

（1）

|  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
|  | max | | | allocation | | | need | | | available | | |
| A | B | C | A | B | C | A | B | C | A | B | C |
| P1 | 5 | 5 | 9 | 2 | 1 | 2 | 3 | 4 | 7 | 2 | 3 | 3 |
| P2 | 5 | 3 | 6 | 4 | 0 | 2 | 1 | 3 | 4 |  |  |  |
| P3 | 4 | 0 | 11 | 4 | 0 | 5 | 0 | 0 | 6 |  |  |  |
| P4 | 4 | 2 | 5 | 2 | 0 | 4 | 2 | 2 | 1 |  |  |  |
| P5 | 4 | 2 | 4 | 3 | 1 | 4 | 1 | 1 | 0 |  |  |  |

是安全状态

安全序列：P4，P2，P3，P5，P1

（2）

用银行家算法进行检查：

Request2(0,3,4)<=Need2(1,3,4)

Request2(0,3,4)>Available(2,3,3)

让P2等待

（3）

用银行家算法进行检查：

Request4(2,0,1)<=Need4(2,2,1)

Request2(2,0,1)<=Available(2,3,3)

系统先假定可以为P4分配资源，并修改有关数据

再利用安全性算法检查此时系统是否安全

安全序列：P4，P2，P3，P5，P1

此时处于安全状态，可以进行分配。

（4）

用银行家算法进行检查：

Request1(0,2,0)<=Need1(3,4,7)

Request1(0,2,0)<=Available(0,3,2)

系统先假定可以为P1分配资源，并修改有关数据

再利用安全性算法检查此时系统是否安全

不处于安全状态，无法分配