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 №5

Задача о перекладке оборудования

название работы

ЛР 09.03.04.23.005 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 2023

# **Аннотация**

В данной лабораторной работе ознакомились с постановкой задачи о переналадке оборудования, сводящейся к задаче коммивояжера. Был изучен и применен алгоритм Литтла для решения задачи коммивояжера, построенный на основе метода ветвей и границ. Для решения задачи о переналадке оборудования была написана программа.

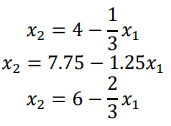
Оглавление

[**Аннотация** 2](#_Toc154646241)

[**Задание** 4](#_Toc154646242)

[**Математическая модель** 5](#_Toc154646243)

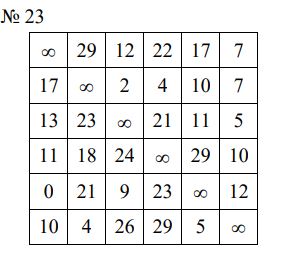
[**Графический метод** 6](#_Toc154646244)

[ 6](#_Toc154646245)

[**Заключение** 7](#_Toc154646246)

[**Вопросы при защите** 7](#_Toc154646247)

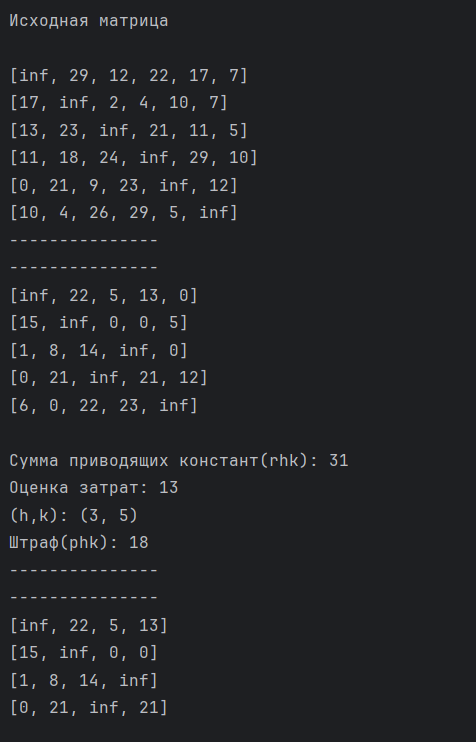
# **Задание**

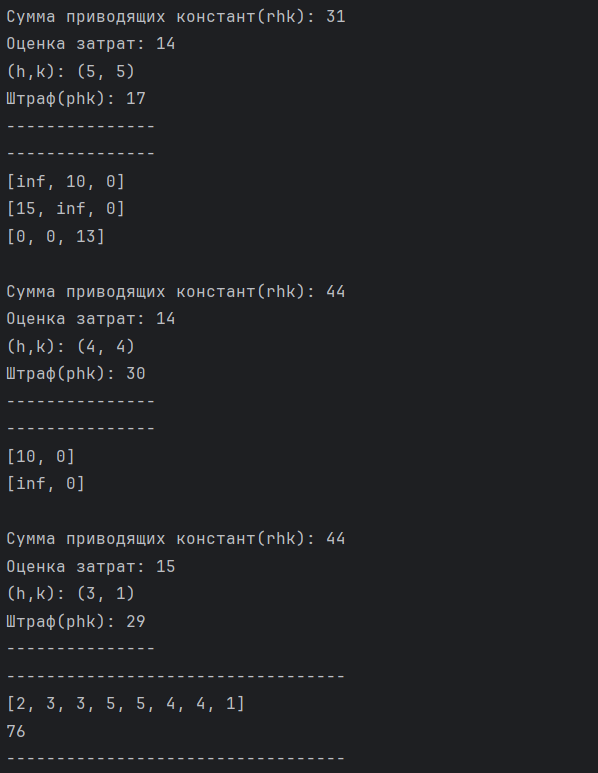


# **Программное решение**

# -\*- coding: windows-1251 -\*-  
#Функция нахождения минимального элемента, исключая текущий элемент  
def Min(lst, myindex):  
 return min(x for idx, x in enumerate(lst) if idx != myindex)  
#функция удаления нужной строки и столбцах  
def Delete(matrix, index1, index2):  
 del matrix[index1]  
 for i in matrix:  
 del i[index2]  
 return matrix  
#Функция вывода матрицы  
def PrintMatrix(matrix, rhk, estimate, h, k, phk):  
 print("---------------")  
 for i in range(len(matrix)):  
 print(matrix[i])  
 print(f"\nСумма приводящих констант(rhk): {rhk}\nОценка затрат: {estimate}\n(h,k): ({h + 1}, {k + 1})\nШтраф(phk): {phk}")  
 print("---------------")  
n = 6  
matrix = [  
 [0, 29, 12, 22, 17, 7],  
 [17, 0, 2, 4, 10, 7],  
 [13, 23, 0, 21, 11, 5],  
 [11, 18, 24, 0, 29, 10],  
 [0, 21, 9, 23, 0, 12],  
 [10, 4, 26, 29, 5, 0]  
]  
H = 0  
PathLength = 0  
Str = []  
Stb = []  
res = []  
result = []  
StartMatrix = []  
#Инициализируем массивы для сохранения индексов  
for i in range(n):  
 Str.append(i)  
 Stb.append(i)  
#Сохраняем изначальную матрицу  
for i in range(n):  
 StartMatrix.append(matrix[i].copy())  
#Присваеваем главной диагонали float(inf)  
for i in range(n):  
 matrix[i][i] = float('inf')  
print("---------------")  
print("Исходная матрица\n")  
for i in range(len(matrix)):  
 print(matrix[i])  
print("---------------")  
while True:  
 #Редуцируем  
 #--------------------------------------  
 #Вычитаем минимальный элемент в строках  
 for i in range(len(matrix)):  
 temp = min(matrix[i])  
 H += temp  
 for j in range(len(matrix)):  
 matrix[i][j] -= temp  
 #Вычитаем минимальный элемент в столбцах  
 for i in range(len(matrix)):  
 temp = min(row[i] for row in matrix)  
 H += temp  
 for j in range(len(matrix)):  
 matrix[j][i] -= temp  
 #--------------------------------------  
 #Оцениваем нулевые клетки и ищем нулевую клетку с максимальной оценкой  
 #--------------------------------------  
 NullMax = 0  
 index1 = 0  
 index2 = 0  
 tmp = 0  
 for i in range(len(matrix)):  
 for j in range(len(matrix)):  
 if matrix[i][j] == 0:  
 tmp = Min(matrix[i], j) + Min((row[j] for row in matrix), i)  
 if tmp >= NullMax:  
 NullMax = tmp  
 index1 = i  
 index2 = j  
 #--------------------------------------  
 #Находим нужный нам путь, записываем его в res и удаляем все ненужное  
 res.append(Str[index1] + 1)  
 res.append(Stb[index2] + 1)  
 oldIndex1 = Str[index1]  
 oldIndex2 = Stb[index2]  
 if oldIndex2 in Str and oldIndex1 in Stb:  
 NewIndex1 = Str.index(oldIndex2)  
 NewIndex2 = Stb.index(oldIndex1)  
 matrix[NewIndex1][NewIndex2] = float('inf')  
 del Str[index1]  
 del Stb[index2]  
 matrix = Delete(matrix, index1, index2)  
 if len(matrix) == 1:  
 break  
 # Вывод информации о текущем шаге  
 PrintMatrix(matrix, H, NullMax, index1, index2, H - NullMax)  
#Формируем порядок пути  
for i in range(0, len(res) - 1, 2):  
 if res.count(res[i]) < 2:  
 result.append(res[i])  
 result.append(res[i + 1])  
for i in range(0, len(res) - 1, 2):  
 for j in range(0, len(res) - 1, 2):  
 if result[len(result) - 1] == res[j]:  
 result.append(res[j])  
 result.append(res[j + 1])  
# Вывод итоговой информации  
print("----------------------------------")  
print(result)  
#Считаем длину пути  
for i in range(0, len(result) - 1, 2):  
 if i == len(result) - 2:  
 PathLength += StartMatrix[result[i] - 1][result[i + 1] - 1]  
 PathLength += StartMatrix[result[i + 1] - 1][result[0] - 1]  
 else:  
 PathLength += StartMatrix[result[i] - 1][result[i + 1] - 1]  
print(PathLength)  
print("----------------------------------")

# **Результат для матрицы**





# **Полученный маршрут**

4-