南京信息工程大学 实验（实习）报告

实验（实习）名称 银行家算法避免死锁的模拟与实现 实验（实习）日期 得分 指导教师

系 计算机学院专业 物联网工程年级 2021 班次 2 姓名 学号

**一、实验目的**

1 进一步理解利用银行家算法避免死锁的问题；

2 掌握银行家算法，了解资源在进程并发执行中的资源分配策略。

3 理解和掌握安全序列、安全性算法。

**二、实验内容与步骤**

1 在了解和掌握银行家算法的基础上，编写一个实现动态资源分配的模拟程序，并能够有效地防止和避免死锁的发生。

2 要求至少五个进程三类资源。将调试结果显示在计算机屏幕上，再检测和笔算的一致性。

1. **实验步骤**

*# 导入numpy模块，用于矩阵运算*

import numpy as np

*# 定义资源的种类和进程的数目*

M = 3 *# 资源的种类*

N = 5 *# 进程的数目*

*# 系统中可利用的资源数目*

Available = np.array([10, 5, 7])

*# 每个进程对每种资源的最大需求*

Max = np.array([[7, 5, 3],

                [3, 2, 2],

                [9, 0, 2],

                [2, 2, 2],

                [4, 3, 3]])

*# 每个进程已分配的各类资源的数目*

Allocation = np.array([[0, 1, 0],

                       [2, 0, 0],

                       [3, 0, 2],

                       [2, 1, 1],

                       [0, 0, 2]])

*# 每个进程还需要的各类资源数*

Need = Max - Allocation

*# 安全序列*

Safe = []

*# 安全性算法*

def SafeAlgorithm():

    global Work, Finish, Safe *# 声明全局变量*

    Work = Available.copy() *# 工作向量，初始值等于Available向量*

    Finish = np.full(N, False) *# 完成向量，初始值全为False*

    k = 0 *# 安全序列的下标*

    while k < N:

        for i in range(N):

            if Finish[i] == False and (Need[i] <= Work).all(): *# 如果进程i未完成且需求小于等于工作向量*

                Work += Allocation[i] *# 更新工作向量*

                Finish[i] = True *# 更新完成向量*

                Safe.append(i) *# 将进程i加入安全序列*

        k += 1 *# 更新下标*

    if Finish.all(): *# 如果完成向量全为True*

        return True *# 返回True，表示系统处于安全状态*

    else:

        return False *# 返回False，表示系统处于不安全状态*

*# 银行家算法*

def BankerAlgorithm():

    global Available, Allocation, Need *# 声明全局变量*

    print("请输入进程号（0-%d）：" % (N - 1), end="")

    i = int(input()) *# 输入进程号*

    print("请输入请求向量：", end="")

    Request = np.array(list(map(int, input().split()))) *# 输入请求向量*

    if (Request > Need[i]).any(): *# 如果请求向量大于需求向量*

        print("请求出错！") *# 输出错误信息*

        return *# 结束函数*

    if (Request > Available).any(): *# 如果请求向量大于可利用向量*

        print("资源不足，进程%d需等待！" % i) *# 输出等待信息*

        return *# 结束函数*

*# 试探性地分配资源*

    Available -= Request

    Allocation[i] += Request

    Need[i] -= Request

*# 执行安全性算法*

    if SafeAlgorithm(): *# 如果系统处于安全状态*

        print("资源分配成功！") *# 输出成功信息*

        print("安全序列为：", end="")

        for j in Safe:

            print("P%d" % j, end=" ") *# 输出安全序列*

        print()

    else: *# 如果系统处于不安全状态*

        print("资源分配失败！") *# 输出失败信息*

*# 撤销试探性的分配*

        Available += Request

        Allocation[i] -= Request

        Need[i] += Request

*# 主函数*

if \_\_name\_\_ == "\_\_main\_\_":

*# 输出初始状态*

    print("初始状态：")

    print("Available:", Available)

    print("Max:", Max)

    print("Allocation:", Allocation)

    print("Need:", Need)

*# 循环执行银行家算法*

    while True:

        BankerAlgorithm()

