Министерство науки и высшего образования Российской Федерации

Федеральное государственное автономное образовательное

учреждение высшего образования

«ЮЖНЫЙ ФЕДЕРАЛЬНЫЙ УНИВЕРСИТЕТ»

(ФГАОУ ВО «ЮФУ»)

Институт компьютерных технологий и информационной безопасности

Кафедра высшей математики

ИНДИВИДУАЛЬНАЯ РАБОТА

по дисциплине «Методы оптимизации»

Программная реализация алгоритма решения транспортной задачи и проведение тестирования разработанной программы на заданной выборке примеров

|  |  |  |
| --- | --- | --- |
| Выполнил  студент группы КТбо3-1 | \_\_\_\_\_\_\_\_\_\_\_ | Беридзе И. Д. |
| Приняла  Доцент ИКТИБ | \_\_\_\_\_\_\_\_\_\_\_ | Липко Ю. Ю. |

Оглавление

[Введение 3](#_Toc151386331)

[Цели и задачи 4](#_Toc151386332)

[Алгоритм 5](#_Toc151386333)

[Структура программы 6](#_Toc151386334)

[Листинг программы 7](#_Toc151386335)

[Выводы 12](#_Toc151386336)

Введение

Транспортные задачи — это специальный класс задач из разделов линейного программирования. В задачах чаще всего описываются перевозки какого-либо товара из пункта отправления (как правило, с места производства) в пункт назначения (склад или магазин). Также при решении транспортных задач должны учитываться ограничения, накладываемые на объёмы грузов в пунктах отправления, потребность в пунктах назначения. Кроме ограничений, стоит учитывать и стоимость перевозки по заданным маршрутам.

Задачей транспортных задач является оптимизация перевозок с учётом всех ограничений. С помощью программной реализации можно существенно сократить время на вычисление целевой функции, описывающей минимальные затраты с учётом всех условий и ограничений.

Цели и задачи

Цель работы — программная реализация алгоритма решения транспортной задачи и тестирование на заданной выборке примеров.

Задачи работы:

1. Реализовать алгоритм Северо-Западного угла для решения транспортной задачи;
2. Реализовать алгоритм минимального транспортного тарифа для решения транспортной задачи;
3. Найти целевую функцию с помощью одного из двух представленных выше алгоритмов.

Алгоритм

Для удобства восприятия алгоритм, переведённый в программу на языке Python3, представлен ниже в виде блок-схемы.

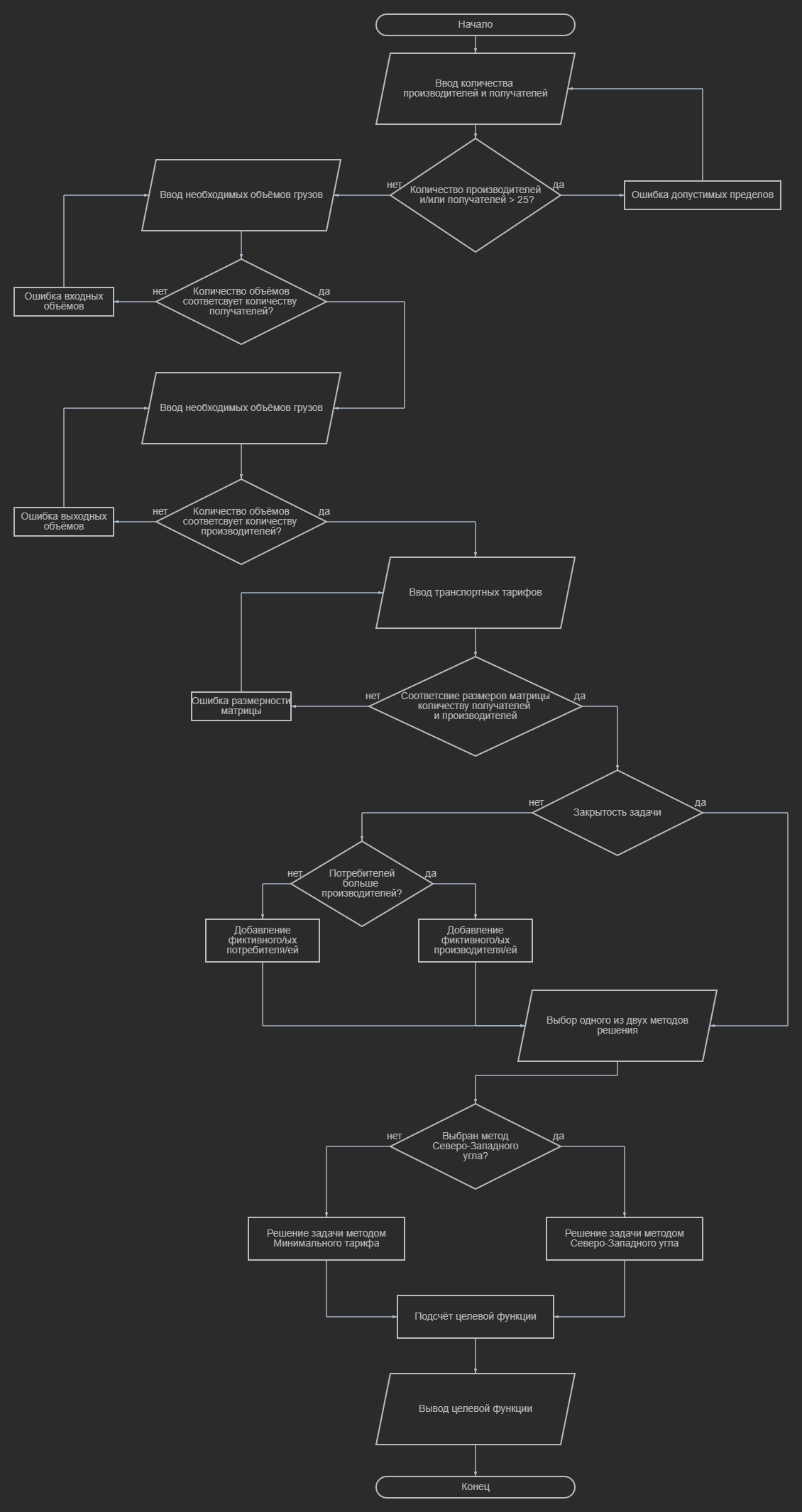


Рисунок 1 - Блок-схема

Структура программы

Проект содержит в себе два файла: файл класса Transport.py и исполняемый файл main.py.

В файле Transport.py реализован класс, включающий необходимые поля (количество получателей, производителей, объёмы входных и выходных грузов, матрица транспортных тарифов, открытость/закрытость задачи, матрица с итоговым планом, целевая функция), методы класса и вспомогательные функции.

В файле main.py реализована работа с пользователем из окна командой строки.

Листинг программы

Файл Transport.py:

# Задание условий

**class** Transport**:**

**def** \_\_init\_\_**(**self**):**

self**.**makers **=** 5 ***# Количество отправителей (производителей)***

self**.**customers **=** 3 ***# Количество получателей (заказчиков)***

self**.**needs **=** **[**60**,** 60**,** 50**]** ***# Необходимые получателям объёмы***

self**.**volume **=** **[**30**,** 20**,** 55**,** 40**,** 25**]** ***# Имеющиеся у отправителей объёмы***

self**.**rate **=** **[]** ***# Транспортный тариф***

self**.**close **=** True ***# Закрытость задачи***

self**.**vol\_out **=** **[]** ***# Итоговый план***

self**.**f\_min **=** 0 ***# Целевая функция***

***# Установка количества производителей и заказчиков***

**def** m\_c\_set**(**self**,** make**,** cust**):**

**if** make **>** 25 **or** cust **>** 25**:**

print**(**'Слишком много получателей и/или отправителей!'**)**

print**(**' Принимается не больше 25!\n'**)**

**return** 0

**else:**

self**.**makers **=** make

self**.**customers **=** cust

**return** 1

***# def extend(self, need\_or\_vol):***

***# t\_need = "Введите необходимые объёмы грузов:"***

***# t\_vol = "Введите имеющиеся объёмы грузов:"***

***#***

***# while True:***

***# print(t\_need)***

***# try:***

***#***

***# Установка объёмов необходимых и имеющихся грузов***

**def** n\_v\_set**(**self**):**

self**.**needs **=** **[]**

self**.**volume **=** **[]**

self**.**separate**()**

**while** True**:**

print**(**"Введите необходимые объёмы грузов:"**)**

**try:**

self**.**needs **=** list**(**map**(**int**,** input**().**split**()))**

**except** ValueError**:**

print**(**" Ошибка! Вы ввели не число!"**)**

**else:**

**if** len**(**self**.**needs**)** **==** self**.**customers**:**

print**(**"Принято!"**)**

self**.**separate**()**

**break**

**else:**

self**.**needs **=** **[]**

print**(**"Недопустимый ввод!"**)**

print**(**self**.**needs**)**

self**.**separate**()**

**while** True**:**

print**(**"Введите имеющиеся объёмы грузов:"**)**

**try:**

self**.**volume **=** list**(**map**(**int**,** input**().**split**()))**

**except** ValueError**:**

print**(**" Ошибка! Вы ввели не число!"**)**

**else:**

**if** len**(**self**.**volume**)** **==** self**.**makers**:**

print**(**"Принято!"**)**

self**.**separate**()**

**break**

**else:**

self**.**volume **=** **[]**

print**(**"Недопустимый ввод!"**)**

print**(**self**.**volume**)**

***# Установка транспортных тарифов***

**def** set\_rate**(**self**):**

self**.**rate **=** **[]**

self**.**separate**()**

**while** True**:**

print**(**"Вводите построчно транспортные тарифы:\n"**)**

**try:**

**for** i **in** range**(**0**,** self**.**customers**):**

self**.**rate**.**append**(**list**(**map**(**int**,** input**().**split**())))**

**if** len**(**self**.**rate**[**i**])** **!=** self**.**makers**:**

self**.**rate **=** **[]**

print**(**" Недопустимый ввод!"**)**

**raise** StopIteration

**except** ValueError**:**

print**(**" Ошибка! Вы ввели не число!"**)**

**except** StopIteration**:**

**pass**

**else:**

print**(**"Принято!"**)**

self**.**separate**()**

**break**

print**(**self**.**rate**)**

***# Проверка на открытость или закрытость задачи***

**def** is\_close**(**self**):**

cust\_vol\_sum **=** 0

make\_vol\_sum **=** 0

**for** L **in** self**.**needs**:**

cust\_vol\_sum **+=** L

**for** R **in** self**.**volume**:**

make\_vol\_sum **+=** R

self**.**separate**()**

**if** cust\_vol\_sum **==** make\_vol\_sum**:** ***# Задача закрыта. Фиктивные отправители или получатели не требуются***

print**(**f'Задача закрыта (сумма = {cust\_vol\_sum}).'**)**

**for** i **in** range**(**0**,** len**(**self**.**rate**)):**

**for** j **in** range**(**0**,** len**(**self**.**rate**[**i**])):**

print**(**self**.**rate**[**i**][**j**],** end**=**' '**)**

print**()**

**elif** cust\_vol\_sum **<** make\_vol\_sum**:** ***# Задача открыта. Необходимо добавить фиктивного заказчика***

self**.**close **=** False

dif **=** make\_vol\_sum **-** cust\_vol\_sum

print**(**f'Задача открыта (лишние {dif} у поставщиков).'**)**

self**.**customers **+=** 1

self**.**needs**.**append**(**dif**)**

self**.**rate**.**insert**(**len**(**self**.**rate**),** **[**0 **\*** i **for** i **in** range**(**self**.**makers**)])**

**for** i **in** range**(**0**,** len**(**self**.**rate**)):**

**for** j **in** range**(**0**,** len**(**self**.**rate**[**i**])):**

print**(**self**.**rate**[**i**][**j**],** end**=**' '**)**

print**()**

**else:** ***# Задача открыта. Необходимо добавить фиктивного поставщика***

self**.**close **=** False

dif **=** cust\_vol\_sum **-** make\_vol\_sum

print**(**f'Задача открыта (недостающие {dif} у заказчиков).'**)**

self**.**makers **+=** 1

self**.**volume**.**append**(**dif**)**

**for** i **in** range**(**0**,** self**.**customers**):**

self**.**rate**[**i**].**append**(**0**)**

**for** i **in** range**(**0**,** len**(**self**.**rate**)):**

**for** j **in** range**(**0**,** len**(**self**.**rate**[**i**])):**

print**(**self**.**rate**[**i**][**j**],** end**=**' '**)**

print**()**

self**.**separate**()**

***# Создание матрицы перевозимого объёма груза***

**def** volume\_now**(**self**):**

self**.**vol\_out **=** **[[-**1**]** **\*** len**(**self**.**rate**[**i**])** **for** i **in** range**(**len**(**self**.**rate**))]**

**for** i **in** range**(**0**,** len**(**self**.**vol\_out**)):**

**for** j **in** range**(**0**,** len**(**self**.**vol\_out**[**i**])):**

self**.**vol\_out**[**i**][**j**]** **=** **-**1

***# Метод Северо-Западного угла***

**def** nort\_west**(**self**):**

print**(**"Метод Северо-Западного угла"**)**

**for** i **in** range**(**0**,** len**(**self**.**vol\_out**)):**

**for** j **in** range**(**0**,** len**(**self**.**vol\_out**[**i**])):**

self**.**check\_set**(**i**,** j**)**

self**.**normal\_set**()**

self**.**separate**()**

self**.**plan\_out**()**

self**.**separate**()**

self**.**func\_out**()**

***# Метод наименьшего транспортного тарифа***

**def** min\_way**(**self**):**

min\_now **=** 1

max\_tt **=** self**.**max\_tt**()**

**while** **(**self**.**volume **!=** **[**0**]** **\*** len**(**self**.**volume**))** **and** **(**self**.**needs **!=** **[**0**]** **\*** len**(**self**.**needs**)):**

**for** i **in** range**(**0**,** len**(**self**.**rate**)):**

**for** j **in** range**(**0**,** len**(**self**.**rate**[**i**])):**

**if** self**.**rate**[**i**][**j**]** **==** min\_now**:**

self**.**check\_set**(**i**,** j**)**

**elif** self**.**rate**[**i**][**j**]** **==** 0 **and** min\_now **==** max\_tt**:**

self**.**check\_set**(**i**,** j**)**

**else:**

**continue**

min\_now **+=** 1

self**.**normal\_set**()**

self**.**separate**()**

self**.**plan\_out**()**

self**.**separate**()**

self**.**func\_out**()**

***# Функция поиска максимального тт***

**def** max\_tt**(**self**):**

tt\_max **=** 0

**for** i **in** range**(**0**,** len**(**self**.**rate**)):**

**for** j **in** range**(**0**,** len**(**self**.**rate**[**i**])):**

**if** self**.**rate**[**i**][**j**]** **>** tt\_max**:**

tt\_max **=** self**.**rate**[**i**][**j**]**

**return** tt\_max

***# Функция распределения объёмов грузов***

**def** check\_set**(**self**,** i**,** j**):**

**if** self**.**needs**[**i**]** **!=** 0 **and** self**.**volume**[**j**]** **!=** 0**:** ***# Проверка необходимости***

**if** self**.**volume**[**j**]** **-** self**.**needs**[**i**]** **<** 0**:** ***# Если у поставщика меньше груза, чем требует получатель***

self**.**vol\_out**[**i**][**j**]** **=** self**.**volume**[**j**]**

self**.**needs**[**i**]** **-=** self**.**volume**[**j**]**

self**.**volume**[**j**]** **=** 0

**elif** self**.**volume**[**j**]** **-** self**.**needs**[**i**]** **==** 0**:** ***# Если имеющийся объём поставщика равен требуемому***

self**.**vol\_out**[**i**][**j**]** **=** self**.**needs**[**i**]**

self**.**needs**[**i**]** **=** 0

self**.**volume**[**j**]** **=** 0

**else:** ***# Если у поставщика больше груза, чем требует получатель***

self**.**vol\_out**[**i**][**j**]** **=** self**.**needs**[**i**]**

self**.**volume**[**j**]** **-=** self**.**needs**[**i**]**

self**.**needs**[**i**]** **=** 0

***# Функция нормализации итогового плана***

**def** normal\_set**(**self**):**

**for** i **in** range**(**0**,** len**(**self**.**vol\_out**)):**

**for** j **in** range**(**0**,** len**(**self**.**vol\_out**[**i**])):**

**if** self**.**vol\_out**[**i**][**j**]** **==** **-**1**:**

self**.**vol\_out**[**i**][**j**]** **=** 0

***# Вывод итогового плана***

**def** plan\_out**(**self**):**

print**(**"Итоговый план:\n"**)**

**for** i **in** range**(**0**,** len**(**self**.**vol\_out**)):**

**for** j **in** range**(**0**,** len**(**self**.**vol\_out**[**i**])):**

print**(**self**.**vol\_out**[**i**][**j**],** end**=**' '**)**

print**()**

***# Вывод целевой функции***

**def** func\_out**(**self**):**

**for** i **in** range**(**0**,** len**(**self**.**vol\_out**)):**

**for** j **in** range**(**0**,** len**(**self**.**vol\_out**[**i**])):**

self**.**f\_min **+=** self**.**vol\_out**[**i**][**j**]** **\*** self**.**rate**[**i**][**j**]**

print**(**f'f(x) -> min = {self**.**f\_min}'**)**

***# Разделитель***

*@staticmethod*

**def** separate**():**

print**(**'-'**\***32**)**

Файл main.py:

**from** Transport **import** Transport

A **=** Transport**()**

A**.**separate**()**

makers **=** 0

customers **=** 0

**while** True**:**

**try:**

makers **=** int**(**input**(**"Введите количество производителей: "**))**

customers **=** int**(**input**(**"Введите количество потребителей: "**))**

**except** ValueError**:**

print**(**" Ошибка! Вы ввели не число!"**)**

**else:**

**if** A**.**m\_c\_set**(**makers**,** customers**)** **==** 1**:**

**break**

A**.**n\_v\_set**()**

A**.**set\_rate**()**

A**.**is\_close**()**

A**.**volume\_now**()**

method **=** **-**1

**while** True**:**

**try:**

print**(**"Выберите один из методов:"**)**

print**(**" Метод Севере-западного угла - 0"**)**

print**(**" Метод минимального тт - 1"**)**

method **=** int**(**input**(**"Выбранный метод - "**))**

**except** ValueError**:**

print**(**" Ошибка! Неправильный ввод!"**)**

**else:**

**if** method **==** 0**:**

A**.**nort\_west**()**

**break**

**elif** method **==** 1**:**

A**.**min\_way**()**

**break**

**else:**

print**(**"Ошибка! Что-то пошло не так..."**)**

**break**

Выводы

В ходе выполнения индивидуальной работы были подробно разобраны алгоритмы решения транспортных задач с помощью методов Северо-Западного угла и минимального транспортного тарифа, что позволило автоматизировать вычисления с помощью переноса в программу.