## 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 №4

По дисциплине: «Эвристические методы и алгоритмы»

Выполнила:

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.

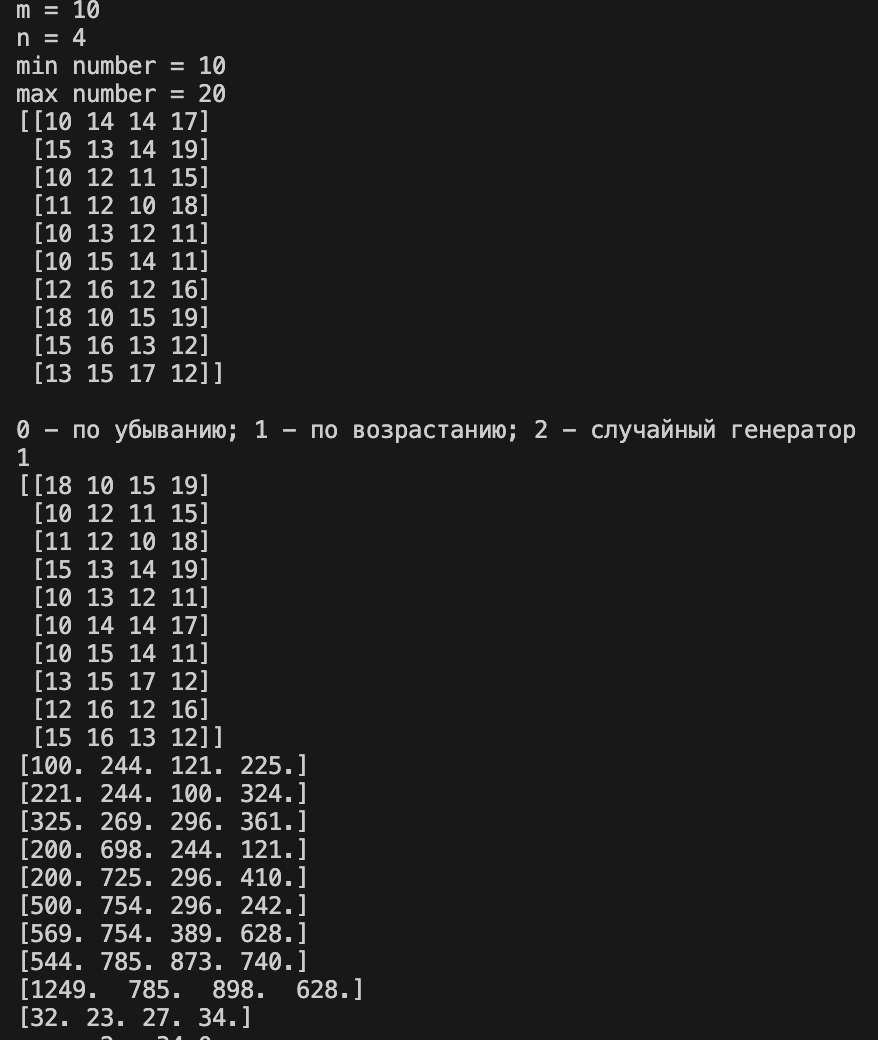
В настоящем руководстве приводятся методы получения расписаний, приводящие к небольшим затратам на вычисление за счет отказа от получения оптимального решения, но в тоже время позволяющие найти приемлемое решение, близкое к оптимальному.

# Постановка задачи

Имеется  независимых работ , которые необходимо распределить на  параллельно работающих разнородных устройств  по критерию , где - время завершения работы процессора . Каждое устройство  выполняет только одну работу в определенный момент времени и выполнение задания не прерывается для передачи на другой процессор. Известно (вес) время выполнения  задания  на любом из устройств . Требуется найти такое распределение заданий по процессорам, при котором суммарное время выполнения заданий на каждом из процессоров было бы минимальным.

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 numpy as np

def descending(matrix):

matrix = matrix[matrix[:, 1].argsort()[::-1]]

return matrix

def ascending(matrix):

matrix = matrix[matrix[:, 1].argsort()]

return matrix

def fun2(matrix, n, m, x):

if x == 0:

matrix = descending(matrix)

elif x == 1:

matrix = ascending(matrix)

elif x == 2:

matrix = matrix

print(matrix)

t = np.zeros(n)

t[matrix[0, ].argmin()] = matrix[0, ].min()

for i in range(1, m):

p = np.zeros(n)

for j in range(n):

p[j] = t[j]\*\*2 + matrix[i, j]\*\*2

p1 = t + matrix[i, ]

p\_i = p.argmin()

print(p)

t[p\_i] = p1[p\_i]

print(t)

return t.max()

def fun3(matrix, n, m, x):

if x == 0:

matrix = descending(matrix)

elif x == 1:

matrix = ascending(matrix)

elif x == 2:

matrix = matrix

print(matrix)

t = np.zeros(n)

t[matrix[0, ].argmin()] = matrix[0, ].min()

for i in range(1, m):

p = np.zeros(n)

for j in range(n):

p[j] = t[j]\*\*2 + matrix[i, j]\*\*3

p1 = t + matrix[i, ]

p\_i = p.argmin()

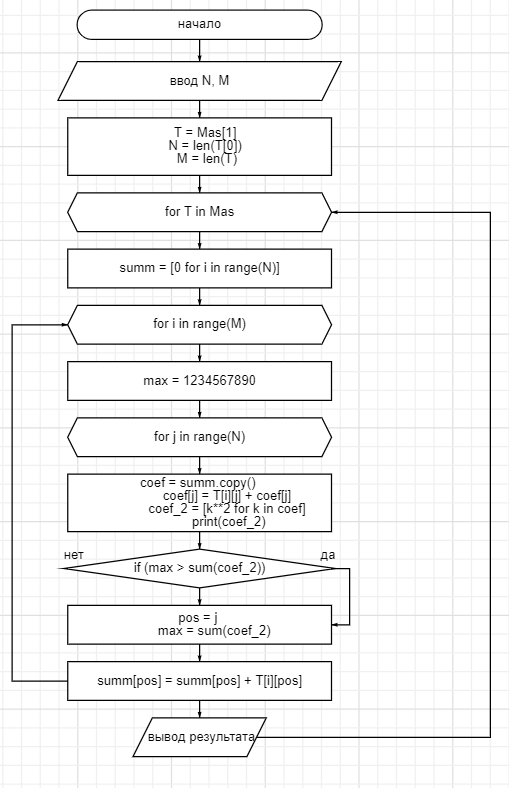
t[p\_i] = p1[p\_i]

print(p)

print(t)

return t.max()

# Блок схема

****

# Вывод

Алгоритм построения расписания с произвольной загрузкой 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.