МИНИСТЕРСТВО ОБРАЗОВАНИЯ И НАУКИ 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 № 4

по дисциплине «Эвристические методы и алгоритмы»

тема: «Алгоритмы списочных расписаний. Алгоритм половинного деления множества заданий»

Выполнил:
ст. гр. ВПР 31 И.С. Недомерков

Проверил:
д.н., профессор В. Г. Кобак

Ростов-на-Дону
2020

1. Введение

Предметом области исследования расписаний является круг задач проектирования и организационного управления в различных системах, в которых требуется найти наилучшее (оптимальное) значение выбранных критериев их функционирования с учетом имеющихся ограничений.

Программирование для многопроцессорных машинных систем связано с распараллеливанием и синхронизацией вычислений и организацией выполнения параллельных вычислительных процессов. Это выдвигает целый ряд сложных задач, среди которых весьма важными являются, расчет характеристик времени и количества операций, требующихся для выполнения параллельных программ, и построения расписаний (планов), выполнения параллельных программ на многопроцессорных и многомашинных вычислительных системах.

Модели параллельных программ и операционные характеристики процессов их выполнения служат основой для планирования параллельных вычислительных процессов, т.е. для построения расписаний указанных процессов. Расписания параллельных вычислительных процессов определяют порядок выполнения программы на вычислительной системе, включая распределение частей программы по процессам. С увеличением числа распределяемых частей программ и количества используемых процессоров сложность построения оптимальных расписаний обычно резко возрастает. Поэтому важное значение имеют простые в построении и удобные в реализации приближенные расписания параллельных вычислительных процессов, близкие к оптимальным с точки зрения времени выполнения параллельных программ.

2. Постановка задачи

Имеется вычислительная система (ВС), состоящая из *N* несвязанных идентичных устройств (приборов, процессоров и т.п.)

$P = \{p_1, p_2, \dots, p_n\}$

На обслуживание в ВС поступает набор из *M* независимых параллельных заданий (работ) $T = \{t_1, t_2, \dots, t_m\}$ известно время решения $\tau(t_i)$ задания t_i на любом из устройств. При этом каждое задание может выполняться на любом из устройств (процессоре), в каждый момент времени отдельный процессор обслуживает не более одного задания и выполнение задания не прерывается для передачи на другой процессор. Требуется найти такое распределение заданий по процессорам, при котором суммарное время выполнения заданий на каждом из процессоров было бы минимальным. Под расписанием следует понимать отображение $A_R: T \to P$, такое что, если $A_R(t_i) = p_j$, то говорят что задание $t_i \in T$, в расписании *R* назначенного на процессор $p_j \in P$. При сделанных выше допущениях, расписание можно представить разбиением множества заданий *T* на *N* непересекающихся подмножеств $T_j; j = 1, \dots, N$

Критерий, используемый для минимизации времени завершения обслуживания заданий, является минимальным критерием и определяется в следующем виде: $f_r = \max_{1 \le j \le n} f_j \to \min$, где

$f_j = \sum_{t_i \in T_j} \tau(t_i)$- время завершения работы процессора p_j.

3. Теоретическая часть.

3.1 Алгоритм критического пути

Простой и эффективный алгоритм списочного расписания: метод критического пути.

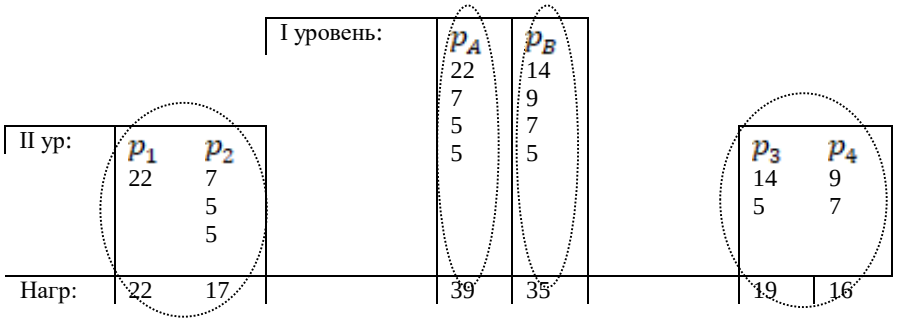
Имеющуюся однородную матрицу упорядочивают по убыванию. Сначала процессы получают задания, идущие по порядку. Значение нагрузки необходимо пересчитывать после каждого добавления задания на один из процессоров. Следующие задания направляются на наименее загруженный из всех процессоров. При наличии одинаковых значений нагрузки на нескольких процессорах, приоритет получения следующего задания принадлежит тому процессору, который ближе к первому (по порядковому номеру).

3.2 Алгоритм половинного деления множества заданий

Приведённый далее алгоритм является одной из модификаций алгоритма критического пути (CMP), но, алгоритм половинного деления состоит из двух уровней, и актуален только для чётного числа процессоров.

Решим задачу №1 с помощью алгоритма половинного деления множества заданий. Как и в CMP, в алгоритме половинного деления (Half Division Multitude Tasks или HDMT) имеющуюся однородную матрицу упорядочить по убыванию, в результате, множество заданий будет иметь вид: T = [22 14 9 7 7 5 5 5]. Далее, как и в алгоритме CMP, распределяем всё множество заданий, но, только, на два процессора - p_A и p_B (В этом заключается первый уровень):

На втором уровне задания с процессоров p_A и p_B разбрасываются на две группы процессоров (по N/2 процессоров в каждой): p_1, p_2 и p_3, p_4 соответственно. Процесс решения:



4. Реализации алгоритма половинного деления множества заданий Python

```
from random import randint


def create_matrix(n, m, T1, T2):
    mass = [randint(T1, T2) for i in range(m)]
    return mass


def output_matrix(matrix, n):
    string = ''
    for i in matrix:
        string += '{}  '.format(i) * n + '\n'

    return string


def crit(matrix, m):
    mass_tmp = matrix.copy()
    p = [[] for i in range(m)]
    for i in mass_tmp:
        p[union(p)].append(i)
    return p


def union(matrix):
    tmp = [sum(i) for i in matrix]
    return tmp.index(min(tmp))


def HDMT(matrix):
    Pa, Pb = crit(matrix, 2)
    print('Уровень 1:')
    print('Pa =', Pa, ' sum =', sum(Pa))
    print('Pb =', Pb, ' sum =', sum(Pb))
    print()
    print('Уровень 2 левая часть:')
    Pa = crit(Pa, n // 2)
    for i in range(n // 2):
        print('P{} = {}, sum = {}'.format(i + 1, Pa[i], sum(Pa[i])))
    print('Уровень 2 правая часть:')
    Pb = crit(Pb, n // 2)
    for i in range(n // 2):
        print('P{} = {}, sum = {}'.format(i + n // 2 + 1, Pb[i], sum(Pb[i])))
    P = Pa + Pb
    sums = list(map(sum, P))
    print(sums)
    print('Tmax = P{}'.format(sums.index(max(sums))+1),
          '=', P[sums.index(max(sums))], '=', max(sums))
    print()
    print()


if __name__ == '__main__':
    n = int(input('Введите количество процессоров: '))
    m = int(input('Введите количество заданий: '))
    T1 = int(input('Введите нижнюю границу: '))
    T2 = int(input('Введите верхнюю границу: '))
    matrix = create_matrix(n, m, T1, T2)
    print('Исходная матрица:')
```

```
print(output_matrix(matrix, n))
HDMT(matrix)
print()
print('Отсортирована по возрастанию: ')
matrix = sorted(matrix)
print(output_matrix(matrix, n))
HDMT(matrix)
print('Отсортирована по убыванию: ')
matrix = sorted(matrix)[::-1]
print(output_matrix(matrix, n))
HDMT(matrix)
```

5. Результаты, выводы

В ходе лабораторной работы была написана программа, реализующая алгоритм половинного деления множества заданий, в которой количество процессоров и количество заданий было определено из вариантов заданий для вариантов из пункта 4 «Варианты заданий». Матрица генерируется случайным образом.

Также было произведено сравнение работы алгоритма при сортированном списке заданий, при случайном и при обратном списке. Выяснили, что алгоритм позволяет получить точной результат, при использовании списка по убыванию. Остальные результаты являются приближенными.

Результат работы программы:

```
Введите количество процессоров: 6
Введите количество заданий: 15
Введите нижнюю границу: 30
Введите верхнюю границу: 45
Исходная матрица:
32  32  32  32  32  32
34  34  34  34  34  34
45  45  45  45  45  45
30  30  30  30  30  30
34  34  34  34  34  34
41  41  41  41  41  41
44  44  44  44  44  44
41  41  41  41  41  41
41  41  41  41  41  41
39  39  39  39  39  39
32  32  32  32  32  32
38  38  38  38  38  38
41  41  41  41  41  41
41  41  41  41  41  41
36  36  36  36  36  36

Уровень 1:
Pa = [32, 45, 41, 41, 39, 38, 41]  sum = 277
Pb = [34, 30, 34, 44, 41, 32, 41, 36]  sum = 292
```

```
Уровень 2 левая часть:
P1 = [32, 41, 41], sum = 114
P2 = [45, 38], sum = 83
P3 = [41, 39], sum = 80
Уровень 2 правая часть:
P4 = [34, 41], sum = 75
P5 = [30, 44, 36], sum = 110
P6 = [34, 32, 41], sum = 107
[114, 83, 80, 75, 110, 107]
Tmax = P1 = [32, 41, 41] = 114



Отсортирована по возрастанию:
30  30  30  30  30  30
32  32  32  32  32  32
32  32  32  32  32  32
34  34  34  34  34  34
34  34  34  34  34  34
36  36  36  36  36  36
38  38  38  38  38  38
39  39  39  39  39  39
41  41  41  41  41  41
41  41  41  41  41  41
41  41  41  41  41  41
41  41  41  41  41  41
41  41  41  41  41  41
44  44  44  44  44  44
45  45  45  45  45  45

Уровень 1:
Pa = [30, 32, 34, 38, 41, 41, 41, 45]  sum = 302
Pb = [32, 34, 36, 39, 41, 41, 44]  sum = 267

Уровень 2 левая часть:
P1 = [30, 38, 41], sum = 109
P2 = [32, 41, 45], sum = 118
P3 = [34, 41], sum = 75
Уровень 2 правая часть:
P4 = [32, 39, 44], sum = 115
P5 = [34, 41], sum = 75
P6 = [36, 41], sum = 77
[109, 118, 75, 115, 75, 77]
Tmax = P2 = [32, 41, 45] = 118
```

```
Отсортирована по убыванию:
45  45  45  45  45  45
44  44  44  44  44  44
41  41  41  41  41  41
41  41  41  41  41  41
41  41  41  41  41  41
41  41  41  41  41  41
41  41  41  41  41  41
39  39  39  39  39  39
38  38  38  38  38  38
36  36  36  36  36  36
34  34  34  34  34  34
34  34  34  34  34  34
32  32  32  32  32  32
32  32  32  32  32  32
30  30  30  30  30  30

Уровень 1:
Pa = [45, 41, 41, 39, 38, 34, 32]  sum = 270
Pb = [44, 41, 41, 41, 36, 34, 32, 30]  sum = 299

Уровень 2 левая часть:
P1 = [45, 34, 32], sum = 111
P2 = [41, 39], sum = 80
P3 = [41, 38], sum = 79
Уровень 2 правая часть:
P4 = [44, 34, 30], sum = 108
P5 = [41, 41], sum = 82
P6 = [41, 36, 32], sum = 109
[111, 80, 79, 108, 82, 109]
Tmax = P1 = [45, 34, 32] = 111
```