**SET A**

Solution for the Bankers Algorithm

3 a. Safe Algorithm:

**1.**Let ***Work*** and ***Finish*** be vectors of length *m* and *n,* respectively. Initialize  
***Work*** = ***Available*** and ***Finish***[*i*] = ***false*** for *i* = 0, 1, ..., *n* − 1.  
**2.** Find an index *i* such that both  
a. ***Finish***[*i*] == ***false***  
b. ***Need****i* ≤ ***Work***  
If no such *i* exists, go to step 4.  
**3. *Work*** = ***Work*** + ***Allocation****i*  
***Finish***[*i*] = ***true***  
Go to step 2.  
**4.** If ***Finish***[*i*] == ***true*** for all *i,* then the system is in a safe state.  
This algorithm may require an order of *m* × *n*2 operations to determine whether  
a state is safe.

**b. Resource-Request Algorithm**  
Next, we describe the algorithm for determining whether requests can be safely  
granted.  
Let ***Request****i* be the request vector for process *Pi*. If ***Request****i* [ *j*] == *k*, then  
process *Pi* wants *k* instances of resource type *Rj*. When a request for resources  
is made by process *Pi*, the following actions are taken:  
**1.** If ***Request****i* ≤ ***Need****i*, go to step 2. Otherwise, raise an error condition, since  
the process has exceeded its maximum claim.  
**2.** If ***Request****i* ≤ ***Available,*** go to step 3. Otherwise, *Pi* must wait, since the  
resources are not available.  
**3.** Have the system pretend to have allocated the requested resources to  
process *Pi* by modifying the state as follows:  
***Available*** = ***Available***–***Request****i* ;  
***Allocation****i* = ***Allocation****i* + ***Request****i*;  
***Need****i* = ***Need****i* –***Request****i* ;  
If the resulting resource-allocation state is safe, the transaction is completed, and process *Pi* is allocated its resources. However, if the new state  
is unsafe, then *Pi* must wait for ***Request****i****,*** and the old resource-allocation  
state is restored.

3. b. Compute the safe state sequence for the given scenario. (5)

Available [A B C D]=[10 5 4 4] , Max and Allocation vectors are given below:

c. Check the request from P2 : < 1 3 1 0 > with respect to the resource types <A B C D > could be granted or not. If the request is granted, give the safe sequence (5)

**Allocation Max**

|  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| process no | a | b | c | d |  | a | b | c | d |
| 1 | 3 | 2 | 1 | 0 |  | 5 | 5 | 1 | 0 |
| 2 | 2 | 1 | 0 | 0 |  | 4 | 2 | 0 | 1 |
| 3 | 2 | 0 | 1 | 1 |  | 4 | 3 | 2 | 2 |
| 4 | 1 | 0 | 0 | 1 |  | 2 | 0 | 1 | 1 |
| 5 | 0 | 1 | 0 | 1 |  | 0 | 1 | 1 | 2 |
| Allocated resources | **8** | **4** | **2** | **3** |  |  |  |  |  |

Consider the initial resources : [A B C D] = [10 5 4 4]

Need vector :

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| PROCESS NO | A | B | C | D |
| 1 | 2 | 3 | 0 | 0 |
| 2 | 2 | 1 | 0 | 1 |
| 3 | 2 | 3 | 1 | 1 |
| 4 | 1 | 0 | 1 | 0 |
| 5 | 0 | 0 | 1 | 1 |

Compute the available vector after allocation :

Available = initial resources – Allocated resources

= [2,1,2,1]

Step 1 : Work = [ 2 1 2 1] : initialize finish [i] = false; for all i where i=1 to 5

Step 2 : Choose an i such that whose finish[i]=false and need i <= available ;

then the following are carried out else carry out step no.4

if so, work=work + allocation i , if no such i exists ; then can carry out the last step.

Step 3 : Start from the beginning i=1; ( Process 1)

(i) Need of Process 1 [2 3 0 0] not <= work [2 1 2 1] ;

(ii) Hence, Process 2 is choosen [ 2 1 0 1] <=work [2 1 2 1] and Finish[2] = False;

Therefore, Work = Work + allocation 2

Work =Work [2 1 2 1] +allocation 2 [2 1 0 0] = [ 4 2 2 1]

**Finish[2]=True;**

(iii) Next, Process 3 is choosen, need of process [ 2 3 1 1] is not < = work [ 4 2 2 1] ,

Hence process 4 is considered;

Need of process 4 is [1 0 1 0 ] < =work [4 2 2 1] and Finish[4]=False;

Therefore : Work = Work +allocation4

(iv) Work =Work [4 2 2 1] + allocation 4[ 1 0 0 1] = [5 2 2 2] and **Finish[4] = True.**

Next Process 5 is choosen ;

Check if Need of process5 [ 0 0 1 1] < = work [5 2 2 2] and Finish[5]=false;

(v) Now, Process 5 can be granted ,

Work = Work + allocation5.

Work= Work[5 2 2 2 ] + allocation5[ 0 1 0 1] = [5 3 2 3 ] and  **Finish [5]= True;**

(vi)Next ,Process 1 is choosen,

Need of process1 [2 3 0 0] < = work [5 3 2 3] and Finish[i] = False;

Work = Work[5 3 2 3] + allocation1[ 3 2 1 0] = [8 5 3 3] and  **Finish [1]=True;**

(vii)Finally, Process 3 can be choosen ,

Need of process 3 [2 3 1 1] <= work[8 5 3 3] and Finish[i] = False;

Work =Work[ 8 5 3 3] +allocation3[2 0 1 1] = [10 5 4 4] and **Finish[3] = True.**

Step 4: **if Finish i = true; for all i= 1 to 4 ; the system is in a safe state.**

**The safe state sequence are : Process 2,Process 4,Process 5 , Process 1 and Process 3.**

3. c. Check the request from P2 : < 1 3 1 0 > with respect to the resource types <A B C D > could be granted or not. If the request is granted, give the safe sequence

Request from p2 < 1 3 1 0 > : Need of Process 2 is < 2 1 0 1>

**: Generally the request raised should be <= <2 1 0 1>**

**[This request cannot be accepted]**