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.

Постановка задачи

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

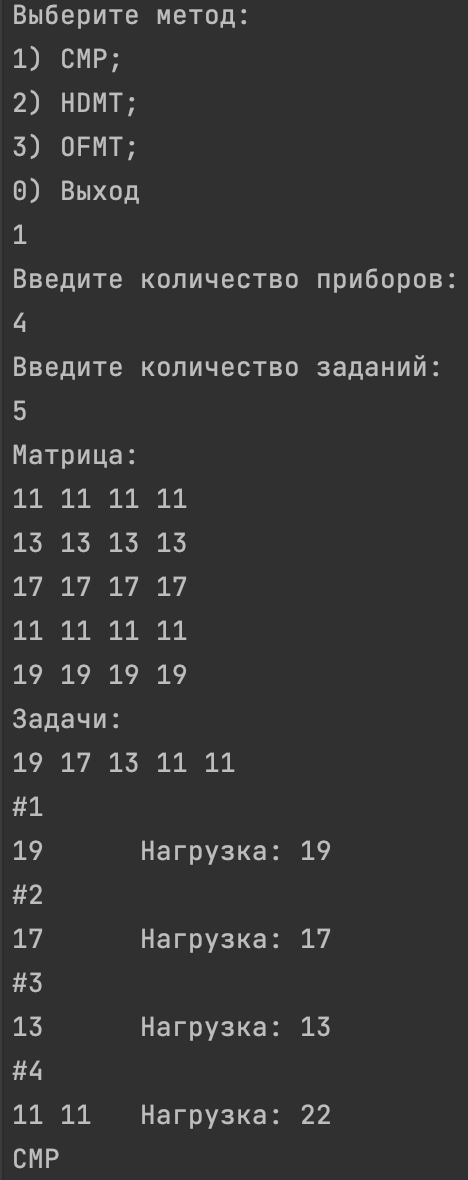


На обслуживание в ВС поступает набор из M независимых параллельных заданий (работ)  известно время решения  задания  на любом из устройств. При этом каждое задание может выполняться на любом из устройств (процессоре), в каждый момент времени отдельный процессор обслуживает не более одного задания и выполнение 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 , такое что, если , то говорят что задание , в расписании R назначенного на процессор . При сделанных выше допущениях, расписание можно представить разбиением множества заданий T на N непересекающихся подмножеств 

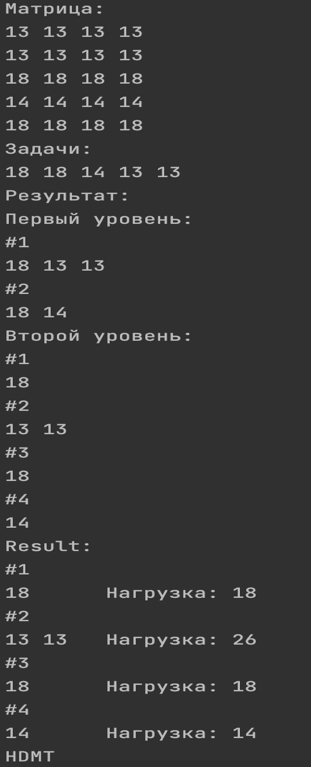
Критерий, используемый для минимизации времени завершения обслуживания заданий, является минимальным критерием и определяется в следующем виде: , где - время завершения работы процессора .

Ручной расчет примера

**Алгоритм критического пути** **(CMP)**

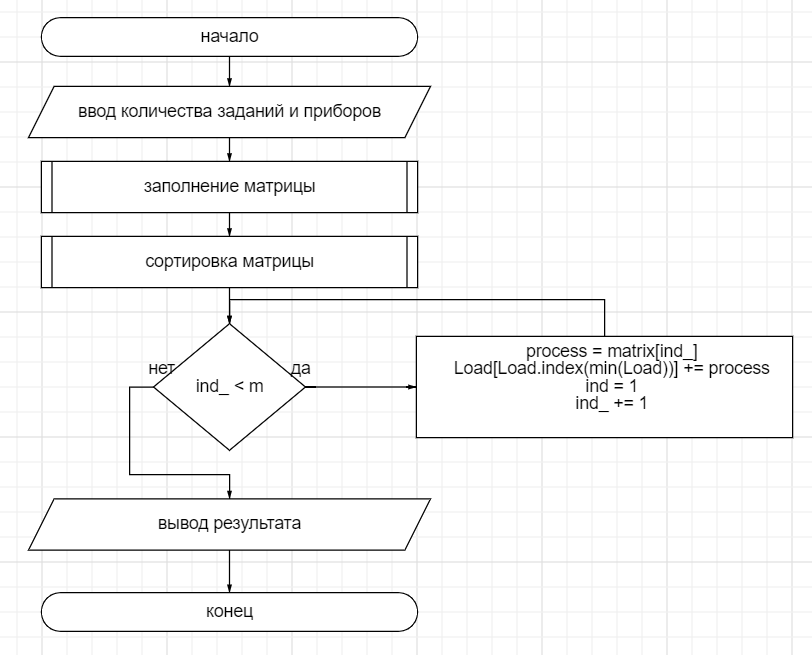


**Алгоритм половинного деления множества заданий (HDMT)**



**Блок схемы**

**Блок схема алгоритма критического пути (CMP)**



**Код программы**

#include <string>

#include <iostream>

#include <cmath>

**using** **namespace** std;

**int** n=4;// количество процессов

**int** nagr[4]={0};

**int** m=10;

**int** mas[8]={0};

**void** sort(){

**int** temp;

**for** (**int** i = 0; i < m - 1; i++) {

**for** (**int** j = 0; j < m - i - 1; j++) {

**if** (mas[j] < mas[j + 1]) {

// меняем элементы местами

temp = mas[j];

mas[j] = mas[j + 1];

mas[j + 1] = temp;

}

}

}

}

**void** show(**int** \*nagr,**int** n){

**for**(**int** i=0;i < n;i++){

cout<<nagr[i]<<" ";

}

}

**void** fillnagr(**int** \*nagr,**int** n){

**for**(**int** i=0;i < n;i++){

nagr[i]=0;

}

}

**void** task1(){

**int** temp,copy=999,time=0;

fillnagr(nagr,n);

**for**(**int** i=0;i<m;i++){

cout<<i+1<<":";

cin>>mas[i];

}

sort();

**for**(**int** i=0;i<m;i++){

cout<<mas[i]<<" ";

}

**for**(**int** i=0;i<m;i++){

cout<<endl;

show(nagr,n);

copy=999;

**for**(**int** j=0;j<n;j++){

**if**(nagr[j]<copy){

copy=nagr[j];

time=j;

}

}

nagr[time]=copy+mas[i];

}

**int** c=nagr[0];

**for**(**int** i=0;i<n;i++){

**if**(nagr[i]>c && nagr[i]!=0)

c=nagr[i];

}

cout<<endl;

cout<<"-----------------"<<endl;

show(nagr,n);

cout<<" max:"<<c<<endl<<"-----------------"<<endl;

}

**void** task2(){

**int** temp,copy=999,time=0;

fillnagr(nagr,n);

**for**(**int** i=0;i<m;i++){

cout<<i+1<<":";

cin>>mas[i];

}

sort();

**for**(**int** i=0;i<m;i++){

cout<<mas[i]<<" ";

}

// // // // // // // // Первый шаг

**int** mas1[m/2];fillnagr(mas1, m/2);

**int** mas2[m/2];fillnagr(mas2, m/2);

**int** sum1=0;**int** s1=0;

**int** sum2=0;

mas1[0]=mas[0];

sum1=sum1+mas[0];

**for**(**int** i=1;i<m;i++){

s1=0;

**if**(sum1<=sum2){

**for**(**int** j=0;j<m/2;j++){

**if**(mas1[j]==0){

mas1[j]=mas[i];

sum1=sum1+mas[i];

s1=1;

**break**;

}

}

}

**if**(sum1>sum2 && s1==0){

**for**(**int** j=0;j<m/2;j++){

**if**(mas2[j]==0){

mas2[j]=mas[i];

sum2=sum2+mas[i];

**break**;

}

}

}

}

// // // // // // // // Второй шаг

**for**(**int** i=0;i<m/2;i++){

cout<<endl;

show(nagr,n);

copy=999;

**for**(**int** j=0;j<n/2;j++){

**if**(nagr[j]<copy){

copy=nagr[j];

time=j;

}

}

nagr[time]=copy+mas1[i];

}

// // // // // // // // Второй шаг

**for**(**int** i=0;i<m/2;i++){

cout<<endl;

show(nagr,n);

copy=999;

**for**(**int** j=n/2;j<n;j++){

**if**(nagr[j]<copy){

copy=nagr[j];

time=j;

}

}

nagr[time]=copy+mas2[i];

}

// // // // // // // //

**int** c=nagr[0];

**for**(**int** i=0;i<n;i++){

**if**(nagr[i]>c && nagr[i]!=0)

c=nagr[i];

}

cout<<endl;

cout<<"-----------------"<<endl;

show(nagr,n);

cout<<" max:"<<c<<endl<<"-----------------"<<endl;

}

**int** main() {

task1();

task2();

**return** 0;

}

}

**Вывод**

Проанализировав данные, которые выдает программа и результаты ручных подсчетов, можно отметить, что предложенный способ решения задач составления списочных расписаний для (четного количества кратных степени 2) устройств с помощью модифицированных алгоритмов OFMT и HDMT является более эффективным, т.к. точность решения в данном случае несколько лучше, чем у CPM.

**Литература**

1. Методические указания

и контрольные задания к практическим, лабораторным занятиям, курсовому проектированию по теме «Теория расписаний», для дисциплин «Алгоритмические языки и программирование», «Вычислительная техника и программирование»



МИНИСТЕРСТВО НАУКИ И ВЫСШЕГО ОБРАЗОВАНИЯ

РОССИЙСКОЙ ФЕДЕРАЦИИ

**ФЕДЕРАЛЬНОЕ ГОСУДАРСТВЕННОЕ БЮДЖЕТНОЕ**

**ОБРАЗОВАТЕЛЬНОЕ УЧРЕЖДЕНИЕ ВЫСШЕГО ОБРАЗОВАНИЯ**

**«ДОНСКОЙ ГОСУДАРСТВЕННЫЙ ТЕХНИЧЕСКИЙ УНИВЕРСИТЕТ»**

**(ДГТУ)**

Факультет «Информатика и вычислительная техника»

Кафедра «Программное обеспечение вычислительной техники и автоматизированных систем»

**ЛАБОРАТОРНАЯ РАБОТА ПО ДИСЦИПЛИНЕ:**

**«Эвристические методы и алгоритмы»**

Выполнил студент группы ВПР31

Андросов И.А

Проверил:

профессор Кобак Валерий Григорьевич

г. Ростов-на-Дону

2023г