Учреждение образования

«БЕЛОРУССКИЙ ГОСУДАРСТВЕННЫЙ ТЕХНОЛОГИЧЕСКИЙ УНИВЕРСИТЕТ»

Дисциплина «Математическое программирование»

Отчёт по лабораторным работ

Студент: Коренчук А.В.

ФИТ 2 курс 2 группа

.

Минск 2023

Содержание

[Лабораторная работа 1 3](#_Toc130641285)

[Лабораторная работа 2 6](#_Toc130641286)

[Лабораторная работа 3 16](#_Toc130641287)

[Лабораторная работа 4 18](#_Toc130641288)

# Лабораторная работа 1

**Задание 1**. Разработайте три функции (start, dget и iget). Три функции будут представлены на рисунке 1.1.

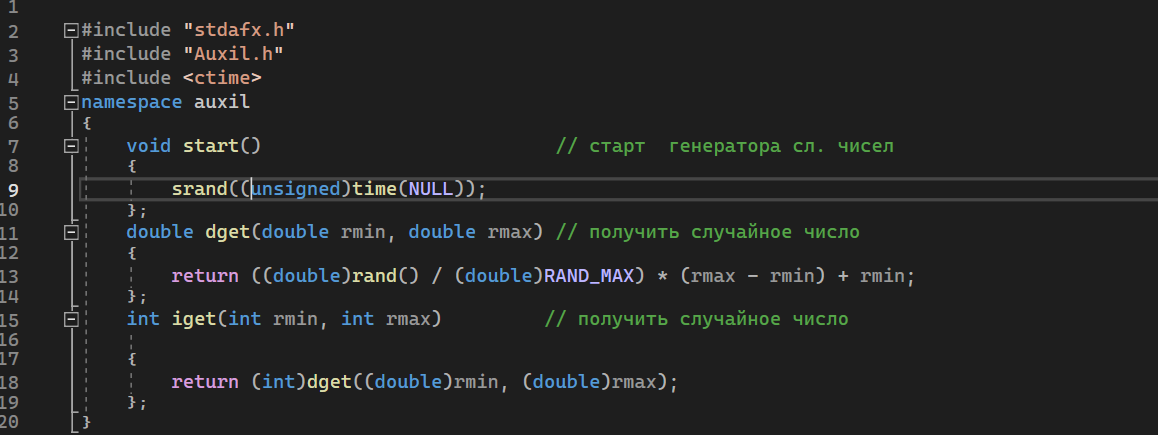


Рисунок 1.1 – Три функции

**Задание 2** будет представлена на Рисунке 1.2

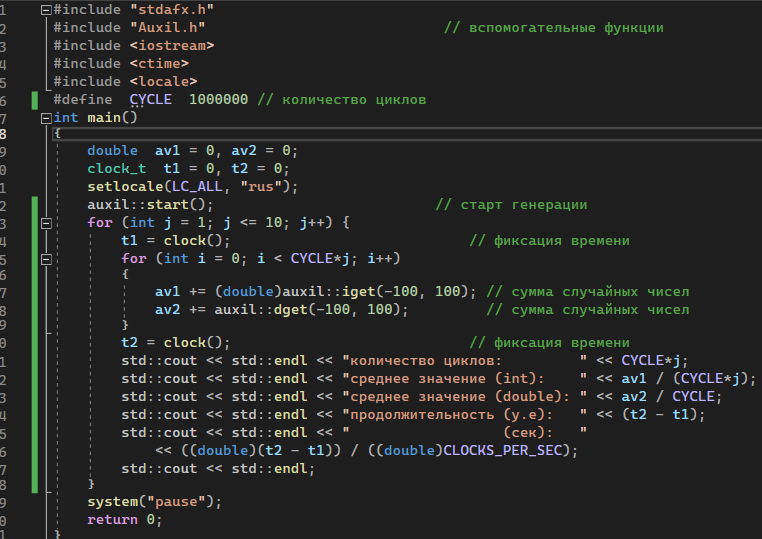


Рисунок 1.2 – Три функции

Вывод данных будет на рисунке 1.3.

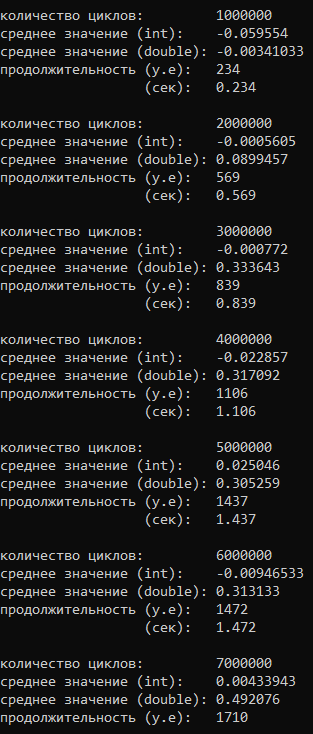


Рисунок 1.3 – Вывод данных

Результаты измерений и соответствующий график приведены ниже на рисунке 1.4.

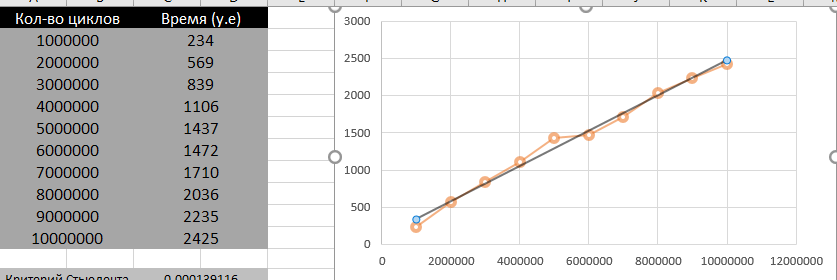


Рисунок 1.4 – Результаты измерений

Критерий согласия Стьюдента был вычислен с помощью формулы =СТЬЮДЕНТ.ТЕСТ(A2:B11;C2:D11;1;1),полученный результат приведен на рисунке 1.5.



Рисунок 1.5 – Критерий согласия Стьюдента

Коэффициент корреляции был вычислен с помощью формулы = =КОРРЕЛ(A2:B11;C2:D11) полученный результат приведен на рисунке 1.6.



Рисунок 1.6 – Коэффициент корреляции

Значение коэффициента корреляции близко к единице. Это означает, что точки корреляционного поля линейно зависимы.

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

# Лабораторная работа 2

**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.

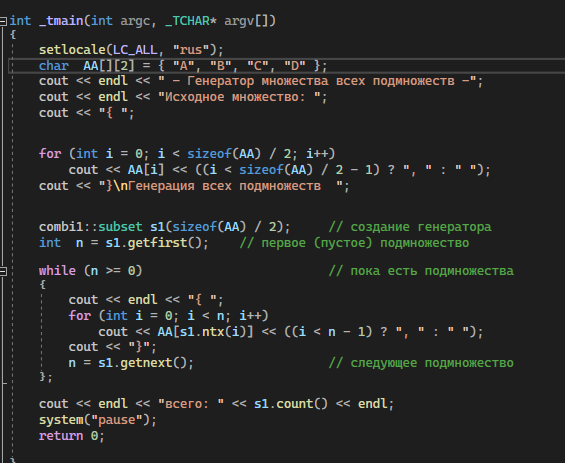


Рисунок 2.1 – Генератор подмножеств

Результат выполнения будет представлен на рисунке 2.2.

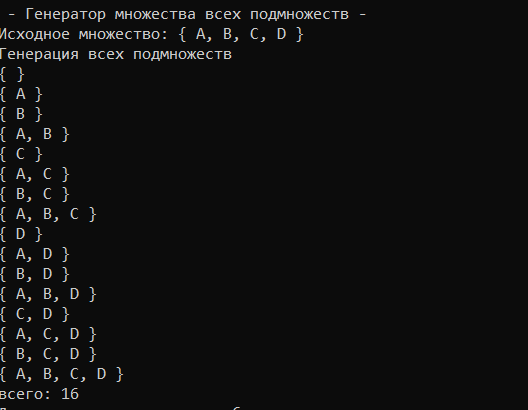


Рисунок 2.2 – Генератор подмножеств заданного множества

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.

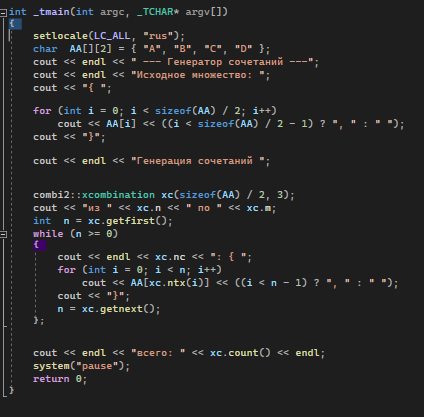


Рисунок 2.4 – Код генератора сочетаний

Результат выполнения будет представлен на рисунке 2.5.

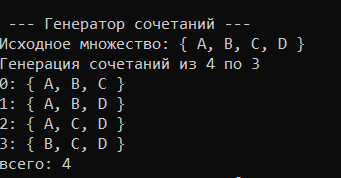


Рисунок 2.5 –Результат генератора сочетаний

Алгоритм будет представлен на рисунке 2.6

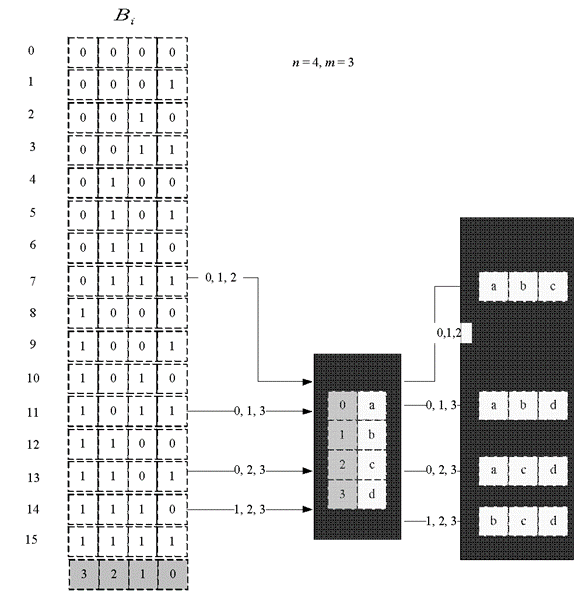


Рисунок 2.6 – Генератор сочетаний

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

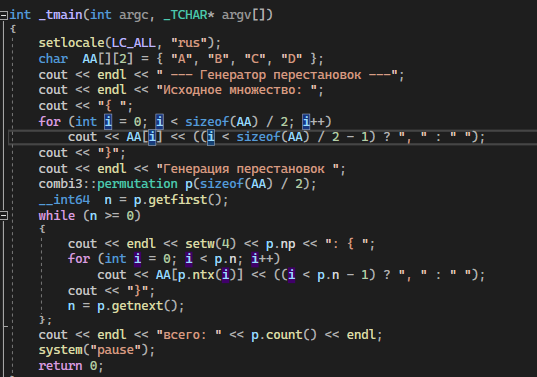


Рисунок 2.7 – Код генератора перестановок

Результат выполнения будет представлен на рисунке 2.8.

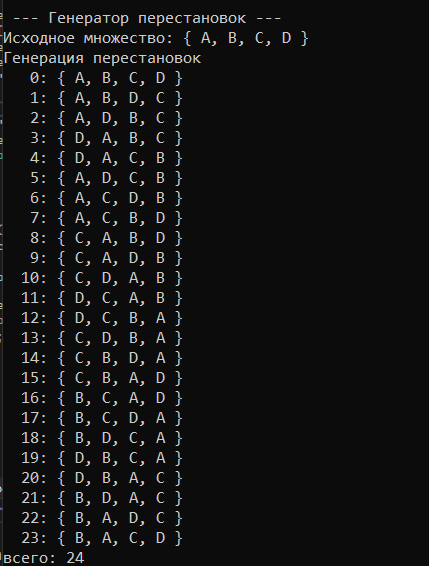


Рисунок 2.8 –Результат генератора перестановок

Алгоритм будет представлен на рисунке 2.9



Рисунок 2.9 – Генератор перестановок

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

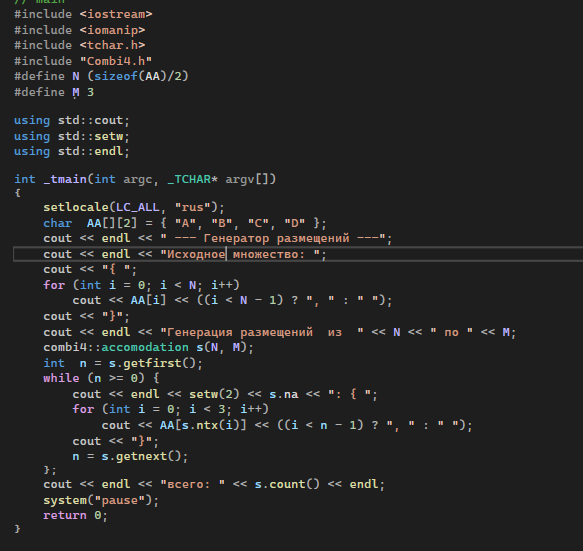


Рисунок 2.10 – Код генератора размещений

Результат выполнения будет представлен на рисунке 2.11.

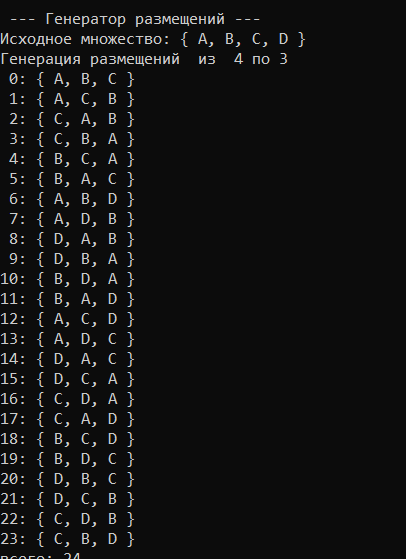


Рисунок 2.11 –Результат генератора размещений

Алгоритм будет представлен на рисунке 2.12.



Рисунок 2.12– Генератор размещений

**Задание 5:** (2, 6, 10, 14) 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.). Код этого задания будет на рисунке 2.13.

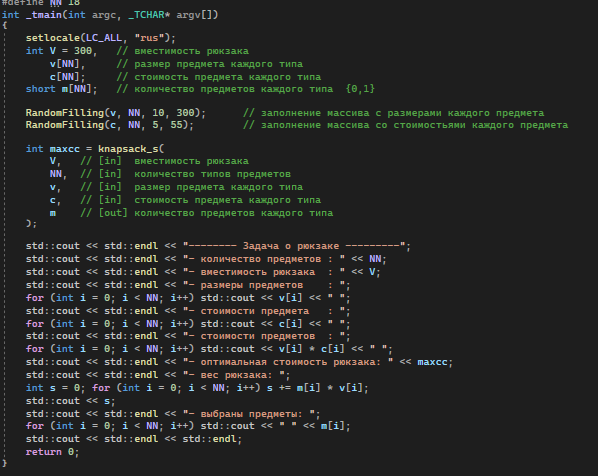


Рисунок 2.13 – Код задания 5

Результат выполнения будет представлен на рисунке 2.14.

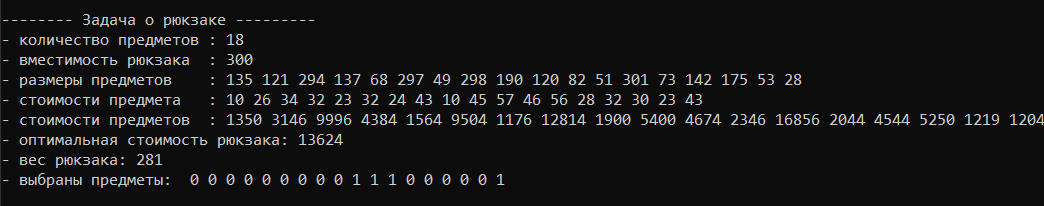


Рисунок 2.14 –Результат задания 5

**Задание 6: (**2, 6, 10, 14) упрощенную о рюкзаке (количество предметов 12 – 20 шт.). Код задания номер 6 будет представлен на рисунке 2.15.

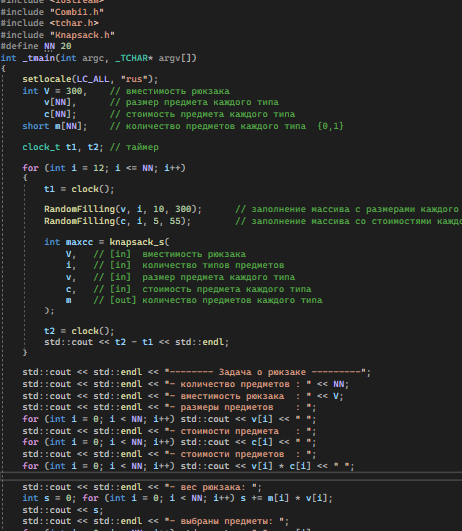


Рисунок 2.15 – Код задания 6

Результат выполнения будет представлен на рисунке 2.16.

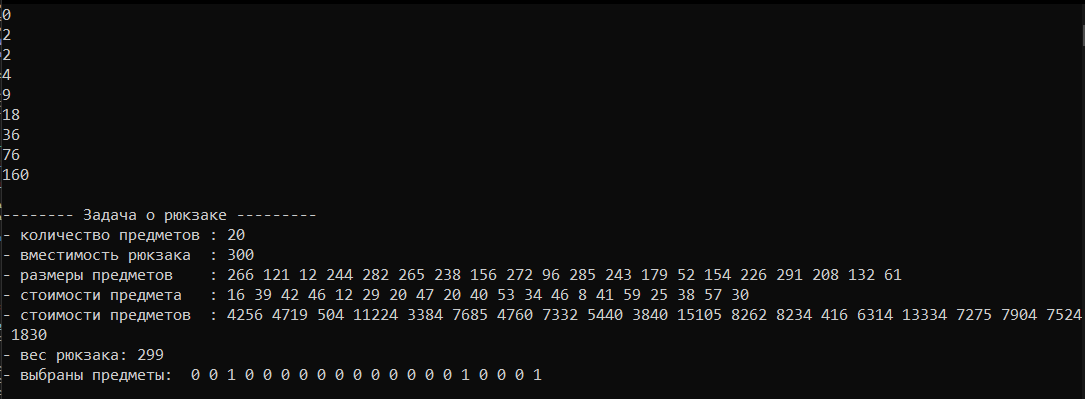


Рисунок 2.16–Результат задания 6

График будет представлен на рисунке 2.17.

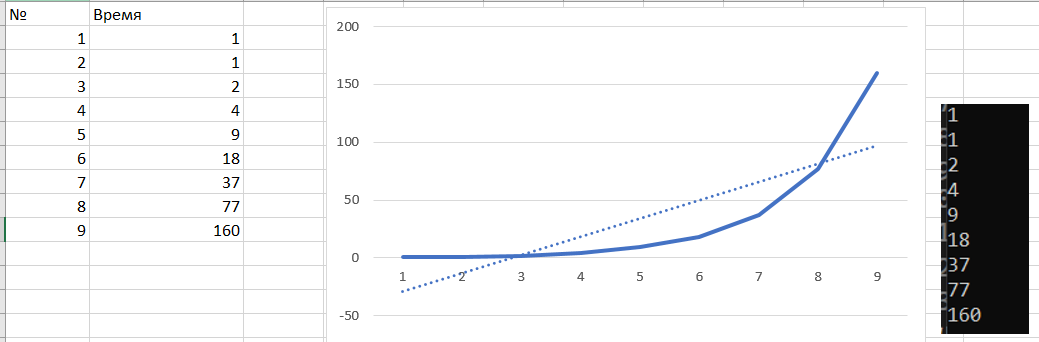


Рисунок 2.17 – График

Алгоритм будет представлен на рисунке 2.18.



Рисунок 2.18– Схема решения задачи о рюкзаке

**Вывод:** приобретены навыки разработки генераторов подмножеств, перестановок, сочетаний и размещений на С++; применены разработанные генераторы об оптимальной загрузке судна, построена зависимость.

# Лабораторная работа 3

**Условие:**

Найти оптимальный маршрут для коммивояжера, если известно, что кол-во городов равно 5, а расстояние между городами задается следующей матрицей d:

где *n* – номер варианта;

n=6.

Задачу следует решить с использованием метода ветвей и границ. Матрица будет представлена на рисунке 3.1

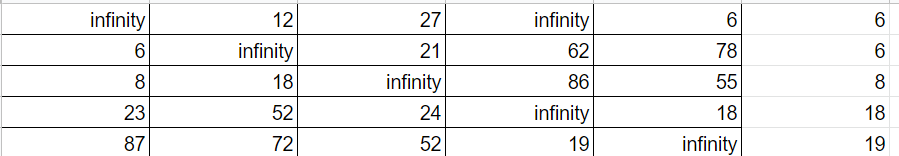
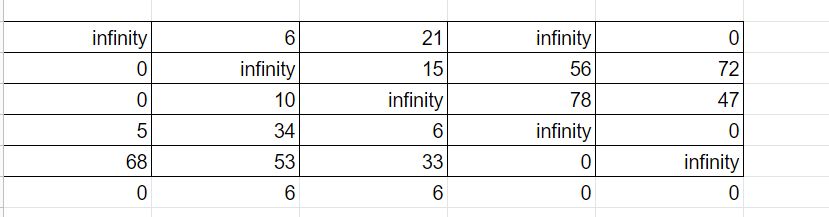
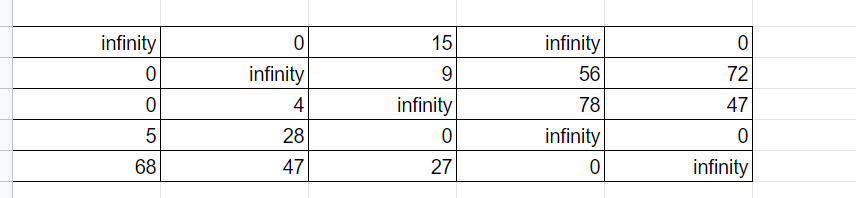


Рисунок 3.1– Матрица

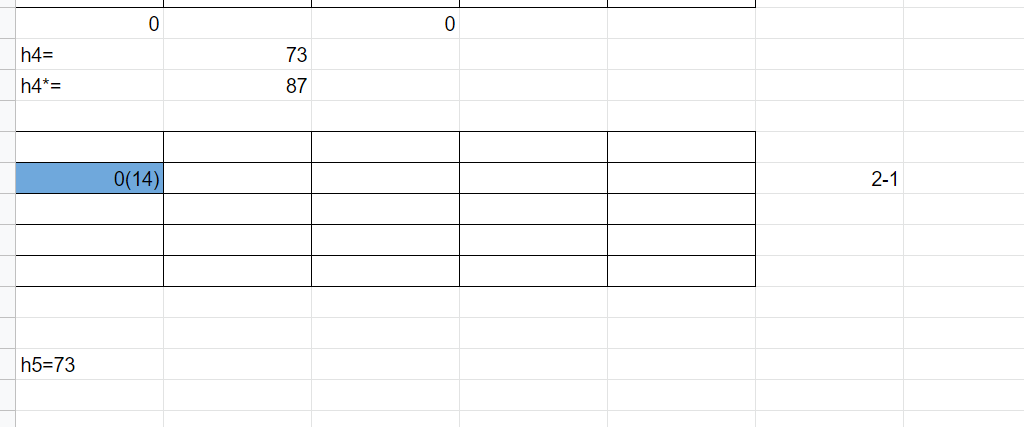
Находим минимальное значение в каждой строке и выписываем его в отдельный столбец: Производим приведение строк



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

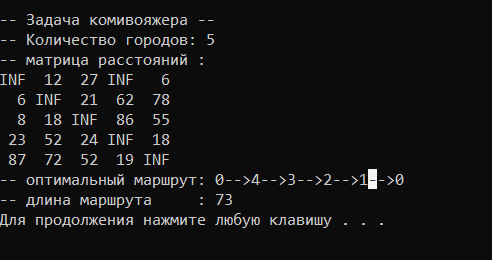


В дальнейшем для каждой нулевой клетки получившейся преобразованной матрицы находим «оценку». Полученную оценку записываем рядом с нулем, в скобках.



**Длина оптимального маршрута: φ=73**

Результат работы программы будет пред



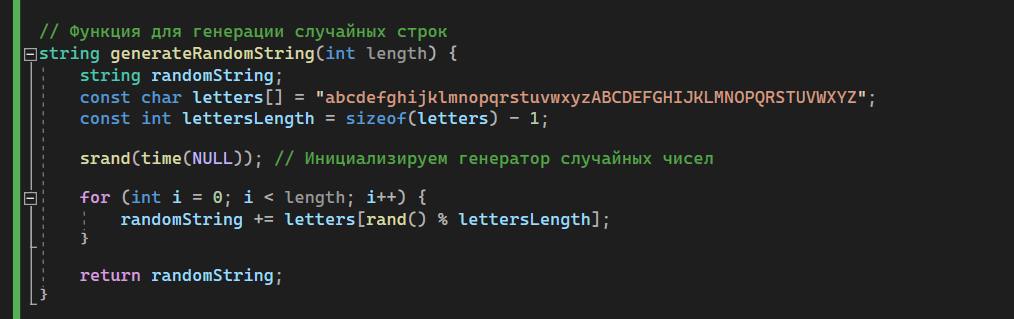
Полученные значения идентичны с результатом работы метода ветвей и границ.

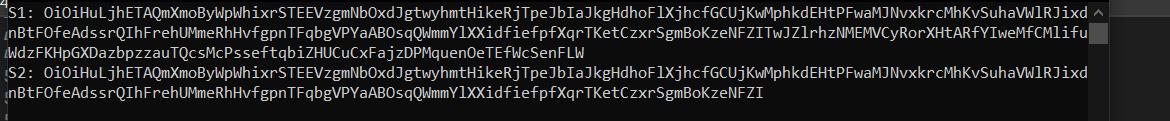
Вывод: освоил общие принципы решения задач методом ветвей и границ, решил задачу о коммивояжере данным методом, сравнил полученное решение задачи с комбинаторным методом перестановок.

# Лабораторная работа 4

***Задание 1.***

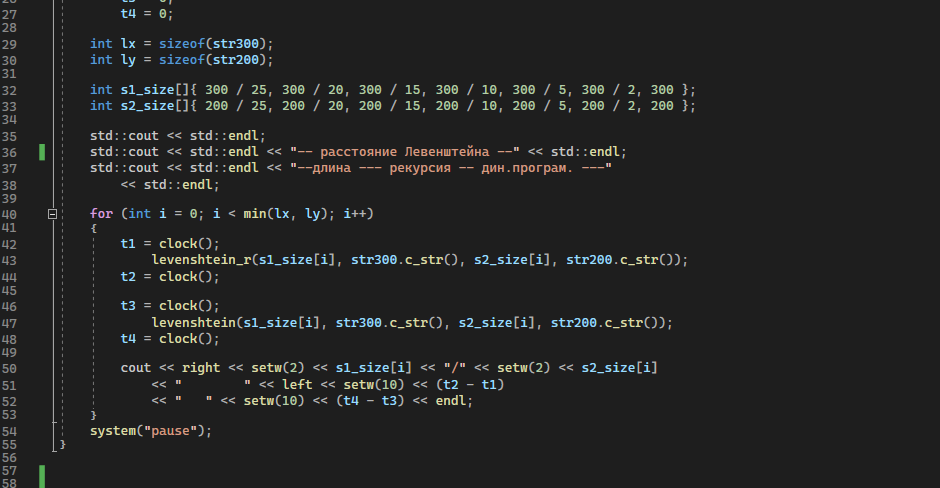
На языке С++ сгенерировать случайным образом строку букв латинского алфавита S1 длиной 300 символов и S2 длиной 200.

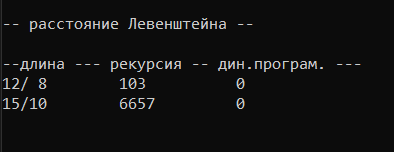




***Задание 2.***

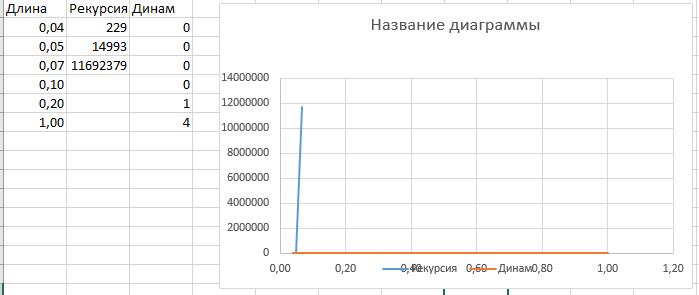
Вычислить двумя способами (рекурсивно и с помощью динамического программирования)  – дистанцию Левенштейна для , где - длина строки ,  - строка состоящая из первых  символов строки . (копии экрана и код вставить в отчет).





***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).

|  |  |  |
| --- | --- | --- |
| 6 | Сом | Домик |

1. Сом, Домик = min(Со, Домик+ 1)(Сом, Доми + 1)(Со, Доми+ 1)

2. Со, Домик= min(С, Домик+ 1)(Со, Доми + 1)(С, Доми+ 1)

3. Сом, Доми= min(Со, Доми + 1)(Сом, Дом+ 1)(Со, Дом+ 1)

4. Со, Доми = min(С, Доми + 1)

(Со, Дом+ 1)

(С, Дом + 1)

5. С, Домик = min(«», Домик) = 5

(С,Доми)

(«»,Доми) = 4

6. С, Доми= min («», Доми) = 4

(С, Дом + 1)

(«», Дом) = 3

7. Сом, Дом = min (С о, Дом) (Сом, До) (Со, До)

8. Со, Дом = min(С, Дом) (Со, До) (С, До)

9. Сом, До = min(Со, До) (Сом, Д) (Со, Д)

10. Сом, Д = min (Со, Д)

(Сом, «») = 3

(Со, «») = 2

11. С, Дом= min («» , Дом) = 3

(С, До)

(«», До) = 2

12. Со, До = min (С, До) (Со, Д) (С, Д)

13. С, До = min («», До) = 2 + 1 = 3

(С, Д) = 1 + 1 = 2 = 2

(«», Д) = 1 + 1 = 2

14.Со, Д= min (C, Д) = 1 + 1 = 2

(Со, «» ) = 2 + 1 = 3 = 2

(С, «» ) = 1 + 1 = 2

15. Со, Д = min (С, Д) = 1 + 1

(Со, «») = 2 + 1 = 2

(С, «») = 1 + 1

16. С, До = min («», ) + 1 = 3

(C, Д) + 1 = 2 = 2

(«», Д) + 1 = 2

17. Со, До = min (С, До) = 2 + 1 = 3

(Со, Д) = 2 + 1 = 3 = 2

(С, Д) = 1 + 1 = 2

18. С, Дом = min («», Дом) + 1 = 4

(С, До) = 2 + 1 = 3 = 3

(«», До) + 1 = 3

19. Сом, Д = min (Со, Д) = 2 + 1 = 3

(Сом, «») + 1 = 4 = 3

(Со, «») + 1 = 3

20. Сом, До = min(Со, До) = 2 + 1 = 3

(Сом, Д) = 3 + 1 = 4 = 3

(Со, Д) = 2 + 1 = 3

21. Со, Дом= min (С, Дом) = 3 + 1 = 4

(Со, До) = 2 + 1 = 3 = 3

(С, До) = 2 + 1 = 3

22. Сом, Дом = min (Со, Дом) = 4 + 1 = 5

(Сом, До) = 3 + 1 = 4 = 3

(Со, До) = 2 + 1 = 3

23. С, Доми = min («», Доми) + 1 = 5

(С,Доми) = 3 + 1 = 4 = 4

(«», Дом) + 1 = 4

24. С, Домик= min («», Домик) + 1 = 6

(А, Доми) = 4 + 1 = 5 = 5

(«», Доми) + 1 = 5

25. Со, Доми = min (С,Доми) = 4 + 1 = 5

(Со, Дом) = 3 + 1 = 4 = 4

(С,Дом) = 3 + 1 = 4

26. Сом, Доми = min (Со,Доми) = 3 + 1 = 4

(Сом, Дом) = 3 + 1 = 4

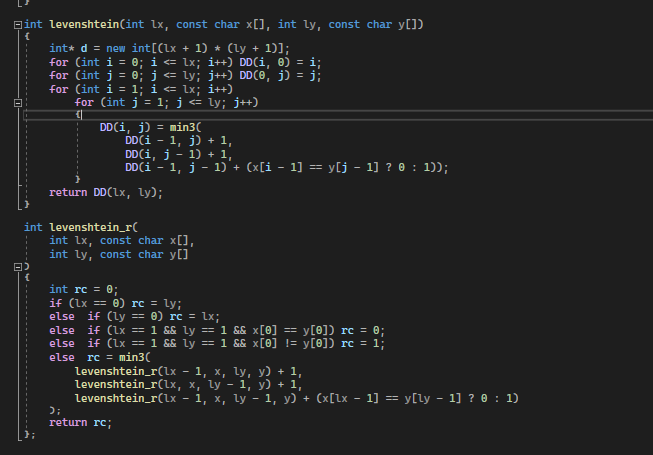
(Со, Дом) = 2 + 1 = 3

27. Со, Домик = min (С, Домик) = 5 + 1 = 6

(Со, Доми) = 3 + 1= 4

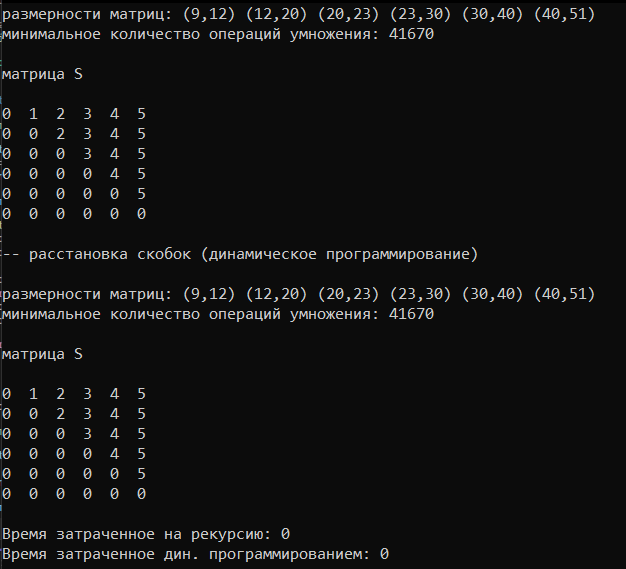
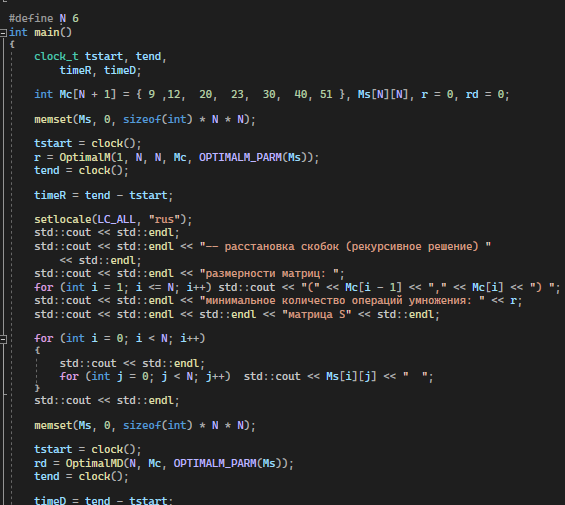
(С, Доми) = 4 + 1= 4

28. Сом, Домик = min(5, 3, 4) = 3



***Задание 5.***

**Нечетные варианты**. Выполнить сравнительный анализ времени затраченного на решение задачи о наибольшей общей подпоследовательности для двух методов решения (рекурсивное решение, динамическое программирование). Две последовательности взять в соответствии с вариантом. Построить графики зависимости времени вычисления от . **Отобразить ход решения в отчете**(по примеру из лекции) + код и копии экрана.



**Вывод:** Для решения задачи о перемножении матриц нужно умножить матрицы последовательно в определенном порядке. В данном случае имеются 6 матриц, и возможны различные порядки их перемножения.

Для определения оптимального порядка перемножения матриц используется алгоритм динамического программирования. Он заключается в том, что матрицы умножаются в определенном порядке в зависимости от их размера. Таким образом, определяется минимальное количество операций умножения, необходимое для перемножения всех матриц.

При рекурсивном методе для вычисления оптимального порядка перемножения матриц используется метод повторного вызова, который имеет экспоненциальную сложность, что означает, что он неэффективен для больших данных. В данном случае, для нахождения оптимального порядка, необходимо рассмотреть все возможные порядки перемножения матриц, что может быть очень трудоемким процессом.

При использовании алгоритма динамического программирования время выполнения сокращается до квадратичной сложности, что дает значительный выигрыш в производительности. В данном случае, оптимальный порядок перемножения матриц следующий:

# Лабораторная работа 5

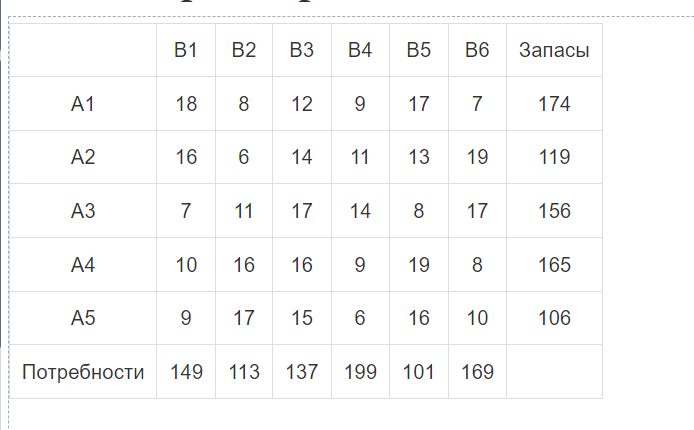
**Задание.** Решить транспортную задачу. Имеется 5 поставщиков продукции и 6 потребителей. Величина запасов, потребностей и стоимость затрат на перевозку продукции взять в соответствии с вариантом (*N*). Оформить отчет.

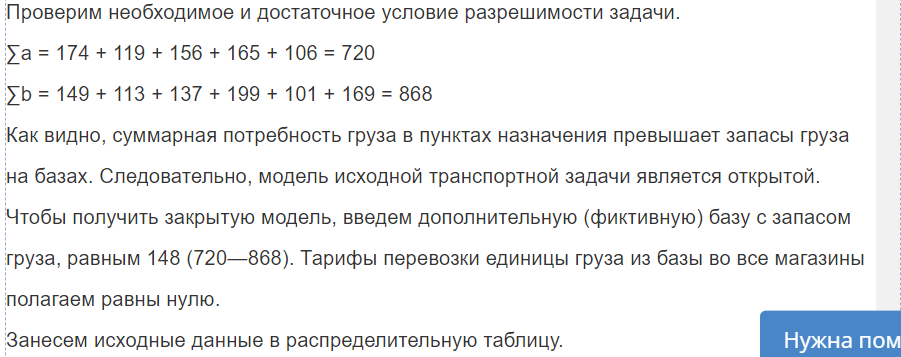
**Ход решения:**

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

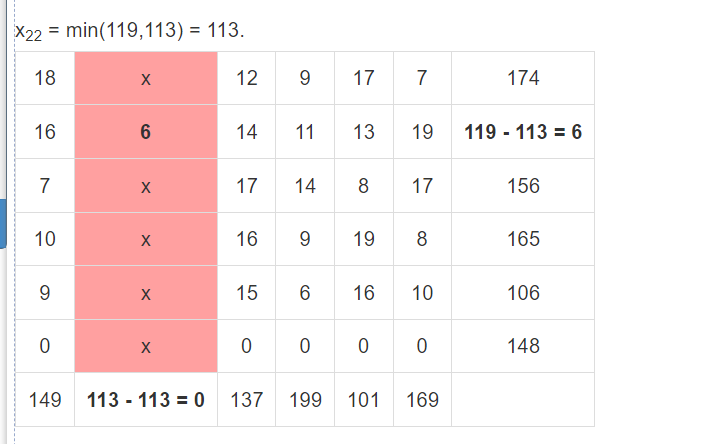
**Вопросы для защиты:**

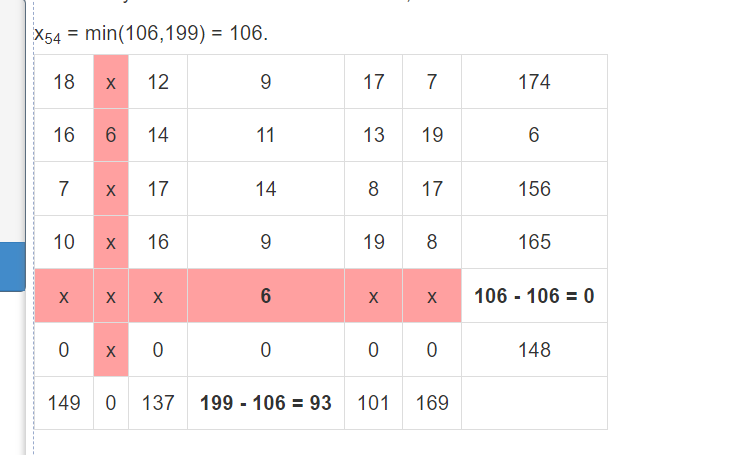
1. Что такое закрытая и открытая транспортная задача?( Метод потенциалов
2. Метод модифицированных потенциалов
3. Метод венгерского алгоритма)
4. Методы решения транспортной задачи.
5. Сфера применения решения транспортной задачи.

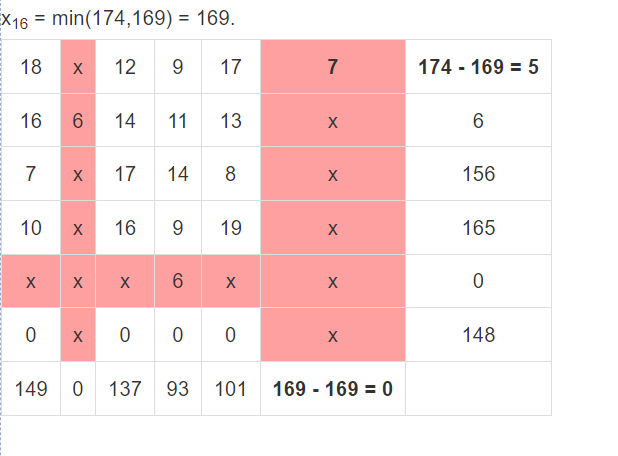


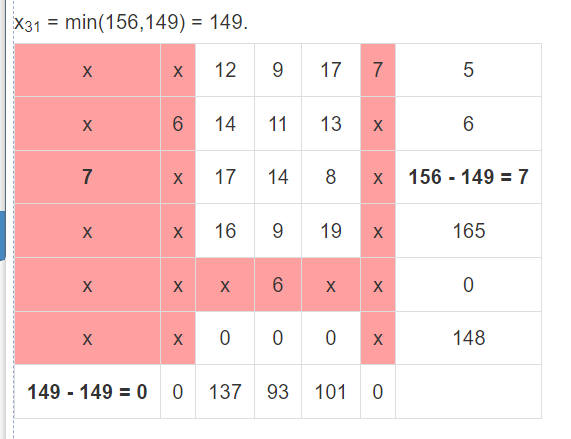


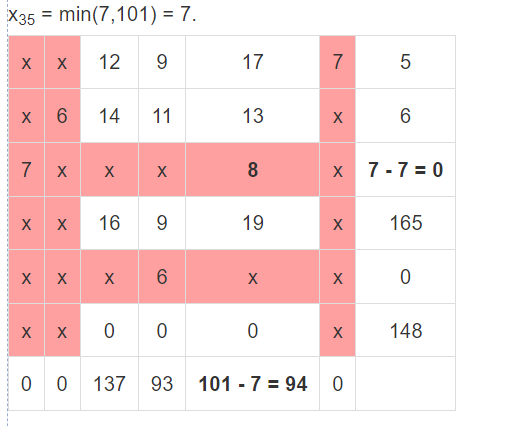


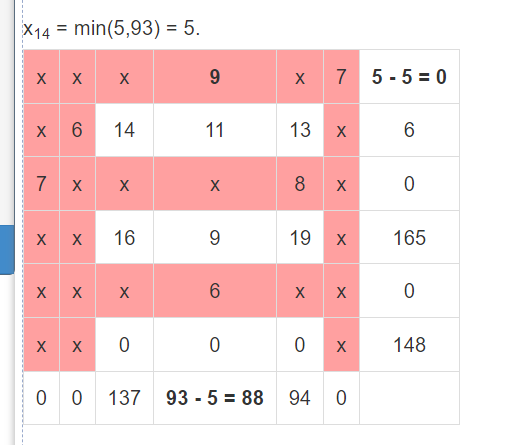


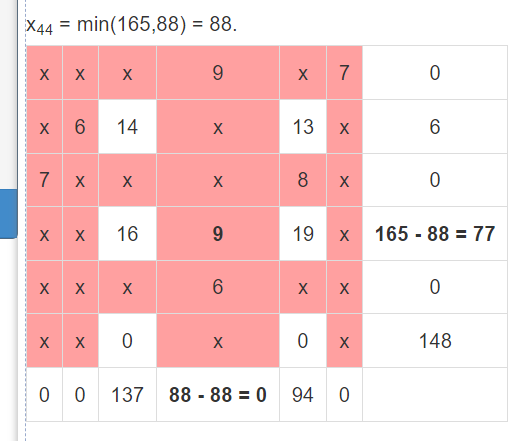


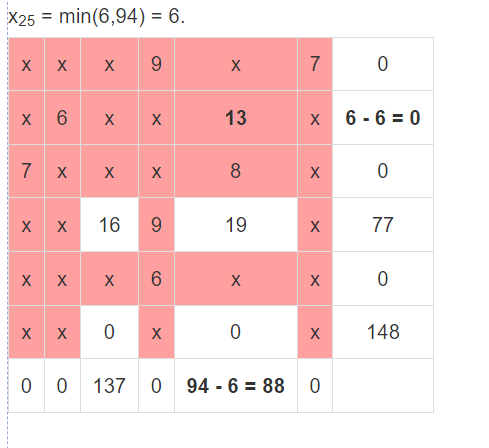


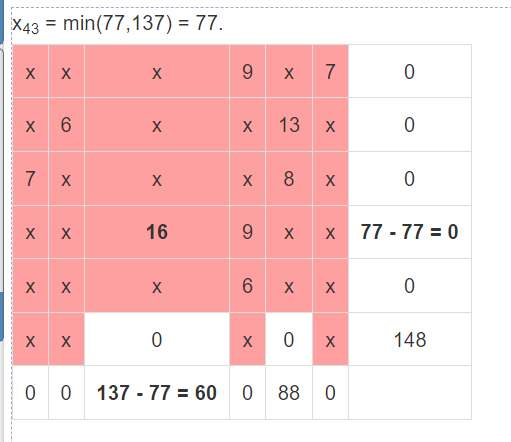


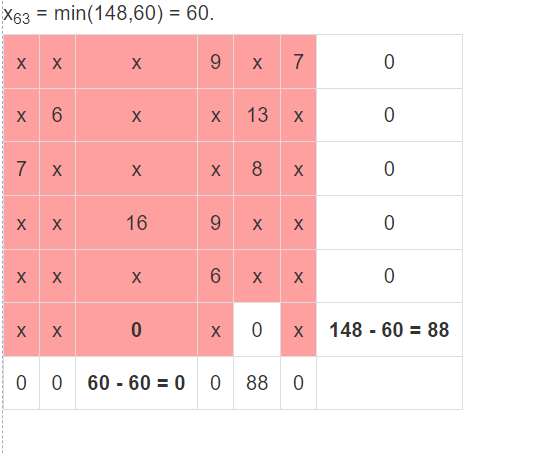


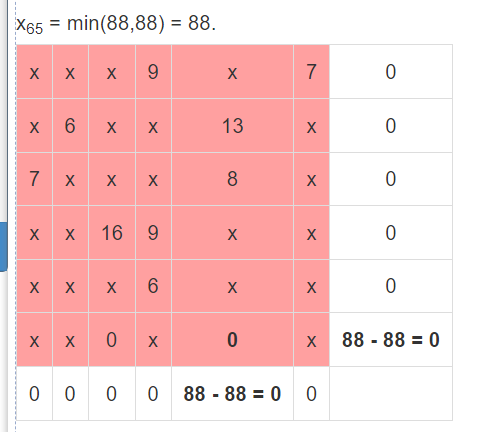


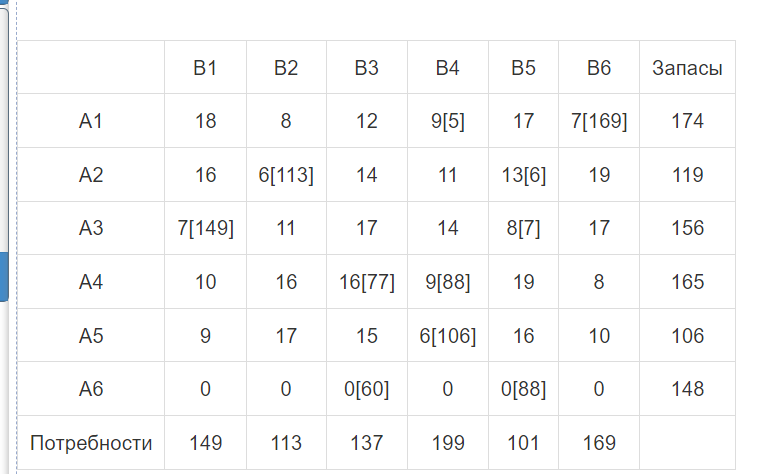






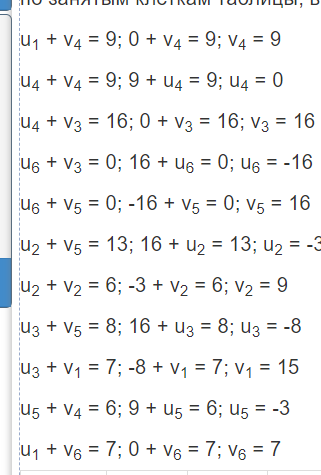




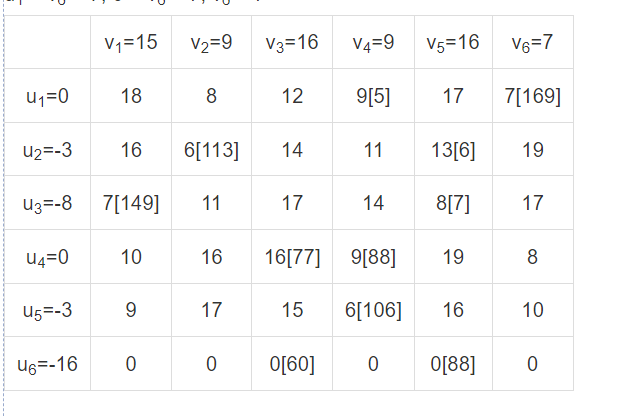


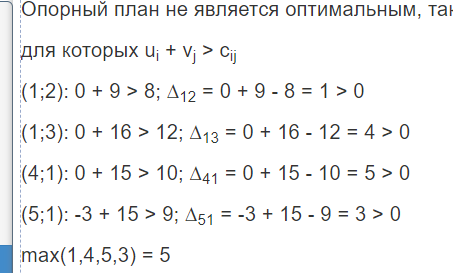
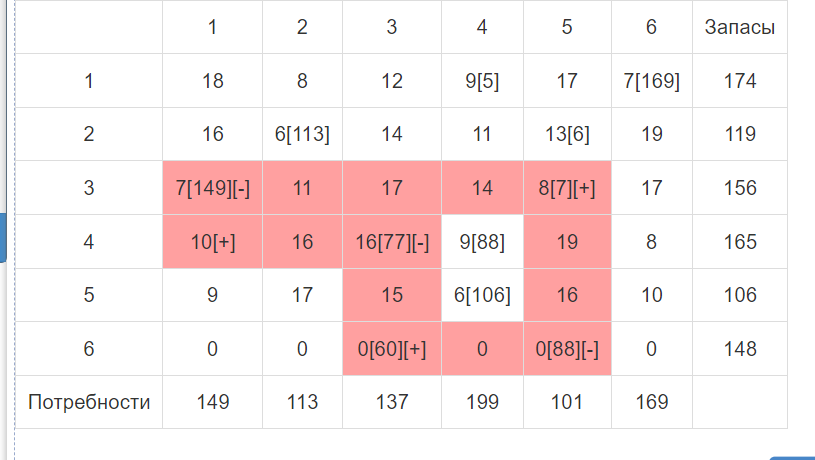


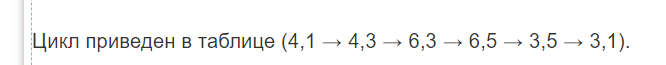


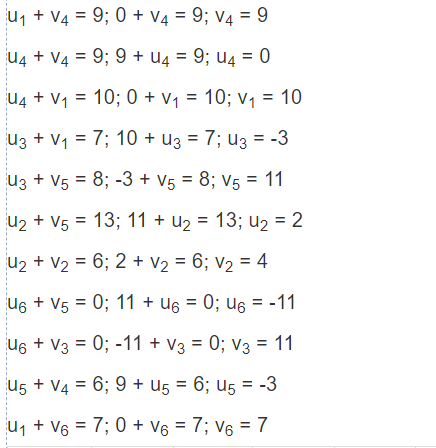


потенциал

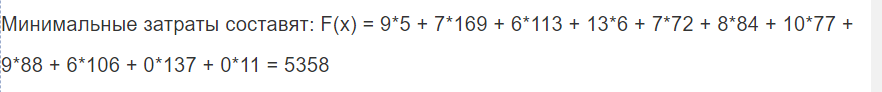










**Вывод:** По результатам решения транспортной задачи, мы можем сделать вывод, что оптимальный план перевозок продукции между поставщиками и потребителями был найден с помощью метода опорных планов и метода потенциалов. Оптимальная стоимость перевозок составила 5358, что является наименьшей возможной стоимостью перевозок при данной конфигурации поставщиков и потребителей.

# Лабораторная работа 6

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

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

|  |  |
| --- | --- |
| Вариант | Граф G |
| 6 |  |

Матрица смежности - это квадратная матрица размера n x n, где n - число вершин в графе. Эта матрица используется для представления графа в виде таблицы, где каждый элемент матрицы a\_ij показывает, существует ли ребро между вершинами i и j.

Если в графе есть ребро между вершинами i и j, то a\_ij равно 1. Если ребра между этими вершинами нет, то a\_ij равно 0. Обратим внимание, что для неориентированного графа матрица смежности будет симметричной относительно главной диагонали.

**Матрица смежности:**

0 1 2 3 4 5 6

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

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

2{0, 0, 0, 1, 0, 1, 0}, 

3{0, 0, 0, 0, 0, 0, 0},

4{0, 1, 0, 1, 0, 0, 0},

5{0, 0, 0, 1, 0, 0, 1},

6{0, 0, 0, 0, 1, 0, 0}



Матрица инцидентности

|  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| Связи Вершины | g(0,1) | g(0,2) | g(0,3) | g(1,3) | g(2,3) | g(2,5) | g(4,1) | g(4,3) | g(5,3) |  | g(6,4) |
| 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 | 1 | 1 | 0 | 0 | 0 |  | 0 |
| 3 | 0 | 0 | -1 | -1 | -1 | 0 | 0 | -1 | -1 |  | 0 |
| 4 | 0 | 0 | 0 | 0 | 0 | 0 | 1 | 1 | 0 |  | -1 |
| 5 | 0 | 0 | 0 | 0 | 0 | -1 | 0 | 0 | 1 |  | 0 |
| 6 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |  | 1 |

**Список смежных вершин:**

S0 = {1, 2, 3}

S1 = {3}

S2 = {3,5}

S3 = ∅

S4 = {1,3}

S5 = {3,6}

S6 = {4}

***Задание 2.*** Осуществить алгоритмы поиска в ширину и глубину, а также алгоритма топологической сортировки аналогично примерам, рассмотренным на лекциях. Оформить отчет, включив в него **каждый** шаг выполнения алгоритмов.

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

**Поиск в ширину:**

Q- для промежуточного хранения вершин(очередь)

массивы:

С- массив окраски вершин ( Б – не добавлена в очередь , С – добавлена в очередь , Ч – вышла из очереди )

D- массив расстояний

Р- массив предшествующих вершин

|  |  |
| --- | --- |
|  | Q={1,2,3} |
|  | Q={2,3} |
|  | Q={3,5} |
|  | Q={5} |
|  | Q={6} |
|  | Q={4} |
|  | Q={} |

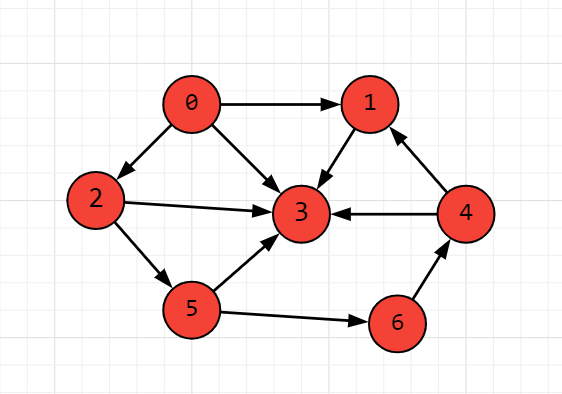
0 1 2 3 5 6 4

**Алгоритм BFS** сводится к следующей последовательности шагов.

1. Инициализировать массивы **С**, **D**, **P**. Стартовую вершину **s** поместить в очередь **Q**. и окрасить в серый цвет: **C[s] = G**. Для стартовой вершины установить расстояние, равное нулю: **D[s] = 0**.
2. Если очередь **Q** пуста, то работа алгоритма завершена, в противном случае перейти к следующему шагу.
3. Выбрать из очереди **Q** вершину **k** и окрасить ее в черный цвет: **С[k] = B**.
4. Построить множества **J** вершин белого цвета смежных вершине **k**. Если таких вершин нет, то перейти к шагу 2, иначе – к следующему шагу.
5. Каждую вершину **j** из множества **J** поместить в очередь **Q**. Обычно (но не обязательно) в очередь вершины помещаются в порядке возрастания номеров.
6. Каждую вершину **j** из множества **J** окрасить в серый цвет: **С[j] = G**.
7. Для каждой вершины **j** из множества **J** вычислитьрасстояние: **D[j] = D[k] + 1**.
8. Для каждой вершины **j** из множества **J** указать предшествующую вершину: **P[j] = k**.
9. Перейти к шагу 3.

**Алгоритм поиска (или обхода) в глубину** (англ. depth-first search, DFS) позволяет построить обход графа, при котором посещаются все вершины, доступные из начальной вершины.

**Алгоритм поиска в глубину:**

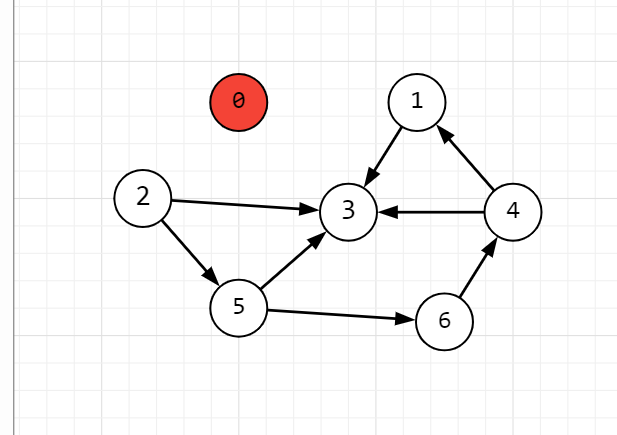
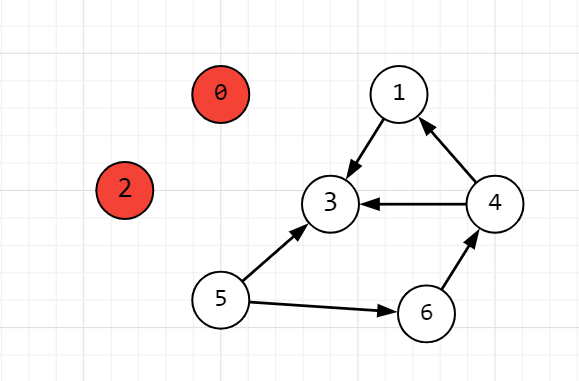
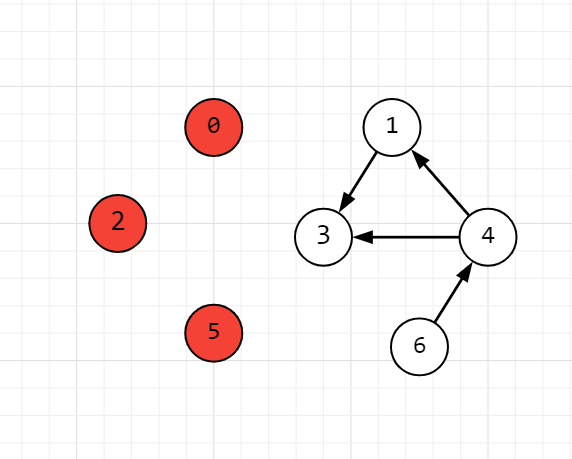


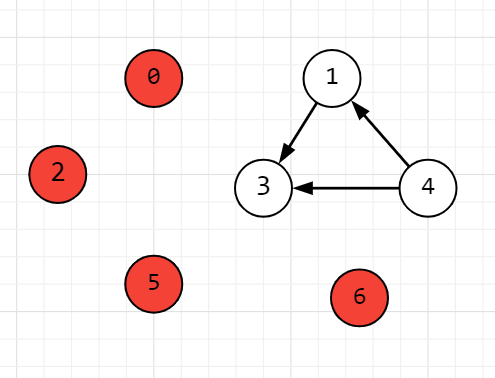
Стек

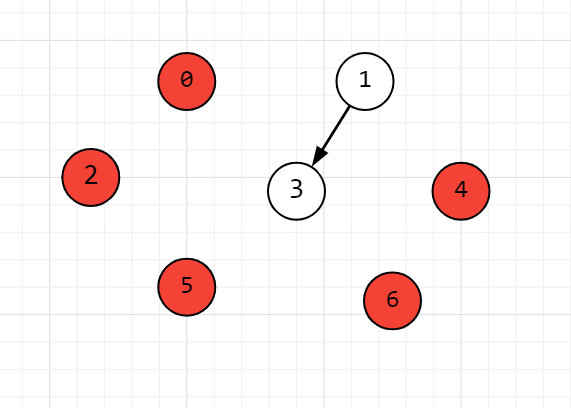
0 3 2 5 6 4 1

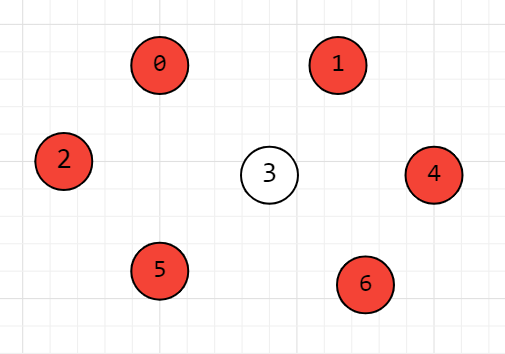
**Топологическая сортировка**

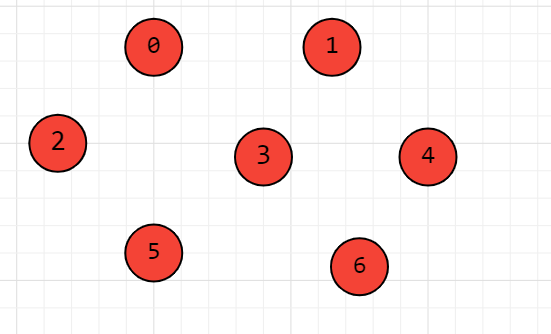
**Топологическая сортировка −** это процедура упорядочивания вершин бесконтурного ориентированного графа, не имеющего циклов (ациклического графа). В результате топологической сортировки для вершин графа определяется такой порядок, что если их расположить на рисунке в соответствии с этим порядком сверху вниз, то дуги будут направлены только от верхних вершин к нижним**.**







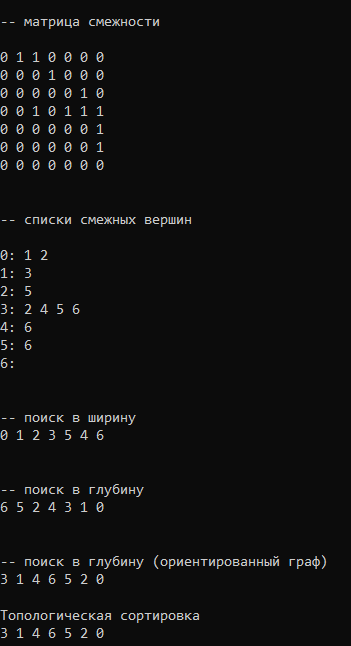


0 2 5 6 4 1 3

***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:

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;

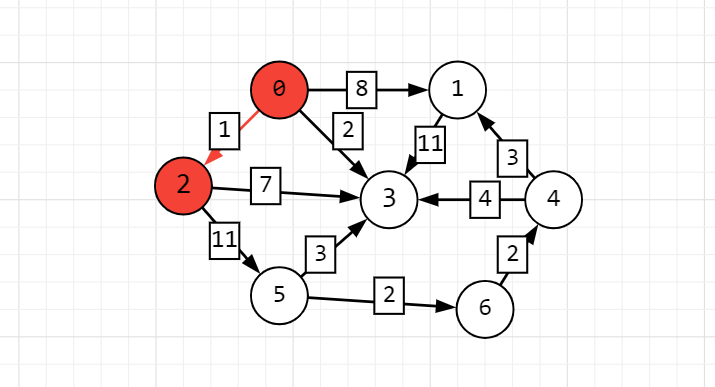
W(e4,3)=4; W(e3,4)=1;

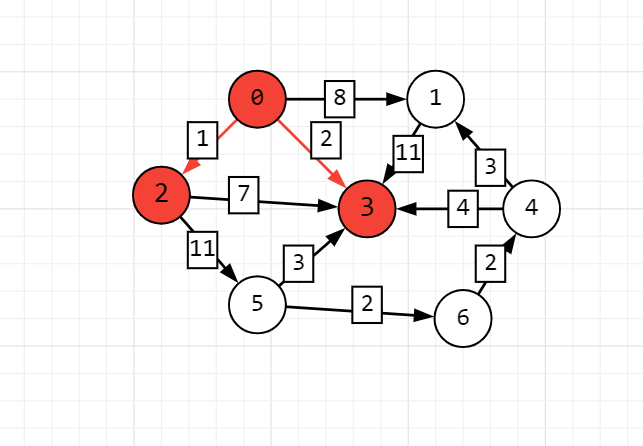
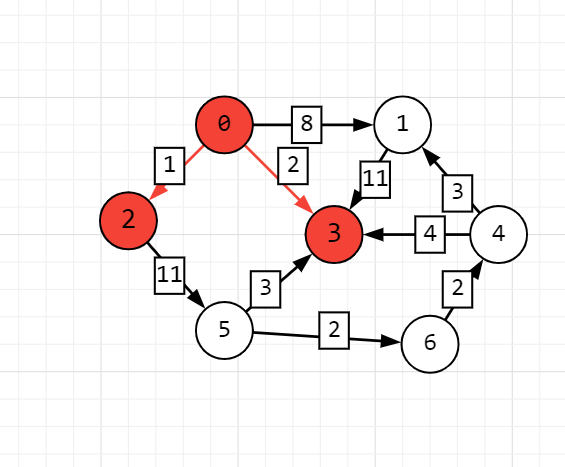
W(e4,6)=10; W(e6,4)=2;

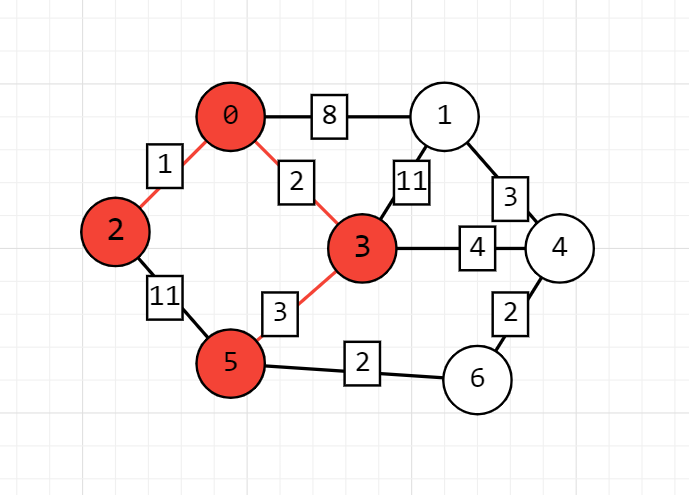
W(e5,6)=2; W(e6,5)=6;

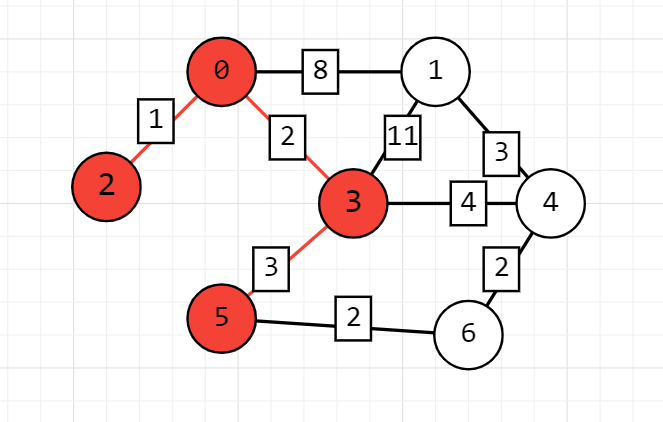
W(e5,3)=3; W(e3,5)=6;

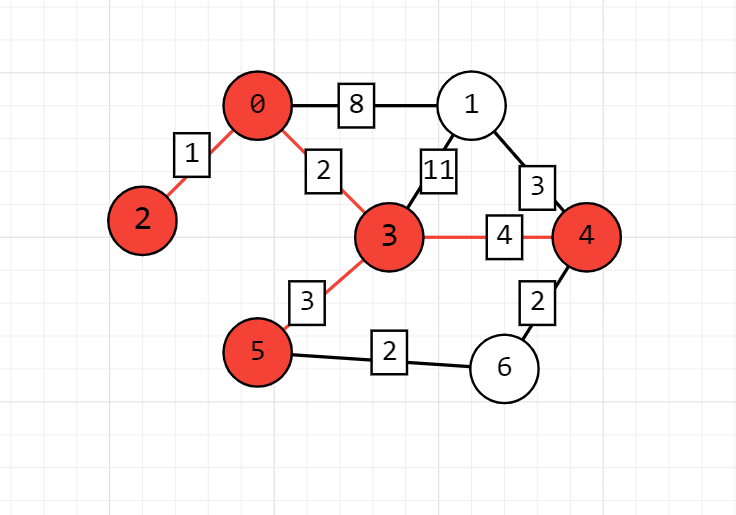
W(e6,3)=7; W(e3,6)=9;

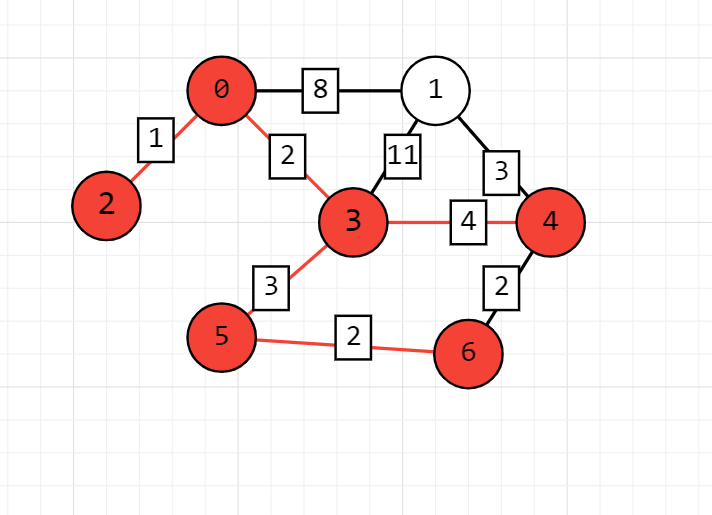


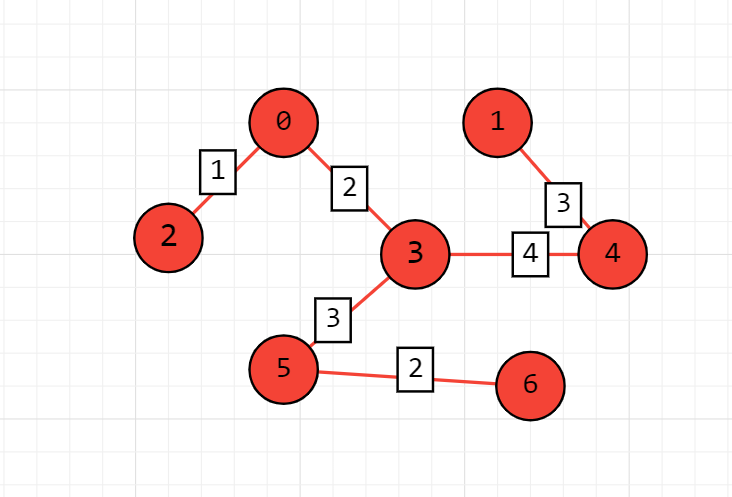
 











***Задание 7.*** По графу, соответствующему варианту составить минимальное остовное дерево по алгоритму Крускала. Шаги построения отразить в отчете.

