**华 南 农 业 大 学 数 学 与 信 息（软 件） 学 院**

《操作系统分析与设计实习》成绩单

开设时间：2018学年第一学期

|  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| 小组成员、组内分工、工作量比例、各成员个人成绩 | | | | | | | | | | | |
| **学号** |  | **姓名** |  | **分工** | 程序设计与报告 | | | **工作量比例** | 100% | **成绩** |  |
| **学号** |  | **姓名** |  | **分工** |  | | | **工作量比例** |  | **成绩** |  |
| **学号** |  | **姓名** |  | **分工** |  | | | **工作量比例** |  | **成绩** |  |
| **学号** |  | **姓名** |  | **分工** |  | | | **工作量比例** |  | **成绩** |  |
| 实 验 题 目 | 单处理器系统的进程调度 | | | | | | | | | | |
| 自 我 评 价 | 通过实现各种进程调度算法，并模拟单处理器系统进程的调度，将上学期所学的理论知识转换为实际操作模拟，更加深入理解处理器的调度机制。 | | | | | | | | | | |
| 教 师 评 语 | 评价指标：   * 题目内容和要求完成情况 优 □ 良 □ 中 □ 差 □ * 对算法原理的理解程度 优 □ 良 □ 中 □ 差 □ * 程序设计水平 优 □ 良 □ 中 □ 差 □ * 程序运行效果及正确性 优 □ 良 □ 中 □ 差 □ * 课程设计报告结构清晰 优 □ 良 □ 中 □ 差 □ * 报告中总结和分析详尽 优 □ 良 □ 中 □ 差 □ | | | | | | | | | | |
|  |  | | | | | **教师签名** | 张丽霞 | | | | |

**1.需求分析**

* 1. **程序功能描述**

编写程序完成单处理器系统中的进程调度，要求实现：

1. 时间片轮转
2. 优先数
3. 最短进程优先
4. 最短剩余时间优先四种调度算法。

实验具体包括：首先确定进程控制块的内容，进程控制块的组成方式；然后完成进程创建原语和进程调度原语；最后编写主函数对所作工作进行测试。模拟程序只对你所设置的“虚拟 PCB”进行相应的调度模拟操作，即每发生“调度”时，显示出当前运行进程的“进程标识符”、“优先数”、“剩余运行时间”等，而不需要对系统中真正的 PCB 等数据进行修改。

* 1. **开发与运行环境**
     1. **开发语言**

本程序选用 Python作为程序设计语言，调度结果由控制台输出。

* + 1. **开发环境**

Windows10，Pycharm 2017.2.1，Python 3.6。

* + 1. **运行环境**

可以在安装了 Python3.6 的系统中运行。

* 1. **输入、输出形式**

由函数随机创建进程就绪队列。控制台输出对应进程数据。

* 1. **数据规范**
     1. **模拟虚拟 PCB**
        1. 模拟进程队列。随机创建 10 个进程。每个进程属性包含： 标志符、进入内存时间、开始执行时间、所需运行时间、进程优先数、进程结束时间、进程运行时间、进程等待时间、进程周转时间、进程状态。
        2. 模拟进程切换上下文信息。上下文信息包括：进程队列、就绪队列、当前进程、结束队列信息。
  2. **实现功能介绍**
     + 1. 创建进程对象

一个进程类用于创建进程对象。可初始化进程标志符、进程到达时间、进程运行所需时间、进程优先数。

* + - 1. 进行进程切换

当调度为抢占式算法或当前进程结束时，进行进程切换。

* + - 1. 算法调度

一个算法类用于保存上下文信息以及调度算法类型。

* + - 1. 信息输出格式规范

信息输出包括：Created、Started、Context switch、Finished、Running。每次进行进程调度输出调度信息。

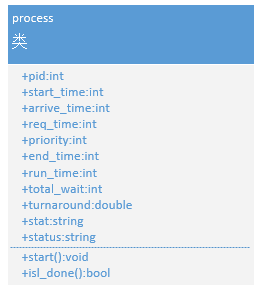
1. **概要设计**

## 项目结构

整个程序是 Pycharm 上创建的 Python 项目。

* 1. **程序模块定义**
     1. 总览
     2. 类定义

**2.2.2.1 process 类**



说明：process 类定义了一个记录基本信息的进程对象。

**类的成员变量**

|  |  |  |  |
| --- | --- | --- | --- |
| 变量名 | 类型 | 数据域 | 描述 |
| pid | int | Public | 进程标志符 |
| start\_time | int | Public | 进程执行开始时间 |
| arrive\_time | int | Public | 进程到达时间 |
| req\_time | int | Public | 进程运行需求时间 |
| priority | int | Public | 进程优先数 |
| end\_time | int | Public | 进程结束时间 |
| run\_time | int | Public | 进程运行时间 |
| total\_wait | int | Public | 进程总等待时间 |
| turnaround | double | Public | 进程周转时间 |
| stat | string | Public | 进程空闲状态 |
| status | String | Public | 进程执行状态 |

方法：

is\_done(self)

描述：判断进程是否终止。

传入参数：无

返回值：无

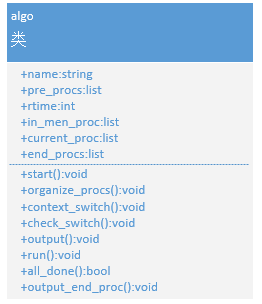
run(self, runtime)

描述：记录进程调度执行时间，一次调度，调度时间增加。

传入参数：时间片长度

返回值:无

**2.2.2.2 algo 类**



说明：

algo类包含对进程的算法调度操作，包含进程切换的上下文信息，用于控制不同类型的算法对不同进程的调度。

|  |  |  |  |
| --- | --- | --- | --- |
| 变量名 | 类型 | 数据域 | 描述 |
| name | string | Public | 调度算法名 |
| pre\_procs | list | Public | 开始进程队列 |
| rtime | int | Public | cpu内存时间 |
| in\_men\_procs | list | Public | 进程运行需求时间 |
| current\_proc | list | Public | 进程优先数 |
| end\_procs | list | Public | 进程完成队列 |
| flag | bool | Public | 初始化进程队列标志 |

**方法：**

organize\_procs(self)

描述：按照各种算法需求

传入参数：无

返回值：无

context\_switch(self, next\_proc)

描述：进行进程切换

传入参数：需切换的下一进程

返回值：无

check\_switch(self)

描述：判断进程是否需要切换

传入参数：无

返回值：无

output(self, args)

描述：信息格式输出控制

传入参数：无

返回值：无

run(self)

描述：模拟进程调度启动

传入参数：无

返回值：无

all\_done(self)

描述：判断所有进程是否调度结束

传入参数：无

返回值：bool 数据类型

output\_end\_proc(self)

描述：统计数据格式输出控制

传入参数：无

返回值：无

def schedule(self)

描述:模拟进程调度操作

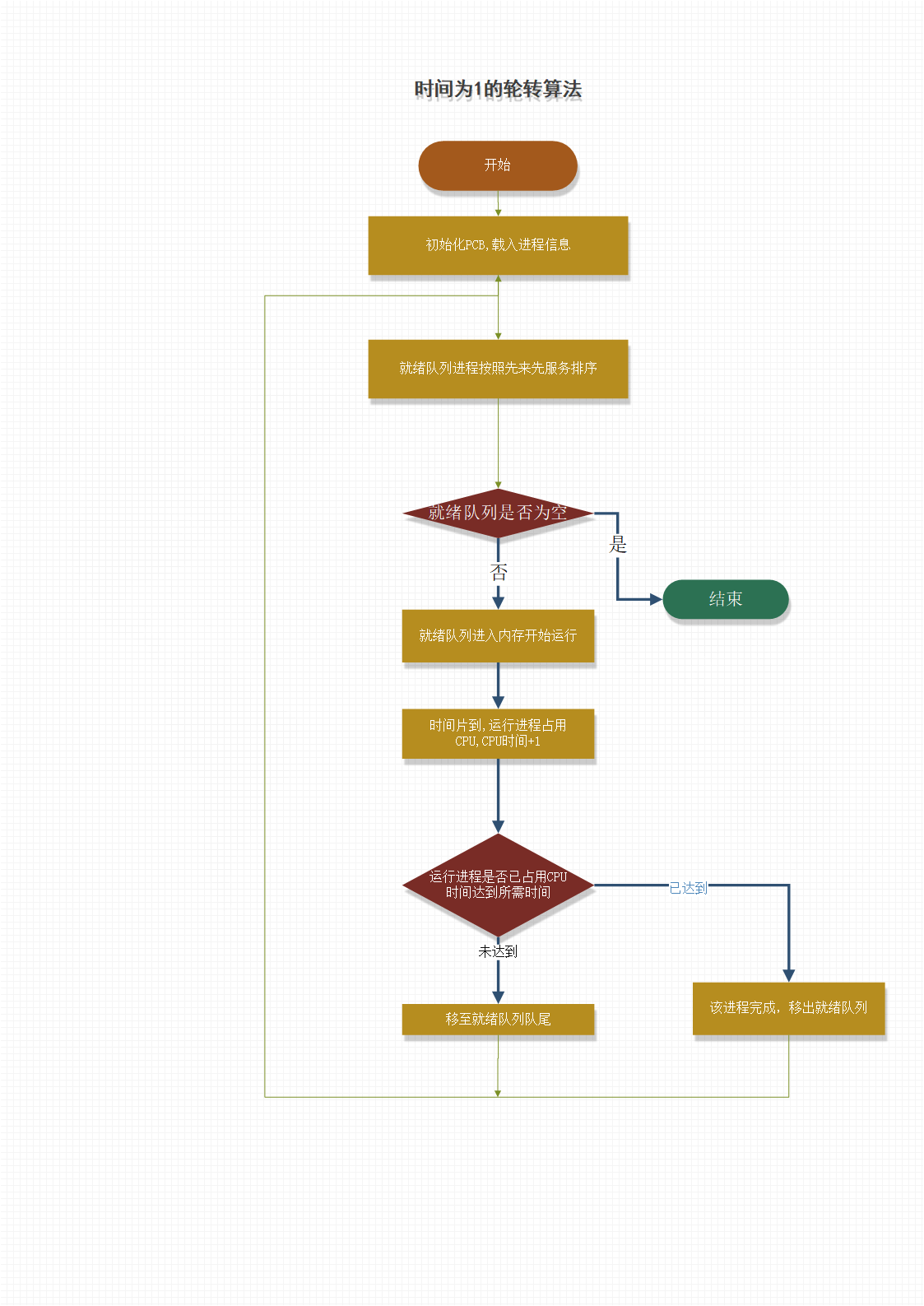
传入参数:无

返回值:无

# 详细设计

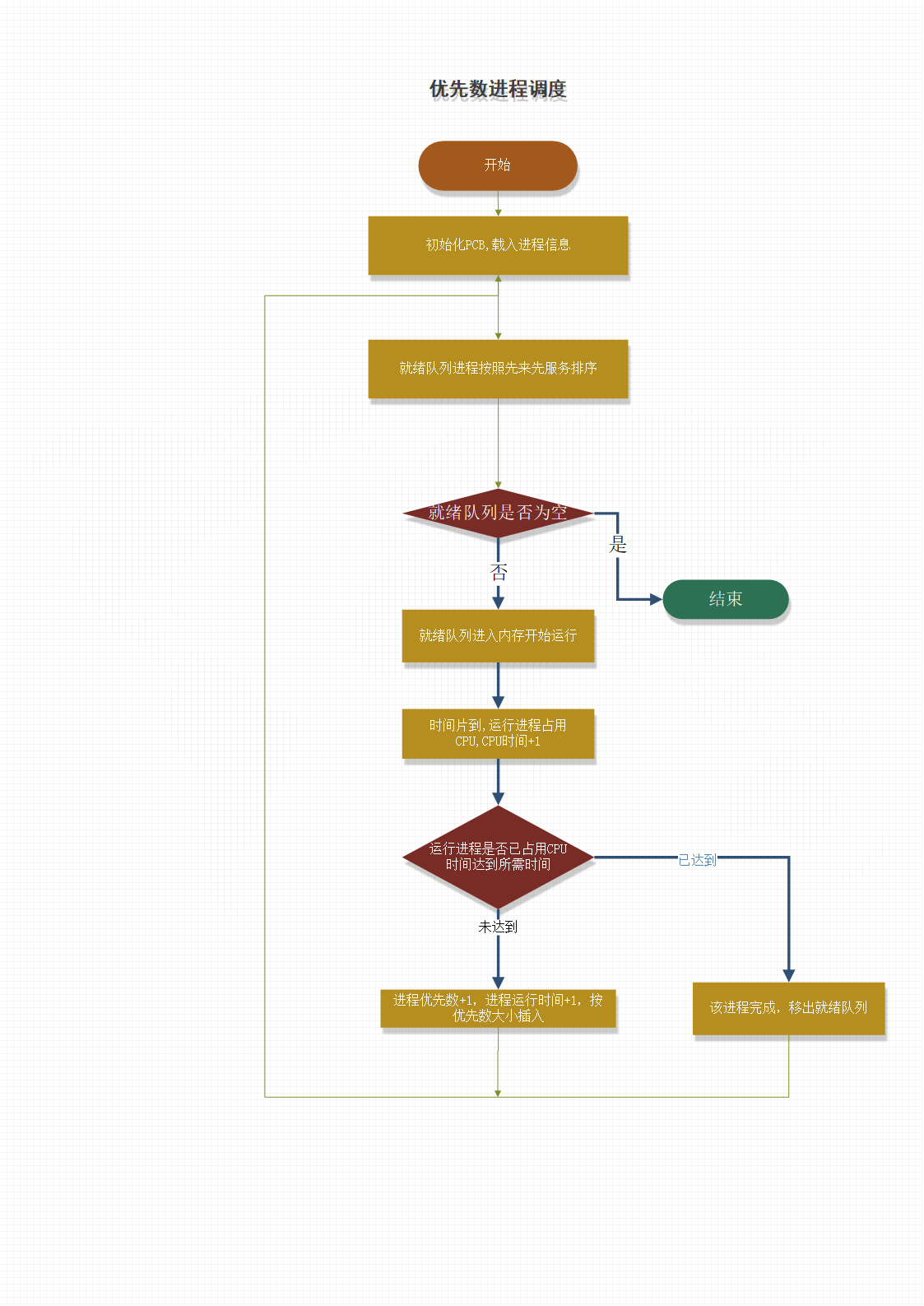
以下介绍了程序内各个主要方法以及不同进程调度算法的设计思路，使用文字及伪代码描述。

3.1 时间片轮转（抢占）



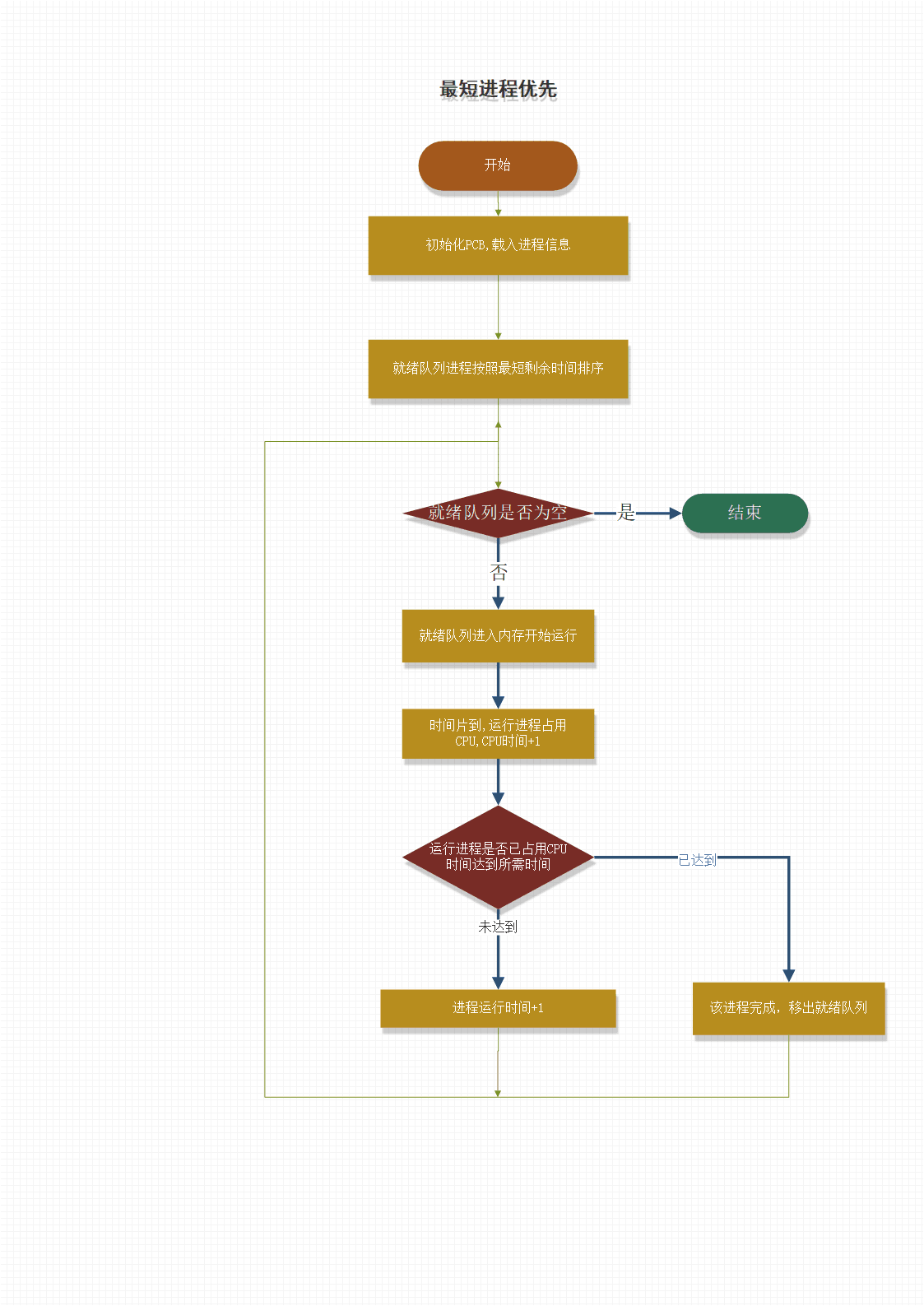
**描述：**将就绪队列的进程按照先来先服务的规则进行排序，每到一个时间片，被调度的进程运行时间+1，当前进程插入到就绪队列队尾，同时当前进程切换为下一就绪队列进程，直至到达进程运行所需时间，移出就绪队列。

3.2 优先数（抢占）



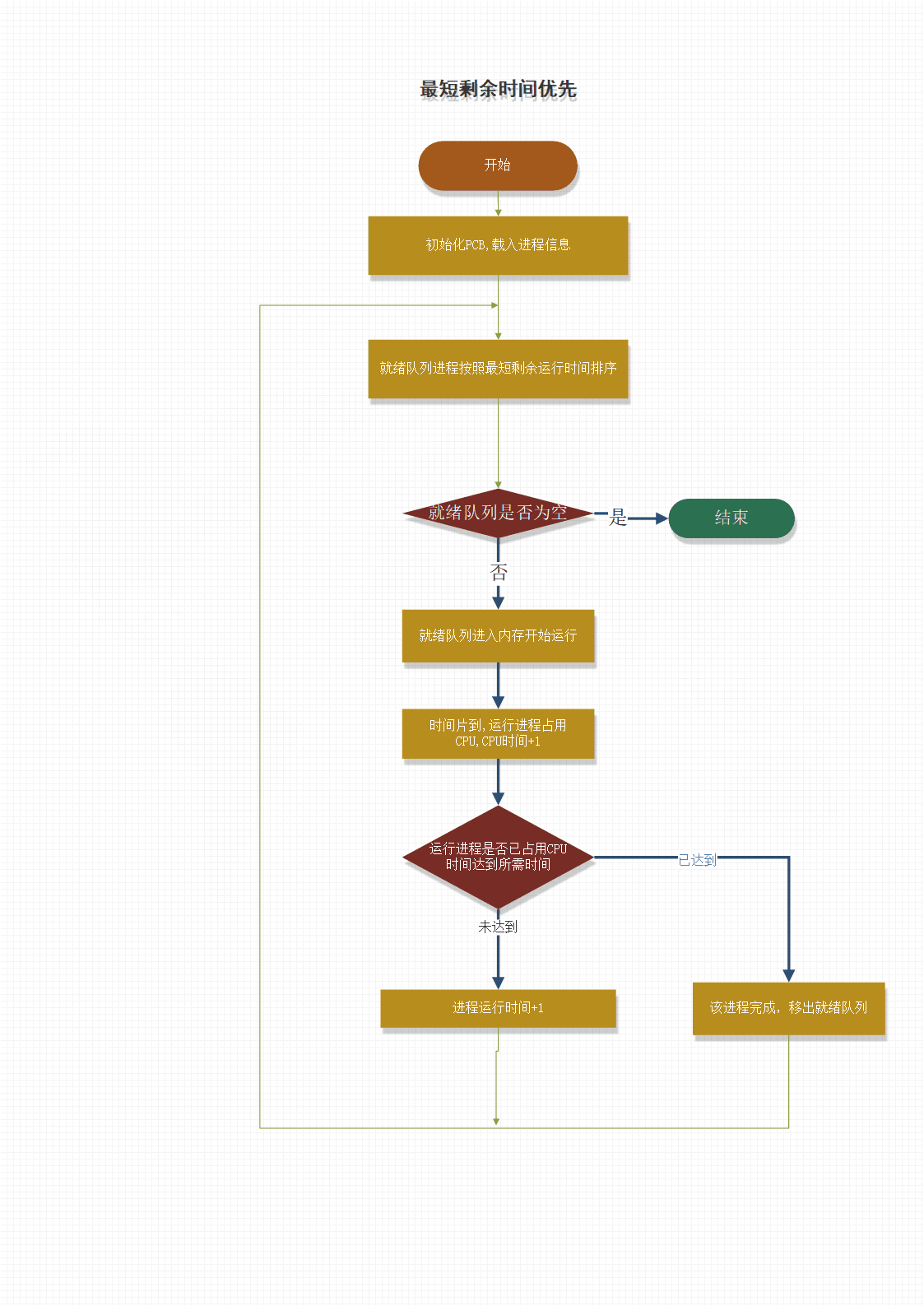
描述：将就绪队列的进程按照优先数大小进行排序（优先数小者优先级大），每到一个时间片，被调度的进程优先数+1，进程运行时间+1，再次将就绪队列的进程按优先数排序，判断当前进程是否需要进行切换。

3.3 最短进程优先（非抢占）



**描述：**将就绪队列的进程按进程最短剩余运行时间进行排序，进程不发生抢占，被调度的进程运行时间+1，直到被调度的进程执行完毕后，才可切换成下一就绪队列进程，进程调度完成则移出就绪队列。

3.4 最短剩余时间优先（抢占）



**描述：**将就绪队列的进程按照进程最短剩余运行时间进行排序，每到一个时间片，被调度的进程运行时间+1，将就绪队列按照最短剩余时间进行排序，进程发生抢占，直到进程运行完毕被移出就绪队列。

# 调试分析

**4.1设计过程经验与体会**

这次在操作系统课设中使用 Python 成功实现了单处理系统对进程的模拟调度算法，让自己对操作系统对进程的调度由理论知识上升到实践认知，是一次非常实用的实践。Python 语言不同于 Java、C++ ，在类和方法的构造上有一定的区别，同时 Python 语言语法的简洁性，让我选择使用 Python 实现。 通过这次的实践过程，由于没注意到Python 语言特性，导致在设计过程中的一段时间没有发现错误源处。总而言之，如果能够很清晰地知道不同进程调度算法的不同之处之后，设计思路是很清晰的。这次设计虽然只选择了一个简易的题目，但也是由于时间有限，但还是有所收获的。

**4.2事先过程中出现的主要问题以及解决办法**

**问题：**

主要问题是，在设计过程中，由于没有意识到 Python 深复制与浅复制的区别和联系，导致当前进程在对应于运行队列中的进程信息无法同步跟新导致，运行结果出错。

**解决：**

在明确了解了 Python 的引用、浅复制、深复制区别之后，改变原有进程队列的复制方式，完成就绪队列对进程队列的赋值。

# 用户使用说明

用户可以在满足 Python 3.6 以上版本的条件下直接运行程序。模拟进程随机生成 10 个进程，进程运行时间、到达时间、优先数随机生成。可以在控制台查看进程具体信息，以及执行每一步的进行调度信息。

# 测试与运行结果

根据课本《操作系统-精髓与设计原理》第七版的第九章单处理器调度的P286-P287 给

出的例子进行程序测试。

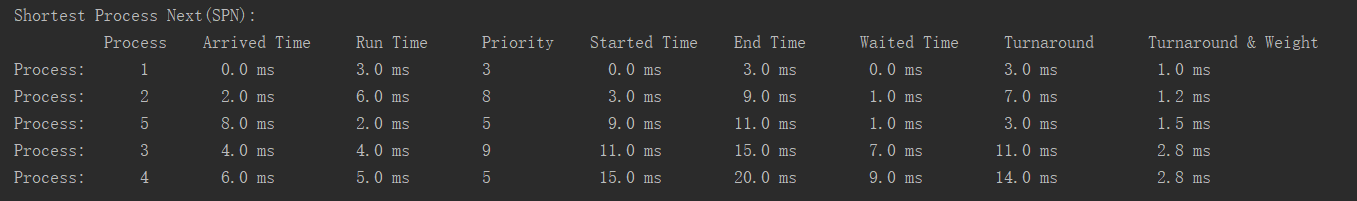
进程队列：

|  |  |  |  |
| --- | --- | --- | --- |
| 进程 | 到达时间 | 服务时间 | 优先数 |
| 1 | 0 | 3 | random.randint(2, 9) |
| 2 | 2 | 6 | random.randint(2, 9) |
| 3 | 4 | 4 | random.randint(2, 9) |
| 4 | 6 | 5 | random.randint(2, 9) |
| 5 | 8 | 2 | random.randint(2, 9) |

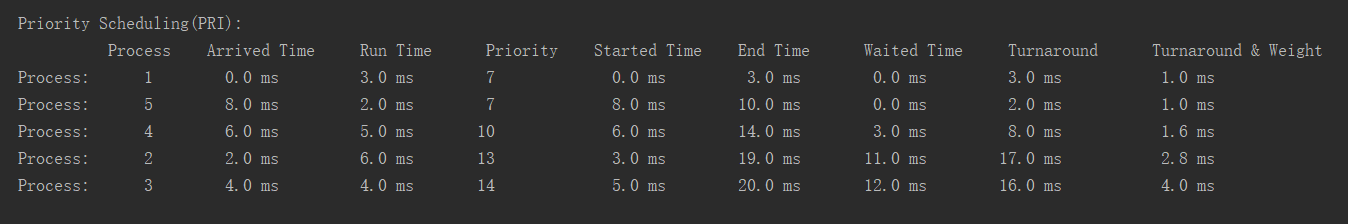
（1）最短剩余时间优先



（2）最短进程优先



（3）优先级



（4）时间片为1 的轮转



**分析：**测试结果与实际相符，程序成功实现模拟单处理系统对进程的调度。