

МИНИСТЕРСТВО НАУКИ И ВЫСШЕГО ОБРАЗОВАНИЯ РОССИЙСКОЙ ФЕДЕРАЦИИ

**ФЕДЕРАЛЬНОЕ ГОСУДАРСТВЕННОЕ БЮДЖЕТНОЕ**

**ОБРАЗОВАТЕЛЬНОЕ УЧРЕЖДЕНИЕ ВЫСШЕГО**

**ПРОФЕССИОНАЛЬНОГО ОБРАЗОВАНИЯ**

**«Донской государственный технический университет»**

**(ДГТУ)**

Кафедра «Программное обеспечение вычислительной техники и

автоматизированных систем»

**ЛАБОРАТОРНАЯ РАБОТА № 3**

по дисциплине «Эвристические методы и алгоритмы»

тема: «Алгоритм Крона»

Выполнил:

ст. гр. ВПР 31 И.С. Недомерков

Проверил:

д.н., профессор В. Г. Кобак

Ростов-на-Дону

2020

# **Введение**

В настоящее время широкое распространение и развитие получили вычислительные устройства с многоядерной архитектурой процессора и многопроцессорной архитектурой. Причем такие устройства могут входить в состав многомашинных комплексов, позволяющих решать сложные задачи путем распределения вычислительного процесса между ресурсами. Это позволяет значительно сократить временные затраты. Однако в процессе распараллеливания вычислений может возникнуть дисбаланс в загрузке доступных вычислительных ресурсов: часть может простаивать, а другая часть будет загружена на сто процентов. Это приведет к неэффективному использованию оборудования и, как следствие, к увеличению времени речения вычислительной задачи. Поэтому важную роль представляет равномерное распределение нагрузки на все вычислительные ресурсы.

1. **Постановка задачи**

Пусть имеется многомашинная вычислительная система, состоящая из n идентичных параллельно функционирующих устройств.

В ходе работы в систему поступает m независимых заданий Q, которые распределяются между процессорами и обрабатываются параллельно. Задача распределения сводится к разбиению исходного множества заданий на n непересекающихся подмножеств. Критерием разбиения, обеспечивающим оптимальность распределения по быстродействию служит минимаксный критерий. Он определяет такое распределение заданий по процессорам, при котором время загрузки Т параллельным заданием минимально.

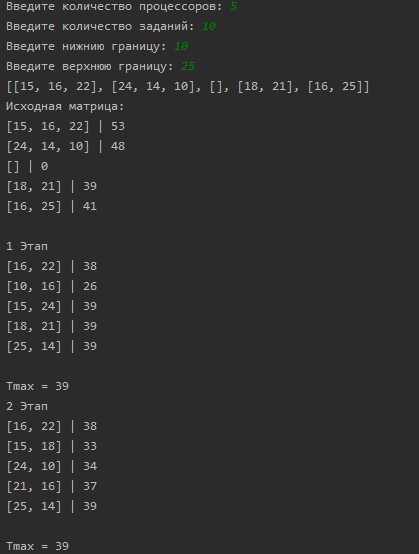
1. **Алгоритм Крона**
2. Каким-либо образом получаем распределение M заданий на N процессоров. Затем ищем Tmax и Tmin (максимальные и минимальные величины столбца). Находим их разность.
3. В столбце, в котором содержится Tmax, находим первый попавшийся элемент, и если он меньше разности Tmax и Tmin, перебрасываем его на столбец, содержащий Tmin, и снова просчитываем Tmax, Tmin и их разность. Процедура продолжается до тех пор, пока не найдется в Tmax ни одного элемента, который можно перебросить в Tmin.
4. В максимальном столбце ищем элемент, от которого можно отнять один из элементов Tmin и разница будет меньше разности Tmax и Tmin.
5. Алгоритм заканчивается, когда в Tmax не находится ни одного такого элемента.

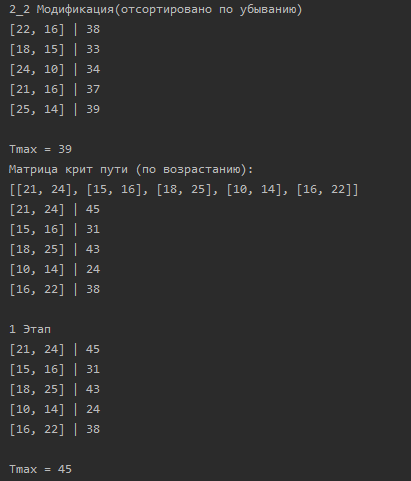
Улучшить алгоритм можно посредством упорядочивания по убыванию элементов в столбце.

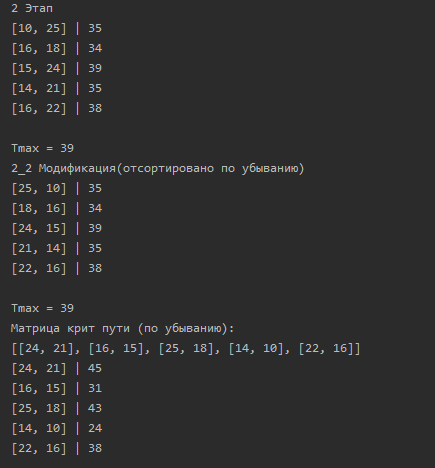
1. **Решение задачи на языке Python**

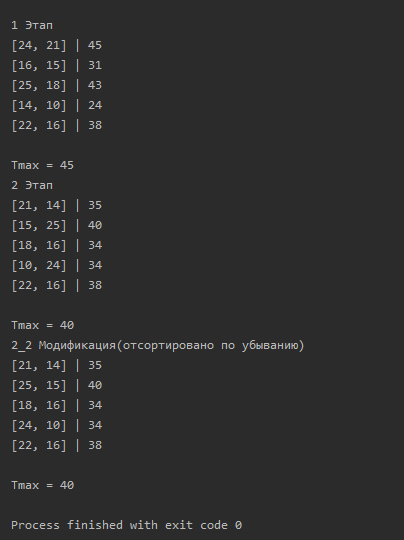
from random import randint  
mass = []  
  
def kron(n, m, T1, T2):  
  
 for i in range(n):  
 mass.append(randint(T1, T2))  
  
 p = [[] for i in range(m)]  
  
 for i in mass:  
 p[randint(0, m - 1)].append(i)  
 print(p)  
 return p  
  
  
def find(mass):  
 sums = []  
 for i in mass:  
 sums.append(sum(i))  
 Tmax = max(sums)  
 for i in mass:  
 if Tmax == sum(i):  
 Tmax = i  
 break  
  
 Tmin = min(sums)  
 for i in mass:  
 if Tmin == sum(i):  
 Tmin = i  
 break  
 return Tmin, Tmax, sum(Tmax) - sum(Tmin)  
  
  
def step(matrix):  
 proverka = True  
 while proverka:  
 Tmin, Tmax, a = find(matrix)  
 proverka = False  
 for i in Tmax:  
 if i < a:  
 Tmin.append(Tmax.pop(Tmax.index(i)))  
 proverka = True  
 break  
 output\_matrix(matrix)  
 print('Tmax = {}'.format(sum(find(matrix)[1])))  
  
  
def step2(matrix):  
 proverka = True  
 while proverka:  
 Tmin, Tmax, a = find(matrix)  
 proverka = False  
 flag = False  
 for i in Tmax:  
 for j in Tmin:  
 if i - j < a and i - j > 0:  
 Tmin.append(Tmax.pop(Tmax.index(i)))  
 Tmax.append(Tmin.pop(Tmin.index(j)))  
 proverka = True  
 flag = True  
 break  
 if flag:  
 flag = False  
 output\_matrix(matrix)  
 print('Tmax = {}'.format(sum(find(matrix)[1])))  
  
  
def step22(matrix):  
 proverka = True  
 while proverka:  
 for i in range(len(matrix)):  
 matrix[i] = sorted(matrix[i])[::-1]  
 Tmin, Tmax, a = find(matrix)  
 output\_matrix(matrix)  
 proverka = False  
 flag = False  
 for i in Tmax:  
 for j in Tmin:  
 if i - j < a and i - j > 0:  
 Tmin.append(Tmax.pop(Tmax.index(i)))  
 Tmax.append(Tmin.pop(Tmin.index(j)))  
 proverka = True  
 flag = True  
 break  
 if flag:  
 flag = False  
 output\_matrix(matrix)  
 print('Tmax = {}'.format(sum(find(matrix)[1])))  
  
  
def output\_matrix(mass):  
 for i in mass:  
 print(i, '|', sum(i))  
 print()  
  
  
def crit(m):  
 mass\_tmp = mass.copy()  
 p = [[] for i in range(m)]  
 for i in mass\_tmp:  
 p[union(p)].append(i)  
  
 for i in range(len(p)):  
 p[i] = sorted(p[i])  
 print(p)  
 return p  
  
def crit2(m):  
 mass\_tmp = mass.copy()  
 p = [[] for i in range(m)]  
 for i in mass\_tmp:  
 p[union(p)].append(i)  
  
 for i in range(len(p)):  
 p[i] = sorted(p[i])[::-1]  
 print(p)  
 return p  
  
  
def union(mass):  
 tmp = [sum(i) for i in mass]  
 return tmp.index(min(tmp))  
  
if \_\_name\_\_ == '\_\_main\_\_':  
 n = int(input('Введите количество процессоров: '))  
 m = int(input('Введите количество заданий: '))  
 T1 = int(input('Введите нижнию границу: '))  
 T2 = int(input('Введите верхнюю границу: '))  
 matrix = kron(m, n, T1, T2)  
 print('Исходная матрица:')  
 output\_matrix(matrix)  
 print('1 Этап')  
 step(matrix)  
 print('2 Этап')  
 step2(matrix)  
 print('2\_2 Модификация(отсортировано по убыванию)')  
 step22(matrix)  
 print('Матрица крит пути (по возрастанию):')  
 matrix2 = crit(n)  
 output\_matrix(matrix2)  
 print('1 Этап')  
 step(matrix2)  
 print('2 Этап')  
 step2(matrix2)  
 print('2\_2 Модификация(отсортировано по убыванию)')  
 step22(matrix2)  
 print('Матрица крит пути (по убыванию):')  
 matrix3 = crit2(n)  
 output\_matrix(matrix3)  
 print('1 Этап')  
 step(matrix3)  
 print('2 Этап')  
 step2(matrix3)  
 print('2\_2 Модификация(отсортировано по убыванию)')  
 step22(matrix3)

**5. Результаты, выводы**









В ходе лабораторной работы был изучен алгоритм Крона, а так же написано программное средство, реализующее его. При сортировке столбцов по убыванию алгоритм работает лучше всего.