银行家算法（Banker's Algorithm）是一个避免死锁（Deadlock）的著名算法。在避免死锁方法中允许进程动态地申请资源,但银行家算法在系统进行资源分配之前,应先计算此次分配资源的安全性,若分配不会导致系统进入不安全状态,则分配,否则等待。

例题：

****

**(1)**T0时刻，资源分配情况如下表：

|  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| 资源情况  进程 | 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 |  |  |  |

进行安全性算法：

初始时work 等于 available 向量为[2,3,3]

1. 分配给P4，Work变为[4,3,7]
2. 分配给P2，Work变为[8,3,9]
3. 分配给P3，Work变为[12,3,14]
4. 分配给P5，Work变为[15,4,18]
5. 分配给P1，Work变为[17,5,20]

存在安全队列：P4→P2→P3→P5→P1

T0时刻是安全状态

**(2)** request[0,3,4]<Need[1,3,4]

request[0,3,4]不小于Available[2,3,3]

尚无足够的资源，请求需要等待，无法分配资源

**(3)** request[2,0,1]<Need[2,2,1]

request[2,0,1]<Available[2,3,3]

先假定可以为P4分配资源，并修改表格值，由此形成的资源变化情况如下表：

|  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| 资源情况  进程 | 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 | 0 | 3 | 2 |
| P2 | 5 | 3 | 6 | 4 | 0 | 2 | 1 | 3 | 4 |  |  |  |
| P3 | 4 | 0 | 11 | 4 | 0 | 5 | 0 | 0 | 6 |  |  |  |
| P4 | 4 | 2 | 5 | 4 | 0 | 5 | 0 | 2 | 0 |  |  |  |
| P5 | 4 | 2 | 4 | 3 | 1 | 4 | 1 | 1 | 0 |  |  |  |

用安全性算法检查此时系统是否安全：

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| 资源情况  进程 | 原来的Work | | | 分配后的Work | | | Finish |
| A | B | C | A | B | C |
| P4 | 2 | 3 | 3 | 4 | 3 | 7 | true |
| P2 | 4 | 3 | 7 | 8 | 3 | 9 | true |
| P3 | 8 | 3 | 9 | 12 | 3 | 14 | true |
| P5 | 12 | 3 | 14 | 15 | 4 | 18 | true |
| P1 | 15 | 4 | 18 | 17 | 5 | 20 | true |

得到安全序列：P4→P2→P3→P5→P1

此时处于安全状态，可以进行分配

**(4)** 在(3)的基础上,Available是[0,3,2]

request1[0,2,0]<Need[3,4,7]

request1[0,2,0]<Available[0,3,2]

先假定可以为P1分配资源，并修改表格值，由此形成的资源变化情况如下表：

|  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| 资源情况  进程 | Max | | | Allocation | | | Need | | | Available | | |
| A | B | C | A | B | C | A | B | C | A | B | C |
| P1 | 5 | 5 | 9 | 2 | 3 | 2 | 3 | 2 | 7 | 0 | 1 | 2 |
| P2 | 5 | 3 | 6 | 4 | 0 | 2 | 1 | 3 | 4 |  |  |  |
| P3 | 4 | 0 | 11 | 4 | 0 | 5 | 0 | 0 | 6 |  |  |  |
| P4 | 4 | 2 | 5 | 4 | 0 | 5 | 0 | 2 | 0 |  |  |  |
| P5 | 4 | 2 | 4 | 3 | 1 | 4 | 1 | 1 | 0 |  |  |  |

Available(0,1,2)无法满足任何一个进程的Need，无法找到安全序列，所以无法分配