1.Please list five scheduling criteria(標準)

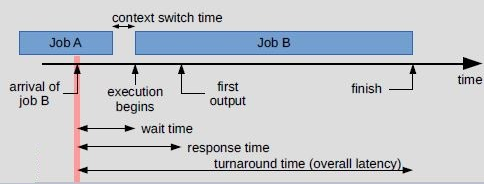
CPU utilization - 使CPU忙碌

Throughput - 每單位時間所完成產量

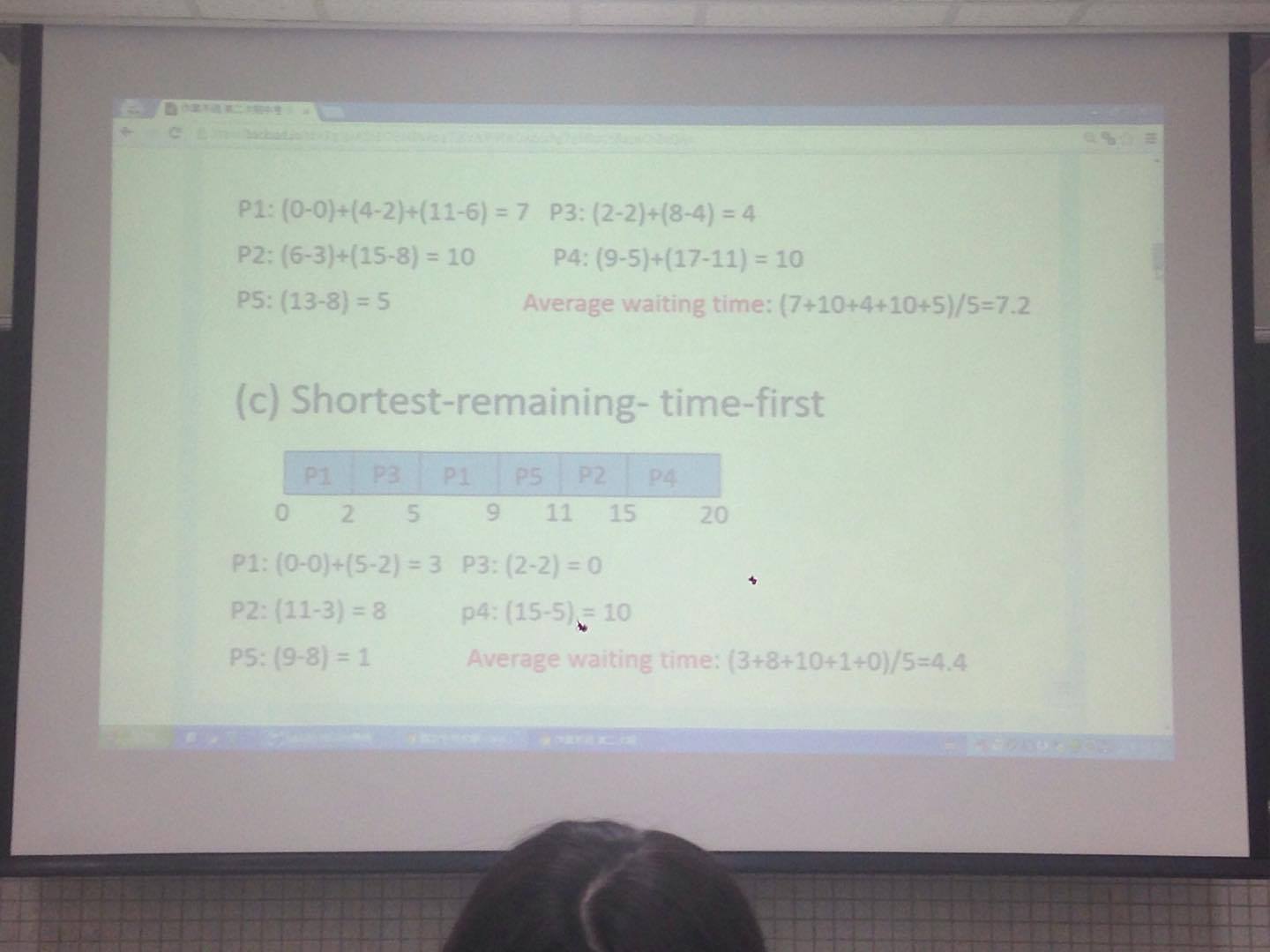
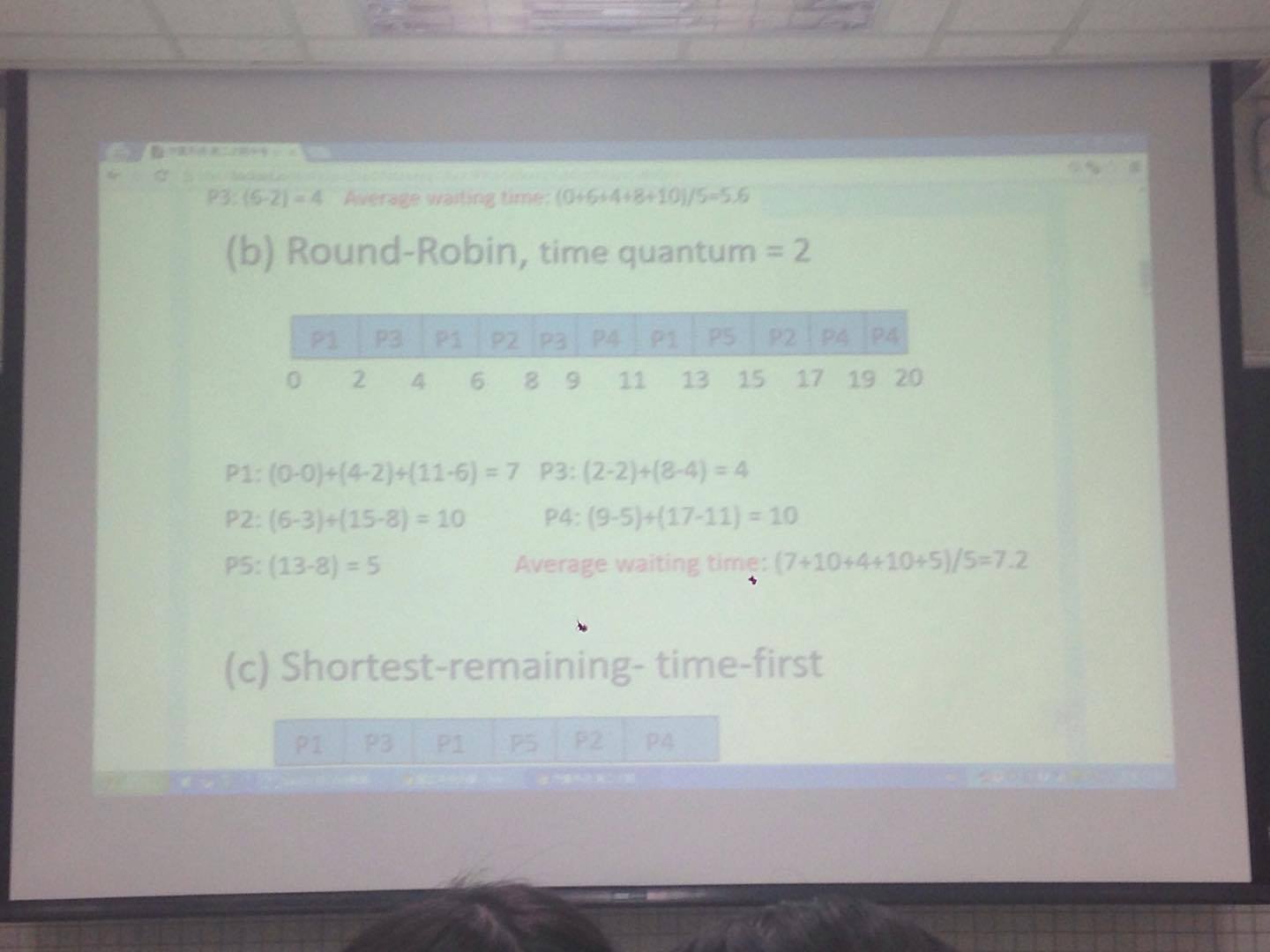
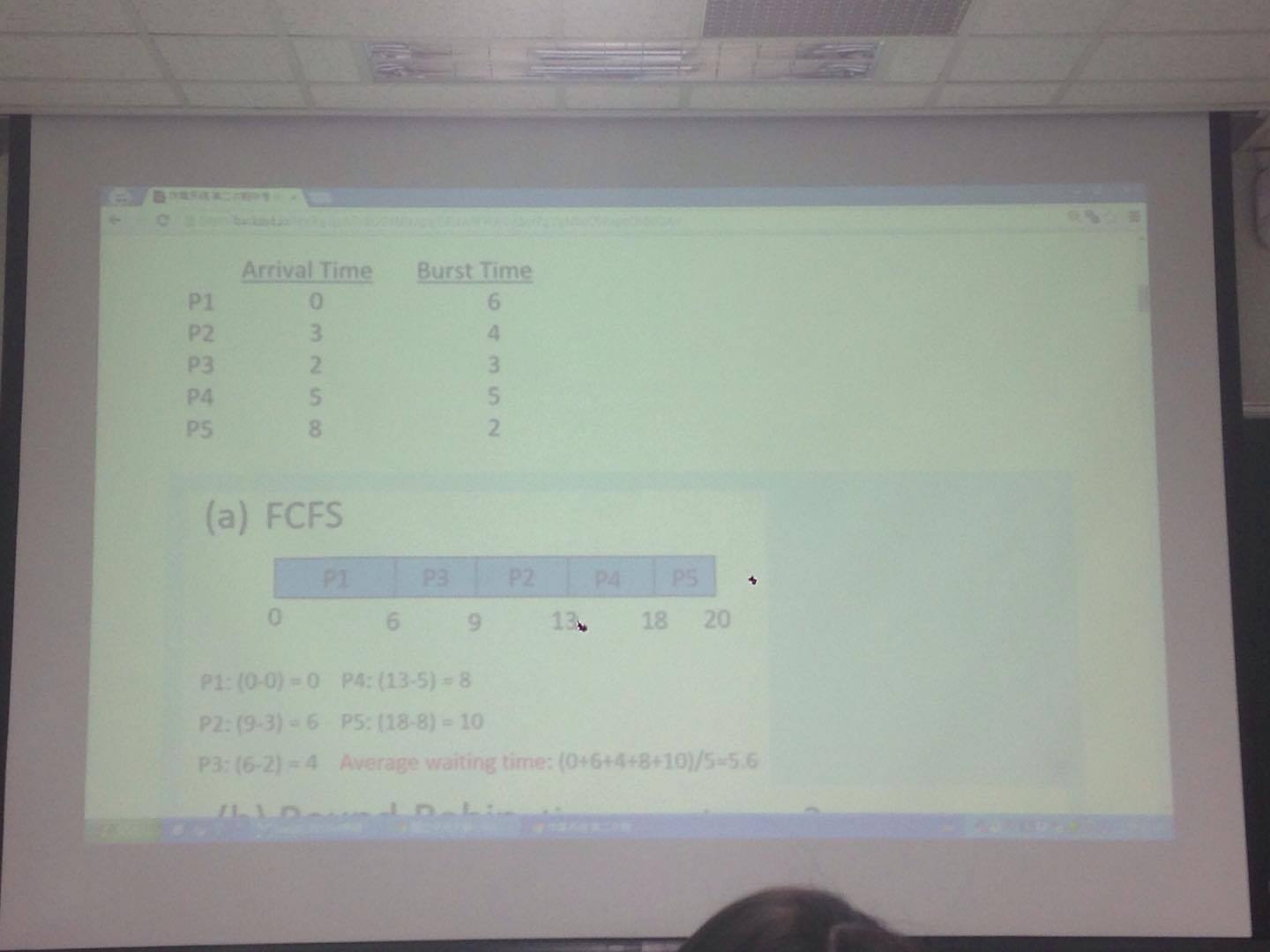
Turnaround time - 工作提出到完成的時間

Waiting time - 在ready queque 等待的時間

Response time - request提出到反應的時間



2.Pls draw the Gantt charts and calculate the average waiting time for the following processes for the problems shown below



4.Pls explain multiple feedback queue

**把ready queue分成**

Foreground(需交談)

Background(需批量處理)

process可以在queue間移動，To solve starvation problem

5.Pls explain the critical-section

process中存取共享資源的程式片段

6.Pls explain how “Disable Interrupt” and “Critical-Section Problem” solve race condition problem.

Disable interrupt : 為確保共享變數之存取敘述執行時不受任何中斷干擾，可以自動地執行(For CPU)

Critical-section design : 提供對共享變數之存取的互斥控制，確保資料的正確性(for data)

7.Pls list and explain the three criteria of critical-section design (solution to critical-section problem)

**Mutual Exclusion(相互排斥)：**同一時間內只有一個process在critical section

**Progress：**決定哪個process可以進入，不想進入critical section的process，不能妨礙其它process進入

**Bounded Waiting：**每個process等待進入critical section的時間是有限的。

8.

Peterson’s solution is a classic software based solution to the critical-section problem.

flag[i]=true;  //自己想進入

turn=j; //將優先權先給對方

while(flag[j] && turn==j ) do

  //當對方想進入且優先權在對方手上時,則自己在while中空轉,而一旦對方不想進入或是對方把優先權讓回來時,自己能立刻進入

10.What is the difference between mutex and semaphore?

Process使用mutex時，process持有mutex執行critical-section來存取資源，然後釋放mutex

Process使用semaphore時，process總是發出信號(signal)或接收信號(wait)，同一個process不會先後進行signal與wait，是為了保護process的執行同步性

Mutex是一把鑰匙，一個人拿了就可進入一個房間，出來的時候把鑰匙交給隊列的第一個。

Semaphore是一件可以容納N人的房間，如果人不滿就可以進去，如果人滿了，就要等待有人出來。對於N=1的情況，稱為binary semaphore。

processes陷入互相等待的情況(Circular waiting)  
造成processes接無法往下執行，使得CPU utilization及Throughput大幅降低

11.Pls explain 4 necessary condition for “Deadlock”

**Mutual exclusion** – 同一時間內，只有一個process使用，其它的process必須wait，直到該process釋放resource為止

**Hold and Wait** – process持有部分資源且又在等待其它processes所持有的資源

**No Preemption** – process不可搶奪其它waiting process所持有的資源，除非其自願釋放

**Circular waiting**

存在一組process

P0→P1→P2→...→Pn→P0

P0~Pn形成Circular waiting

12.Pls explain three solutions to deadlock

**Deadlock prevention:**

打破 Mutual Exclusion : 不可能

打破 hold & wait : 保證一個process在要求一項資源時，不可以佔用任何其它的資源。

打破 no preemption : 改為preemption即可

打破 circular waiting: 確保circular waiting的條件不成立，強迫安排一個線性的順序。

**Deadlock Avoidance:**

當process提資源申請(Request)時，則OS依據

1.系統目前可用的資源數量

2.各process對資源的需求量

3.各process目前持有的資源量

執行Banker's Algorithm(內含Safety Algorithm)判斷Safe state核准

申請，不行則否決此次申請，稍後再重新申請。

**Deadlock Detection & Recovery:**

偵測死結是否存在，若死結存在，則必須打破死結，恢復正常的機制。

優點：resource utilization較高，Throughput↑

缺點：cost太高

13.Pls explain two ways Deadlock Recovery work

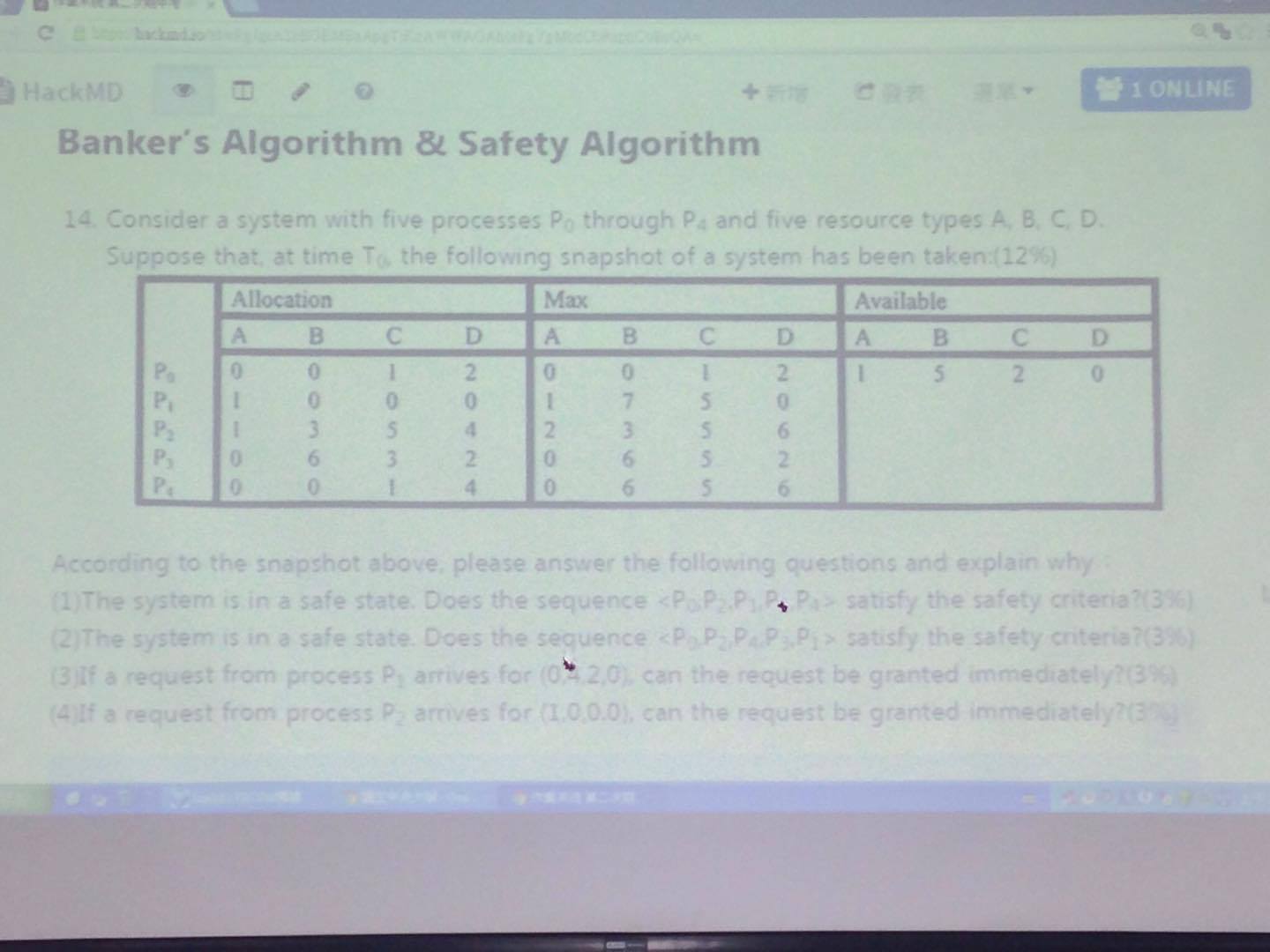
1. Kill process: kill all process in deadlock:

Kill one process, then apply Detection algorithm 若仍有deadlock repeat

2. Resource preemption:

先排出victim，rollback回到safe state，重新將process回復到該狀態

14.Condiser a system with five processes P0 through P4 and five resource types A, B, C, D. Suppose that at time T0 the following snapshot of a system has been taken.



1. The system is in a safe state. Does the sequence 0 2 1 3 4 satisfy the safety criteria?

在P0->P2->P3->P4可以，因為P0&P2取完資源就可以符合所有process需求

1. The system is in a safe state. Does the sequence 3 2 4 3 1 satisfy the safety criteria?

同1

1. If a request from process P1 arrives for (0,4,2,0) can the request be granted immediately?

取完可以在P0->P2->P3->P4->P1下做完

1. If a request from process P2 arrives for (1,0,0,0) can the request be granted immediately?

同上