7.6 (a) safe，Available 增加讓 deadlock 更不容易發生

(b) unsafe，Available 減少使需要特定數量的 available resource 可能找不到 safe sequence 而進入 unsafe state

(c) unsafe，Max 增加会使 process "need" 也增加，可能找不到 safe sequence 而進入 unsafe

(d) safe，與前面敘述情形相反，deadlock 不会發生

(e) safe，假設 resource 是分配給還未進入 unsafe state 的 new process 就為 safe

(f) safe，processes 減少沒有進入 deadlock 的風險

7.13 Total instance (A,B,C,D) = (12, 12, 8, 10)

	Allocation A B C D	Max A B C D	Available A B C D	Need A B C D
① P0	~~2 0 0 1~~	~~4 2 1 2~~	① 3 3 2 1	2 2 1 1 √①
③ P1	~~3 1 2 1~~	~~5 2 5 2~~	② 5 3 2 2	2 1 3 1 √③
④ P2	~~2 1 0 3~~	~~2 3 1 6~~	③ 6 6 3 4	0 2 1 3 √④
② P3	~~1 3 1 2~~	~~1 4 2 4~~	④ 9 9 5 5	0 1 1 2 √②
⑤ P4	1 4 3 2	3 6 6 5		2 2 3 3
	t) 9 9 6 9			
	t) 3 3 2 1			
	12,12,8,10			

(a) one of safe sequence: P0 → P3 → P1 → P2 → P4 ※

(b) request P1 = (1,1,0,0)

	Allocation A B C D	Max A B C D	Available A B C D	Need A B C D
① P0	~~2 0 0 1~~ 3 1 0 1	4 2 1 2	~~3 3 2 1~~ 2 2 2 1	1 1 1 1 √①
P1	3 1 2 1	5 2 5 2	5 3 2 2	2 1 3 1
P2	2 1 0 3	2 3 1 6	可以：	0 2 1 3
P3	1 3 1 2	1 4 2 4		0 1 1 2
P4	1 4 3 2	3 6 6 5		2 2 3 3

Ans: Yes, safe sequence: P0 → P3 → P1 → P2 → P4 ※

(c) Request P4 = (0,0,2,0)

	Allocation A B C D	Max A B C D	Available A B C D	Need A B C D
P0	2 0 0 1	4 2 1 2	~~3 3 2 1~~ 3 3 0 1	2 2 1 1
P1	3 1 2 1	5 2 5 2		2 1 3 1
P2	2 1 0 3	2 3 1 6		0 2 1 3
P3	1 3 1 2	1 4 2 4		0 1 1 2
P4	~~1 4 3 2~~ 1 4 5 2	3 6 6 5		~~2 2 3 3~~ 2 2 1 3

no one can be chosen → no safe sequence

Ans: the request won't be granted ※

7.15
```
semaphore ok_to_cross = 1;

void enter_bridge() {
    ok_to_cross.wait();
}

void exit_bridge() {
    ok_to_cross.signal();
}
```

8.1 Internal fragmentation ① 發生在 process 被 allocate 給比自己需要的 memory 空間更大的 memory block 時，剩餘的 memory 無法被其它 process 所用.

② 發生在 fixed-partition allocation

External fragmentation ① 則發生在有足夠的總空間讓其它 process 執行，但因空間不連續使其它 process 不能使用.

② 發生在 variable-size allocation

8.9. paging 需要 paging table 來存每個 entry/page 對應到哪個 physical address，因此存在更多 memory overhead.

segmentation 在每一 segment 只需存2個 register，一個存 base of the segment，另一個存 extent of the segment

8.16 32bit logical address, 4KB page size (2^{12}) 12bit, 512 MB ($2^9 \times 2^{20}$) physical memory (29bit, 2^{29})

How many entries?

(a) single-level page table: $2^{32}/2^{12} = 2^{20}$ entries ※ (1,048,576 entries)

(b) inverted page table: 512 MB = $2^9 \times 2^{20} = 2^{29}$, $2^{29}/2^{12} = 2^{17}$ entries ※ (131,072 entries)

9.8 7,2,3,1,2,5,3,4,6,7,7,1,0,5,4,6,2,3,0,1, 3 frames

(a) LRU

7	1	3	7	5	2	1
2	4	1	4	3		
3	5	6	0	6	0	

⇒ 18 次 ※

(b) FIFO

7	1	6	0	6	0	
2	5	7	5	2	1	
3	4	1	4	3		

⇒ 17 次 ※

(c) optimal replacement

7	1					
2	5	4	6	2	3	
3	4	6	7	0		

⇒ 13 次 ※

9.11

1) # page fault LFU < LRU

ex: situation 1 2 3 4 5 1, 4 pages in memory

1	5	1
2	5	
3	.	
4	.	

當 page 5 要放入 memory 時, LFU 會因 1 access 過 2 次而不會被 replace,
但若是 LRU, 則因最久沒使用而 replace 1. 如此, 下一個 1 就會多 1 次
page fault. 此時, LFU 比 LRU 的 page fault 少。

2) # page fault LFU > LRU

ex: situation 1 2 3 4 5 2, 4 pages in memory

1	5	
2	.	
3	.	5
4	.	

當 page 5 要放入 memory 時, LRU 會 replace 1 的位置, LFU 則可能 replace 2 而
這就多 1 次 page fault。此時, LRU 比 LFU 的 page fault 少。

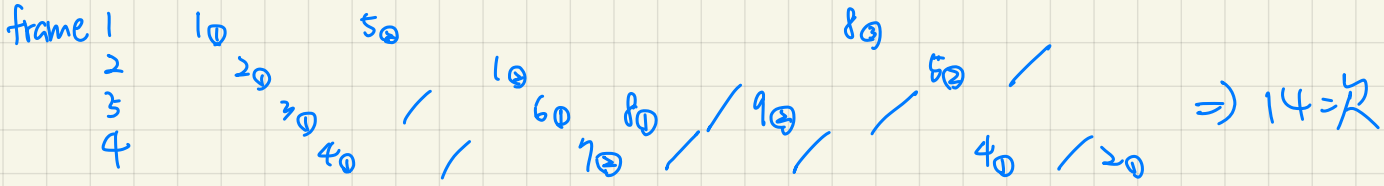
9.17 (a) ① init value = 0,

② counter increase when a new page is associated with that frame.

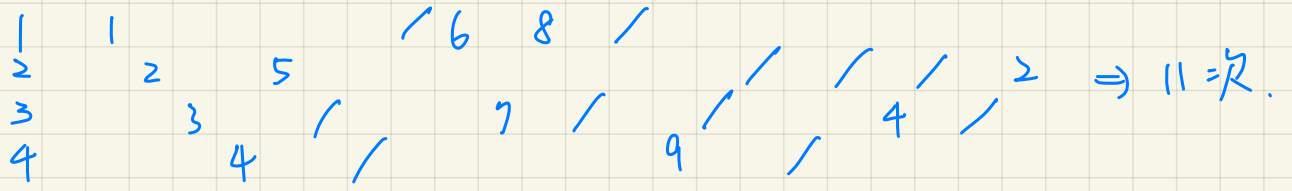
③ " decrease when one of the pages associated with that frame is no longer required

④ find a frame with the smallest counter. Use FIFO for breaking ties.

b. 1 2 3 4 5 2 4 1 6 7 8 7 8 9 7 8 9 5 4 5 4 2 , 4 pages



C.



9.19 ① Thrashing is caused by underallocation of the minimum number of pages required by a process forcing it to continuously page fault.

② The system can detect thrashing by evaluating the level of CPU utilization as compared to the level of multiprogramming.

③ It can be eliminated by reducing the level of multiprogramming.