Banker's Algorithm

→ named as "banker's" since could be used in banking system to ensure that bank never allocated its available cash in such a way that it could no longer satisfy the needs of all customers.

Datastructures used

* Available: indicates available resources of each type.

* Max: gives max demand of each process

Notes * Allocation: defines no of resources currently allocated to each process

* Need: indicates remaining resources needed by each process to complete its task

Need[i][j] = Max(i) − Allocation(i)

Safety Algorithm

1. Let us initialize
   work = Available
   finish[i] = false for i = 0, 1, ---, n−1

2. Find an i such that
   a) finish[i] = false
   b) $Need_i \leq work$
   if no such i exists goto 4

3. $work = work + Allocation_i$
   finish[i] = true
   goto step 2

4. if finish[i] = true for all i
   then system is in safe state.

Resource Request Algorithm for P_i

1- if Request $\leq$ Need goto 2
otherwise error.

2- if Request $\leq$ Available goto 3
otherwise P_i must wait
since no resources available

3- Assume that resources are
allocated.

Available = Available − Request$_i$

Allocation$_i$ = Allocation + Request$_i$

Need$_i$ = Need$_i$ − Request$_i$

Notes

Example

* Consider 5 Processes P_0 to P_4
* 3 resource types with instances
  A : 10 ; B = 5 ; C = 7

Snapshot at time T_0 :

	Allocation			Max			Available		
	A	B	C	A	B	C	A	B	C
P_0	0	1	0	7	5	3	3	3	2
P_1	2	0	0	3	2	2			
P_2	3	0	2	9	0	2			
P_3	2	1	1	2	2	2			
P_4	0	0	2	4	3	3			

Note: $\boxed{\text{Available} = \Sigma\, \text{Alloc}_i - \text{Total}_i}$

13 MONDAY JUNE 2022

164-201 Week 25

	Alloc			Max			Avail			Need = Max − Alloc		
	A	B	C	A	B	C	A	B	C	A	B	C
P0	0	1	0	7	5	3	3	3	2	7	4	3
P1	2	0	0	3	2	2				1	2	2
P2	3	0	2	9	0	2				6	0	0
P3	2	1	1	2	2	2				0	1	1
P4	0	0	2	4	3	3				4	3	1

Apply Safety Algorithm

9

1) Initially work = Available

10 $\therefore$ work = 3 3 2

finish[i] = false $\forall i$

11

P0:

12 2) Finish[0] = false

need$_0$ $\leq$ work

1 7 4 3 $\leq$ 3 3 2 (NO)

2 so goto 4

4) Finish[0] = false (still not true)

3

P1:

4 2) Finish[1] = false

need$_1$ $\leq$ work

5 1 2 2 $\leq$ 3 3 2 (yes)

6

3) work = work + Alloc$_1$

= 3 3 2 + 2 0 0

Notes

$\boxed{\text{work} = 5\ 3\ 2}$

Finish[1] = True

9 $\underline{\underline{P_2}}$:

10 2) Finish[2] = false

11 $need_2 \leq work$

12 $6 \; 0 \; 0 \leq 5 \; 3 \; 2$ (NO)

1 4) Finish[2] = false

2 $\underline{\underline{P_3}}$:

3 2) Finish[3] = false

$need_3 \leq work$

4 $0 \; 1 \; 1 \leq 5 \; 3 \; 2$ (yes)

5 3) $Work = work + Alloc_3$

6 $= 5 \; 3 \; 2 + 2 \; 1 \; 1$

$\boxed{work = 7 \; 4 \; 3}$

Notes

Finish[3] = True

P4:

2) Finish[4] = false
$need_4 \le work$
4 3 1 ≤ 7 4 3 (yes)

3) $work = work + Alloc_4$
= 7 4 3 + 0 0 2
$\boxed{work = 7\ 4\ 5}$
Finish[4] = True

P0:

2) Finish[0] = false
$need_0 \le work$
7 4 3 ≤ 7 4 5 (yes)
3) $work = work + Alloc_0$
= 745 + 0 1 0
$\boxed{work = 7\ 5\ 5}$
Finish[0] = True

Safe sequence
<P1, P3, P4, P0, P2>

P2:

2) Finish[2] = false
$need_2 \le work$
6 0 0 ≤ 7 5 5 (yes)
3) $work = work + Alloc_2$
= 7 5 5 + 3 0 2
= 10 5 7

Finish[2] = True

Consider P_1 makes additional request (1, 0, 2)
Check can it be granted

Apply Resource Request algorithm

1- $Request_i \leq need_i$

1 0 2 $\leq$ 7 4 3 (yes)

2) $Request_i \leq Avail$

1 0 2 $\leq$ 3 3 2 (yes)

3) So allocated

* $Avail = Avail - Req_i$

= 3 3 2 - 1 0 2

$\boxed{Avail = 2\ 3\ 0}$

* $Alloc_i = Alloc_i + Req_i$

= 2 0 0 + 1 0 2

$\boxed{Alloc_1 = 3\ 0\ 2}$

* $Need_1 = Need_1 - Req_1$

= 1 2 2 - 1 0 2

$\boxed{Need_1 = 0\ 2\ 0}$

Notes

Request can be granted

Revised table

	Alloc A	B	C	Need A	B	C	Avail A	B	C
P_0	0	1	0	7	4	3	2	3	0
P_1	3	0	2	0	2	0			
P_2	3	0	2	9	0	2			
P_3	2	1	1	2	2	2			
P_4	0	0	2	4	3	3			

Notes