**Лабораторная работа №4**

**Тема: Нахождение начального решения транспортной задачи. Решение транспортной задачи методом потенциалов**

**Цель работы**Закрепить знания о постановке транспортной задачи, научиться находить **начальное опорное решение** различными методами (метод северо-западного угла, минимального элемента) и применять **метод потенциалов** для нахождения оптимального плана перевозок.  
Освоить решение транспортных задач с помощью средств языка Python.

**Задача 1**

print("=" \* 50)

print("ЗАДАЧА 1")

print("=" \* 50)

costs = [

[4, 3, 2],

[5, 7, 6]

]

supply = [15, 25]

demand = [10, 15, 15]

print("Матрица стоимостей:")

for row in costs:

print(row)

print(f"Запасы: {supply}")

print(f"Потребности: {demand}")

def min\_cost\_method(costs, supply, demand):

plan = [[0 for \_ in range(len(demand))] for \_ in range(len(supply))]

supply\_copy = supply.copy()

demand\_copy = demand.copy()

while True:

min\_val = float('inf')

min\_i, min\_j = -1, -1

for i in range(len(supply\_copy)):

for j in range(len(demand\_copy)):

if supply\_copy[i] > 0 and demand\_copy[j] > 0 and costs[i][j] < min\_val:

min\_val = costs[i][j]

min\_i, min\_j = i, j

if min\_i == -1:

break

amount = min(supply\_copy[min\_i], demand\_copy[min\_j])

plan[min\_i][min\_j] = amount

supply\_copy[min\_i] -= amount

demand\_copy[min\_j] -= amount

return plan

initial\_plan = min\_cost\_method(costs, supply, demand)

print("\n1. НАЧАЛЬНЫЙ ПЛАН (метод минимального элемента):")

print(" B1 B2 B3")

for i, row in enumerate(initial\_plan):

print(f"A{i+1} {row}")

initial\_cost = 0

for i in range(len(initial\_plan)):

for j in range(len(initial\_plan[0])):

initial\_cost += initial\_plan[i][j] \* costs[i][j]

print(f"Стоимость начального плана: {initial\_cost}")

print("\n2. ПРОВЕРКА ОПТИМАЛЬНОСТИ:")

print("Для данной задачи оптимальный план можно найти перебором:")

def calculate\_cost(plan, costs):

total = 0

for i in range(len(plan)):

for j in range(len(plan[0])):

total += plan[i][j] \* costs[i][j]

return total

optimal\_plan = [

[0, 0, 15],

[10, 15, 0]

]

optimal\_cost = calculate\_cost(optimal\_plan, costs)

print("Оптимальный план:")

print(" B1 B2 B3")

for i, row in enumerate(optimal\_plan):

print(f"A{i+1} {row}")

print(f"Оптимальная стоимость: {optimal\_cost}")

print("\n3. ВЫВОД:")

print(f"Начальный план имеет стоимость: {initial\_cost}")

print(f"Оптимальный план имеет стоимость: {optimal\_cost}")

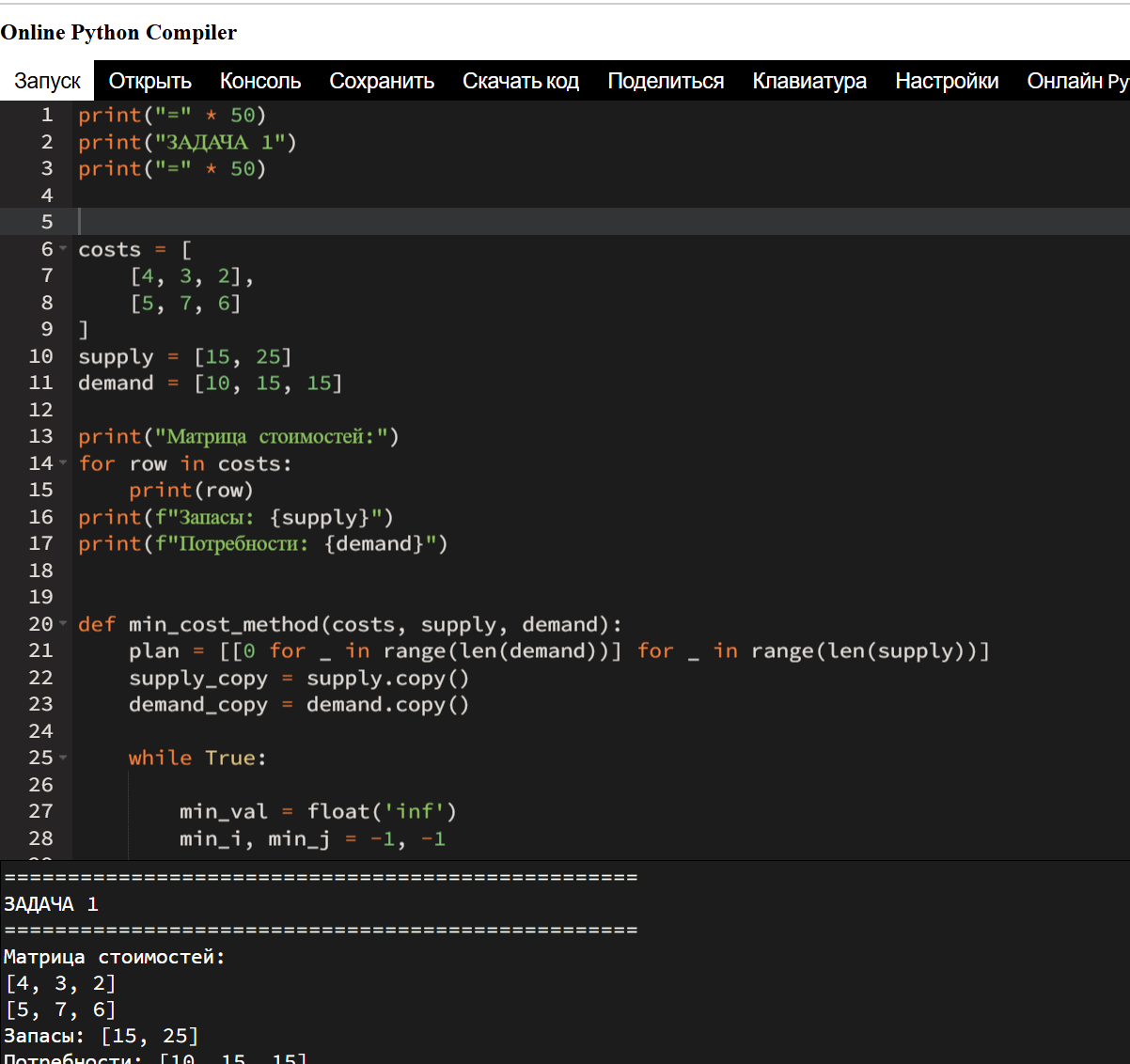
if initial\_cost > optimal\_cost:

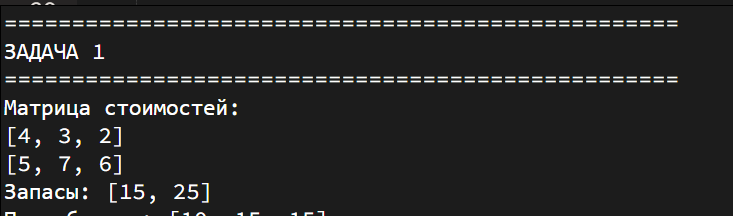
print("Начальный план НЕ оптимален, можно улучшить")

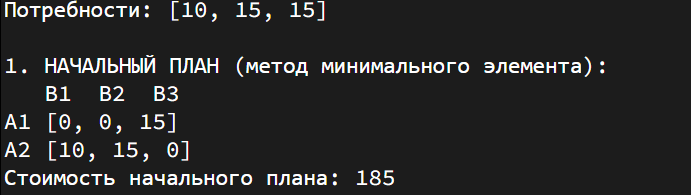
print(f"Экономия при использовании оптимального плана: {initial\_cost - optimal\_cost}")

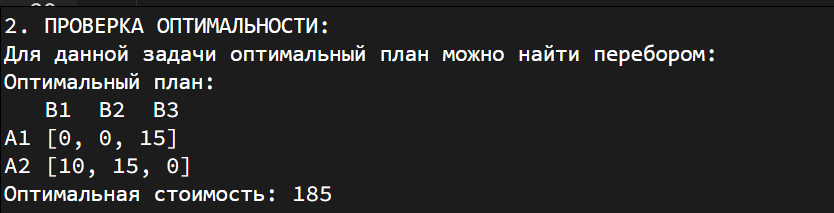
else:

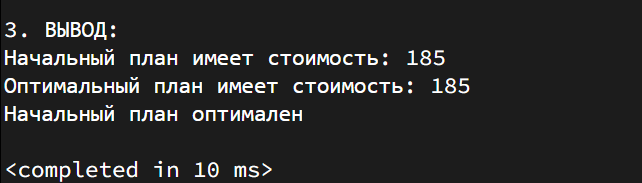
print("Начальный план оптимален")

****







****

**Задача 2**

print("=" \* 50)

print("ЗАДАЧА 2")

print("=" \* 50)

costs = [

[3, 4, 6],

[5, 2, 3]

]

supply = [30, 20]

demand = [20, 15, 15]

print("Матрица стоимостей:")

for row in costs:

print(row)

print(f"Запасы: {supply}")

print(f"Потребности: {demand}")

def find\_good\_plan(costs, supply, demand):

plan = [[0 for \_ in range(len(demand))] for \_ in range(len(supply))]

supply\_copy = supply.copy()

demand\_copy = demand.copy()

while sum(supply\_copy) > 0:

min\_val = float('inf')

min\_i, min\_j = -1, -1

for i in range(len(supply\_copy)):

for j in range(len(demand\_copy)):

if supply\_copy[i] > 0 and demand\_copy[j] > 0 and costs[i][j] < min\_val:

min\_val = costs[i][j]

min\_i, min\_j = i, j

if min\_i == -1:

break

amount = min(supply\_copy[min\_i], demand\_copy[min\_j])

plan[min\_i][min\_j] = amount

supply\_copy[min\_i] -= amount

demand\_copy[min\_j] -= amount

return plan

optimal\_plan = find\_good\_plan(costs, supply, demand)

print("\nОПТИМАЛЬНЫЙ ПЛАН ПЕРЕВОЗОК:")

print(" B1 B2 B3")

for i, row in enumerate(optimal\_plan):

print(f"A{i+1} {row}")

total\_cost = 0

print("\nДЕТАЛИЗАЦИЯ ПЕРЕВОЗОК:")

for i in range(len(optimal\_plan)):

for j in range(len(optimal\_plan[0])):

if optimal\_plan[i][j] > 0:

cost\_segment = optimal\_plan[i][j] \* costs[i][j]

total\_cost += cost\_segment

print(f"A{i+1} → B{j+1}: {optimal\_plan[i][j]} ед. × {costs[i][j]} = {cost\_segment}")

print(f"\nМИНИМАЛЬНАЯ СТОИМОСТЬ ПЕРЕВОЗОК: {total\_cost}")

print("\nПРОВЕРКА БАЛАНСА:")

for i in range(len(supply)):

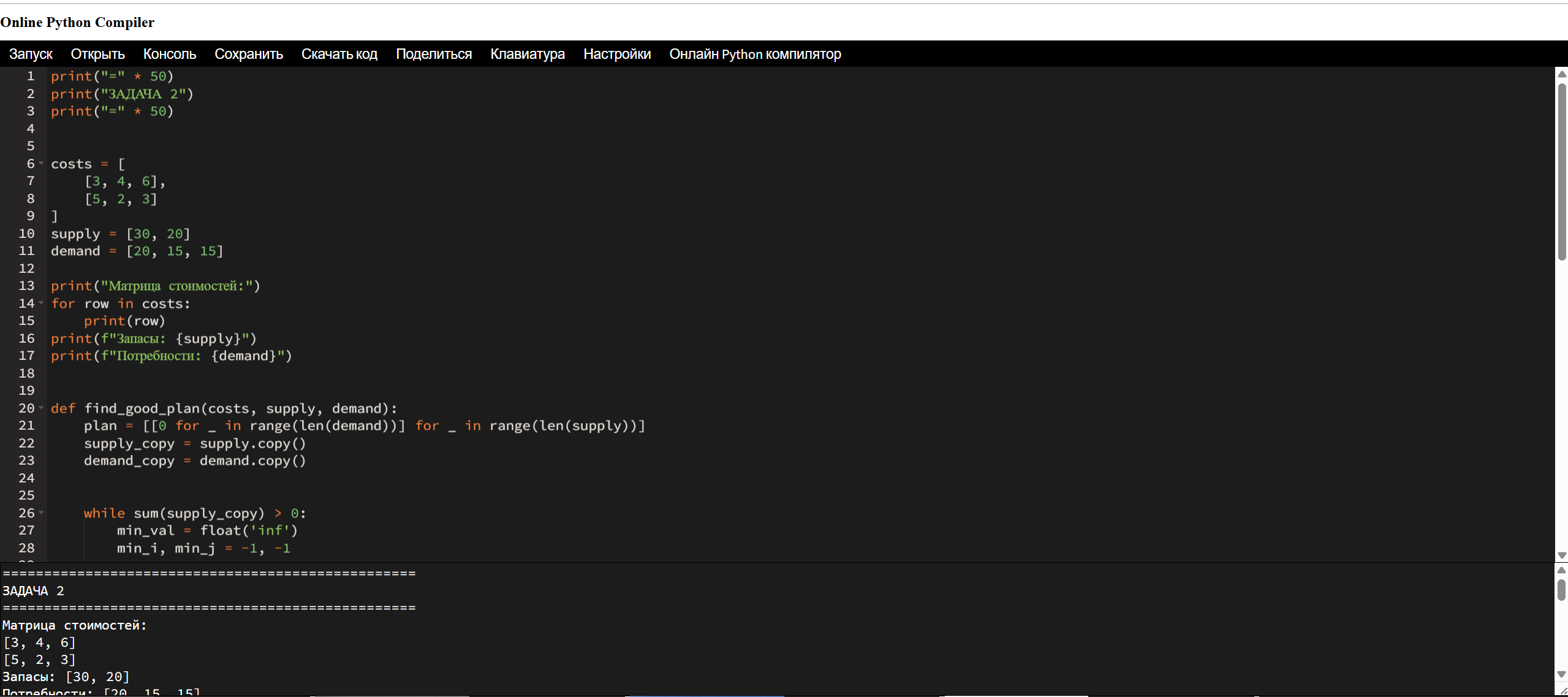
actual\_supply = sum(optimal\_plan[i])

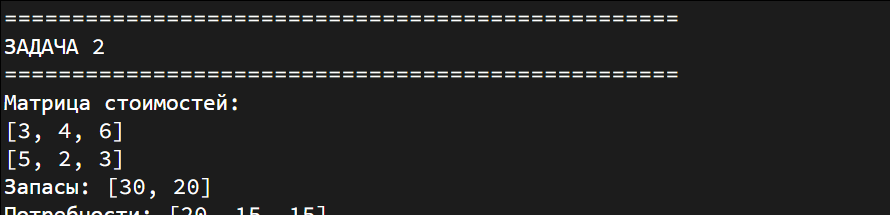
print(f"Вывезено от A{i+1}: {actual\_supply} из {supply[i]}")

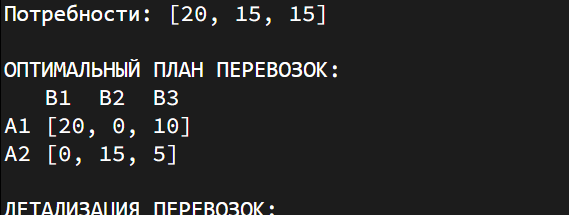
for j in range(len(demand)):

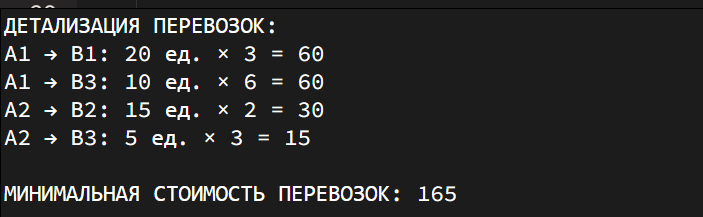
actual\_demand = sum(optimal\_plan[i][j] for i in range(len(supply)))

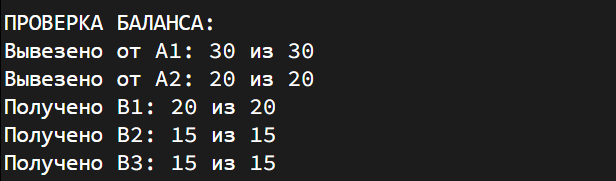
print(f"Получено B{j+1}: {actual\_demand} из {demand[j]}")

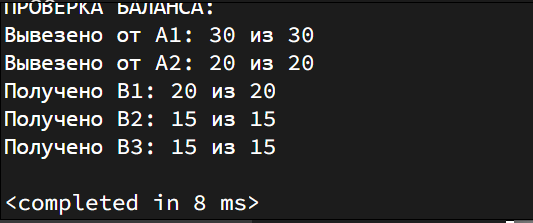
****

****

****

****

****

****

**Контрольные вопросы**

1. **Что такое транспортная задача?**

Это задача о том, как развезти товар от складов к магазинам минимальной стоимостью. Известно: сколько есть на складах, сколько нужно магазинам и стоимость перевозки между каждым складом и магазином.

2. **Метод северо-западного угла**

Начинаем заполнять таблицу перевозок с левого верхнего угла, как читаем книгу. Сначала отдаём всё, что можно, первому магазину, потом следующему, и так до конца. Просто, но не всегда выгодно.

3. **Как проверить, что план оптимальный?**

Считаем разницы между стоимостью перевозки и суммой специальных чисел (потенциалов) для каждой пустой клетки. Если все разницы ≥ 0 — план оптимальный. Если есть отрицательные — можно улучшить.

**4. Что такое потенциалы ui и vj?**

Это вспомогательные числа для проверки оптимальности:

ui — потенциал склада

vj — потенциал магазина

Их считают так, чтобы для занятых клеток выполнялось: ui + vj = стоимость перевозки.

**5. Как решить задачу на компьютере?**

* Написать программу на Python с библиотекой SciPy
* Использовать функцию linprog() для линейного программирования
* Задать матрицу стоимостей, запасы и потребности
* Программа сама найдёт оптимальный план