Министерство науки и высшего образования Российской Федерации

Государственное образовательное учреждение высшего профессионального образования

«Алтайский государственный технический университет им. И. И. Ползунова»

Факультет информационных технологий

Кафедра Прикладная математика

Отчет защищен с оценкой\_\_\_\_\_\_\_\_\_\_\_\_\_\_

Преподаватель         А. В. Сорокин

(подпись) (и.о.фамилия)

“\_\_\_\_” \_\_\_\_\_\_\_\_\_\_\_ 2022 г.

дата

Отчет

по дисциплине

ОСНОВЫ МОДЕЛИРОВАНИЯ

лабораторная работа №4

Математические модели в задаче линейного программирования

название работы

ЛР 09.03.04.23.004 О

обозначение документа

Студент группы гр. ПИ-02                                                     Чередов Р.А.

*(подпись*)

Преподаватель доцент, к.т.н.               А. В. Сорокин

должность, ученое звание и.о., фамилия

БАРНАУЛ 2022

# **Аннотация**

В данной лабораторной работе были рассмотрены модели транспортной задачи. Общая постановка транспортной задачи состоит в определении оптимального плана перевозок некоторого однородного груза из m пунктов отправления в n пунктов назначения. Также были рассмотрены методы северо-западного угла и наименьшей стоимости, совместно с методом потенциалов.

Оглавление

[**Аннотация** 2](#_Toc155776386)

[**Задание** 4](#_Toc155776387)

[**Математическая модель** 5](#_Toc155776388)

[**Решение средствами Microsoft Excel** 6](#_Toc155776389)

[**Программа** 9](#_Toc155776390)

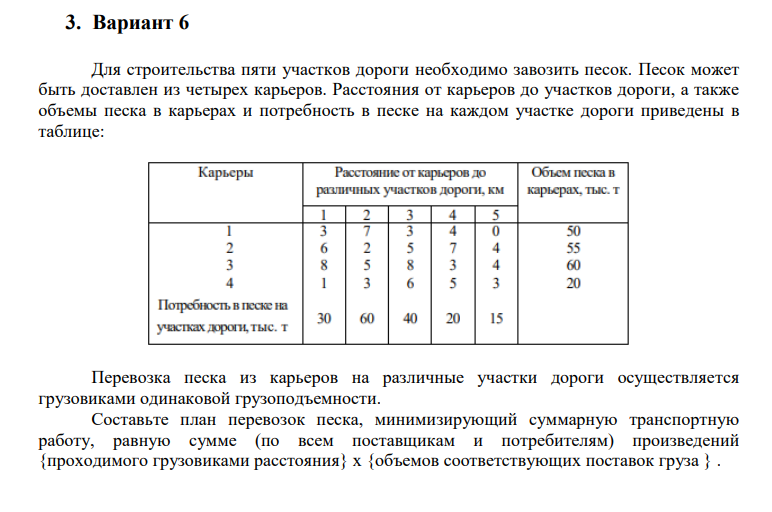
[**Программа** 10](#_Toc155776391)

[**Результат работы** 15](#_Toc155776392)

[**Заключение** 18](#_Toc155776393)

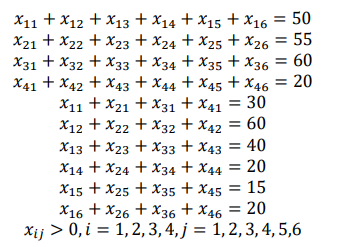
[**Список использованных источников** 18](#_Toc155776394)

# **Задание**

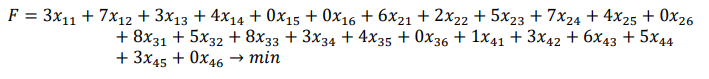


# **Математическая модель**

Обозначим через xij количество тыс. т песка, перевозимого от i-го карьера до j-го участка дороги. Задача является открытой, так как сумма запасов песка 50 + 55 + 60 + 20 = 185 в карьерах больше суммы потребностей на участках дорог 30 + 60 + 40 + 20 + 15 = 165. Для того чтобы она стала закрытой, введем фиктивный участок дороги с номером 6, потребность в песке у которого будет равна 185 – 165 = 20. При этом все транспортные работы от каждого карьера до участка дороги с номером 6 будут равны 0:



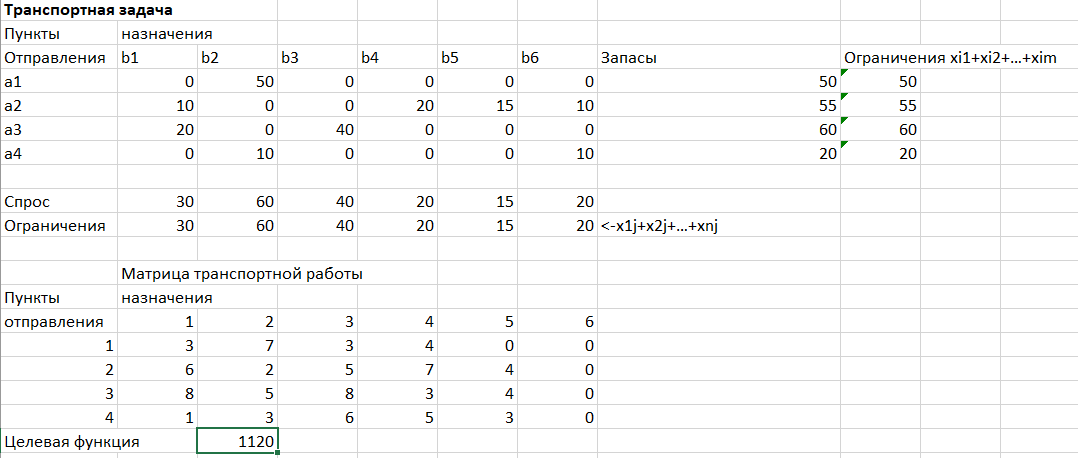
При данном плане перевозок суммарная их транспортная работа составит (целевая функция):



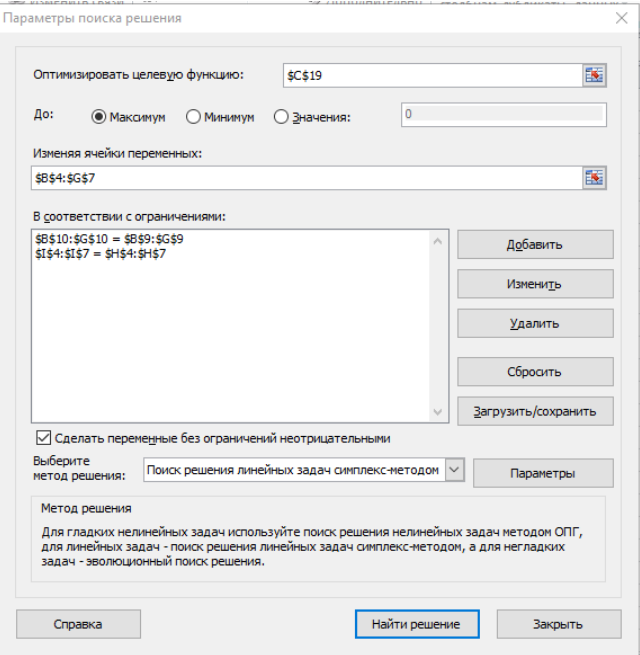
Таким образом, математическая постановка задачи состоит в нахождении такого неотрицательного решения системы линейных равенств-ограничений, при котором целевая функция принимает минимальное значение

# **Решение средствами Microsoft Excel**

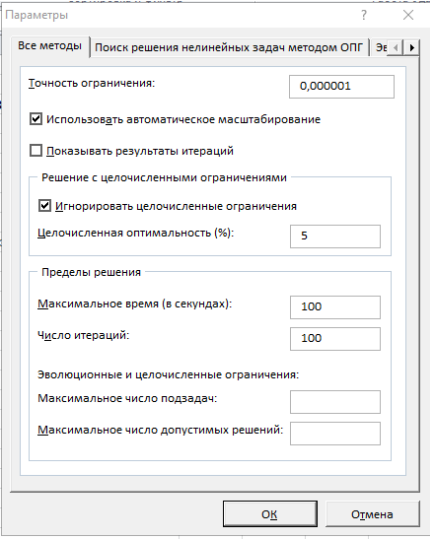
Построим таблицу :



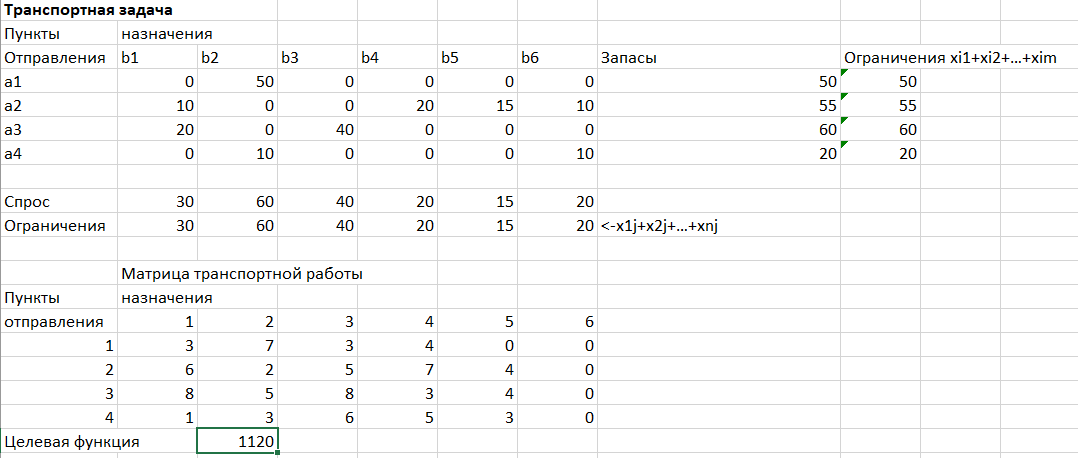
Ставим курсор мыши в ячейку C19, нажимаем левую кнопку мыши. Далее во вкладке «Данные» выбираем «Поиск решения». Заполняем поля как показано на скриншоте ниже



Выбираем пункт «Параметры» -> «Все методы» и заполняем поля как показано ниже:



Нажимаем «ОК» -> «Найти решение». В результате получим решение:



# **Программа**

Для решения задачи необходимо выполнение следующих условий:

1. cуммарные запасы продукции у поставщиков должны равняться суммарной потребности потребителей.
2. количество задействованных маршрутов = количество поставщиков + количество потребителей ‐ 1. (Поэтому если возникнет ситуация, в которой будет необходимо исключить столбец и строку одновременно, мы исключим что‐то одно.)

**Метод северо-западного угла**

Дадим переменной x11 максимально возможное значение или, иными словами, максимально возможную поставку в клетку (1,1) – «Северо-западный» угол таблицы поставок. Далее ищем новый «северо-западный» угол. Когда мощность поставщика полностью реализована – убираем из рассмотрения строку. Когда потребность потребителя полностью удовлетворена – убираем столбец. Число заполненных клеток должно оказаться равным n+m-1, т.е. числу основных (базисных) переменных

**Метод наименьших затрат**

Вместо того, чтобы на каждом шаге максимально возможную поставку давать в «северо-западную» клетку, можно эту поставку давать в клетку с наименьшим коэффициентом затрат. Такой метод называется методом наименьших затрат

**Метод потенциалов**

Для проверки оптимальности решения используется метод потенциалов:

Каждому поставщику A i ставим в соответствие некоторое число U i , называемое потенциалом поставщика

Каждому потребителю B j ставим в соответствие некоторое число V j , называемое потенциалом потребителя.

Для задействованного маршрута: потенциал поставщика + потенциал потребителя = тариф задействованного маршрута.

После этого ищем значения потенциалов. Задаем значение одного из потенциалов нулем. Далее ищем оценки незадействованных маршрутов, и если есть отрицательные оценки – это означает, что можно найти новое решение, как минимум, не хуже имеющегося

Используя только горизонтальные и вертикальные линии, соединияем непрерывной линией заполненные ячейки так, чтобы вернуться в исходную ячейку с отрицательной оценкой

Ячейки, расположенные в вершинах построенной ломаной линии, образуют цикл для выбранной ячейки.

Пусть выбранная ячейка имеет порядковый номер один. Последовательно пронумеруем ячейки, из которых состоит цикл. Среди значений ячеек, находящихся на четных позициях выберем наименьшее.

Далее из ячеек, стоящих на четных позициях, вычитаем минимальное значение, а к ячейкам, стоящим на нечетных позициях, прибавляем.

Данное преобразование не изменит баланса. А вот общая стоимость доставки продукции изменится в меньшую сторону.

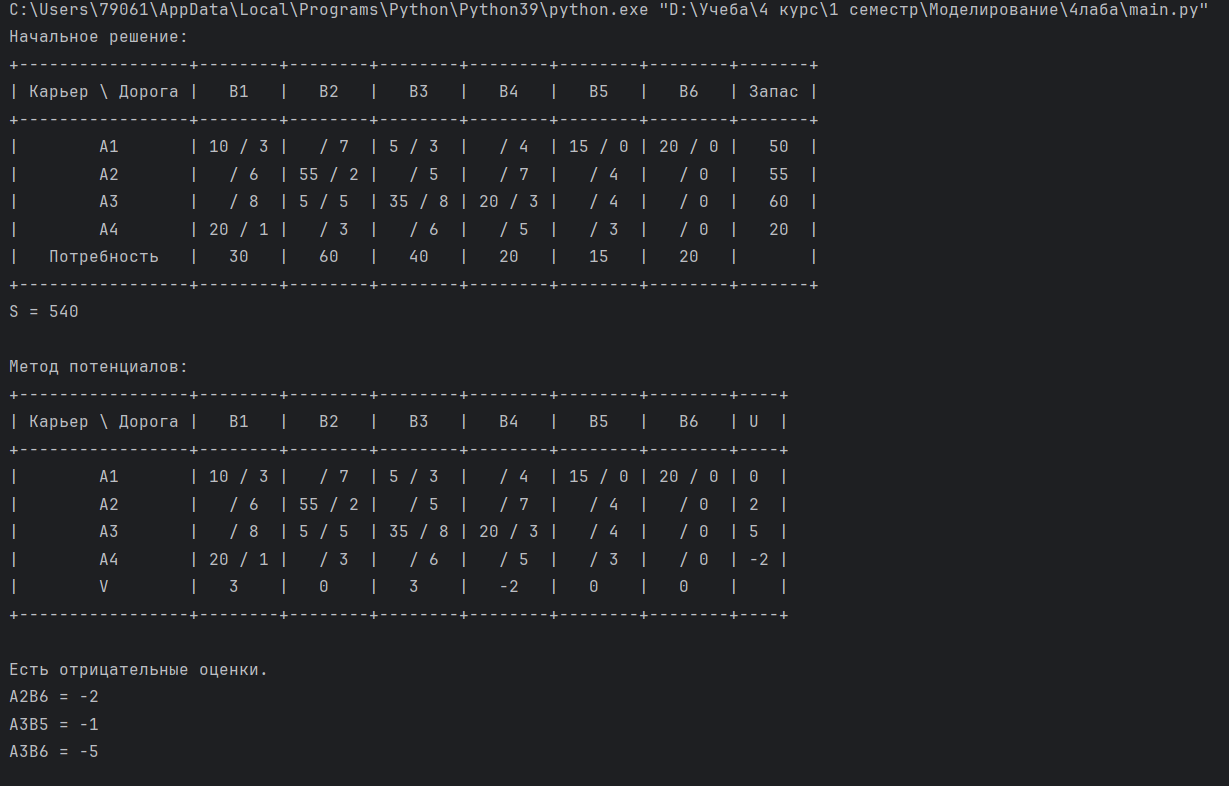
Данная закономерность справедлива для любого цикла. Можно сделать выводы:

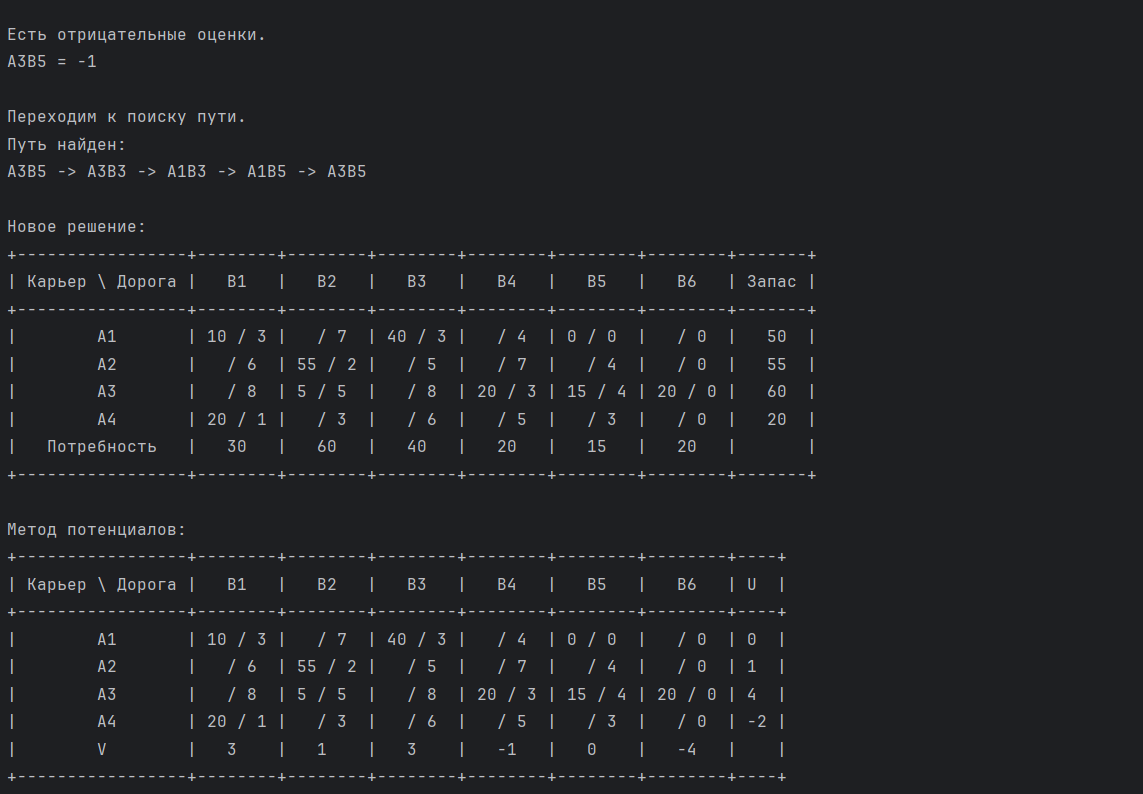
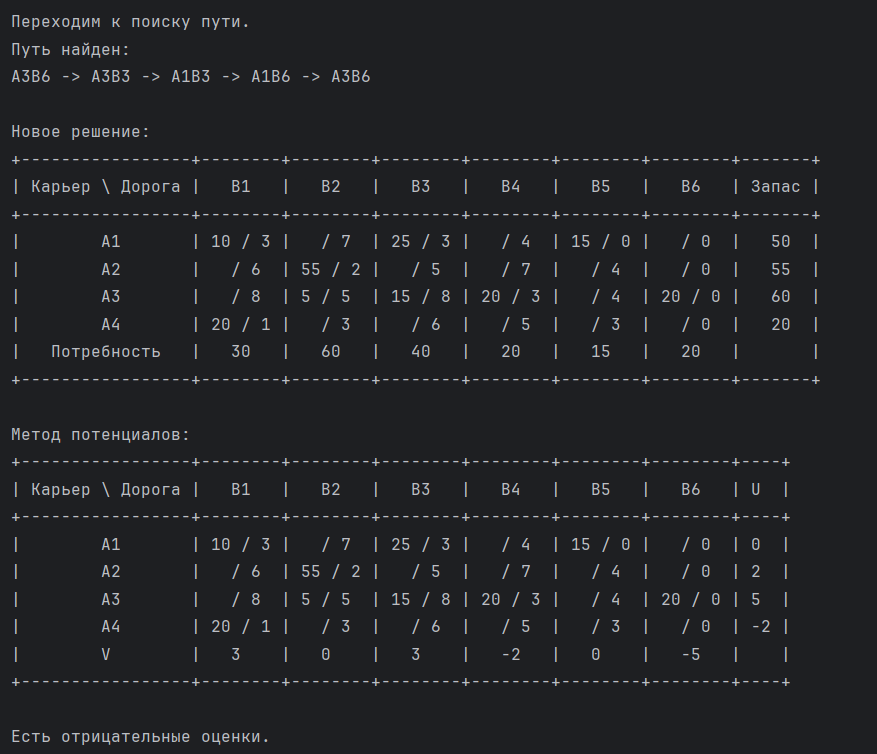
* Если нет отрицательных оценок, то уменьшить общую стоимость доставки невозможно.
* Если наименьшее значение ячеек цикла, расположенных на четных позициях, равно 0, то общая стоимость доставки не изменится на данном шаге.

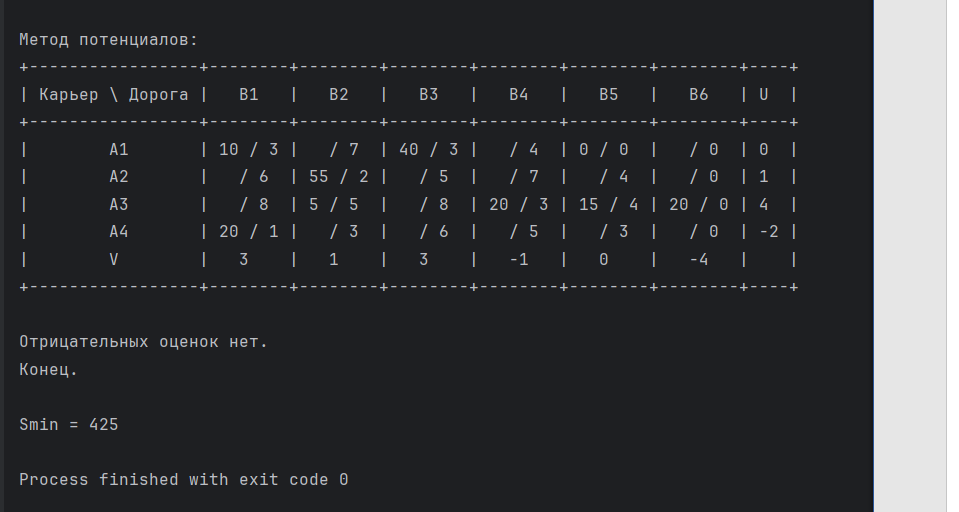
# **Программа**

from prettytable import PrettyTable  
# Метод северо-западного угла  
def north\_west(A, B):  
 n = len(A)  
 m = len(B)  
 result = [[None] \* m for \_ in range(n)]  
  
 for i in range(n):  
 for j in range(m):  
 if A[i] == 0:  
 break  
 elif B[j] == 0:  
 continue  
 elif A[i] == B[j] and j != m-1:  
 if i == n-1:  
 result[i][j+1] = 0  
 else:  
 result[i+1][j] = 0  
 min\_AB = min(A[i], B[j])  
 result[i][j] = min\_AB  
 A[i] -= min\_AB  
 B[j] -= min\_AB  
 return result  
# Метод наименьшей стоимости  
def min\_price(A, B, tariff):  
 n = len(A)  
 m = len(B)  
 tariff\_copy = [[True] \* m for \_ in range(n)]  
 result = [[None] \* m for \_ in range(n)]  
 while max(A) > 0 or max(B) > 0:  
 min\_el = (9999999, 0, 0)  
 for i in range(n):  
 for j in range(m):  
 if tariff\_copy[i][j]:  
 if tariff[i][j] < min\_el[0] and A[i] != 0 and B[j] != 0:  
 min\_el = (tariff[i][j], i, j)  
 i, j = min\_el[1], min\_el[2]  
 tariff\_copy[i][j] = False  
 min\_AB = min(A[i], B[j])  
 result[i][j] = min\_AB  
 A[i] -= min\_AB  
 B[j] -= min\_AB  
 return result  
# Метод потенциалов (поиск отрицательных потенциалов)  
def potential\_method(base\_matrix, A, B, tariff\_matrix):  
 n = len(A)  
 m = len(B)  
 count = 0  
 for i in range(n):  
 for j in range(m):  
 if base\_matrix[i][j] != None:  
 count+=1  
 if count != n+m-1:  
 return False  
 ocenki = []  
 V = [None] \* m  
 U = [None] \* n  
 U[0] = 0  
 while None in V or None in U:  
 for i in range(n):  
 for j in range(m):  
 if (U[i] == None and V[j] == None) or (U[i] != None and V[j] != None):  
 continue  
 elif U[i] != None and base\_matrix[i][j] != None:  
 V[j] = tariff\_matrix[i][j] - U[i]  
 elif base\_matrix[i][j] != None and V[j] != None:  
 U[i] = tariff\_matrix[i][j] - V[j]  
 for i in range(n):  
 for j in range(m):  
 if base\_matrix[i][j] == None:  
 ocenki.append((i, j, tariff\_matrix[i][j] - (U[i] + V[j])))  
 table = PrettyTable()  
 field\_names = ["Карьер \\ Дорога"]  
 for i in range(m):  
 field\_names.append("B{}".format(i+1))  
 field\_names.append("U")  
 table.field\_names = field\_names  
 for i in range(n):  
 table\_row = []  
 table\_row.append("A{}".format(i+1))  
 for j in range(m):  
 if base\_matrix[i][j] != None:  
 table\_row.append(str(base\_matrix[i][j]) + ' / ' + str(tariff\_matrix[i][j]))  
 else:  
 table\_row.append(' ' + ' / ' + str(tariff\_matrix[i][j]))  
 table\_row.append(U[i])  
 table.add\_row(table\_row)  
 table\_row = []  
 table\_row.append("V")  
 for i in range(m):  
 table\_row.append(V[i])  
 table\_row.append('')  
 table.add\_row(table\_row)  
 print("\nМетод потенциалов:")  
 print(table)  
 return [value for value in ocenki if value[2] < 0]  
# Конвертация матрицы в граф  
def matrix\_to\_graph(matrix, start):  
 graph = {}  
 for i in range(len(matrix)):  
 for j in range(len(matrix[i])):  
 if matrix[i][j] is not None:  
 neighbors = []  
 # Добавляем всех соседей в пределах матрицы  
 for x in range(len(matrix)):  
 if matrix[x][j] != None and (x, j) != (i, j):  
 neighbors.append((x, j))  
 for y in range(len(matrix[i])):  
 if matrix[i][y] is not None and (i, y) != (i, j):  
 neighbors.append((i, y))  
 graph[(i, j)] = neighbors  
 neighbors = []  
 i, j = start[0], start[1]  
 # Добавляем всех соседей в пределах матрицы  
 for x in range(len(matrix)):  
 if matrix[x][j] != None and (x, j) != (i, j):  
 neighbors.append((x, j))  
 for y in range(len(matrix[i])):  
 if matrix[i][y] is not None and (i, y) != (i, j):  
 neighbors.append((i, y))  
 graph[(i, j)] = neighbors  
 return graph, neighbors  
# Поиск пути в графе  
def find\_path(graph, start, target):  
 stack = [(start, [start])]  
 visited = set()  
 while stack:  
 (vertex, path) = stack.pop()  
 if vertex not in visited:  
 for neighbor in graph[vertex]:  
 if neighbor == target and len(path) < 3:  
 continue  
 if (len(path) > 1 and ((vertex[0] == neighbor[0] and neighbor[0] ==path[len(path)-2][0]) or (vertex[1] == neighbor[1] and neighbor[1] == path[len(path)-2][1]))):  
 continue  
 if neighbor == target:  
 return path + [neighbor]  
 else:  
 stack.append((neighbor, path + [neighbor]))  
 visited.add(vertex)  
 return None  
# Главный цикл (Поиск подходящего пути)  
def main\_cycle(base\_matrix, start):  
 n = len(base\_matrix)  
 m = len(base\_matrix[0])  
 start = start[:-1]  
 graph, targets = matrix\_to\_graph(base\_matrix, start)  
 paths = []  
 for target in targets:  
 path = find\_path(graph, start, target)  
 if path:  
 paths.append(path)  
 for path in paths:  
 even = [path[i] for i in range(1, len(path), 2)]  
 min\_idx = min(even, key=lambda x: base\_matrix[x[0]][x[1]])  
 min\_even = base\_matrix[min\_idx[0]][min\_idx[1]]  
 if min\_even == 0:  
 continue  
 one\_None = True  
 base\_matrix[start[0]][start[1]] = min\_even  
 for i in range(1, len(path)):  
 if i % 2 != 0:  
 base\_matrix[path[i][0]][path[i][1]] -= min\_even  
 if base\_matrix[path[i][0]][path[i][1]] == 0 and one\_None:  
 one\_None = False  
 base\_matrix[path[i][0]][path[i][1]] = None  
 else:  
 base\_matrix[path[i][0]][path[i][1]] += min\_even  
 return base\_matrix, path  
 return False  
A = [50, 55, 60, 20]  
B = [30, 60, 40, 20, 15, 20]  
A\_copy = A.copy()  
B\_copy = B.copy()  
tariff\_matrix = [  
[3, 7, 3, 4, 0, 0],  
[6, 2, 5, 7, 4, 0],  
[8, 5, 8, 3, 4, 0],  
[1, 3, 6, 5, 3, 0]  
]  
base = min\_price(A\_copy, B\_copy, tariff\_matrix)  
#base = north\_west(A\_copy, B\_copy)  
table = PrettyTable()  
field\_names = ["Карьер \\ Дорога"]  
for i in range(len(B)):  
 field\_names.append("B{}".format(i+1))  
field\_names.append("Запас")  
table.field\_names = field\_names  
for i in range(len(A)):  
 table\_row = []  
 table\_row.append("A{}".format(i+1))  
 for j in range(len(B)):  
 if base[i][j] != None:  
 table\_row.append(str(base[i][j]) + ' / ' + str(tariff\_matrix[i][j]))  
 else:  
 table\_row.append(' ' + ' / ' + str(tariff\_matrix[i][j]))  
 table\_row.append(A[i])  
 table.add\_row(table\_row)  
table\_row = []  
table\_row.append("Потребность")  
for i in range(len(B)):  
 table\_row.append(B[i])  
table\_row.append('')  
table.add\_row(table\_row)  
print("Начальное решение:")  
print(table)  
result = 0  
for i in range(len(A)):  
 for j in range(len(B)):  
 if base[i][j] != None:  
 result += base[i][j] \* tariff\_matrix[i][j]  
print("S = " + str(result))  
while True:  
 ocenki = potential\_method(base, A, B, tariff\_matrix)  
 print()  
 if ocenki:  
 print("Есть отрицательные оценки.")  
 for value in ocenki:  
 print("A{}B{} = {}".format(value[0]+1, value[1]+1, value[2]))  
 print("\nПереходим к поиску пути.")  
 else:  
 print("Отрицательных оценок нет.\nКонец.")  
 break  
 start = min(ocenki, key=lambda x: x[2])  
 test = main\_cycle(base, start)  
 while not test:  
 index = ocenki.index(start)  
 ocenki.pop(index)  
 if len(ocenki) == 0:  
 break  
 else:  
 start = min(ocenki, key=lambda x: x[2])  
 test = main\_cycle(base, start)  
 if not test:  
 print("Невозможно найти путь!")  
 break  
 base = test[0]  
 print("Путь найден:")  
 for value in test[1]:  
 print("A{}B{} -> ".format(value[0]+1, value[1]+1), end="")  
 print("A{}B{}\n".format(test[1][0][0]+1, test[1][0][1]+1))  
 print("Новое решение:")  
 table.clear\_rows()  
 for i in range(len(A)):  
 table\_row = []  
 table\_row.append("A{}".format(i+1))  
 for j in range(len(B)):  
 if base[i][j] != None:  
 table\_row.append(str(base[i][j]) + ' / ' + str(tariff\_matrix[i][j]))  
 else:  
 table\_row.append(' ' + ' / ' + str(tariff\_matrix[i][j]))  
 table\_row.append(A[i])  
 table.add\_row(table\_row)  
 table\_row = []  
 table\_row.append("Потребность")  
 for i in range(len(B)):  
 table\_row.append(B[i])  
 table\_row.append('')  
 table.add\_row(table\_row)  
 print(table)  
result = 0  
for i in range(len(A)):  
 for j in range(len(B)):  
 if base[i][j] != None:  
 result += base[i][j] \* tariff\_matrix[i][j]  
print()  
print("Smin = " + str(result))

# **Результат работы**







# **Заключение**

В данной лабораторной работе были рассмотрены модели транспортной задачи. Общая постановка транспортной задачи состоит в определении оптимального плана перевозок некоторого однородного груза из m пунктов отправления в n пунктов назначения. Также были рассмотрены методы северо-западного угла и наименьшей стоимости, совместно с методом потенциалов..

В процессе выполнения лабораторной работы возникли трудности с поиском пути в методе потенциалов, проблема была решена созданием графа и применением поиска в глубину.

Применение метода наименьшей оценки показало лучший результат в плане количества итераций по сравнению с методом северо-западного угла. Но в итоге результат оказался одинаковым для обоих методов.

# **Список использованных источников**

1. Сорокин А.В. Использование алгоритма Джонсона для решения задачи упорядочения. Методические указания. - Барнаул: АлтГТУ, 2015. – 32 с. – [Электронный ресурс]. – url: <http://elib.altstu.ru/eum/download/pm/Sorokin_alg_Johnson.pdf>
2. Алпатов, Ю.Н. Математическое моделирование производственных процессов [Электронный ресурс] : учеб. пособие / Ю.Н. Алпатов. — Электрон. дан. — Санкт Петербург : Лань, 2018. — 136 с. — Режим доступа: https://e.lanbook.com/book/107271. — Загл. с экрана.
3. Голубева, Н.В. Математическое моделирование систем и процессов [Электронный ресурс] : учебное пособие / Н.В. Голубева. — Электрон. дан. — Санкт-Петербург : Лань, 2016. — 192 с. — Режим доступа: https://e.lanbook.com/book/76825. — Загл. с экрана
4. Кудрявцев, Е.М. GPSS World. Основы имитационного моделирования различных систем [Электронный ресурс] / Е.М. Кудрявцев. — Электрон. дан. — Москва : ДМК Пресс, 2008. — 317 с. — Режим доступа: https://e.lanbook.com/book/1213. — Загл. с экрана.
5. Березовская, Е.А. Имитационное моделирование : учебное пособие / Е.А. Березовская ; Министерство образования и науки Российской Федерации, Федеральное государственное автономное образовательное учреждение высшего образования «Южный федеральный университет», Экономический факультет. - Ростов-на-Дону ; Таганрог : Издательство Южного федерального университета, 2018. - 76 с. : ил. - Библиогр. в кн. - ISBN 978-5 9275- 2426-6 ; То же [Электронный ресурс]. - URL: http://biblioclub.ru/index.php?page=book&id=499496 (21.04.2019)
6. Лисяк, Н.К. Моделирование систем : учебное пособие / Н.К. Лисяк, В.В. Лисяк ; Министерство образования и науки Российской Федерации, Федеральное государственное автономное образовательное учреждение высшего образования «Южный федеральный университет», Инженерно-технологическая академия. - Ростов-на-Дону ; Таганрог : Издательство Южного федерального университета, 2017. - Ч. 1. - 107 с. : ил. - Библиогр.: с. 101 - 102 - ISBN 978-5-9275- 2504-1 ; То же [Электронный ресурс]. - URL: http://biblioclub.ru/index.php?page=book&id=499733 (21.04.2019)
7. Аверченков, В.И. Основы математического моделирования технических систем : учебное пособие / В.И. Аверченков, В.П. Федоров, М.Л. Хейфец. - 3-е изд., стер. - Москва : Издательство «Флинта», 2016. - 271 с. : схем., ил. - Библиогр. в кн. - ISBN 978-5-9765-1278- 8 ; То же [Электронный ресурс]. - URL: http://biblioclub.ru/index.php?page=book&id=93344 (21.04.2019).
8. Теория систем массового обслуживания : учебное пособие / сост. А.В. Шапошников, В.В. Бережной, А.М. Лягин, А.А. Плетухина и др. - Ставрополь : СКФУ, 2017. - 134 с. : ил. - Библиогр. в кн. ; То же [Электронный ресурс]. - URL: http://biblioclub.ru/index.php?page=book&id=483842 (24.04.2019).
9. Поршнев, С.В. Компьютерное моделирование физических процессов в пакете MATLAB [Электронный ресурс] : учебное пособие / С.В. Поршнев. — Электрон. дан. — Санкт Петербург : Лань, 2011. — 736 с. — Режим доступа: https://e.lanbook.com/book/650. — Загл. с экрана.
10. Материалы сайта "Интернет университет информационных технологий" , сетевой адрес: http://www.intuit.ru/, Костюкова Н. Основы математического моделирования. - НГУ, сетевой адрес: https://www.intuit.ru/studies/courses/66/66/info Материалы сайта "Единое окно доступа к образовательным ресурсам", Андриевский А.Б.,
11. Андриевский Б.Р., Капитонов А.А., Фрадков А.Л. Решение инженерных задач в среде Scilab. Учебное пособие. - СПб.: НИУ ИТМО, 2013. — 97 с. – [Электронный ресурс]. – url:http://window.edu.ru/resource/044/80044/files/itmo1329.pdf