## МИНИСТЕРСТВО НАУКИ И ВЫСШЕГО ОБРАЗОВАНИЕ РФ

**ФЕДЕРАЛЬНОЕ ГОСУДАРСТВЕННОЕ БЮДЖЕТНОЕ ОБРАЗОВАТЕЛЬНОЕ УЧРЕЖДЕНИЕ ВЫСШЕГО ОБРАЗОВАНИЯ**

## “ДОНСКОЙ ГОСУДАРСТВЕННЫЙ ТЕХНИЧЕСКИЙ УНИВЕРСИТЕТ”

Кафедра «Программное обеспечение вычислительной техники и автоматизированных систем»

Лабораторная работа №3

По дисциплине: «Эвристические методы и алгоритмы» На тему «Теория расписаний»

Выполнил:

Студент группы ВПР31

Андросов И.А.

Проверил:

Проф. Кобак В.Г.

Ростов-на-Дону 2023 г.

# Введение

Задачи проектирования и управления в системах, для которых необходимо распределение работы между параллельно работающими разнородными вычислительными устройствами занимают значимое место в теории построения расписаний. Практическая актуальность таких задач определяется существенными возможностями экономии машинного времени и вытекающими функциональными и эксплуатационными преимуществами.

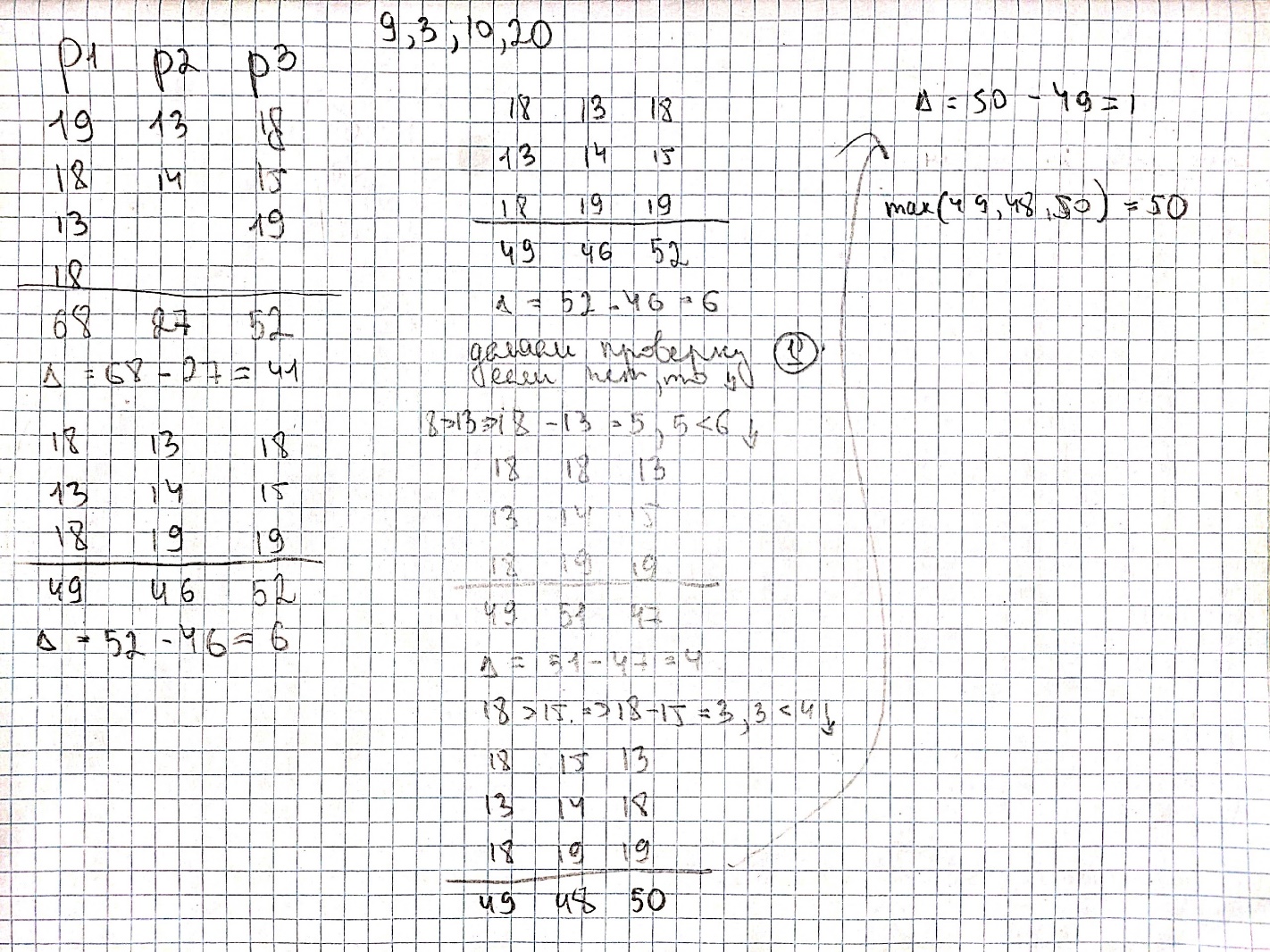
Теоретическая сложность нахождения наилучшего распределения связана с необходимостью решения экстремальных задач комбинаторного типа, требующих больших вычислительных ресурсов, так что эффект от нахождения близкого к оптимальному, с точки зрения времени выполнения, распределения может быть сведен на нет затратами на его получение.

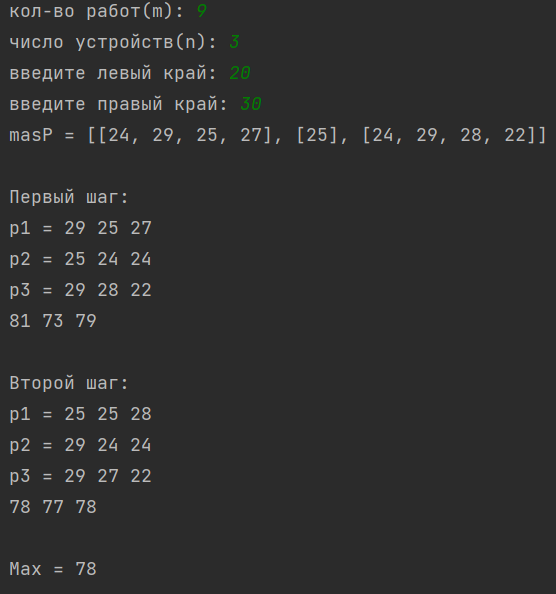
В настоящем руководстве приводятся методы получения расписаний, приводящие к небольшим затратам на вычисление за счет отказа от получения оптимального решения, но в тоже время позволяющие найти приемлемое решение, близкое к оптимальному.

# Постановка задачи

Имеется  независимых работ , которые необходимо распределить на  параллельно работающих разнородных устройств  по критерию .Получение оптимального распределения в такой постановке приводит к громоздким вычислениям, требующим значительного времени машинного счета, поэтому цель – продемонстрировать алгоритмы, с помощью которого можно находить с малыми затратами достаточно приемлемое решение.

# Результаты тестирования программы





# Код программы

import random

import copy

m, n = int(input('количество задач: ')), int(input('число устройств: '))

T1, T2 = int(input('от: ')), int(input('до: '))

mas, masP, masSum = [], [], [0] \* n

for i in range(n):

masP.append([])

while len(mas) < m:

acc = random.randint(T1, T2)

mas.append(acc)

#mas = [12, 13, 14, 15, 19, 10, 11, 13, 18]

#print(mas)

for i in mas:

acc = random.randint(0, n - 1)

masP[acc].append(i)

#masP = [[12, 10, 13], [13, 19, 18], [14, 15, 11]]

#print(masP)

first\_modern, second\_modern = copy.deepcopy(mas), copy.deepcopy(masP)

print('masP =', masP)

for i in masP:

for k in masP: masSum[masP.index(k)] = sum(k)

delt = max(masSum) - min(masSum)

for j in masP[masSum.index(max(masSum))]:

if j < delt:

masP[masSum.index(min(masSum))].append(j)

masP[masSum.index(max(masSum))].remove(j)

break

print()

print('После сравнения прибора с максимальной нагрузкой с дельта: ')

acc\_for\_print = 1

for \_ in masP:

print(f'p{acc\_for\_print} =', \*\_)

acc\_for\_print += 1

print(\*masSum)

print()

print('После сравнений между приборами с наименьшей и наибольшей нагрузкой:')

second, new\_prov, acc1, acc2, delt = True, True, -1, -1, max(masSum) - min(masSum)

while second:

for \_ in masP: masSum[masP.index(\_)] = sum(\_)

while new\_prov:

masPprov = copy.deepcopy(masP)

for \_ in masP[masSum.index(max(masSum))]:

if \_ < delt:

masP[masSum.index(min(masSum))].append(\_)

masP[masSum.index(max(masSum))].remove(\_)

for \_ in masP: masSum[masP.index(\_)] = sum(\_)

delt = max(masSum) - min(masSum)

break

if masPprov == masP:

new\_prov = False

delt = max(masSum) - min(masSum)

masPprov = copy.deepcopy(masP)

for i in masP[masSum.index(max(masSum))]:

acc1 += 1

for j in masP[masSum.index(min(masSum))]:

acc2 += 1

if (i > j) and (i - j < delt):

masP[masSum.index(max(masSum))][acc1], masP[masSum.index(min(masSum))][acc2] = \

masP[masSum.index(min(masSum))][acc2], masP[masSum.index(max(masSum))][acc1]

acc1, acc2 = -1, -1

break

if masPprov != masP:

break

else:

acc2 = -1

new\_prov = True

else:

second = False

break

acc\_for\_print = 1

for \_ in masP:

print(f'p{acc\_for\_print} =', \*\_)

acc\_for\_print += 1

print(\*masSum)

print()

print('Max =', max(masSum))

# 1 модификация

print()

print('----1 модификация----')

mas, masSum, masNumbers, masx2 = first\_modern, [0] \* n, [], []

print(f'mas = {mas}')

for \_ in range(n):

masx2.append([])

print('Mas:')

for i in mas:

for j in range(n):

print(i, end=' ')

print()

print('\nSort:')

mas.sort()

mas.reverse()

for i in mas:

for j in range(n):

print(i, end=' ')

print()

print('\nSums of appliances:')

acc, full\_acc = 0, 0

print(\*masSum)

while full\_acc < m:

number = mas[full\_acc]

if acc < n:

masx2[masSum.index(min(masSum))].append(number)

masSum[masSum.index(min(masSum))] += number

acc += 1

full\_acc += 1

print(\*masSum)

continue

if acc == n:

masx2[masSum.index(min(masSum))].append(number)

masSum[masSum.index(min(masSum))] += number

acc = 1

full\_acc += 1

print(\*masSum)

print(f'\nMas = {mas}')

print(f'End number = {max(masSum)}')

print(f'masx2 = {masx2}')

mas, masP, masSum = [], copy.deepcopy(masx2), [0] \* n

print('masP =', masP)

for i in masP:

for k in masP: masSum[masP.index(k)] = sum(k)

delt = max(masSum) - min(masSum)

for j in masP[masSum.index(max(masSum))]:

if j < delt:

masP[masSum.index(min(masSum))].append(j)

masP[masSum.index(max(masSum))].remove(j)

break

print()

print('Первый шаг: ')

acc\_for\_print = 1

for \_ in masP:

print(f'p{acc\_for\_print} =', \*\_)

acc\_for\_print += 1

print(\*masSum)

print()

print('Второй шаг:')

second, new\_prov, acc1, acc2, delt = True, True, -1, -1, max(masSum) - min(masSum)

while second:

for \_ in masP: masSum[masP.index(\_)] = sum(\_)

while new\_prov:

masPprov = copy.deepcopy(masP)

for \_ in masP[masSum.index(max(masSum))]:

if \_ < delt:

masP[masSum.index(min(masSum))].append(\_)

masP[masSum.index(max(masSum))].remove(\_)

for \_ in masP: masSum[masP.index(\_)] = sum(\_)

delt = max(masSum) - min(masSum)

break

if masPprov == masP:

new\_prov = False

delt = max(masSum) - min(masSum)

masPprov = copy.deepcopy(masP)

for i in masP[masSum.index(max(masSum))]:

acc1 += 1

for j in masP[masSum.index(min(masSum))]:

acc2 += 1

if (i > j) and (i - j < delt):

masP[masSum.index(max(masSum))][acc1], masP[masSum.index(min(masSum))][acc2] = \

masP[masSum.index(min(masSum))][acc2], masP[masSum.index(max(masSum))][acc1]

acc1, acc2 = -1, -1

break

if masPprov != masP:

break

else:

acc2 = -1

new\_prov = True

else:

second = False

break

acc\_for\_print = 1

for \_ in masP:

print(f'p{acc\_for\_print} =', \*\_)

acc\_for\_print += 1

print(\*masSum)

print()

print('Max =', max(masSum))

# 2 модификация

print('----2 модификация----')

mas, masP, masSum = [], second\_modern, [0] \* n

print('masP =', masP)

for i in masP:

for \_ in masP: \_.sort(reverse = True)

for k in masP: masSum[masP.index(k)] = sum(k)

delt = max(masSum) - min(masSum)

for j in masP[masSum.index(max(masSum))]:

if j < delt:

masP[masSum.index(min(masSum))].append(j)

masP[masSum.index(max(masSum))].remove(j)

break

print()

print('Первый шаг: ')

acc\_for\_print = 1

for \_ in masP:

print(f'p{acc\_for\_print} =', \*\_)

acc\_for\_print += 1

print(\*masSum)

print()

print('Второй шаг:')

second, new\_prov, acc1, acc2, delt = True, True, -1, -1, max(masSum) - min(masSum)

while second:

for \_ in masP: \_.sort(reverse = True)

for \_ in masP: masSum[masP.index(\_)] = sum(\_)

while new\_prov:

masPprov = copy.deepcopy(masP)

for \_ in masP[masSum.index(max(masSum))]:

if \_ < delt:

masP[masSum.index(min(masSum))].append(\_)

masP[masSum.index(max(masSum))].remove(\_)

for \_ in masP: masSum[masP.index(\_)] = sum(\_)

delt = max(masSum) - min(masSum)

break

if masPprov == masP:

new\_prov = False

delt = max(masSum) - min(masSum)

masPprov = copy.deepcopy(masP)

for i in masP[masSum.index(max(masSum))]:

acc1 += 1

for j in masP[masSum.index(min(masSum))]:

acc2 += 1

if (i > j) and (i - j < delt):

masP[masSum.index(max(masSum))][acc1], masP[masSum.index(min(masSum))][acc2] = \

masP[masSum.index(min(masSum))][acc2], masP[masSum.index(max(masSum))][acc1]

acc1, acc2 = -1, -1

break

if masPprov != masP:

break

else:

acc2 = -1

new\_prov = True

else:

second = False

break

acc\_for\_print = 1

for \_ in masP:

print(f'p{acc\_for\_print} =', \*\_)

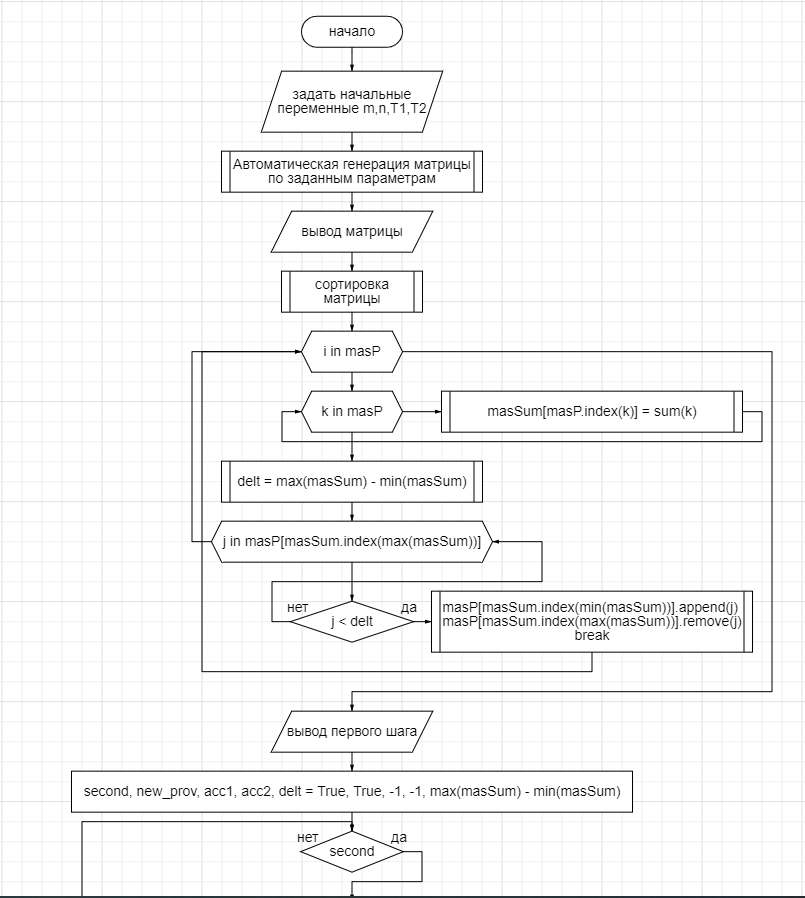
acc\_for\_print += 1

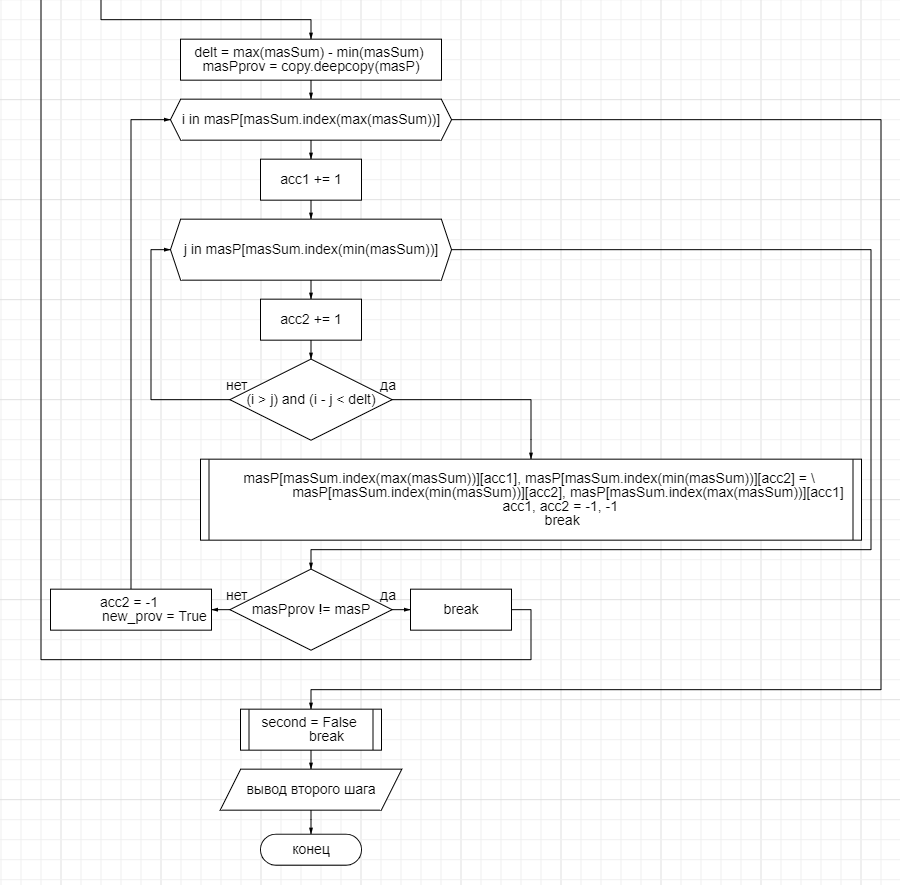
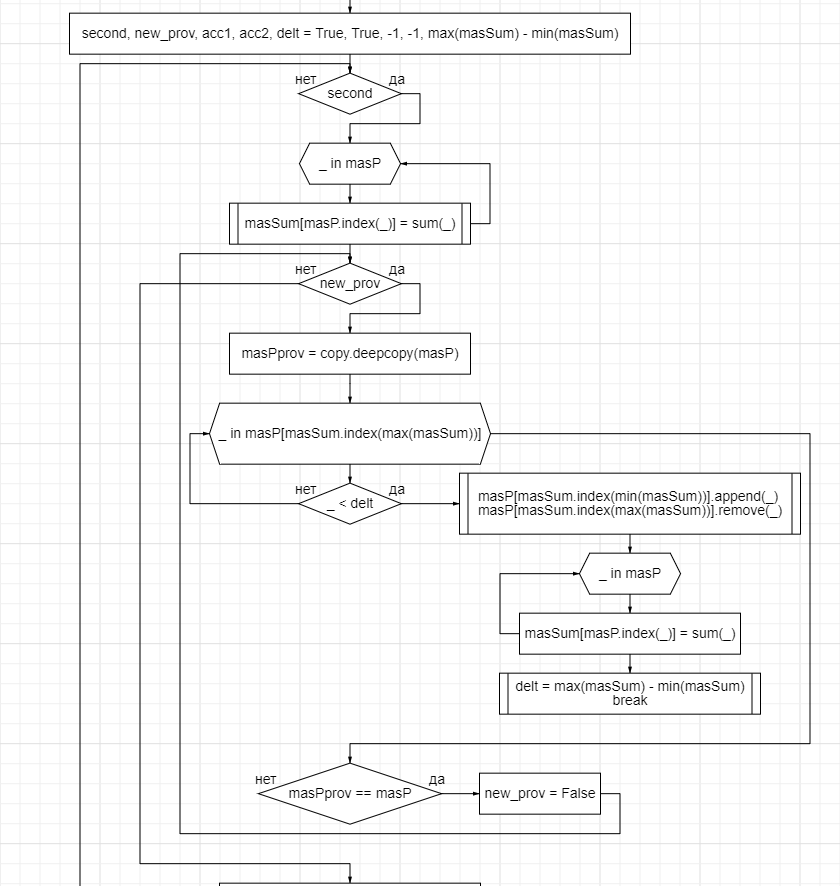
print(\*masSum)

print()

print('Max =', max(masSum))

# Блок схема

****

****

# Вывод

Алгоритм построения расписания с методом наименьших квадратов достаточно эффективен по скорости поиска приемлемого по точности решения.

# Литература

1. Коффман Э.Г. “Теория расписания и вычислительные машины” – M.: “Наука”, 1987
2. Романовский И.В. “Алгоритмы решения экстремальных задач” – М.: “Наука”, 1977
3. Пашкеев С.Д., Минязов Р.И., Могилевский В.Д. “Машинные методы оптимизации в технике связи” – М.: “Связь”, 1976.