Федеральное агентство связи
Федеральное государственное бюджетное образовательное учреждение высшего образования
«Сибирский государственный университет телекоммуникаций и информатики»

Кафедра прикладной математики и кибернетики

Лабораторная работа № 3
«Нахождение начального опорного плана транспортной задачи»
по дисциплине «Алгоритмы и вычислительные методы оптимизации»

Бригада № 1

Выполнил:
студент 3 курса
группы ИП-811
Мироненко К. А

Проверил:
ассистент кафедры ПМиК
Новожилов Д.И.

Новосибирск 2020-21 уч. год

Оглавление

1. Постановка задачи

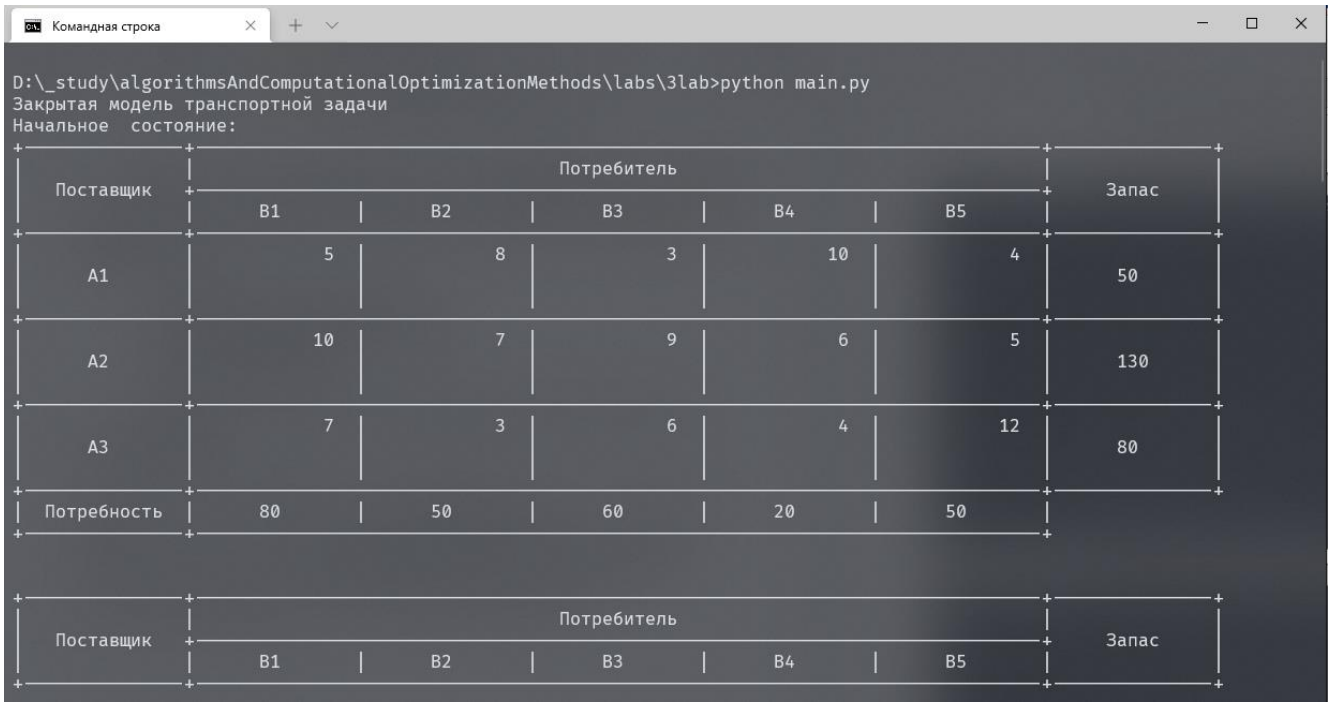
Написать программу, находящую начальный опорный план транспортной задачи одним из указанных методов (номер метода находится как n mod 3, где n – номер бригады).

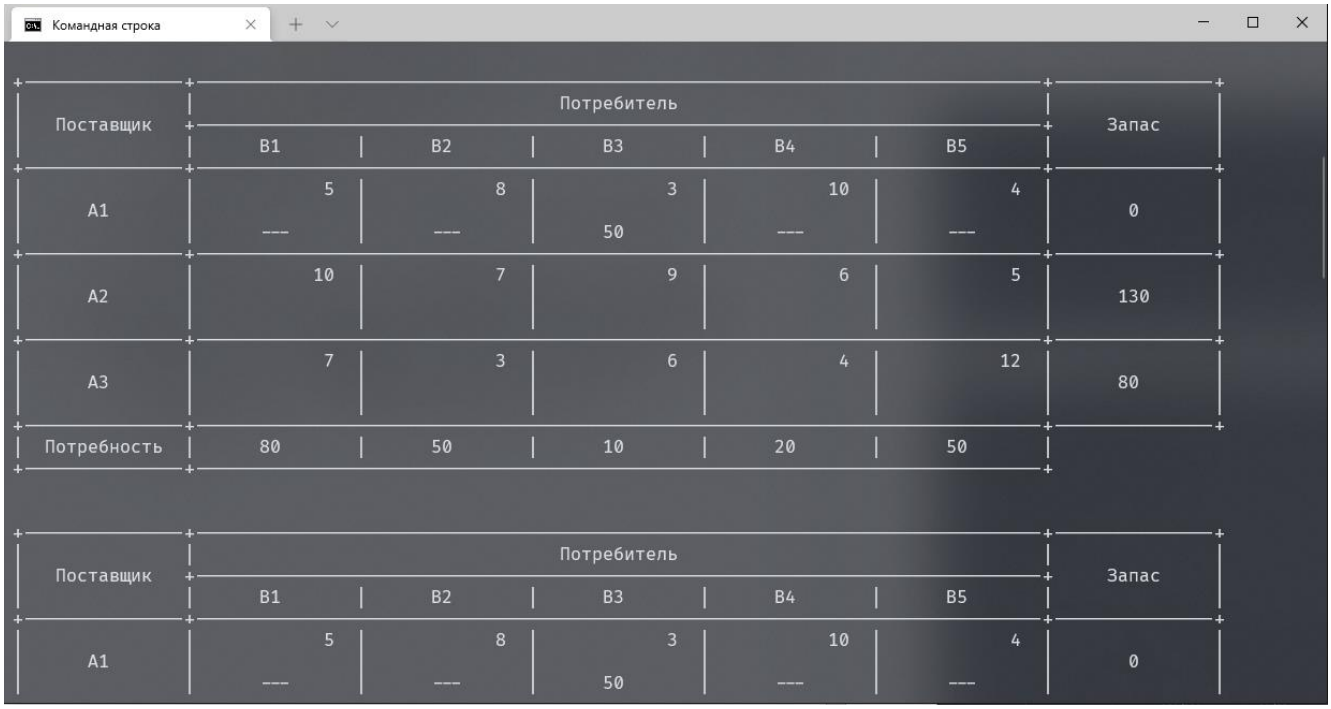
0. Метод северо-западного угла.
1. Метод минимальной стоимости.
2. Метод Фогеля

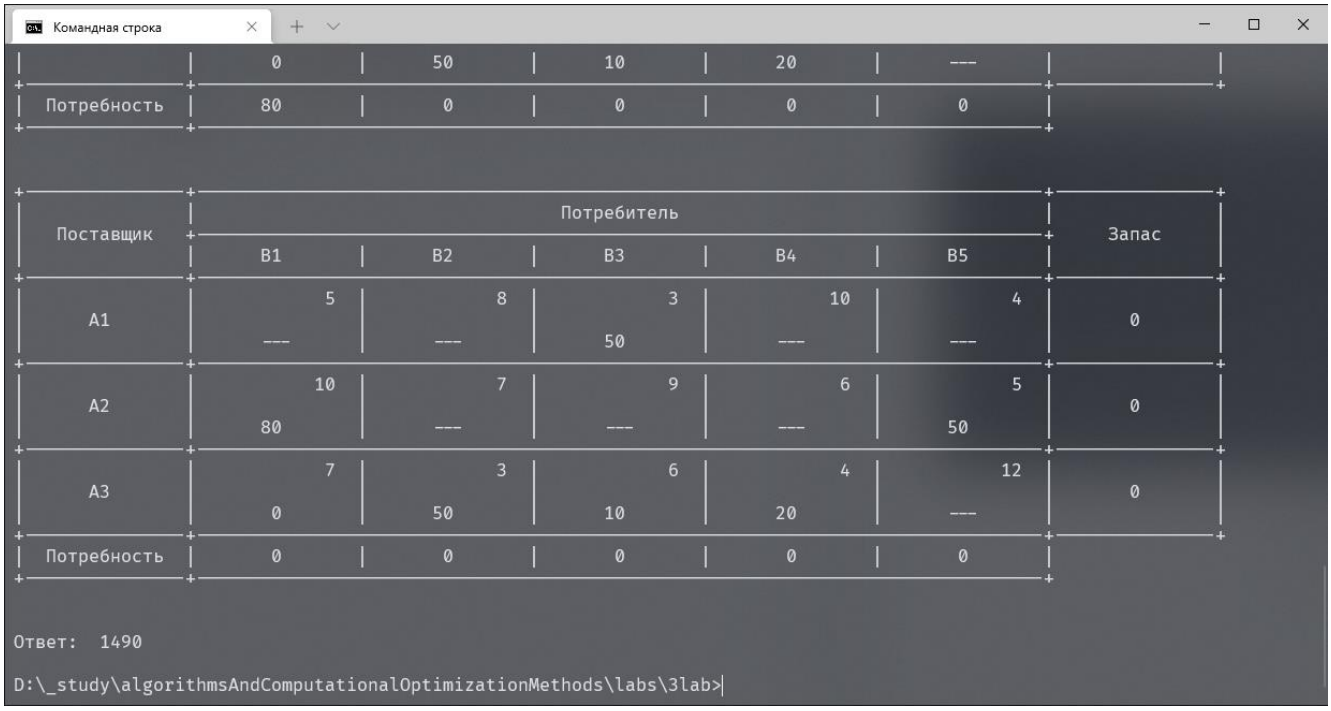
Матрицу тарифов, запасы поставщиков и потребности потребителей вводить из файла.

Программа должна работать как с открытой, так и закрытой моделью транспортной задачи. Предусмотреть программное нахождение вырожденного плана. Вывести распределение перевозок и затраты. Для тестирования использовать несколько заданий из практических занятий.

2. Примеры работы программы






Командная строка
	0	50	10	20	---	
Потребность	80	0	0	0	0	
Поставщик	Потребитель					Запас
	B1	B2	B3	B4	B5	
A1	5 ---	8 ---	3 50	10 ---	4 ---	0
A2	10 80	7 ---	9 ---	6 ---	5 50	0
A3	7 0	3 50	6 10	4 20	12 ---	0
Потребность	0	0	0	0	0	
Ответ: 1490
D:_study\algorithmsAndComputationalOptimizationMethods\labs\3lab>

Приложение Листинг

```
import sys


def print_step(stocs, needs, matrix):
    cell_width = 15
    print(("+{:-^15}+{:-^" + str(cell_width * len(matrix[0]) + len(matrix[0]) - 1) + "}+{:-^" + str(cell_width)
+"}+").format(", ", "))
    print(("|{:^15}|{:^" + str(cell_width * len(matrix[0]) + len(matrix[0]) - 1) + "}|{:^" + str(cell_width) +
"}|").format(", "Потребитель", "))
    print(("|{:^15}+{:-^" + str(cell_width * len(matrix[0]) + len(matrix[0]) - 1) + "}+{:^" + str(cell_width)
+"}|").format("Поставщик", ", "Запас"))

    print("|{:^15}|".format(''), end='')
    for i in range(len(matrix[0])):
        print(("{:^" + str(cell_width) + "}|").format("B" + str(i + 1)), end='')
    print(("{:^" + str(cell_width) + "}|").format(''))
    print(("+{:-^15}+{:-^" + str(cell_width * len(matrix[0]) + len(matrix[0]) - 1) + "}+{:-^" + str(cell_width) +
"}+").format(", ", "))
    for line in enumerate(matrix):
        print("|{:^15}|".format(''), end='')
        for el in line[1]:
            print(("{:>" + str(cell_width - 2) + "}  |").format("(0)" if el["tariff"] == (sys.maxsize - 1) else
el["tariff"]), end='')
        print(("{:^" + str(cell_width) + "}|").format(''))
        print("|{:^15}|".format("A" + str(line[0] + 1)), end='')
        for el in line[1]:
            print(("{:^" + str(cell_width) + "}|").format(''), end='')
        print(("{:^" + str(cell_width) + "}|").format(stocs[line[0]]))
        print("|{:^15}|".format(''), end='')
        for el in line[1]:
            print(("{:^" + str(cell_width) + "}|").format("" if el["quantity"] is None else ('---' if el["quantity"] == -1
else el["quantity"])), end='')
        print(("{:^" + str(cell_width) + "}|").format(''))
        print(("+{:-^15}+{:-^" + str(cell_width * len(matrix[0]) + len(matrix[0]) - 1) + "}+{:-^" + str(cell_width) +
"}+").format(", ", "))

    print("|{:^15}|".format("Потребность"), end='')
    for i in needs:
        print(("{:^" + str(cell_width) + "}|").format(str(i)), end='')
    print(("\n+{:-^15}+{:-^" + str(cell_width * len(matrix[0]) + len(matrix[0]) - 1) + "}+").format(", "))


def main():
    # Чтение из файла
    with open('input.txt', 'r', encoding="utf-8") as f:
        lines = list(filter(lambda x: x != '' and '#' not in x, list(map(lambda x: x.strip(), f.readlines()))))
    f.close()
    # print(lines)

    # "Распарс" по переменным
    stocks = list(map(int, lines[0].split(' ')))  # Запасы
    needs = list(map(int, lines[1].split(' ')))  # Потребности
    matrix = list(list(dict(tariff=int(y), quantity=None, used=False) for y in x.split(' ')) for x in lines[2:])
    # print(stocks, needs, '', *matrix, sep='\n')
```

```
res = 0  # Ответ

# Проверка на тип модели и добавление в случае чего
if sum(stocks) < sum(needs):
    print("Открытая модель транспортной задачи (фиктивный поставщик)")
    matrix.append(list(dict(tariff=sys.maxsize - 1, quantity=None, used=False) for _ in range(len(matrix[0]))))
    stocks.append(sum(needs) - sum(stocks))
elif sum(stocks) > sum(needs):
    print("Открытая модель транспортной задачи (фиктивный потребитель)")
    for i in range(len(matrix)):
        matrix[i].append(dict(tariff=sys.maxsize - 1, quantity=None, used=False))
    needs.append(sum(stocks) - sum(needs))
else:
    print("Закрытая модель транспортной задачи")

print("Начальное  состояние:")
print_step(stocks, needs, matrix)
print('\n')

while sum(stocks) != 0 and sum(needs) != 0:
    min_el = sys.maxsize
    row = -1
    column = -1

    # Поиск минимального тарифа
    for i in range(len(matrix)):
        for j in range(len(matrix[i])):
            if matrix[i][j]["tariff"] < min_el and not matrix[i][j]["used"]:
                min_el = matrix[i][j]["tariff"]
                row = i
                column = j

    min_el = min(needs[column], stocks[row])
    matrix[row][column]["used"] = True
    matrix[row][column]["quantity"] = min_el
    needs[column] -= min_el
    stocks[row] -= min_el

    # "Вычеркивание" строк/столбцов
    if sum(stocks) != 0 and sum(needs) != 0:
        if needs[column] < stocks[row]:
            for i in matrix:
                if not i[column]["used"]:
                    i[column]["used"] = True
                    i[column]["quantity"] = -1
        elif needs[column] > stocks[row]:
            for i in matrix[row]:
                if not i["used"]:
                    i["used"] = True
                    i["quantity"] = -1
        else:
            min_tmp = sys.maxsize
            row_tmp = -1
            column_tmp = -1
            for i in range(len(matrix)):
```

```
                if not matrix[i][column]["used"]:
                    matrix[i][column]["used"] = True
                    matrix[i][column]["quantity"] = -1
                    if matrix[i][column]["tariff"] < min_tmp:
                        min_tmp = matrix[i][column]["tariff"]
                        row_tmp = i
                        column_tmp = column
            for j in range(len(matrix[row])):
                if not matrix[row][j]["used"]:
                    matrix[row][j]["used"] = True
                    matrix[row][j]["quantity"] = -1
                    if matrix[row][j]["tariff"] < min_tmp:
                        min_tmp = matrix[row][j]["tariff"]
                        row_tmp = row
                        column_tmp = j
            matrix[row_tmp][column_tmp]["quantity"] = 0

        print_step(stocks, needs, matrix)
        print('\n')
        res += 0 if matrix[row][column]["tariff"] == (sys.maxsize - 1) else matrix[row][column]["tariff"] *
matrix[row][column]["quantity"]

    print("Ответ:  " + str(res))
    return


if __name__ == '__main__':
    main()
    # test()
```