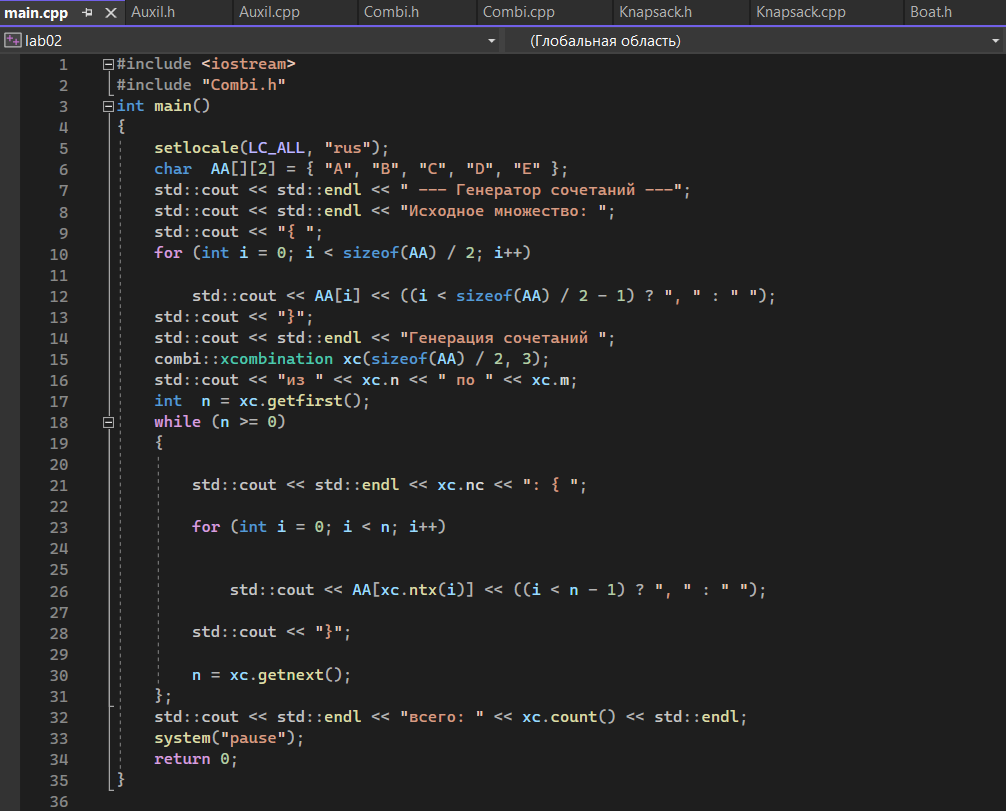


Рисунок 2.4 – Код генератора сочетаний

Результат выполнения будет представлен на рисунке 2.5.

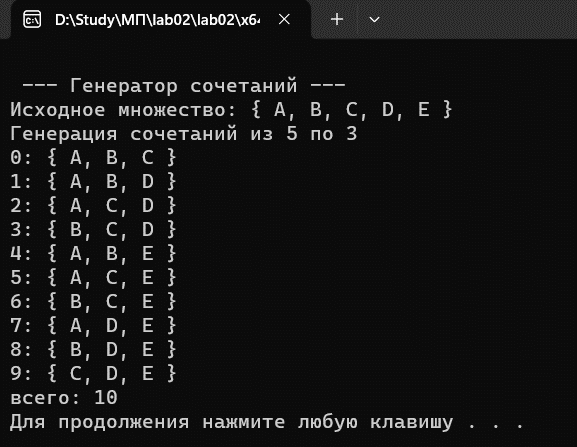


Рисунок 2.5 –Результат генератора сочетаний

Алгоритм будет представлен на рисунке 2.6

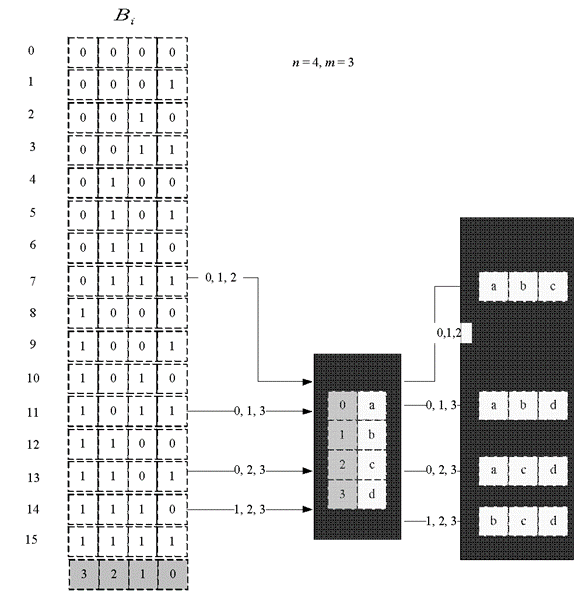


Рисунок 2.6 – Генератор сочетаний

**Задание 3:** Разработать генератор перестановок. Код будет представлен на рисунке 2.7.

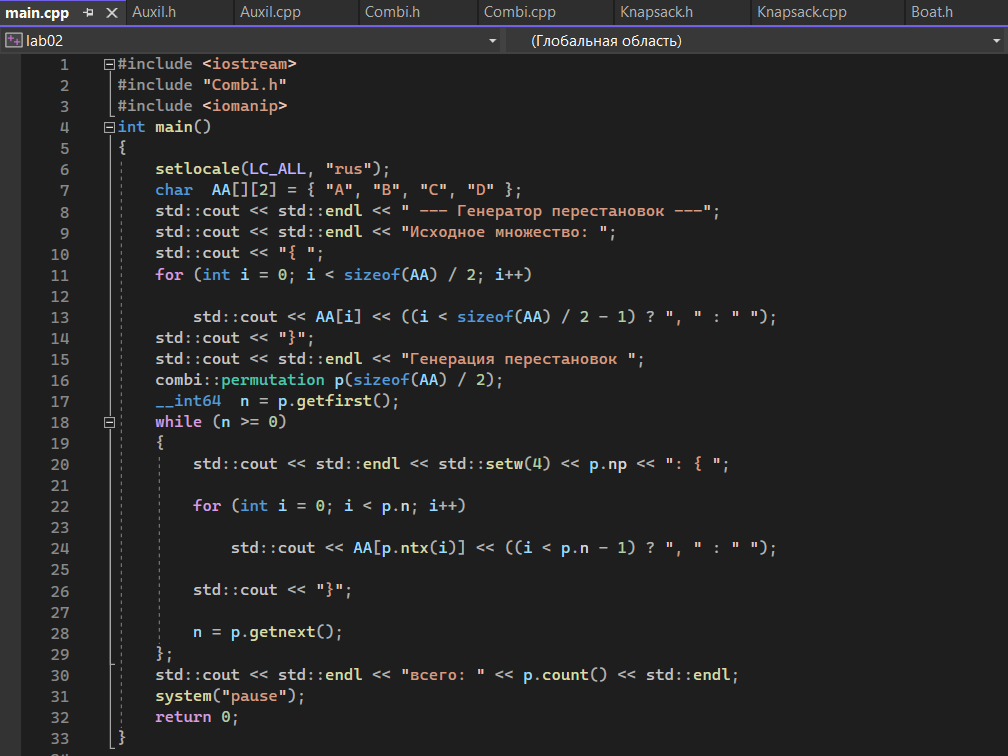


Рисунок 2.7 – Код генератора перестановок

Результат выполнения будет представлен на рисунке 2.8.

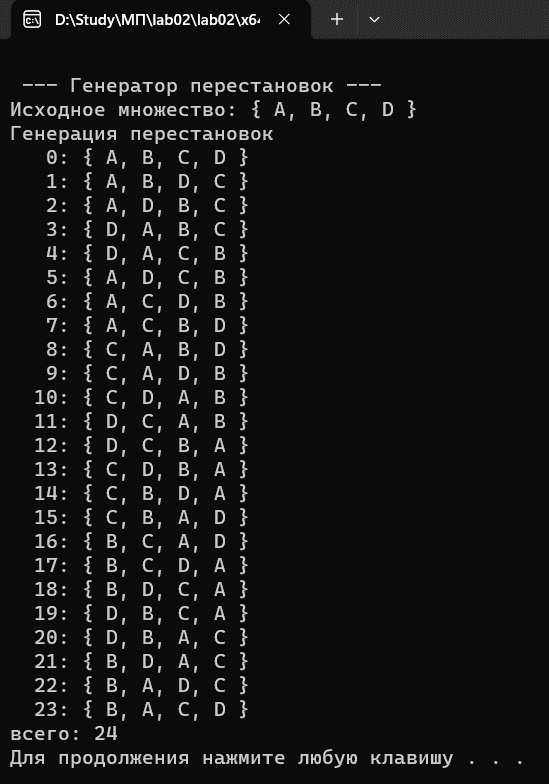


Рисунок 2.8 –Результат генератора перестановок

Алгоритм будет представлен на рисунке 2.9



Рисунок 2.9 – Генератор перестановок

**Задание 4:** Разработать генератор размещений. Код будет представлен на рисунке 2.10.

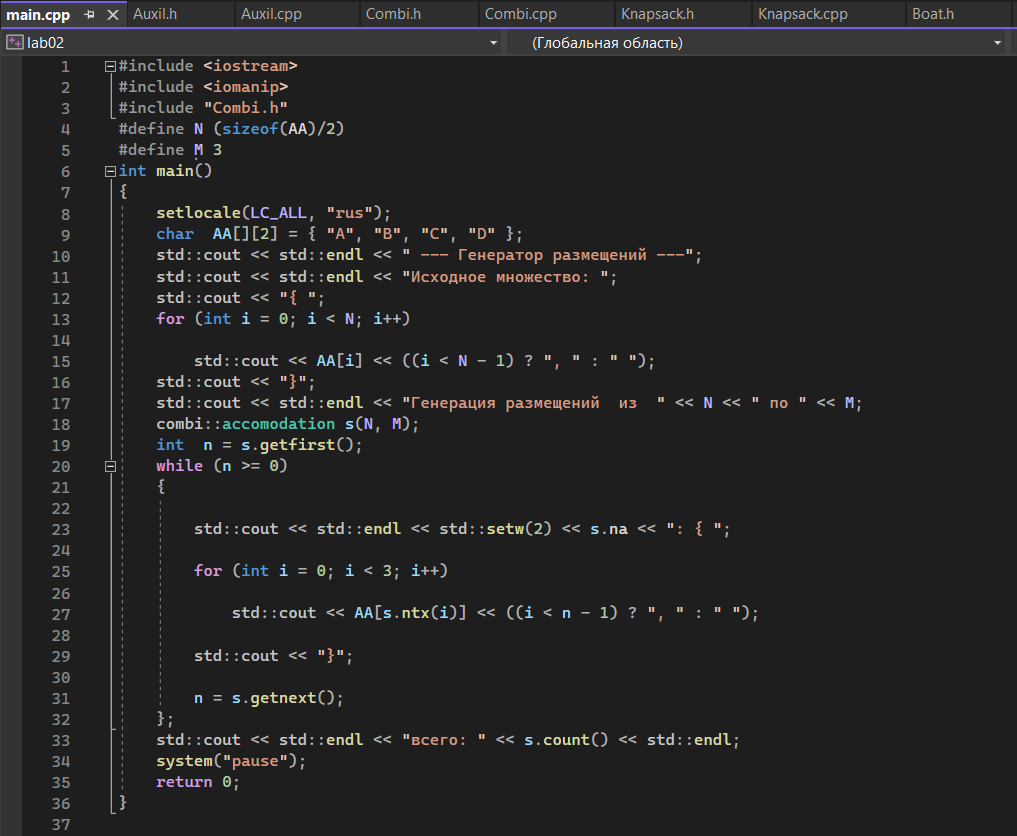


Рисунок 2.10 – Код генератора размещений

Результат выполнения будет представлен на рисунке 2.11.

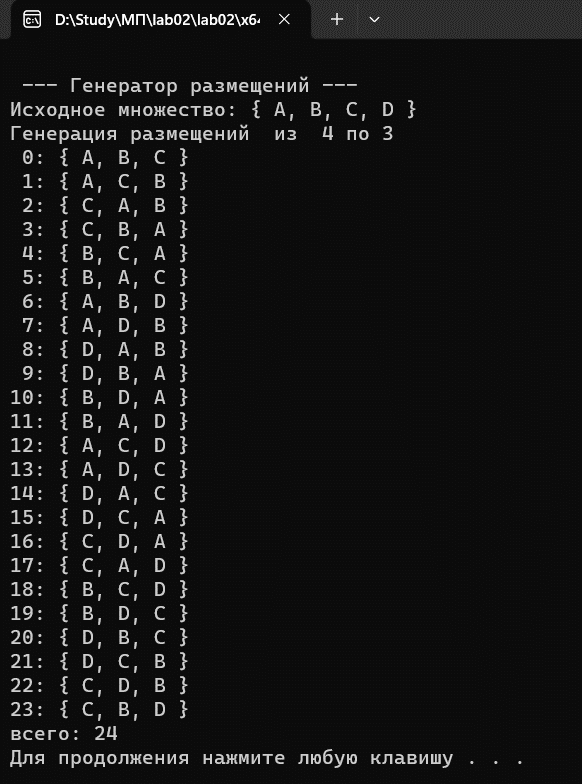


Рисунок 2.11 –Результат генератора размещений

Алгоритм будет представлен на рисунке 2.12.



Рисунок 2.12– Генератор размещений

**Задание 5:** Решить в соответствии с вариантом задачу и результат занести в отчет (Вариант распределяется по списку): (14) упрощенную о рюкзаке (веса предметов и их стоимость сгенерировать случайным образом: вместимость рюкзака 300 кг, веса предметов 10 – 300 кг, предметов 5 – 55 у.е., количество предметов – 18 шт.).

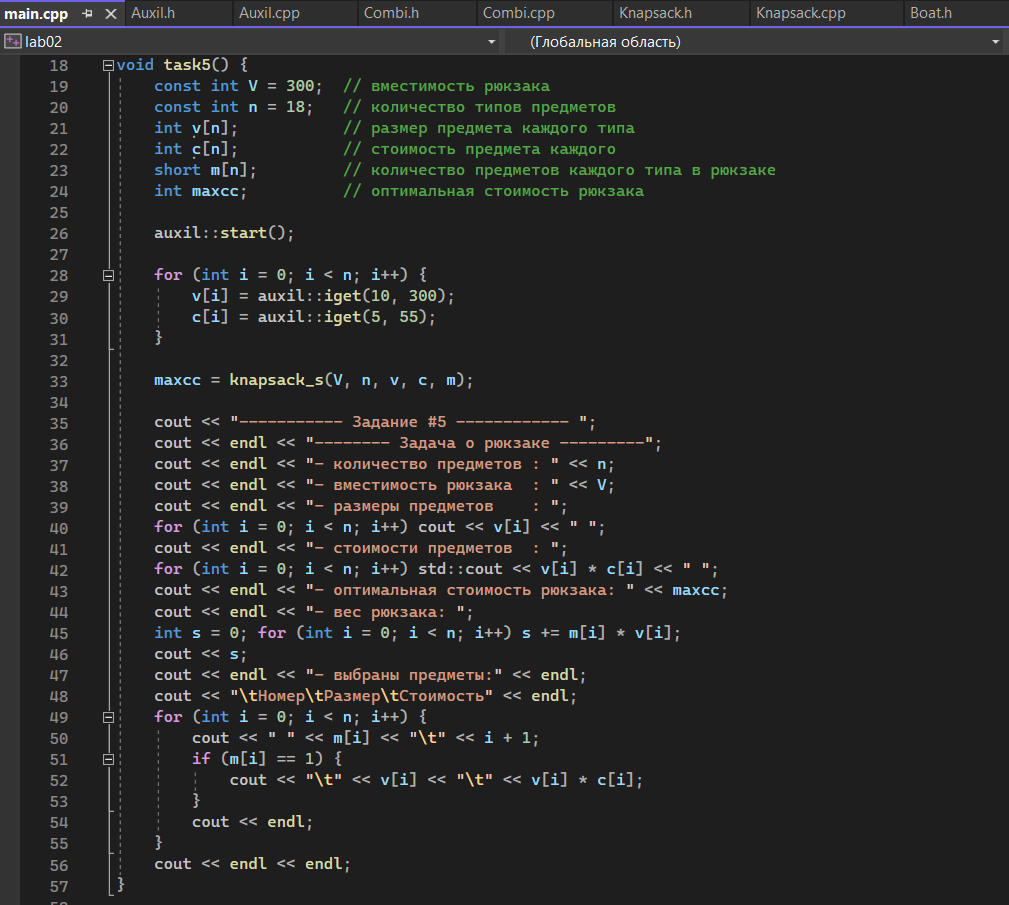


Рисунок 2.13 – Код задания 5

Результат выполнения будет представлен на рисунке 2.14.

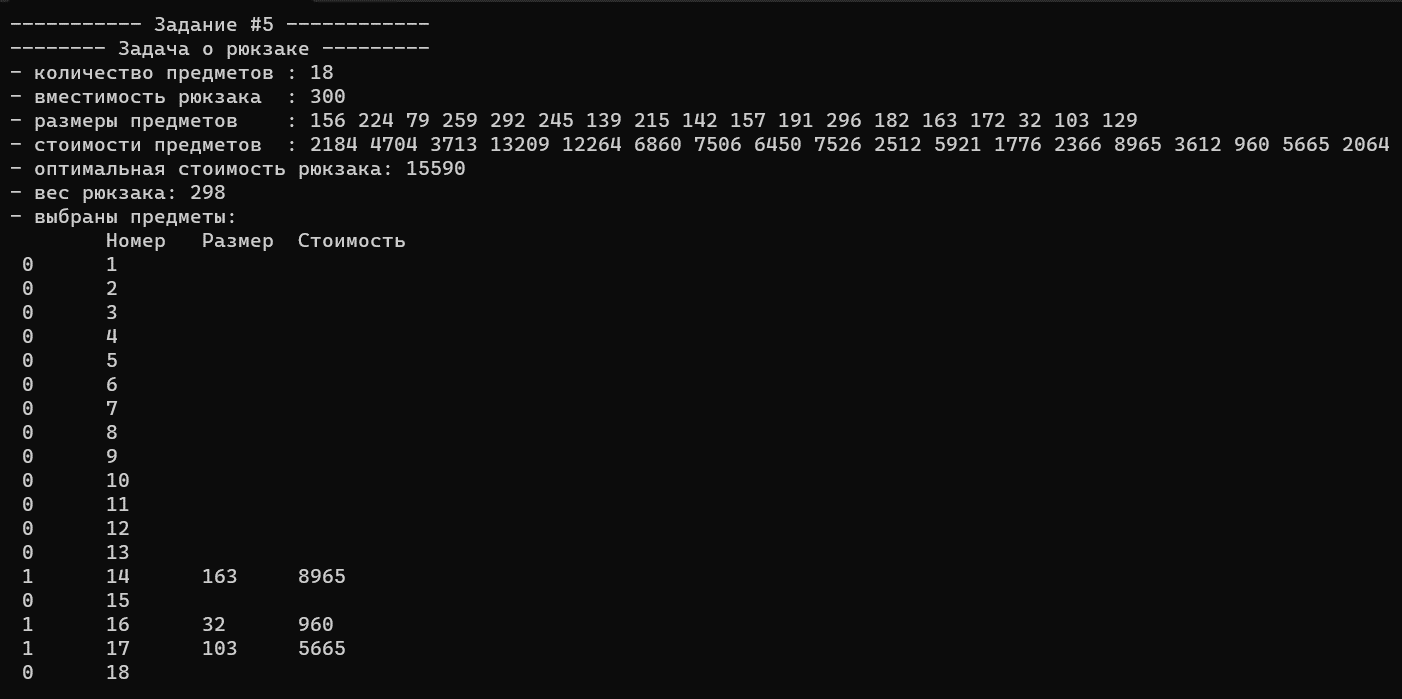


Рисунок 2.14 –Результат задания 5

**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: (14) упрощенную о рюкзаке (количество предметов 12 – 20 шт.)

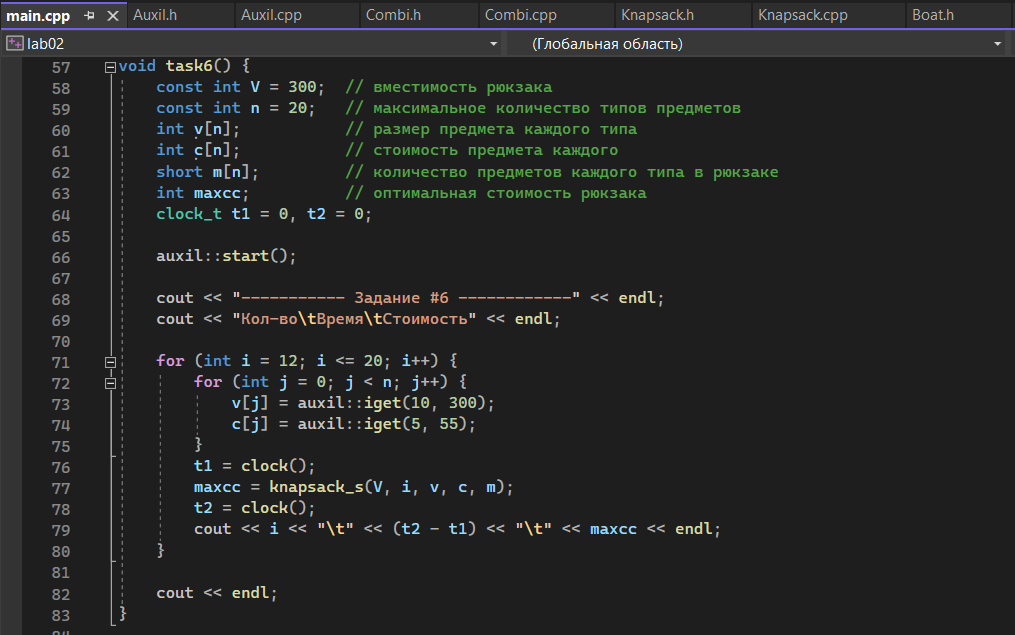


Рисунок 2.15 – Код задания 6

Результат выполнения будет представлен на рисунке 2.16.

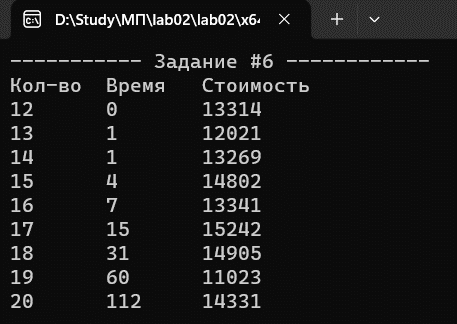


Рисунок 2.16–Результат задания 6

График будет представлен на рисунке 2.17.



Рисунок 2.17 – График

Исходя из данного графика, можно сделать вывод о том, что зависимость времени решения задачи от количества элементов является экспоненциальной. Алгоритм будет представлен на рисунке 2.18.



Рисунок 2.18– Схема решения задачи о рюкзаке

**Вывод:** В ходе работы были приобретены навыки разработки генераторов подмножеств, перестановок, сочетаний и размещений на С++; применены разработанные генераторы для решения упрощенной задачи о рюкзаке, а также построен график зависимости времени решения задачи от количества элементов.

# Лабораторная работа 3. Метод ветвей и границ

**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** |  | 2 \* n | 21 + n |  | n |
| **2** | n |  | 15 + n | 68 - n | 84 - n |
| **3** | 2 + n | 3 \* n |  | 86 | 49 + n |
| **4** | 17 + n | 58 - n | 4 \* n |  | 3 \* n |
| **5** | 93 - n | 66 + n | 52 | 13 + n |  |

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

Вариант №14.

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| Город | 1 | 2 | 3 | 4 | 5 |
| 1 | INF | 28 | 35 | INF | 14 |
| 2 | 14 | INF | 29 | 54 | 70 |
| 3 | 16 | 42 | INF | 86 | 63 |
| 4 | 31 | 44 | 56 | INF | 42 |
| 5 | 79 | 80 | 52 | 27 | INF |

**Задание 2.**

Решить сформулированную задачу методом ветвей и границ.

Для определения нижней границы множества приведем матрицу по строкам, для чего необходимо в каждой строке матрицы найти минимальный элемент и вычесть его из всех элементов строки. Такую же операцию проводим по столбцам, для чего в каждом столбце находим минимальный элемент.

1. Приведение по строкам.

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| Город | 1 | 2 | 3 | 4 | 5 | MIN |
| 1 | INF | 14 | 21 | INF | 0 | 14 |
| 2 | 0 | INF | 15 | 40 | 56 | 14 |
| 3 | 0 | 26 | INF | 70 | 47 | 16 |
| 4 | 0 | 13 | 25 | INF | 11 | 31 |
| 5 | 52 | 53 | 25 | 0 | INF | 27 |

2. Приведение по столбцам.

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| Город | 1 | 2 | 3 | 4 | 5 | MIN |
| 1 | INF | 1 | 6 | INF | 0 | 14 |
| 2 | 0 | INF | 0 | 40 | 56 | 14 |
| 3 | 0 | 13 | INF | 70 | 47 | 16 |
| 4 | 0 | 0 | 10 | INF | 11 | 31 |
| 5 | 52 | 40 | 10 | 0 | INF | 27 |
| MIN | 0 | 13 | 15 | 0 | 0 |  |

3. Определение нижней границы.

После вычитания минимальных элементов получаем полностью приведенную матрицу, где величины MIN называются **константами приведения**. Сумма констант приведения определяет нижнюю границу H = 14 + 14 + 16 + 31 + 27 + 0 + 13 + 15 + 0 + 0 = 130.

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| Город | 1 | 2 | 3 | 4 | 5 | MIN |
| 1 | INF | 1 | 6 | INF | 0 | 14 |
| 2 | 0 | INF | 0 | 40 | 56 | 14 |
| 3 | 0 | 13 | INF | 70 | 47 | 16 |
| 4 | 0 | 0 | 10 | INF | 11 | 31 |
| 5 | 52 | 40 | 10 | 0 | INF | 27 |
| MIN | 0 | 13 | 15 | 0 | 0 | 130 |

Полностью приведенная матрица:

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| Город | 1 | 2 | 3 | 4 | 5 |
| 1 | INF | 1 | 6 | INF | 0 |
| 2 | 0 | INF | 0 | 40 | 56 |
| 3 | 0 | 13 | INF | 70 | 47 |
| 4 | 0 | 0 | 10 | INF | 11 |
| 5 | 52 | 40 | 10 | 0 | INF |

4. Матрица 5х5

**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.

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| Город | 1 | 2 | 3 | 4 | 5 |
| 1 | INF | 1 | 6 | INF | 0 (12) |
| 2 | 0 (0) | INF | 0 (6) | 40 | 56 |
| 3 | 0 (13) | 13 | INF | 70 | 47 |
| 4 | 0 (0) | 0 (1) | 10 | INF | 11 |
| 5 | 52 | 40 | 10 | 0 (50) | INF |

Наибольшая сумма констант приведения равна 50 для ребра (5,4), 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\*) = 130 + 50 = 180.

**Включение ребра** (5,4) проводится путем исключения всех элементов 5-ой строки и 4-го столбца, в которой элемент d45 заменяем на бесконечность, для исключения образования негамильтонова цикла.

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) = 130 + 0 = 130 ≤ 180. Поскольку нижняя граница этого подмножества (5,4) меньше, чем подмножества (5\*,4\*), то ребро (5,4) включаем в маршрут с новой границей H = 130.

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| Город | 1 | 2 | 3 | 4 | 5 | MIN |
| 1 | INF | 1 | 6 | INF | 0 | 0 |
| 2 | 0 | INF | 0 | 40 | 56 | 0 |
| 3 | 0 | 13 | INF | 70 | 47 | 0 |
| 4 | 0 | 0 | 10 | INF | INF | 0 |
| 5 | 52 | 40 | 10 | 0 | INF |  |
| MIN | 0 | 0 | 0 |  | 0 | 0 |

Полностью приведенная матрица:

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Город | 1 | 2 | 3 | 5 |
| 1 | INF | 1 | 6 | 0 |
| 2 | 0 | INF | 0 | 56 |
| 3 | 0 | 13 | INF | 47 |
| 4 | 0 | 0 | 10 | INF |

Аналогичные действия повторяются для матриц размеров 4x4 и 5x5.

5. Матрица 4x4

Определение ребра ветвления.

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Город | 1 | 2 | 3 | 5 |
| 1 | INF | 1 | 6 | 0 (48) |
| 2 | 0 (0) | INF | 0 (6) | 56 |
| 3 | 0 (13) | 13 | INF | 47 |
| 4 | 0 (0) | 0 (1) | 10 | INF |

Включение ребра (1, 5).

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| Город | 1 | 2 | 3 | 5 | MIN |
| 1 | INF | 1 | 6 | 0 |  |
| 2 | 0 | INF | 0 | 56 | 0 |
| 3 | 0 | 13 | INF | 47 | 0 |
| 4 | 0 | 0 | 10 | INF | 0 |
| MIN | 0 | 0 | 0 |  | 0 |

Полностью приведенная матрица:

|  |  |  |  |
| --- | --- | --- | --- |
| Город | 1 | 2 | 3 |
| 2 | 0 | INF | 0 |
| 3 | 0 | 13 | INF |
| 4 | 0 | 0 | 10 |

6. Матрица 3x3.

Определение ребра ветвления.

|  |  |  |  |
| --- | --- | --- | --- |
| Город | 1 | 2 | 3 |
| 2 | 0 (0) | INF | 0 (10) |
| 3 | 0 (13) | 13 | INF |
| 4 | 0 (0) | 0 (13) | 10 |

Включение ребра (4, 2).

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Город | 1 | 2 | 3 | MIN |
| 2 | 0 | INF | 0 | 0 |
| 3 | 0 | 13 | INF | 0 |
| 4 | 0 | 0 | 10 |  |
| MIN | 0 |  | 0 | 0 |

Полностью приведенная матрица:

|  |  |  |
| --- | --- | --- |
| Город | 1 | 3 |
| 2 | 0 | 0 |
| 3 | 0 | INF |

7. Матрица 2x2.

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

|  |  |  |
| --- | --- | --- |
| Город | 1 | 3 |
| 2 | 0 | 0 |
| 3 | 0 | INF |

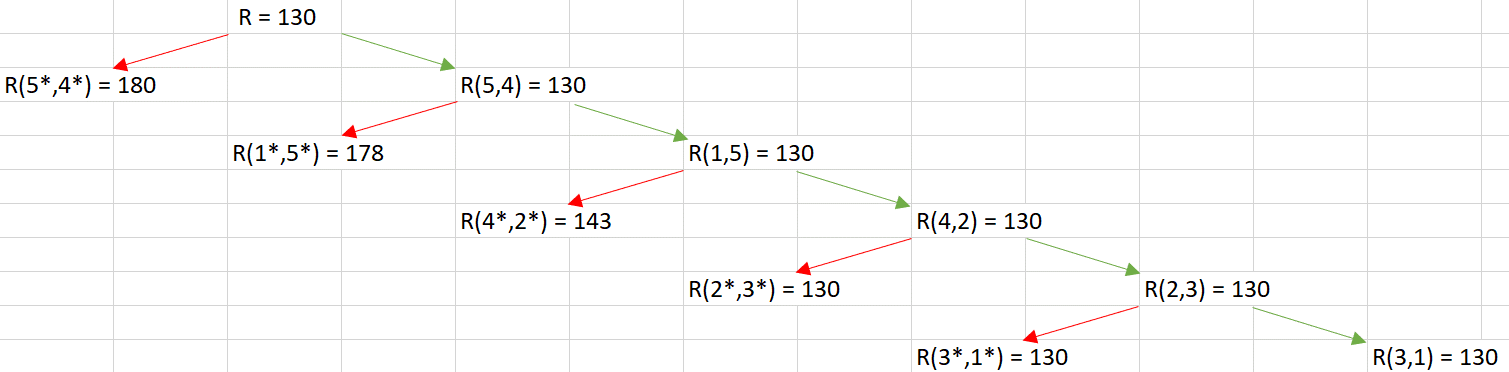
8. Длина маршрута.

Длина маршрута равна F(∞) = 130.

|  |  |
| --- | --- |
| Отрезок | Длина |
| 5 -> 4 | 27 |
| 1 -> 5 | 14 |
| 4 -> 2 | 44 |
| 2 -> 3 | 29 |
| 3 -> 1 | 16 |
| Всего | 130 |

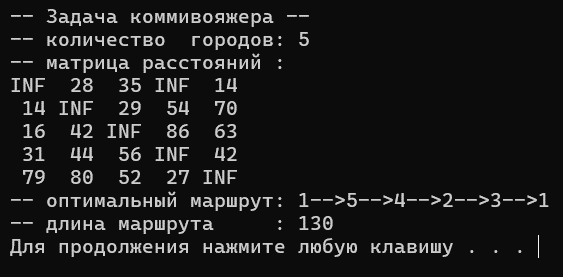
9. Дерево ветвлений.

В результате по дереву ветвлений гамильтонов цикл образуют ребра:  
(5, 4), (1, 5), (4, 2), (2, 3), (3, 1). Оптимальный маршрут 1->5->4->2->3->1.



**Задание 3.**

Вывод программы.



**Вывод**: решение, полученное при помощи генератора перестановок, совпадает с решением, полученным при использовании метода ветвей и границ.

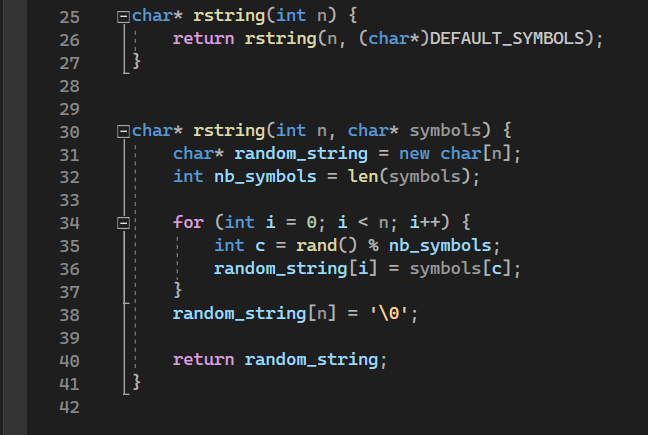
# Лабораторная работа 4. Динамическое программирование

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

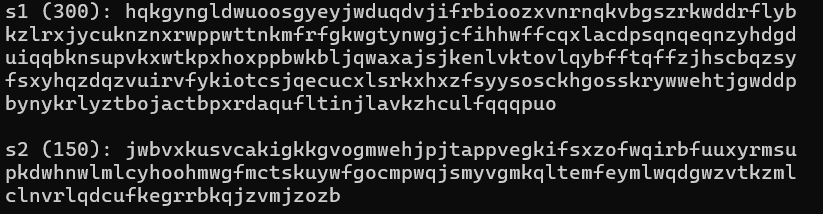
**Задание 1.**

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

Код генерации случайных строк заданной длины:



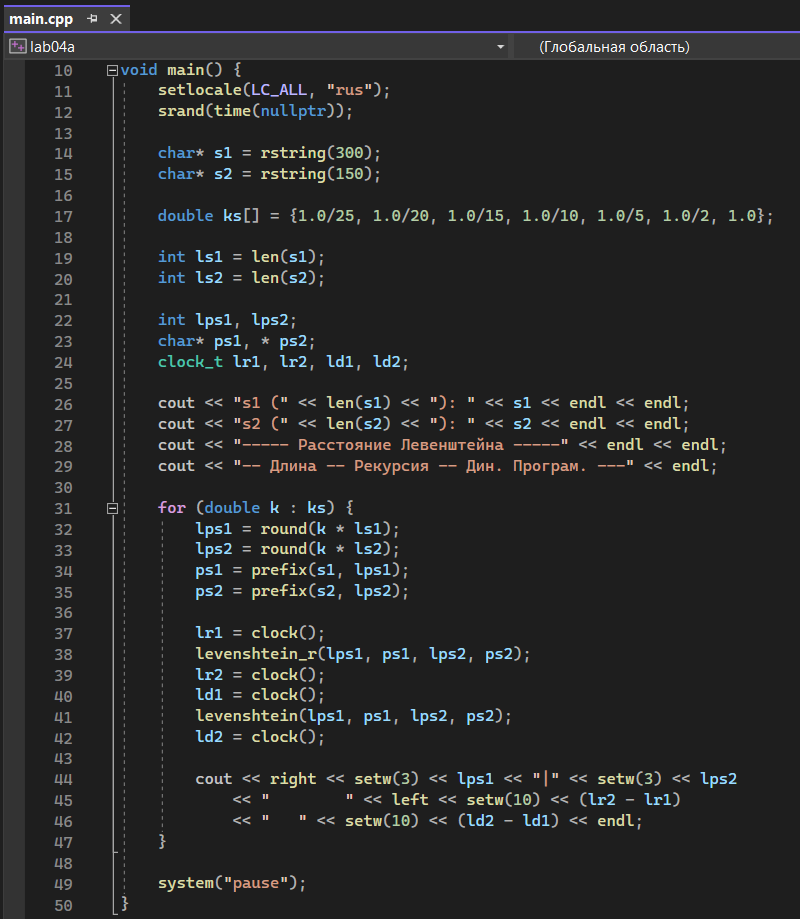
Пример генерации случайных строк заданной длины:



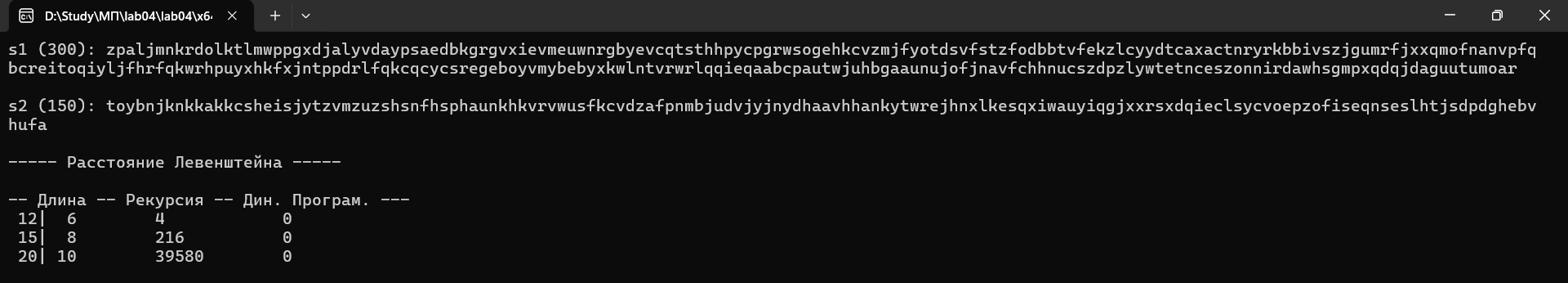
**Задание 2.**

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).

Код программы:



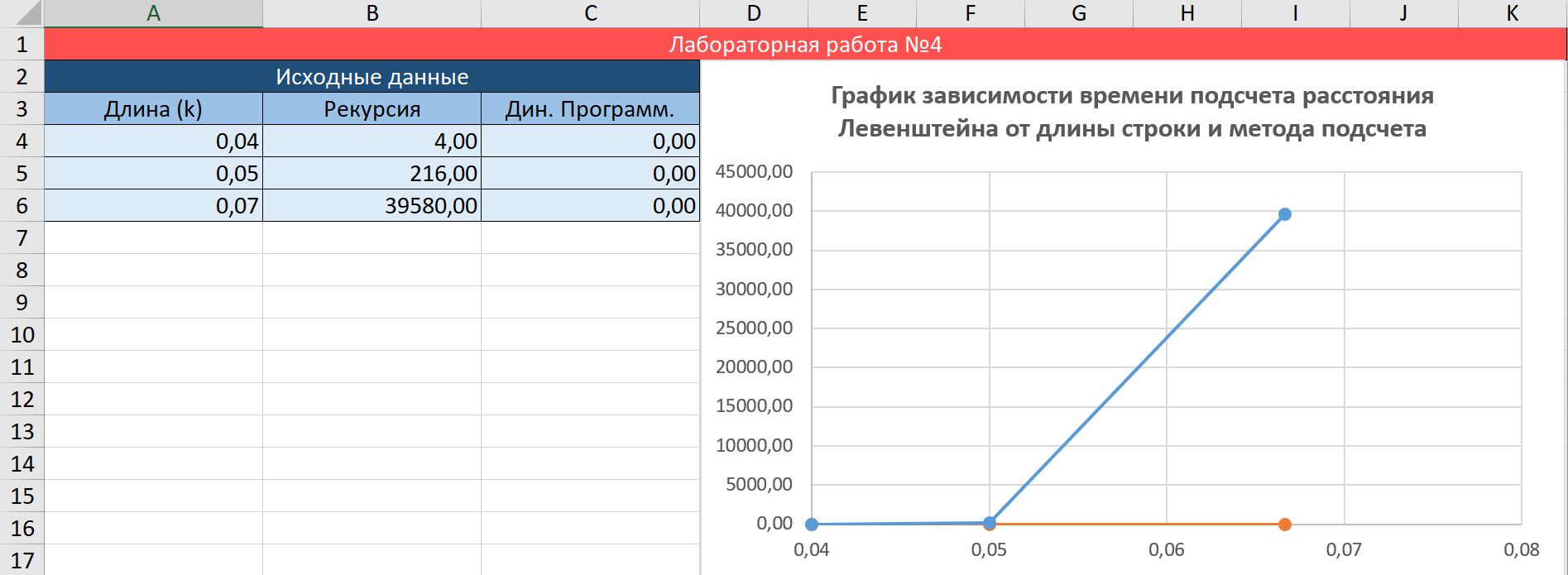
Окно вывода:



**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).

График зависимости времени вычисления от длин строк:



Исходя из графика зависимости видно, что метод рекурсии выполняется значительно медленнее метода динамического программирования, на большей длине строк этот эффект проявляется сильнее.

**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).

Первое слово: Сан; Второе слово: Сонар (Вариант 14).

* + - 1. L(“сан”, “сонар”) =
      2. L(“са”, “сонар”) =
      3. L(“сан”, “сона”) =
      4. L(“са”, “сона”) =
      5. L(“с”, “сонар”) =

L(“”, “сонар”) = 5

L(“”, “сона”) = 4

* + - 1. L(“с”, “сона”) =

L(“”, “сона”) = 4

L(“”, “сон”) = 3

* + - 1. L(“сан”, “сон”) =
      2. L(“са”, “сон”) =
      3. L(“сан”, “со”) =
      4. L(“сан”, “с”) =

L(“сан”, “”) = 3

L(“са”, “”) = 2

* + - 1. L(“с”, “сон”) =

L(“”, “сон”) = 3

L(“”, “со”) = 2

* + - 1. L(“са”, “со”) =

L(“с”, “с”) = 0

* + - 1. L(“с”, “со”) =

L(“”, “со”) = 2

L(“с”, “с”) = 0

L(“”, “с”) = 1

* + - 1. L(“са”, “с”) =

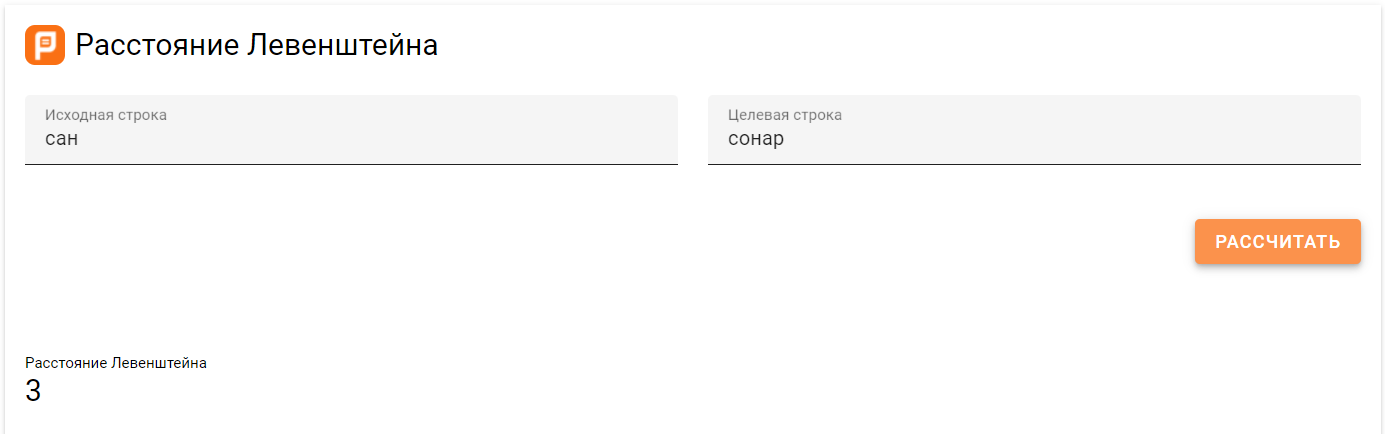
L(“с”, “с”) = 0

L(“са”, “”) = 2

L(“с”, “”) = 1

* + - 1. L(“са”, “с”) = min(1, 3, 2) = 1
      2. L(“с”, “со”) = min(3, 1, 2) = 1
      3. L(“са”, “со”) = min(2, 2, 1) = 1
      4. L(“с”, “сон”) = min(4, 2, 3) = 2
      5. L(“сан”, “с”) = min(2, 4, 3) = 2
      6. L(“сан”, “со”) = min(2, 3, 2) = 2
      7. L(“са”, “сон”) = min(3, 2, 2) = 2
      8. L(“сан”, “сон”) = min(3, 3, 1) = 1
      9. L(“с”, “сона”) = min(5, 3, 4) = 3
      10. L(“с”, “сонар”) = min(6, 4, 5) = 4
      11. L(“са”, “сона”) = min(4, 3, 3) = 3
      12. L(“сан”, “сона”) = min(4, 2, 3) = 2
      13. L(“са”, “сонар”) = min(5, 4, 4) = 4
      14. L(“сан”, “сонар”) = min(5, 3, 4) = 3

Проверка



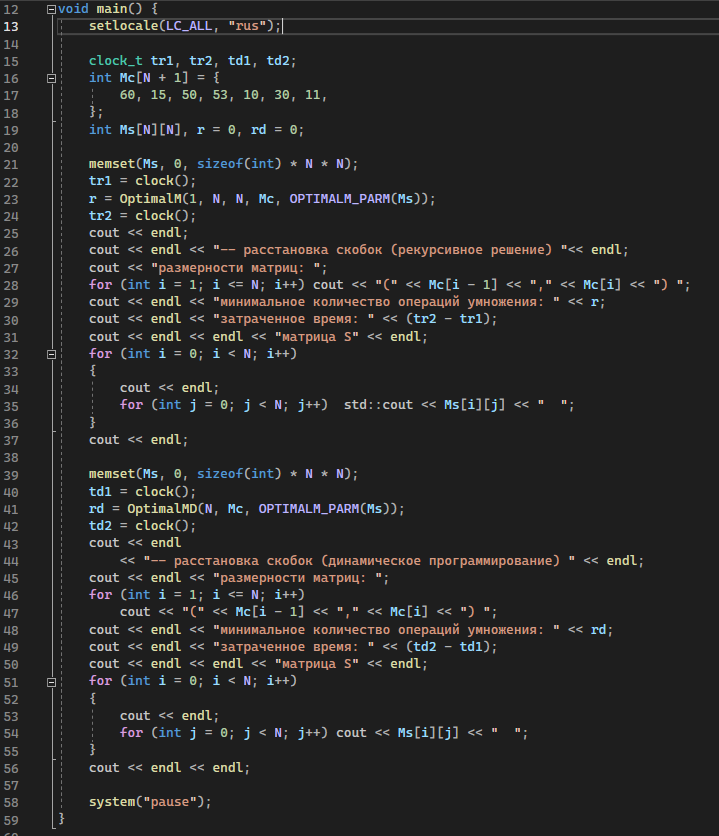
Таким образом, ручное решение рекурсивным способом дало правильный результат.

**Задание 5.**

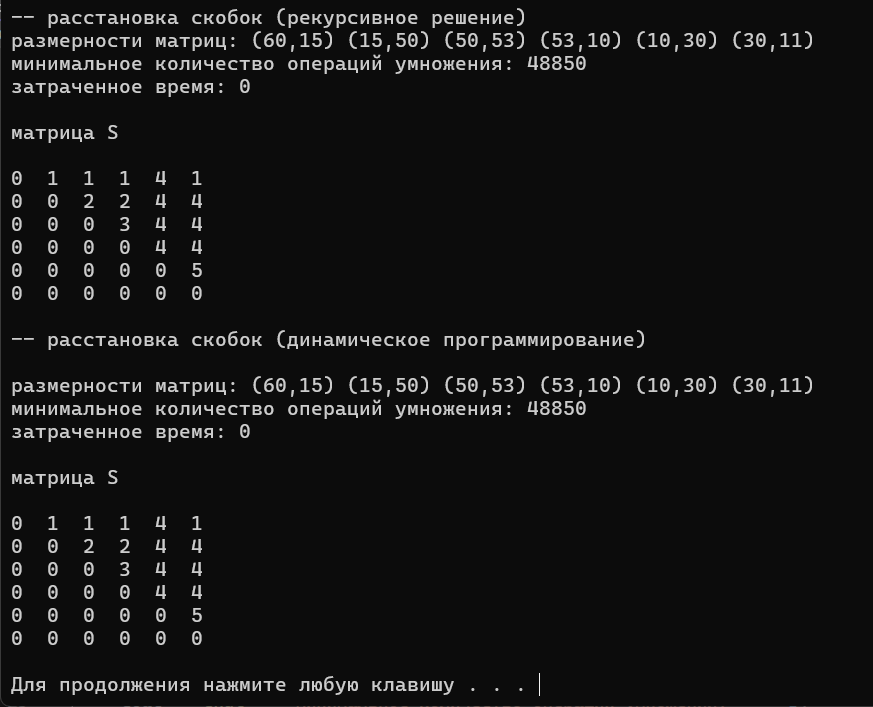
Выполнить сравнительный анализ времени, затраченного на решение задачи об оптимальной расстановке скобок при умножении нескольких матриц для двух методов решения (рекурсивное решение, динамическое программирование). Размерность матриц взять в соответствии с вариантом. Объяснить в отчете принцип расставления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

Условие: 60\*15, 15\*50, 50\*53, 53\*10, 10\*30, 30\*11 (Вариант 14)

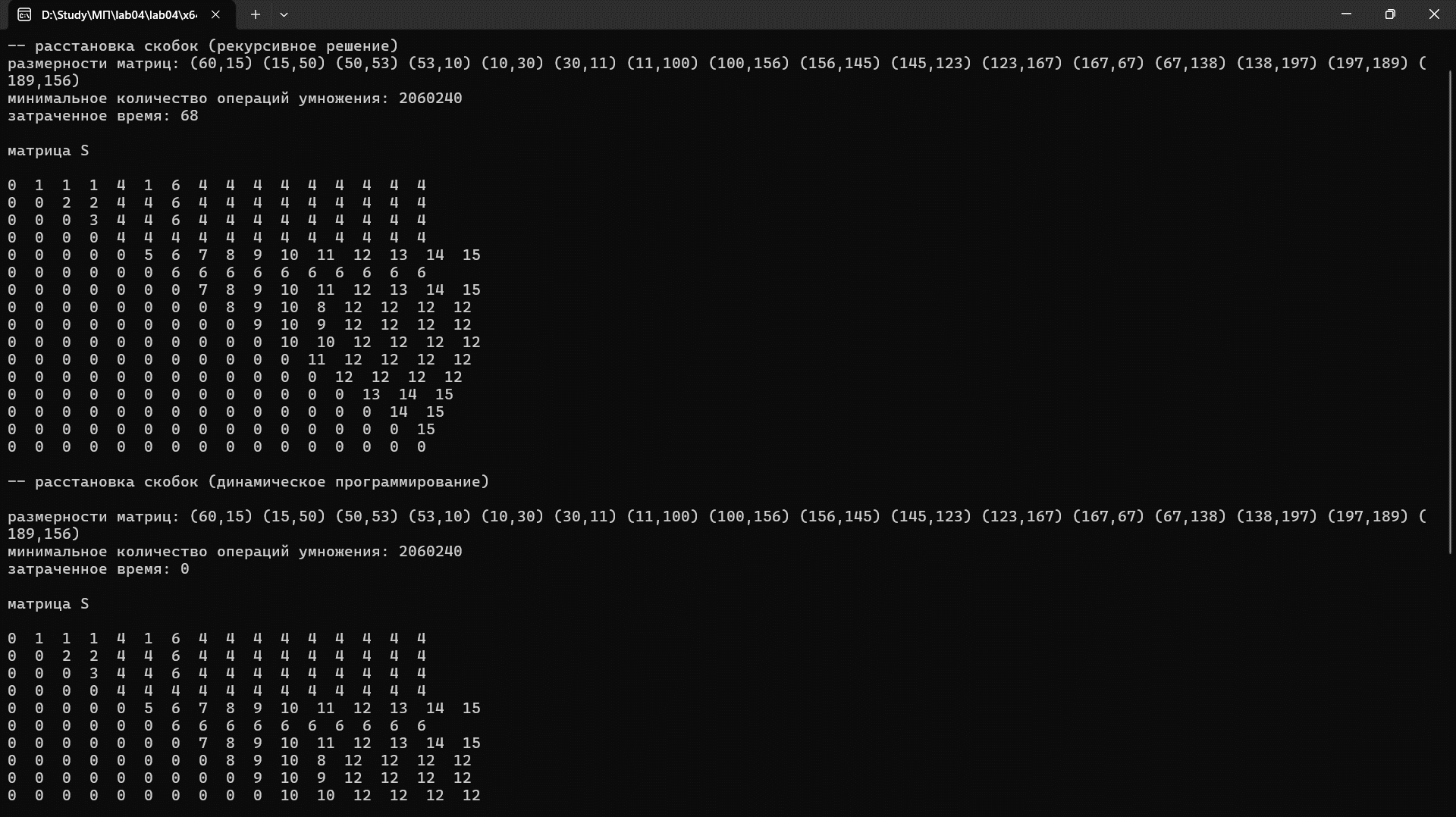
Код программы:



Окно вывода:



Для относительно небольшого количества матриц оба метода выполняются достаточно быстро. При увеличении количества матриц видно, что метод рекурсии выполняется существенно медленнее, в то время как метод динамического программирования не теряет в своей производительности:



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: (60,15); (15,50); (50,53); (53,10); (10,30); (30,11).

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

Совершая те же действия, получается следующая последовательность перемножений:

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

**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.**

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.

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| ПОТРЕБИТЕЛИ  ПОСТАВЩИКИ | 1 | 2 | 3 | 4 | 5 | 6 | ЗАПАСЫ |
| 1 | **N+12** | **N+2** | **N+6** | **N+3** | **N+11** | **N+1** | **168+N** |
| 2 | **N+10** | **N** | **N+8** | **N+5** | **N+7** | **N+13** | **113+N** |
| 3 | **N+1** | **N+5** | **N+11** | **N+8** | **N+2** | **N+11** | **150+N** |
| 4 | **N+4** | **N+10** | **N+10** | **N+3** | **N+13** | **N+2** | **159+N** |
| 5 | **N+3** | **N+11** | **N+9** | **N** | **N+10** | **N+4** | **100+N** |
| ПОТРЕБНОСТИ | **143+N** | **107+N** | **131+N** | **193+N** | **95+N** | **163+N** |  |

Вариант №14

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| Потребители | 1 | 2 | 3 | 4 | 5 | 6 | Запасы |
| Поставщики |
| 1 | 26 | 16 | 20 | 17 | 25 | 15 | 182 |
| 2 | 24 | 14 | 22 | 19 | 21 | 27 | 127 |
| 3 | 15 | 19 | 25 | 22 | 16 | 25 | 164 |
| 4 | 18 | 24 | 24 | 17 | 27 | 16 | 173 |
| 5 | 17 | 25 | 23 | 14 | 24 | 18 | 114 |
| Потребности | 157 | 121 | 145 | 207 | 109 | 177 |  |

**Решение.**

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

ΣЗапасы = 182 + 127 + 164 + 173 + 114 = 760

ΣПотребности = 157 + 121 + 145 + 207 + 109 + 177 = 916

Так как запасы поставщиков меньше потребности потребителей, задача является открытой. Введем фиктивного поставщика 6, с запасом продукции равным 916 -760=156. Стоимость доставки единицы продукции от фиктивного поставщика ко всем потребителям примем равной нулю.

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| Поставщик | Потребитель | | | | | | Запас |
| 1 | 2 | 3 | 4 | 5 | 6 |
| 1 | 26 | 16 | 20 | 17 | 25 | 15 | 182 |
| 2 | 24 | 14 | 22 | 19 | 21 | 27 | 127 |
| 3 | 15 | 19 | 25 | 22 | 16 | 25 | 164 |
| 4 | 18 | 24 | 24 | 17 | 27 | 16 | 173 |
| 5 | 17 | 25 | 23 | 14 | 24 | 18 | 114 |
| 6 | 0 | 0 | 0 | 0 | 0 | 0 | 156 |
| Потребность | 157 | 121 | 145 | 207 | 109 | 177 |  |

Теперь выполняется условие.

*1. 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.

1.1. Искомый элемент равен с22 = 14. x22 = min(121, 127) = 121.

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| Поставщик | Потребитель | | | | | | Запас |
| 1 | 2 | 3 | 4 | 5 | 6 |
| 1 | 26 | 16 | 20 | 17 | 25 | 15 | 182 |
| 2 | 24 | **14 | 121** | 22 | 19 | 21 | 27 | 127-121=6 |
| 3 | 15 | 19 | 25 | 22 | 16 | 25 | 164 |
| 4 | 18 | 24 | 24 | 17 | 27 | 16 | 173 |
| 5 | 17 | 25 | 23 | 14 | 24 | 18 | 114 |
| 6 | 0 | 0 | 0 | 0 | 0 | 0 | 156 |
| Потребность | 157 | 121-121=0 | 145 | 207 | 109 | 177 |  |

1.2. Искомый элемент равен с54 = 14. x54 = min(207, 114) = 114.

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| Поставщик | Потребитель | | | | | | Запас |
| 1 | 2 | 3 | 4 | 5 | 6 |
| 1 | 26 | 16 | 20 | 17 | 25 | 15 | 182 |
| 2 | 24 | 14 | 121 | 22 | 19 | 21 | 27 | 6 |
| 3 | 15 | 19 | 25 | 22 | 16 | 25 | 164 |
| 4 | 18 | 24 | 24 | 17 | 27 | 16 | 173 |
| 5 | 17 | 25 | 23 | **14 | 114** | 24 | 18 | 114-114=0 |
| 6 | 0 | 0 | 0 | 0 | 0 | 0 | 156 |
| Потребность | 157 | 0 | 145 | 207-114=93 | 109 | 177 |  |

1.3. Искомый элемент равен с16 = 15. x16 = min(177, 182) = 177.

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| Поставщик | Потребитель | | | | | | Запас |
| 1 | 2 | 3 | 4 | 5 | 6 |
| 1 | 26 | 16 | 20 | 17 | 25 | **15 | 177** | 182-177=5 |
| 2 | 24 | 14 | 121 | 22 | 19 | 21 | 27 | 6 |
| 3 | 15 | 19 | 25 | 22 | 16 | 25 | 164 |
| 4 | 18 | 24 | 24 | 17 | 27 | 16 | 173 |
| 5 | 17 | 25 | 23 | 14 | 114 | 24 | 18 | 0 |
| 6 | 0 | 0 | 0 | 0 | 0 | 0 | 156 |
| Потребность | 157 | 0 | 145 | 93 | 109 | 177-177=0 |  |

1.4. Искомый элемент равен с31 = 15. x31 = min(157, 164) = 157.

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| Поставщик | Потребитель | | | | | | Запас |
| 1 | 2 | 3 | 4 | 5 | 6 |
| 1 | 26 | 16 | 20 | 17 | 25 | 15 | 177 | 5 |
| 2 | 24 | 14 | 121 | 22 | 19 | 21 | 27 | 6 |
| 3 | **15 | 157** | 19 | 25 | 22 | 16 | 25 | 164-157=7 |
| 4 | 18 | 24 | 24 | 17 | 27 | 16 | 173 |
| 5 | 17 | 25 | 23 | 14 | 114 | 24 | 18 | 0 |
| 6 | 0 | 0 | 0 | 0 | 0 | 0 | 156 |
| Потребность | 157-157=0 | 0 | 145 | 93 | 109 | 0 |  |

1.5. Искомый элемент равен с35 = 16. x35 = min(109, 7) = 7.

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| Поставщик | Потребитель | | | | | | Запас |
| 1 | 2 | 3 | 4 | 5 | 6 |
| 1 | 26 | 16 | 20 | 17 | 25 | 15 | 177 | 5 |
| 2 | 24 | 14 | 121 | 22 | 19 | 21 | 27 | 6 |
| 3 | 15 | 157 | 19 | 25 | 22 | **16 | 7** | 25 | 7-7=0 |
| 4 | 18 | 24 | 24 | 17 | 27 | 16 | 173 |
| 5 | 17 | 25 | 23 | 14 | 114 | 24 | 18 | 0 |
| 6 | 0 | 0 | 0 | 0 | 0 | 0 | 156 |
| Потребность | 0 | 0 | 145 | 93 | 109-7=102 | 0 |  |

1.6. Искомый элемент равен с14 = 17. x14 = min(17, 5) = 5.

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| Поставщик | Потребитель | | | | | | Запас |
| 1 | 2 | 3 | 4 | 5 | 6 |
| 1 | 26 | 16 | 20 | **17 | 5** | 25 | 15 | 177 | 5-5=0 |
| 2 | 24 | 14 | 121 | 22 | 19 | 21 | 27 | 6 |
| 3 | 15 | 157 | 19 | 25 | 22 | 16 | 7 | 25 | 0 |
| 4 | 18 | 24 | 24 | 17 | 27 | 16 | 173 |
| 5 | 17 | 25 | 23 | 14 | 114 | 24 | 18 | 0 |
| 6 | 0 | 0 | 0 | 0 | 0 | 0 | 156 |
| Потребность | 0 | 0 | 145 | 93-5=88 | 102 | 0 |  |

1.7. Искомый элемент равен с44 = 17. x44 = min(88, 173) = 88.

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| Поставщик | Потребитель | | | | | | Запас |
| 1 | 2 | 3 | 4 | 5 | 6 |
| 1 | 26 | 16 | 20 | 17 | 5 | 25 | 15 | 177 | 0 |
| 2 | 24 | 14 | 121 | 22 | 19 | 21 | 27 | 6 |
| 3 | 15 | 157 | 19 | 25 | 22 | 16 | 7 | 25 | 0 |
| 4 | 18 | 24 | 24 | **17 | 88** | 27 | 16 | 173-88=85 |
| 5 | 17 | 25 | 23 | 14 | 114 | 24 | 18 | 0 |
| 6 | 0 | 0 | 0 | 0 | 0 | 0 | 156 |
| Потребность | 0 | 0 | 145 | 88-88=0 | 102 | 0 |  |

1.8. Искомый элемент равен с25 = 21. x25 = min(102, 6) = 6.

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| Поставщик | Потребитель | | | | | | Запас |
| 1 | 2 | 3 | 4 | 5 | 6 |
| 1 | 26 | 16 | 20 | 17 | 5 | 25 | 15 | 177 | 0 |
| 2 | 24 | 14 | 121 | 22 | 19 | **21 | 6** | 27 | 6-6=0 |
| 3 | 15 | 157 | 19 | 25 | 22 | 16 | 7 | 25 | 0 |
| 4 | 18 | 24 | 24 | 17 | 88 | 27 | 16 | 85 |
| 5 | 17 | 25 | 23 | 14 | 114 | 24 | 18 | 0 |
| 6 | 0 | 0 | 0 | 0 | 0 | 0 | 156 |
| Потребность | 0 | 0 | 145 | 0 | 102-6=96 | 0 |  |

1.9. Искомый элемент равен с43 = 24. x43 = min(145, 85) = 85.

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| Поставщик | Потребитель | | | | | | Запас |
| 1 | 2 | 3 | 4 | 5 | 6 |
| 1 | 26 | 16 | 20 | 17 | 5 | 25 | 15 | 177 | 0 |
| 2 | 24 | 14 | 121 | 22 | 19 | 21 | 6 | 27 | 0 |
| 3 | 15 | 157 | 19 | 25 | 22 | 16 | 7 | 25 | 0 |
| 4 | 18 | 24 | **24 | 85** | 17 | 88 | 27 | 16 | 85-85=0 |
| 5 | 17 | 25 | 23 | 14 | 114 | 24 | 18 | 0 |
| 6 | 0 | 0 | 0 | 0 | 0 | 0 | 156 |
| Потребность | 0 | 0 | 145-85=60 | 0 | 96 | 0 |  |

1.10. Искомый элемент равен с63 = 0. x63 = min(60, 156) = 60.

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| Поставщик | Потребитель | | | | | | Запас |
| 1 | 2 | 3 | 4 | 5 | 6 |
| 1 | 26 | 16 | 20 | 17 | 5 | 25 | 15 | 177 | 0 |
| 2 | 24 | 14 | 121 | 22 | 19 | 21 | 6 | 27 | 0 |
| 3 | 15 | 157 | 19 | 25 | 22 | 16 | 7 | 25 | 0 |
| 4 | 18 | 24 | 24 | 85 | 17 | 88 | 27 | 16 | 0 |
| 5 | 17 | 25 | 23 | 14 | 114 | 24 | 18 | 0 |
| 6 | 0 | 0 | **0 | 60** | 0 | 0 | 0 | 156-60=96 |
| Потребность | 0 | 0 | 60-60=0 | 0 | 96 | 0 |  |

1.11. Искомый элемент равен с65 = 0. x65 = min(96, 96) = 96.

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| Поставщик | Потребитель | | | | | | Запас |
| 1 | 2 | 3 | 4 | 5 | 6 |
| 1 | 26 | 16 | 20 | 17 | 5 | 25 | 15 | 177 | 0 |
| 2 | 24 | 14 | 121 | 22 | 19 | 21 | 6 | 27 | 0 |
| 3 | 15 | 157 | 19 | 25 | 22 | 16 | 7 | 25 | 0 |
| 4 | 18 | 24 | 24 | 85 | 17 | 88 | 27 | 16 | 0 |
| 5 | 17 | 25 | 23 | 14 | 114 | 24 | 18 | 0 |
| 6 | 0 | 0 | 0 | 60 | 0 | **0 | 96** | 0 | 96-96=0 |
| Потребность | 0 | 0 | 0 | 0 | 96-96=0 | 0 |  |

1.12. Опорный план.

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.

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| Поставщик | Потребитель | | | | | | Запас |
| 1 | 2 | 3 | 4 | 5 | 6 |
| 1 | 26 | 16 | 20 | 17 | 5 | 25 | 15 | 177 | 0 |
| 2 | 24 | 14 | 121 | 22 | 19 | 21 | 6 | 27 | 0 |
| 3 | 15 | 157 | 19 | 25 | 22 | 16 | 7 | 25 | 0 |
| 4 | 18 | 24 | 24 | 85 | 17 | 88 | 27 | 16 | 0 |
| 5 | 17 | 25 | 23 | 14 | 114 | 24 | 18 | 0 |
| 6 | 0 | 0 | 0 | 60 | 0 | 0 | 96 | 0 | 0 |
| Потребность | 0 | 0 | 0 | 0 | 0 | 0 |  |

Подсчитаем число базисных переменных. Их должно быть m + n - 1 = 6 + 6 - 1 = 11.

Значение целевой функции для этого опорного плана равно:

F(x) = 17\*5 + 15\*177 + 14\*121 + 21\*6 + 15\*157 + 16\*7 + 24\*85 + 17\*88 + 14\*114 + 0\*60 + 0\*96 = 12159.

*2. 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.

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| Поставщик | Потребитель | | | | | | U |
| b1 | b2 | b3 | b4 | b5 | b6 |
| a1 | 26 | 16 (1) | 20 (4) | 17 | 5 | 25 | 15 | 177 | u1=0 |
| a2 | 24 | 14 | 121 | 22 | 19 | 21 | 6 | 27 | u2=-3 |
| a3 | 15 | 157 | 19 | 25 | 22 | 16 | 7 | 25 | u3=-8 |
| a4 | 18 (5) | 24 | 24 | 85 | 17 | 88 | 27 | 16 | u4=0 |
| a5 | 17 (3) | 25 | 23 | 14 | 114 | 24 | 18 | u5=-3 |
| a6 | 0 | 0 | 0 | 60 | 0 | 0 | 96 | 0 | u6=-24 |
| V | v1=23 | v2=17 | v3=24 | v4=17 | v5=24 | v6=15 |  |

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

(1;2): ∆12 = 0 + 17 - 16 = 1 > 0

(1;3): ∆13 = 0 + 24 - 20 = 4 > 0

(4;1): ∆41 = 0 + 23 - 18 = 5 > 0

(5;1): ∆51 = -3 + 23 - 17 = 3 > 0

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

Выбираем максимальную оценку свободной клетки (4;1): 15. Для этого в перспективную клетку (4;1) поставим знак «+», а в остальных вершинах многоугольника чередующиеся знаки «-», «+», «-».

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| Поставщик | Потребитель | | | | | | Запас |
| b1 | b2 | b3 | b4 | b5 | b6 |
| a1 | 26 | 16 | 20 | 17 | 5 | 25 | 15 | 177 | 0 |
| a2 | 24 | 14 | 121 | 22 | 19 | 21 | 6 | 27 | 0 |
| a3 | **15 | 157 [-]** | 19 | 25 | 22 | **16 | 7 [+]** | 25 | 0 |
| a4 | **18 [+]** | 24 | **24 | 85 [-]** | 17 | 88 | 27 | 16 | 0 |
| a5 | 17 | 25 | 23 | 14 | 114 | 24 | 18 | 0 |
| a6 | 0 | 0 | **0 | 60 [+]** | 0 | **0 | 96 [-]** | 0 | 0 |
| Потребность | 0 | 0 | 0 | 0 | 0 | 0 |  |

Из грузов хij стоящих в минусовых клетках, выбираем наименьшее, т.е. прибавляем 85 к объемам грузов, стоящих в плюсовых, и вычитаем 85 из xij, стоящих в минусовых клетках. В результате получим новый опорный план.

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| Поставщик | Потребитель | | | | | | Запас |
| b1 | b2 | b3 | b4 | b5 | b6 |
| a1 | 26 | 16 | 20 | 17 | 5 | 25 | 15 | 177 | 0 |
| a2 | 24 | 14 | 121 | 22 | 19 | 21 | 6 | 27 | 0 |
| a3 | **15 | 72 [-]** | 19 | 25 | 22 | **16 | 92 [+]** | 25 | 0 |
| a4 | **18 | 85 [+]** | 24 | **24 | 0 [-]** | 17 | 88 | 27 | 16 | 0 |
| a5 | 17 | 25 | 23 | 14 | 114 | 24 | 18 | 0 |
| a6 | 0 | 0 | **0 | 145 [+]** | 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.

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| Поставщик | Потребитель | | | | | | U |
| b1 | b2 | b3 | b4 | b5 | b6 |
| a1 | 26 | 16 | 20 | 17 | 5 | 25 | 15 | 177 | u1=0 |
| a2 | 24 | 14 | 121 | 22 | 19 | 21 | 6 | 27 | u2=2 |
| a3 | 15 | 72 | 19 | 25 | 22 | 16 | 92 | 25 | u3=-3 |
| a4 | 18 | 85 | 24 | 24 | 17 | 88 | 27 | 16 | u4=0 |
| a5 | 17 | 25 | 23 | 14 | 114 | 24 | 18 | u5=-3 |
| a6 | 0 | 0 | 0 | 145 | 0 | 0 | 11 | 0 | u6=-19 |
| V | v1=18 | v2=12 | v3=19 | v4=17 | v5=19 | v6=15 |  |

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

Минимальные затраты составят: F(x) = 17\*5 + 15\*177 + 14\*121 + 21\*6 + 15\*72 + 16\*92 + 18\*85 + 17\*88 + 14\*114 + 0\*145 + 0\*11 = 11734.

Небазисные переменные:

X11 = u1 + v1 - c11 = 0 + 18 - 26 = -8

X12 = u1 + v2 - c12 = 0 + 12 - 16 = -4

X13 = u1 + v3 - c13 = 0 + 19 - 20 = -1

X15 = u1 + v5 - c15 = 0 + 19 - 25 = -6

X21 = u2 + v1 - c21 = 2 + 18 - 24 = -4

X23 = u2 + v3 - c23 = 2 + 19 - 22 = -1

X24 = u2 + v4 - c24 = 2 + 17 - 19 = 0

X26 = u2 + v6 - c26 = 2 + 15 - 27 = -10

X32 = u3 + v2 - c32 = -3 + 12 - 19 = -10

X33 = u3 + v3 - c33 = -3 + 19 - 25 = -9

X34 = u3 + v4 - c34 = -3 + 17 - 22 = -8

X36 = u3 + v6 - c36 = -3 + 15 - 25 = -13

X42 = u4 + v2 - c42 = 0 + 12 - 24 = -12

X43 = u4 + v3 - c43 = 0 + 19 - 24 = -5

X45 = u4 + v5 - c45 = 0 + 19 - 27 = -8

X46 = u4 + v6 - c46 = 0 + 15 - 16 = -1

X51 = u5 + v1 - c51 = -3 + 18 - 17 = -2

X52 = u5 + v2 - c52 = -3 + 12 - 25 = -16

X53 = u5 + v3 - c53 = -3 + 19 - 23 = -7

X55 = u5 + v5 - c55 = -3 + 19 - 24 = -8

X56 = u5 + v6 - c56 = -3 + 15 - 18 = -6

X61 = u6 + v1 - c61 = -19 + 18 - 0 = -1

X62 = u6 + v2 - c62 = -19 + 12 - 0 = -7

X64 = u6 + v4 - c64 = -19 + 17 - 0 = -2

X66 = u6 + v6 - c66 = -19 + 15 - 0 = -4

*3. Анализ опорного плана.*

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| Поставщик | Потребитель | | | | | | Запас |
| b1 | b2 | b3 | b4 | b5 | b6 |
| a1 | 26 | 16 | 20 | 17 | 5 | 25 | 15 | 177 | 0 |
| a2 | 24 | 14 | 121 | 22 | 19 | 21 | 6 | 27 | 0 |
| a3 | 15 | 72 | 19 | 25 | 22 | 16 | 92 | 25 | 0 |
| a4 | 18 | 85 | 24 | 24 | 17 | 88 | 27 | 16 | 0 |
| a5 | 17 | 25 | 23 | 14 | 114 | 24 | 18 | 0 |
| a6 | 0 | 0 | 0 | 145 | 0 | 0 | 11 | 0 | 0 |
| Потребность | 0 | 0 | 0 | 0 | 0 | 0 |  |

Из 1-го склада необходимо доставить 5 ед. товара 4-му потребителю и 177 ед. товара 6-му потребителю.

Из 2-го склада необходимо доставить 121 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 92 ед. товара 5-му потребителю.

Из 4-го склада необходимо доставить 85 ед. товара 1-му потребителю и 88 ед. товара 4-му потребителю.

Из 5-го склада необходимо доставить 114 ед. товара 4-му потребителю.

Потребность 3-го потребителя остается неудовлетворенной на 145 ед.

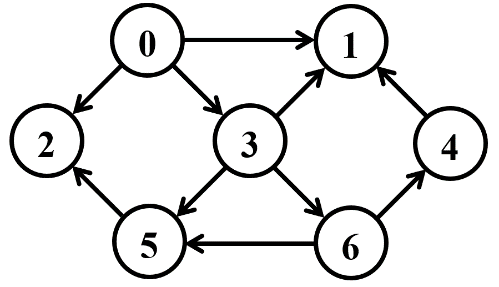
Потребность 5-го потребителя остается неудовлетворенной на 11 ед.

**Вывод:** в ходе работы были приобретены навыки решения транспортной задачи. Были составлены опорные планы с применением методов наименьшей стоимости и потенциалов, а также произведен анализ полученных опорных планов.

# 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.

**Вариант 14.**

****

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

Матрица инцидентности:

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

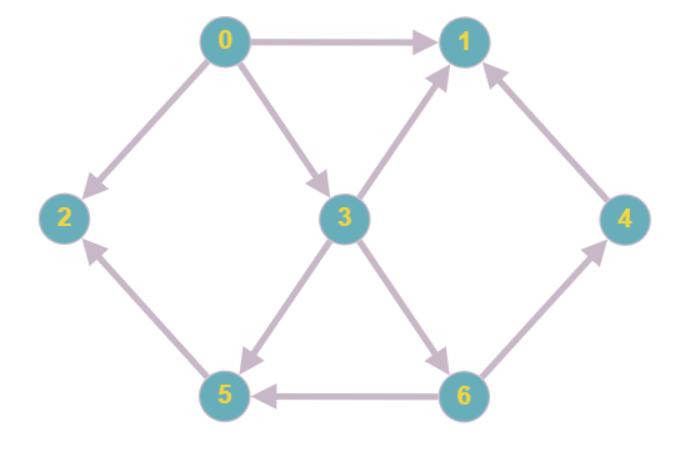
Список смежных вершин:

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

**Задание 2.**

Осуществить алгоритмы поиска в ширину и глубину, а также алгоритма топологической сортировки аналогично примерам, рассмотренным на лекциях. Оформить отчет, включив в него каждый шаг выполнения алгоритмов.

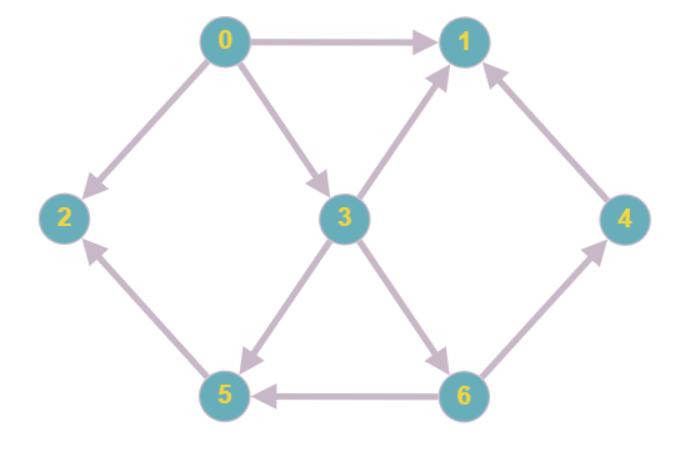
**Поиск в ширину**

****

1. Выбираем стартовую вершину (0) и добавляем ее в очередь.

Очередь: [0]

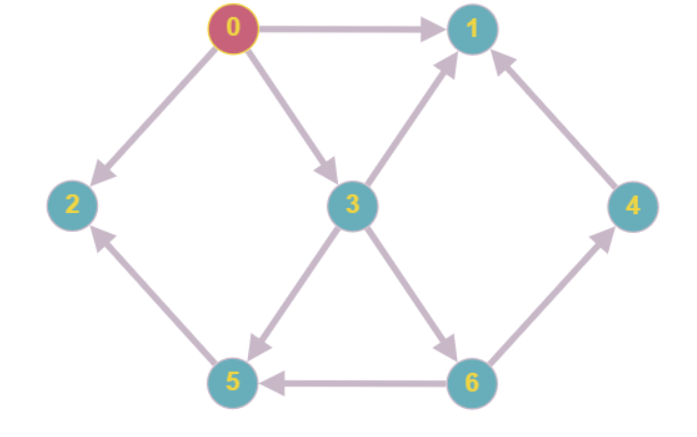
Посещенные вершины: []



1. Извлекаем из очереди вершину 0 и заносим в список посещения. У вершины 0 имеются смежные вершины – 1, 2 и 3. Добавляем их в очередь.

Очередь: [1, 2, 3]

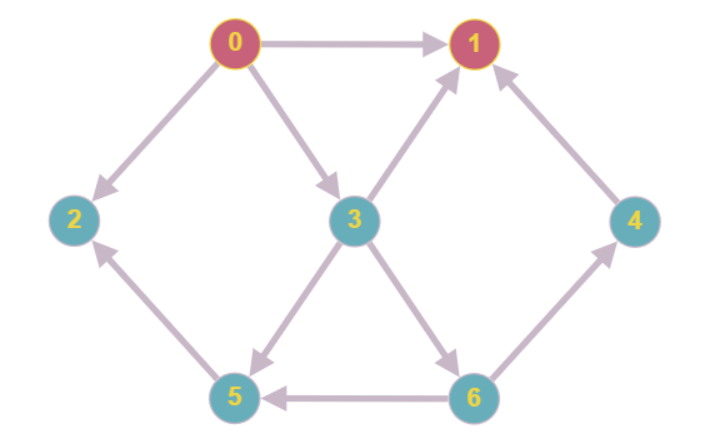
Посещенные вершины: [0]



1. Извлекаем из очереди первую вершину (1) и добавляем ее в список посещения. У вершины 1 не имеется смежных вершин.

Очередь: [2, 3]

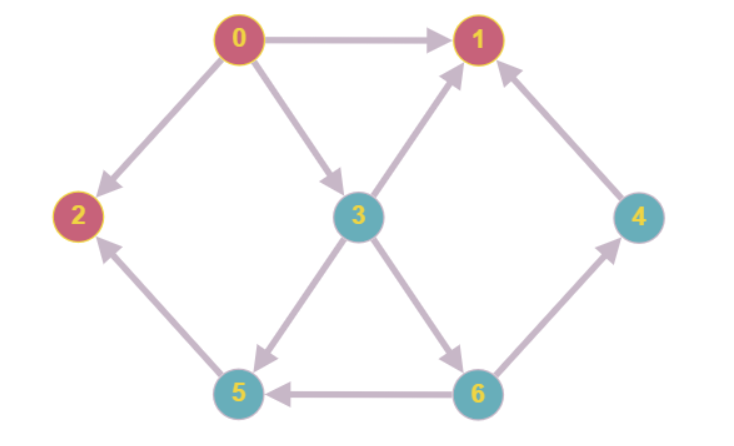
Посещенные вершины: [0, 1]



1. Извлекаем из очереди первую вершину (2) и добавляем ее в список посещения. У вершины 2 не имеется смежных вершин.

Очередь: [3]

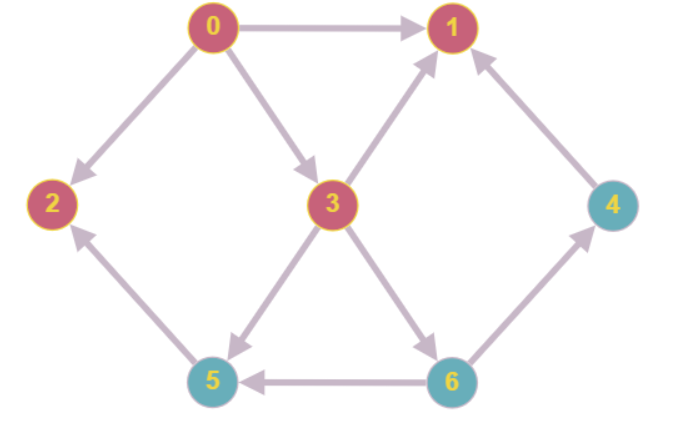
Посещенные вершины: [0, 1, 2]



1. Извлекаем из очереди первую вершину (3) и добавляем ее в список посещения. У вершины 3 имеются смежные вершины 5 и 6. Добавляем их в очередь.

Очередь: [5, 6]

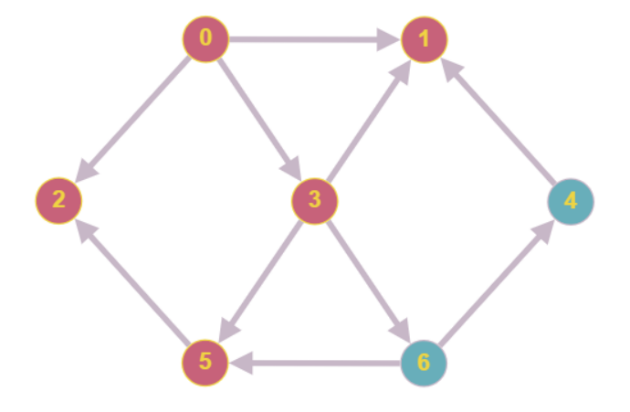
Посещенные вершины: [0, 1, 2, 3]



1. Извлекаем из очереди первую вершину (5) и добавляем ее в список посещения. У вершины 5 не имеется смежных вершин.

Очередь: [6]

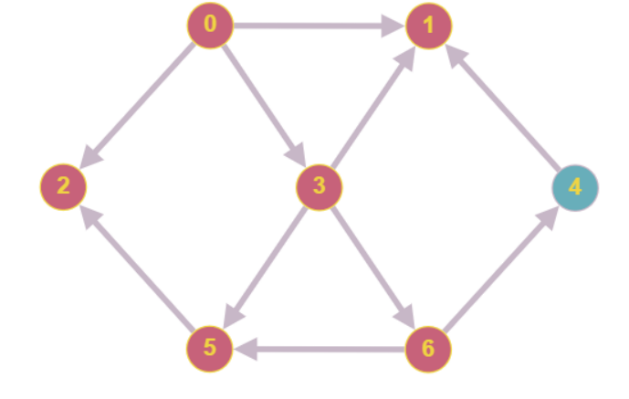
Посещенные вершины: [0, 1, 2, 3, 5]



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

Очередь: [4]

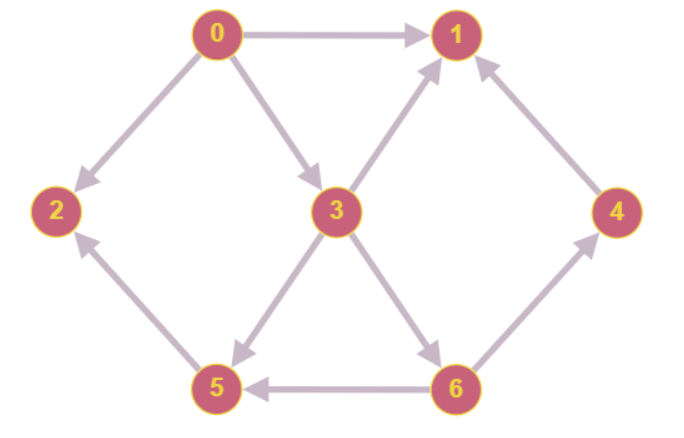
Посещенные вершины: [0, 1, 2, 3, 5, 6]



1. Извлекаем из очереди первую вершину (4) и добавляем ее в список посещения. У вершины 4 не имеется смежных вершин.

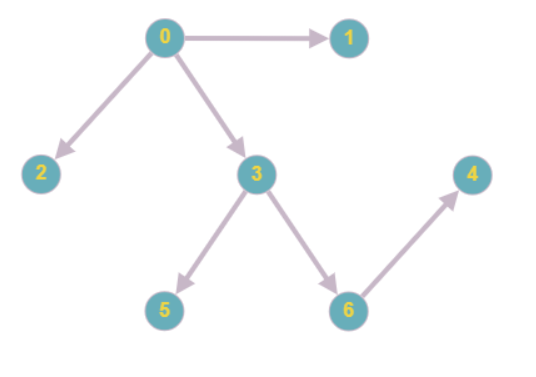
Очередь: []

Посещенные вершины: [0, 1, 2, 3, 5, 6, 4]

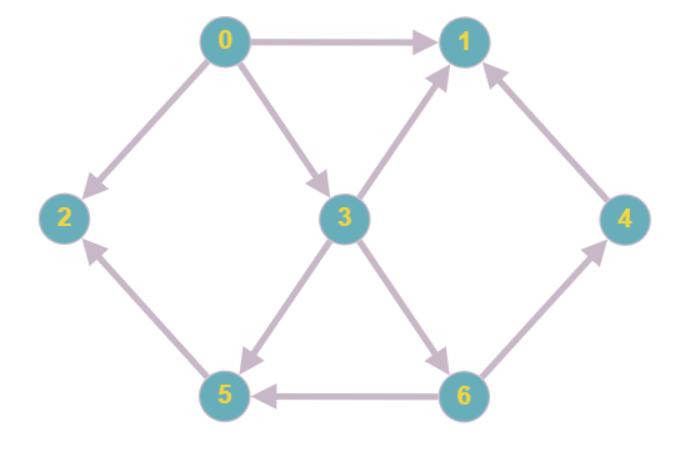


1. Очередь пуста, все вершины пройдены, конец алгоритма.

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



**Поиск в глубину**

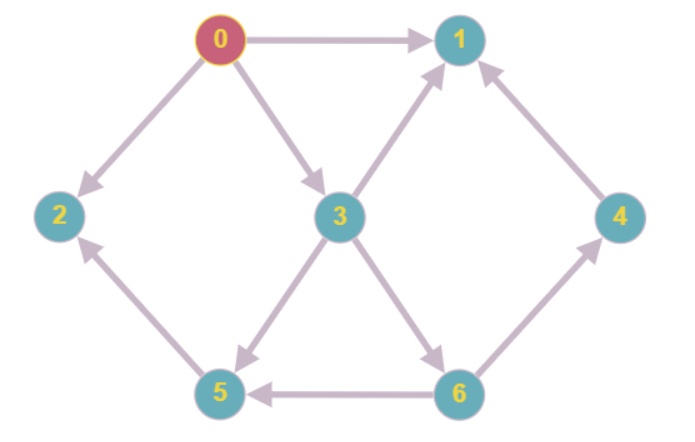


1. Создаем стек и помещаем в него начальную вершину 0. Помечаем вершину 0 как посещенную.

Стек: [0]

Посещенные вершины: {0}

Список поиска: []



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

Стек: [0, 1]

Посещенные вершины: {0}

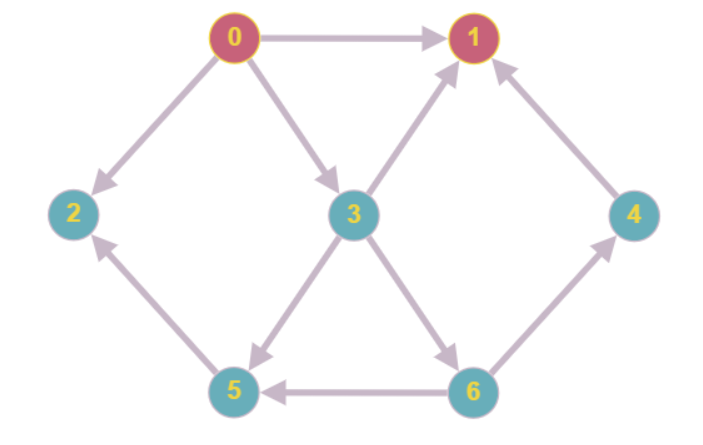
Список поиска: []

1. У вершины 1 не имеется смежных вершин. Отмечаем 1 как посещенную и удаляем из стека.

Стек: [0]

Посещенные вершины: {0, 1}

Список поиска: [1]



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

Стек: [0, 2]

Посещенные вершины: {0, 1}

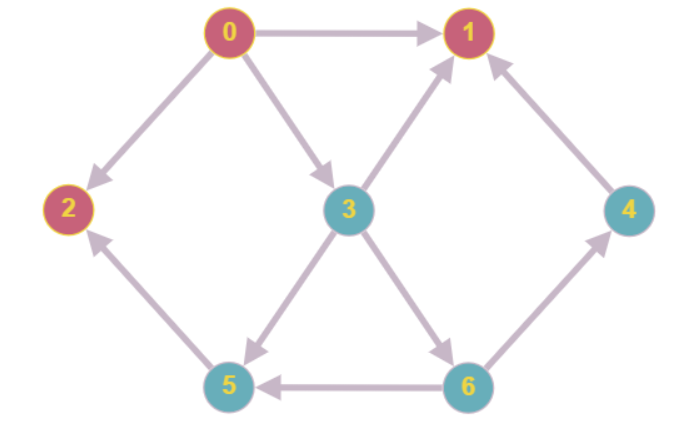
Список поиска: [1]

1. У вершины 2 не имеется смежных вершин. Отмечаем 2 как посещенную и удаляем из стека.

Стек: [0]

Посещенные вершины: {0, 1, 2}

Список поиска: [1, 2]



1. У вершины 3 имеется 1 смежная вершина – 3. Добавляем ее в стек.

Стек: [0, 3]

Посещенные вершины: {0, 1, 2}

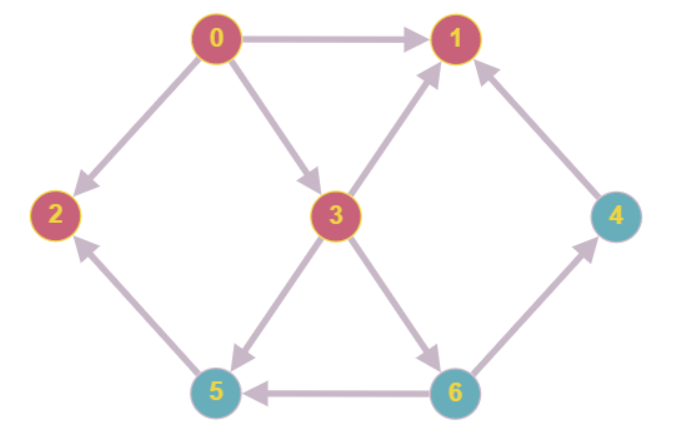
Список поиска: [1, 2]

1. У вершины 3 имеется смежная вершина 5, добавляем ее в стек, отмечаем 3 как посещенную.

Стек: [0, 3, 5]

Посещенные вершины: {0, 1, 2, 3}

Список поиска: [1, 2]

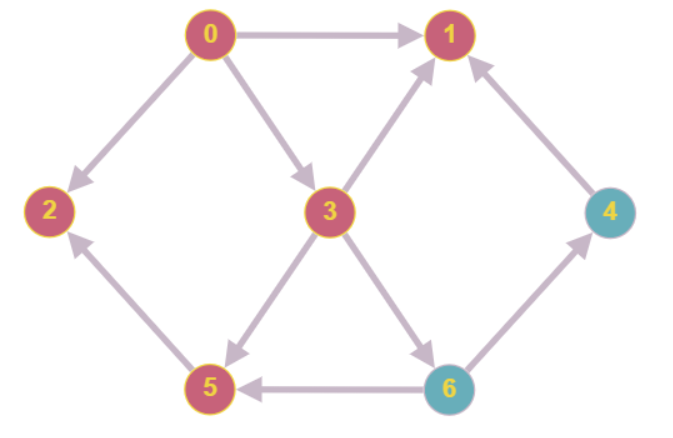


1. У вершины 5 не имеется смежных вершин. Отмечаем 5 как посещенную и удаляем из стека.

Стек: [0, 3]

Посещенные вершины: {0, 1, 2, 3, 5}

Список поиска: [1, 2, 5]



1. У вершины 3 имеется смежная вершина 6, добавляем ее в стек.

Стек: [0, 3, 6]

Посещенные вершины: {0, 1, 2, 3, 5}

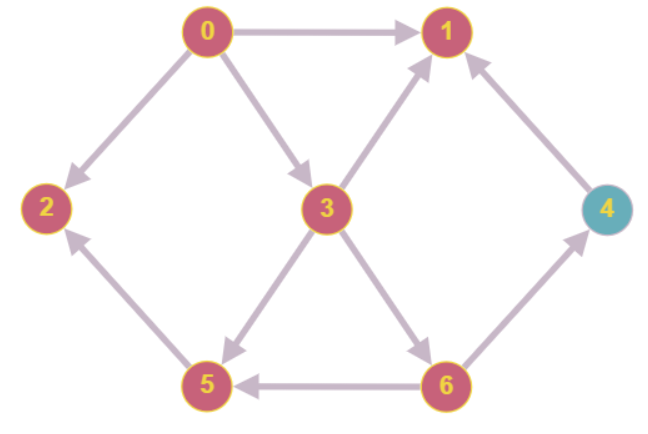
Список поиска: [1, 2, 5]

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

Стек: [0, 3, 6, 4]

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

Список поиска: [1, 2, 5]

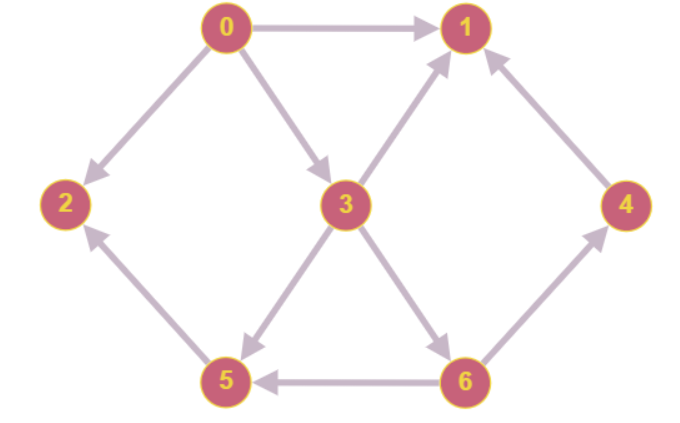


1. У вершины 4 не имеется смежных вершин. Отмечаем 4 как посещенную и удаляем из стека.

Стек: [0, 3, 6]

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

Список поиска: [1, 2, 5, 4]



1. У вершины 6 не имеется смежных вершин, удаляем из стека.

Стек: [0, 3]

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

Список поиска: [1, 2, 5, 4, 6]

1. У вершины 3 не имеется смежных вершин, удаляем из стека.

Стек: [0]

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

Список поиска: [1, 2, 5, 4, 6, 3]

1. У вершины 0 не имеется смежных вершин, удаляем из стека.

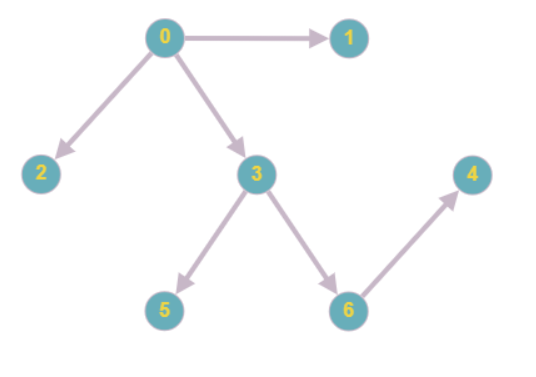
Стек: []

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

Список поиска: [1, 2, 5, 4, 6, 3, 0]

1. Стек пуст, все вершины пройдены, конец алгоритма.

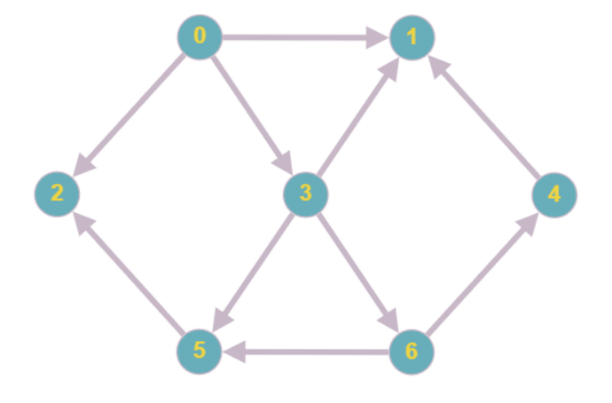
Поиск в глубину: 1 -> 2 -> 5 -> 4 -> 6 -> 3 -> 0



**Топологическая сортировка**

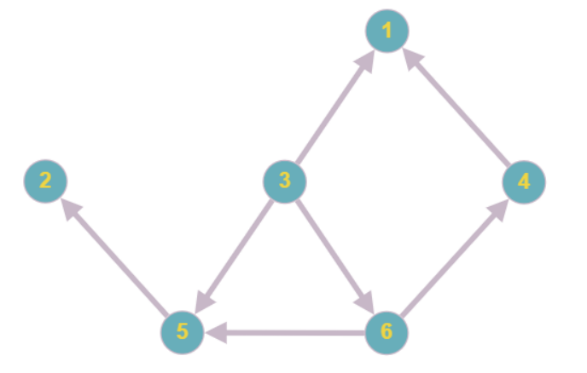
Топологическая сортировка будет выполнена с помощью метода Демукрона.

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.



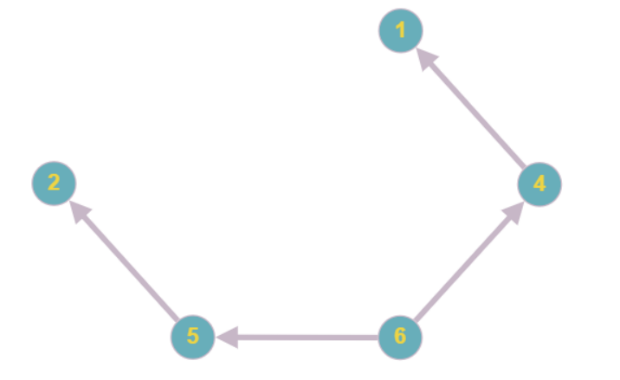
1. В исходном графе исток – нулевая вершина. Ее и вычеркиваем. Записываем эту вершину на нулевой уровень отсортированного графа.

Удаленные вершины: [0]



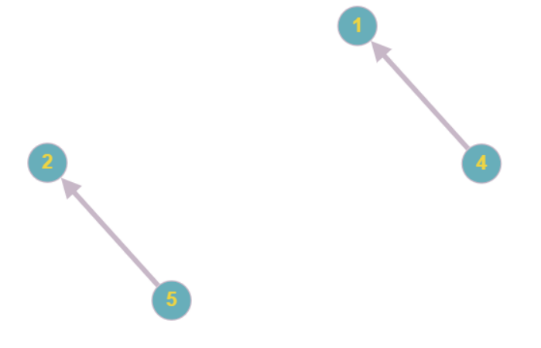
1. После этого убираем вершины без вхождений – в данном случае 3.

Удаленные вершины: [0, 3]

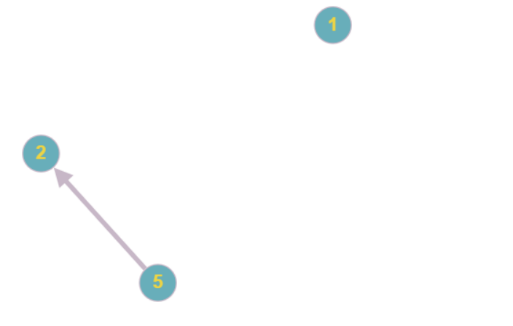


1. Проделываем все тоже самое до того момента пока не останется одна вершина.

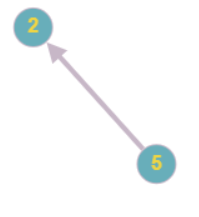
Удаленные вершины: [0, 3, 6]



Удаленные вершины: [0, 3, 6, 4]



Удаленные вершины: [0, 3, 6, 4, 1]

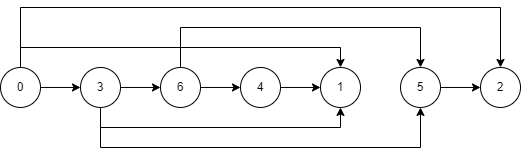


Удаленные вершины: [0, 3, 6, 4, 1, 5]



Удаленные вершины: [0, 3, 6, 4, 1, 5, 2]

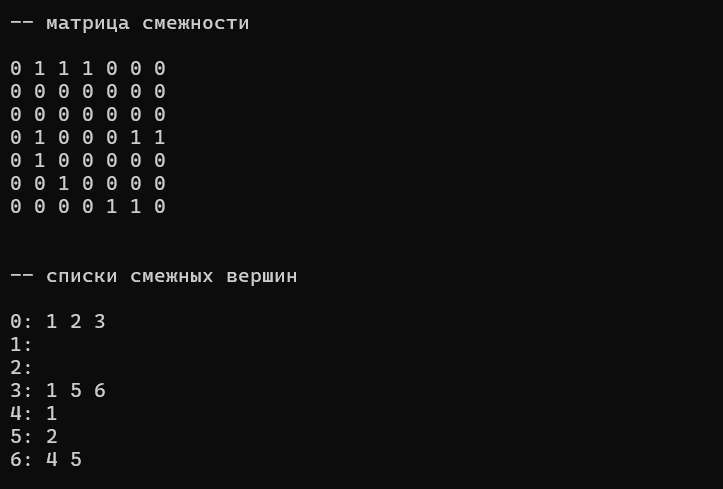
1. После исключения всех истоков получим отсортированный граф. В этом графе все вершины будут соединены точно так же, как и в исходном.

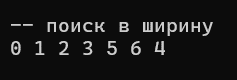


Топологическая сортировка: 0 -> 3 -> 6 -> 4 -> 1 -> 5 -> 2

**Задание 3.**

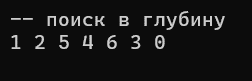
Осуществить программную реализацию алгоритмов на C++. Разработать структуры AMatrix и АList для представления ориентированного графа матричным и списковым способом. Разработать функции преобразования из одного способа представления в другой. Разработать функцию BFS обхода вершин графа, используя метод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





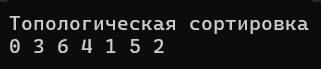
**Задание 4.**

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



**Задание 5.**

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



**Задание 6.**

По графу, соответствующему варианту составить минимальное остовное дерево по алгоритму Прима. Шаги построения отразить в отчете. Веса ребер принять:

W(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;

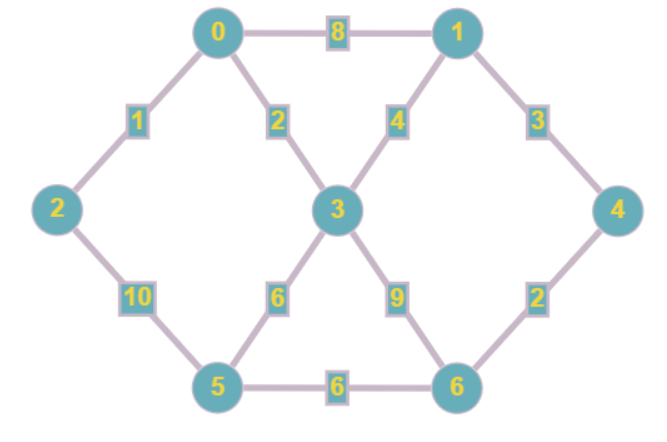
W(e4,3)=4; W(e3,4)=1;

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.

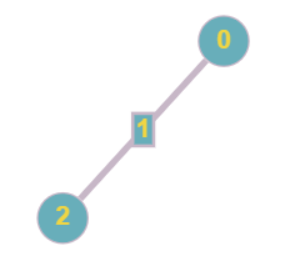


1. Выбираем стартовую вершину, пусть она будет 0. Добавляем вершину 0 в дерево.

Дерево: {0}

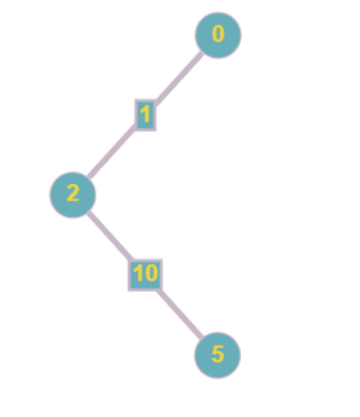
1. Доступны три ребра: (0, 1), (0, 2) и (0, 3), выбираем ребро с наименьшим весом 1 (0, 2), добавляем вершину 2 в дерево.

Дерево: {0, 2}



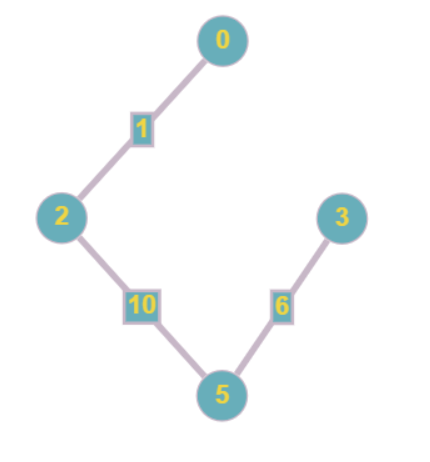
1. Из доступных ребер выбираем ребро с наименьшим весом 10 (2, 5), добавляем вершину 5 в дерево.

Дерево: {0, 2, 5}



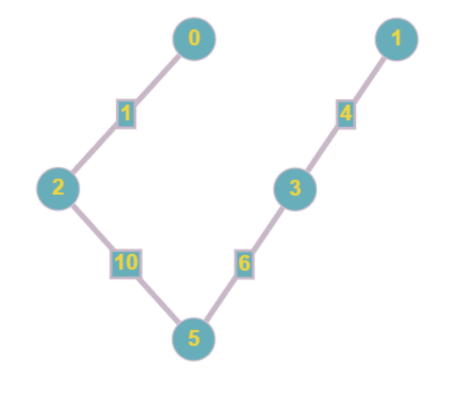
1. Из доступных ребер выбираем ребро с наименьшим весом 6 (5, 3), добавляем вершину 3 в дерево.

Дерево: {0, 2, 5, 3}



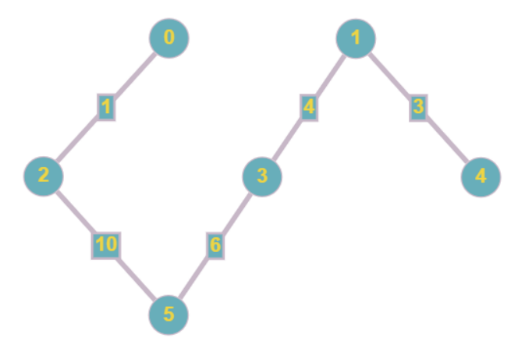
1. Из доступных ребер выбираем ребро с наименьшим весом 4 (3, 1), добавляем вершину 1 в дерево.

Дерево: {0, 2, 5, 3, 1}



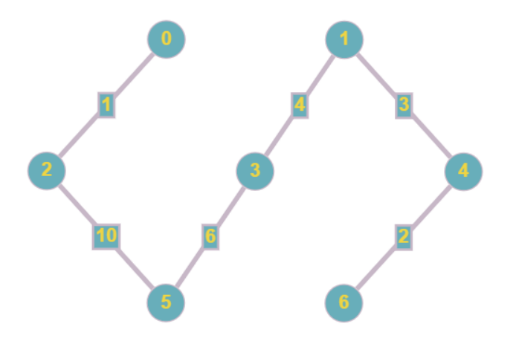
1. Из доступных ребер выбираем ребро с наименьшим весом 3 (1, 4), добавляем вершину 4 в дерево.

Дерево: {0, 2, 5, 3, 1, 4}



1. Из доступных ребер выбираем ребро с наименьшим весом 2 (4, 6), добавляем вершину 4 в дерево.

Дерево: {0, 2, 5, 3, 1, 4, 6}

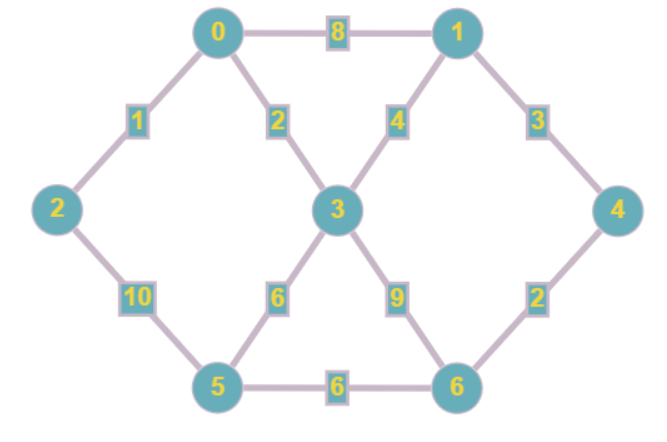


1. Все вершины пройдены, конец алгоритма.

Вес минимального остовного дерева: 26

**Задание 7.**

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



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.

Ребра, отсортированные по возрастанию веса:

0-2: 1

0-3: 2

4-6: 2

1-4: 3

1-3: 4

3-5: 6

5-6: 6

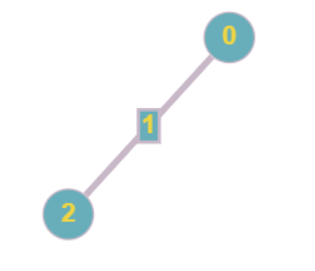
0-1: 8

3-6: 9

2-5: 10

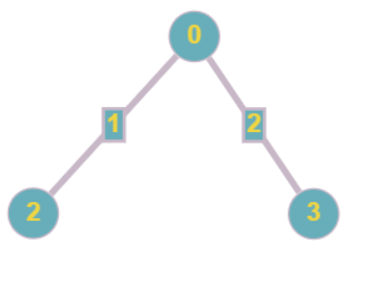
1. Выбираем вершину 0, так как к ней прилегает ребро с минимальным весом 1. Создаем дерево с корнем в вершине 0. Ребро ведет к вершине 2, записываем эту вершину как вершину первого дерева.

Дерево: {0, 2}



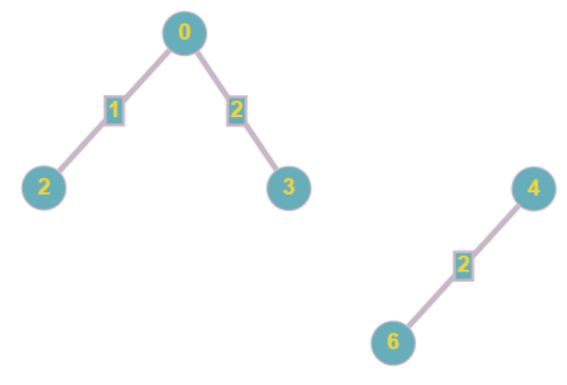
1. Выбираем следующее ребро с минимальным весом, это ребро с весом 2, которое идет между вершинами 0 и 3, в котором уже имеется дерево. Записываем эту вершину как вершину первого дерева.

Дерево: {0, 2, 3}



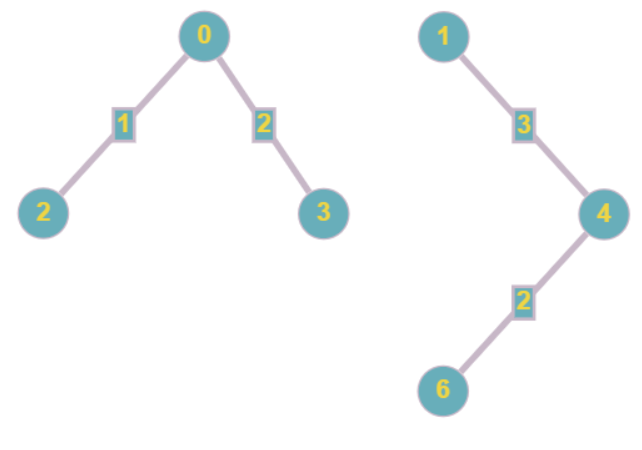
1. Выбираем следующее ребро с минимальным весом, это ребро с весом 2, которое идет между вершинами 4 и 6. Данных вершин еще нет в дереве, поэтому записываем их отдельно.

Дерево: {0, 2, 3, 4, 6}



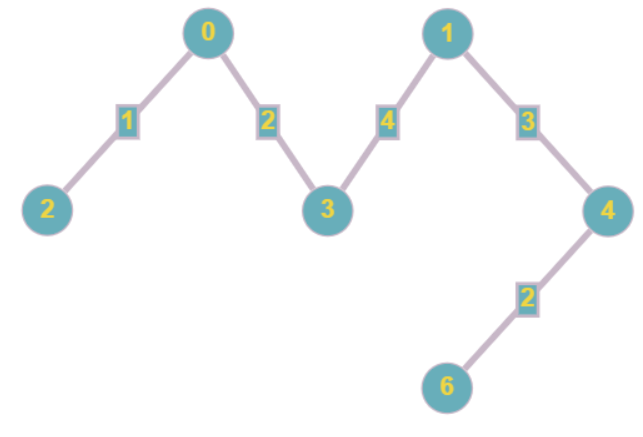
1. Выбираем следующее ребро с минимальным весом, это ребро с весом 3, которое идет между вершинами 1 и 4. Добавляем его ко второму дереву.

Дерево: {0, 2, 3, 4, 6, 1}



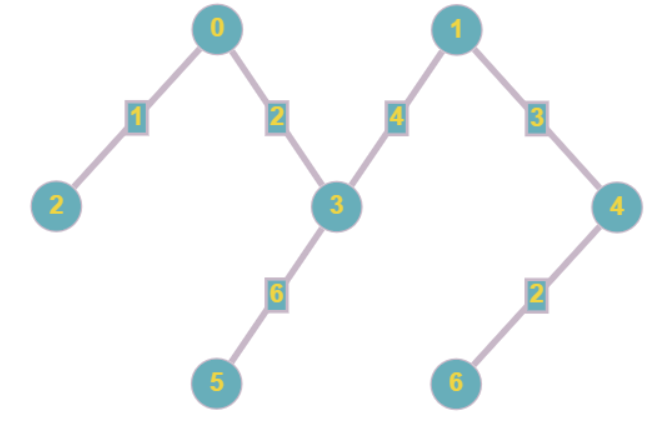
1. Выбираем следующее ребро с минимальным весом, это ребро с весом 4, которое идет между вершинами 1 и 3. Данное ребро соединяет два предыдущих дерева.

Дерево: {0, 2, 3, 4, 6, 1}



1. Выбираем следующее ребро с минимальным весом, это ребро с весом 6, которое идет между вершинами 3 и 5. Добавляем вершину 5 к дереву.

Дерево: {0, 2, 3, 4, 6, 1, 5}



1. Все вершины пройдены, конец алгоритма.

Вес минимального остовного дерева: 18

**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.