Sistemi Operativi
a.a. 2004-2005

Esercizi - 2

Renzo Davoli
Alberto Montresor

Esercizio - 24/09/02, es. 3

- Sia dato uno scheduler Round Robin per un sistema monoprocessore con time slice di 4ms. Sia data la storia esecutiva dei seguenti processi:

```
P1: 5ms CPU, 10 ms I/O un.1, 4ms CPU, 10 ms I/O un 1, 3 ms CPU

P2: 6ms CPU, 10 ms I/O un.1, 5ms CPU, 10 ms I/O un 2, 3 ms CPU

P3: 5ms CPU, 10 ms I/O un.1, 2ms CPU, 10 ms I/O un 2, 3 ms CPU

P4: 5ms CPU, 10 ms I/O un.2, 8ms CPU, 10 ms I/O un 2, 4 ms CPU
```

- mostrare il diagramma di Gannt e calcolare il tempo totale di esecuzione.

Esercizio - 14/02/02, es. 1

- Sia dato uno scheduler Round Robin per un sistema monoprocessore con time slice di 3 ms e incremento del time slice di 1ms per ogni time slice completamente utilizzato (senza I/O). L'incremento deve essere calcolato processo per processo e dopo che un processo ha terminato l'I/O, il time slice torna a 3 ms. Sia data la storia esecutiva dei seguenti processi:

```
P1: 3ms CPU, 10 ms I/O un.1, 14ms CPU, 10 ms I/O un 1, 3 ms CPU

P2: 6ms CPU, 10 ms I/O un.1,  5ms CPU, 10 ms I/O un 2, 3 ms CPU

P3: 7ms CPU, 10 ms I/O un.2, 12ms CPU, 10 ms I/O un 2, 3 ms CPU

P4: 5ms CPU, 10 ms I/O un.2,  8ms CPU, 10 ms I/O un 1, 4 ms CPU
```

- mostrare il diagramma di Gannt e calcolare il tempo totale di esecuzione.

Esercizio - 24/09/02, es. 2

- **Disegnare un grafo di Holt che rappresenti un deadlock e nel quale occorra terminare non meno di tre processi per eliminare completamente il blocco.**

Esercizio - 24/09/02, es. 1

- L'algoritmo del Banchiere multivaluta non è equivalente a molteplici istanze dell'algoritmo monovaluta.
- Mostrare con un semplice esempio che uno stato unsafe di un Banchiere a due valute può risultare safe se esaminiamo due Banchieri che indipendentemente gestiscono le due differenti valute.

Esercizio - 17/01/02, es. 3

- **Dimostrare che l'algoritmo dell'orologio non è a stack**
- **Suggerimento:**
  - mostrare che soffre dell'anomalia di Belady

Esercizio - 24/09/02, es. 1

- **Un algoritmo di rimpiazzamento è definito come segue.**
  - Detto *t=timetick()* il numero di tick dalla accensione della macchina sia detto *f = t % NF* (dove *NF* è il numero di frame disponibili).
  - Qualora sia necessario liberare un frame, si decide di eliminare la pagina nel frame f.
- **E' un algoritmo a stack? Dare una risposta e motivarla.**

Esercizio - 14/02/02, es. 3

- Fissato il numero di frame, mostrare che esistono stringhe di riferimenti per le quali LFU (Least Frequently Used) ha un numero minore di page fault di LRU (Least Recently Used) e viceversa (non è sempre migliore LRU né LFU)

Esercizio - 02/07/02, es. 2

- Mostrare un semplice caso nel quale gli algoritmi LRU e MIN abbiano lo stesso numero di page fault (per non incorrere in casi banali il numero dei page fault deve essere maggiore del doppio del numero di frame)

Esercizio - 02/07/02, es. 3

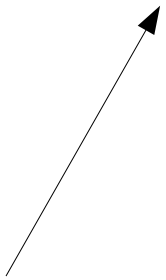
- **Un costruttore di sistemi operativi decide di usare un algoritmo di scheduling per la CPU che utilizza time slice e priorità. Alla fine di ogni time slice ogni processo tranne quello correntemente running ha la propria priorità aumentata di un'unità.**
- **Siano dati tre processi che non svolgono I/O in un sistema con time slice posto a 1ms.**
  - Il processo P1 ha priorità 3 e ha necessità di 10 ms di CPU.
  - Il processo P2 ha priorità 2 e ha necessità di 5 ms di CPU.
  - Il processo P3 ha priorità 1 e ha necessità di 2 ms di CPU.
- **Mostrare il relativo diagramma di Gannt. A quale tempo termineranno rispettivamente i processi?**

Esercizio - 04/06/02, es. 3

- Mostrare un esempio nel quale l'algoritmo di rimpiazzamento MRU (most recently used) compia meno page fault dell'algoritmo LRU (least recently used)

Esercizio - 04/06/02, es. 1

- **Sia dato un disco con velocità di seek di 1 traccia per ms e che impiega 1 ms a leggere/scrivere dati nella traccia corrente (indipendentemente dal numero delle richieste, ha un buffer di traccia)**
- **Sia data la seguente sequenza di richieste**
  - tempo t= 0 ms, traccia 1
  - tempo t=10 ms, traccia 20
  - tempo t=15 ms, traccia 5
  - tempo t=25 ms, traccia 5
  - tempo t=30 ms, traccia 4
  - tempo t=35 ms, traccia 10
  - tempo t=40 ms, traccia 10
  - tempo t=45 ms, traccia 20
  - tempo t=50 ms, traccia 1



- **Si mostri il percorso delle testine e si calcoli il tempo necessario per completare tutte le richieste nel caso si utilizzino gli algoritmi LOOK e C-LOOK.**

Esercizio 1 - Parte A

- **Si consideri un banchiere che gestisca due valute, e si consideri lo stato seguente:**

Valuta 1:

IC=54; COH 4

i	C_i	P_i	N_i
1	20	10	10
2	14	10	4
3	10	8	2
4	22	12	10
5	28	10	18

Valuta 2:

IC=54; COH=6

C_i	P_i	N_i
28	14	14
14	8	6
24	16	8
8	6	2
20	4	16

A) Dimostrare che tale stato è safe per l'algoritmo del banchiere multivaluta

Esercizio 1 - Parte A - Soluzione

- **Si consideri un banchiere che gestisca due valute, e si consideri lo stato seguente:**

Valuta 1: IC=54; COH 4					Valuta 2: IC=54; COH=6			
i	C_i	P_i	N_i	$avail_i$	C_i	P_i	N_i	$avail_i$
2	14	10	4	4	14	8	6	6
1	20	10	10	14	28	14	14	14
3	10	8	2	24	24	16	8	28
4	22	12	10	32	8	6	2	44
5	28	10	18	44	20	4	16	50

E' safe perchè è possibile trovare una sequenza (2-1-3-4-5) tale per cui $n_{s(j)} \leq avail[j]$, con $j=1 \ldots N$

Esercizio 1 - Parte B

- Mostrare un esempio di richiesta (della forma ``il processo *x* richiede *y* unità della valuta *z*'') tale per cui (i) la richiesta sia soddisfacibile, dato il saldo di cassa attuale e (ii) la richiesta debba essere rifiutata dall'algoritmo del banchiere multivaluta, in quanto porterebbe ad uno stato unsafe. Mostrare lo stato risultante ed evidenziare il motivo del problema.

Esercizio 1 - Parte B - Soluzione

- **Il processo 4 richiede 2 unità della valuta 2**

Valuta 1:

IC=54; COH 4

i	C_i	P_i	N_i
1	20	10	10
2	14	10	4
3	10	8	2
4	22	12	10
5	28	10	18

Valuta 2:

IC=54; COH=4

C_i	P_i	N_i
28	14	14
14	8	6
24	16	8
8	8	0
20	4	16

La richiesta è soddisfacibile e porta a questo stato. Non è safe perchè non è possibile trovare una sequenza *s* tale per cui

$n_{s(j)} \leq avail[j]$, con $j=1 \ldots N$

Esercizio 2

- Si consideri l'algoritmo Rate Monotonic per processi real-time periodici.
- Si mostri un esempio di almeno quattro processi che NON SIANO schedulabili secondo la condizione di Liu and Layland, ma che SIANO effettivamente schedulabili dall'algoritmo.
- A tal scopo, mostrare il grafo di Ganntt relativo allo scheduling di tale algoritmo.
- Per evitare casi banali, ogni processo deve avere un rapporto C/T (costo computazionale/periodicità) di almeno 0.14.
- Si ricorda che $4(2^{1/4}-1)$ è circa uguale a 0.743.

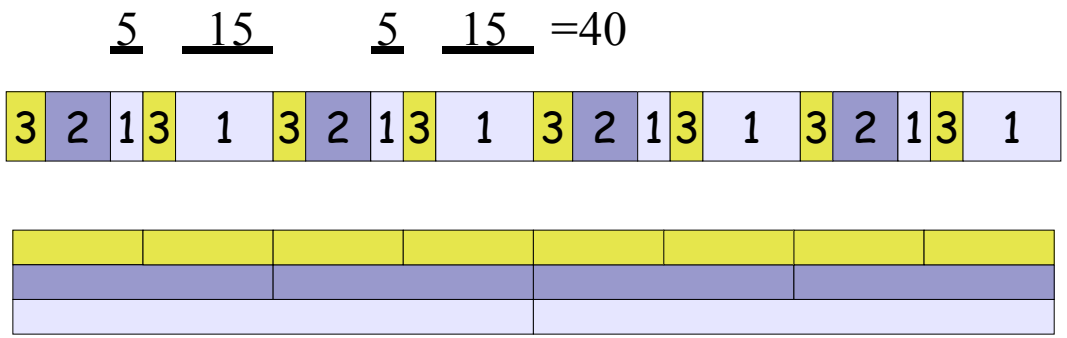
Esercizio 2 - Esempio da cui partire (3 processi)

	T	C	Priorità	C/T
Task 1	80	40	1 (min)	0.5
Task 2	40	10	2	0.25
Task 3	20	5	3 (max)	0.25
				1.00

$\geq 3(2^{1/3}-1)=0.778$

Condizione Non OK

Ma in realtà è schedulabile



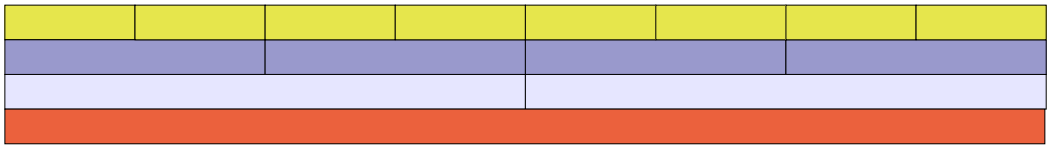
Esercizio 2 - Soluzione

	T	C	Priorità	C/T
Task 1	160	40	1 (min)	0.25
Task 2	80	20	2	0.25
Task 3	40	10	3	0.25
Task 4	20	5	4 (max)	0.25
				1.00

$\geq 4(2^{1/4}-1)=0.743$

Condizione Non OK

Ma in realtà è schedulabile



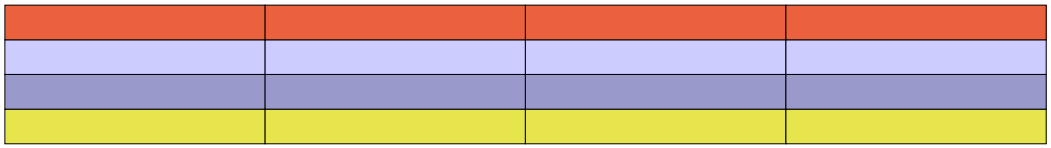
Esercizio 2 - Soluzione

	T	C	Priorità	C/T
Task 1	4	1	1 (min)	0.25
Task 2	4	1	2	0.25
Task 3	4	1	3	0.25
Task 4	4	1	4 (max)	0.25
				1.00

$\geq 4(2^{1/4}-1)=0.743$

Condizione Non OK

Ma in realtà è schedulabile



Esercizio 3

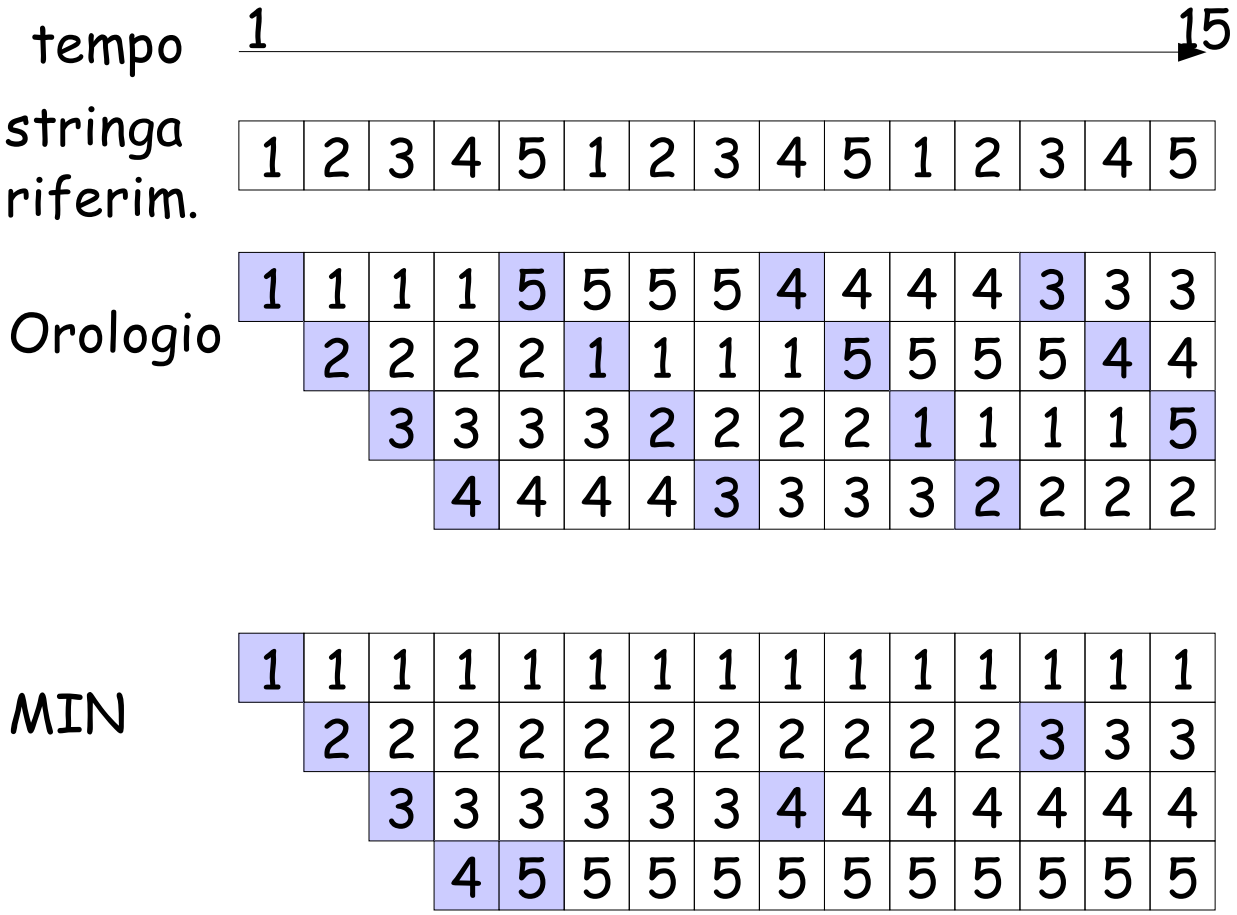
- Si consideri una memoria virtuale composta da 5 pagine, che devono essere allocate su una memoria fisica composta da 4 frame. Date queste condizioni, si mostrino tre casi distinti tali per cui:
- (A) l'algoritmo dell'orologio e MIN generino entrambi 7 page fault
  (o dimostrare che tale caso non esiste)
- (B) l'algoritmo dell'orologio abbia un numero di page fault maggiore del doppio di quelli generati da MIN
  (o dimostrare che tale caso non esiste)
- (C) l'algoritmo dell'orologio abbia un numero di page fault minore della metà di quelli generati da MIN
  (o dimostrare che tale caso non esiste)

Esercizio 3 - Parte C - Soluzione

- MIN è l'algoritmo ottimale. Questo significa che, data qualunque stringa di riferimenti, MIN produce il minor numero di page fault. Qualunque altro algoritmo può al limite uguagliare MIN.

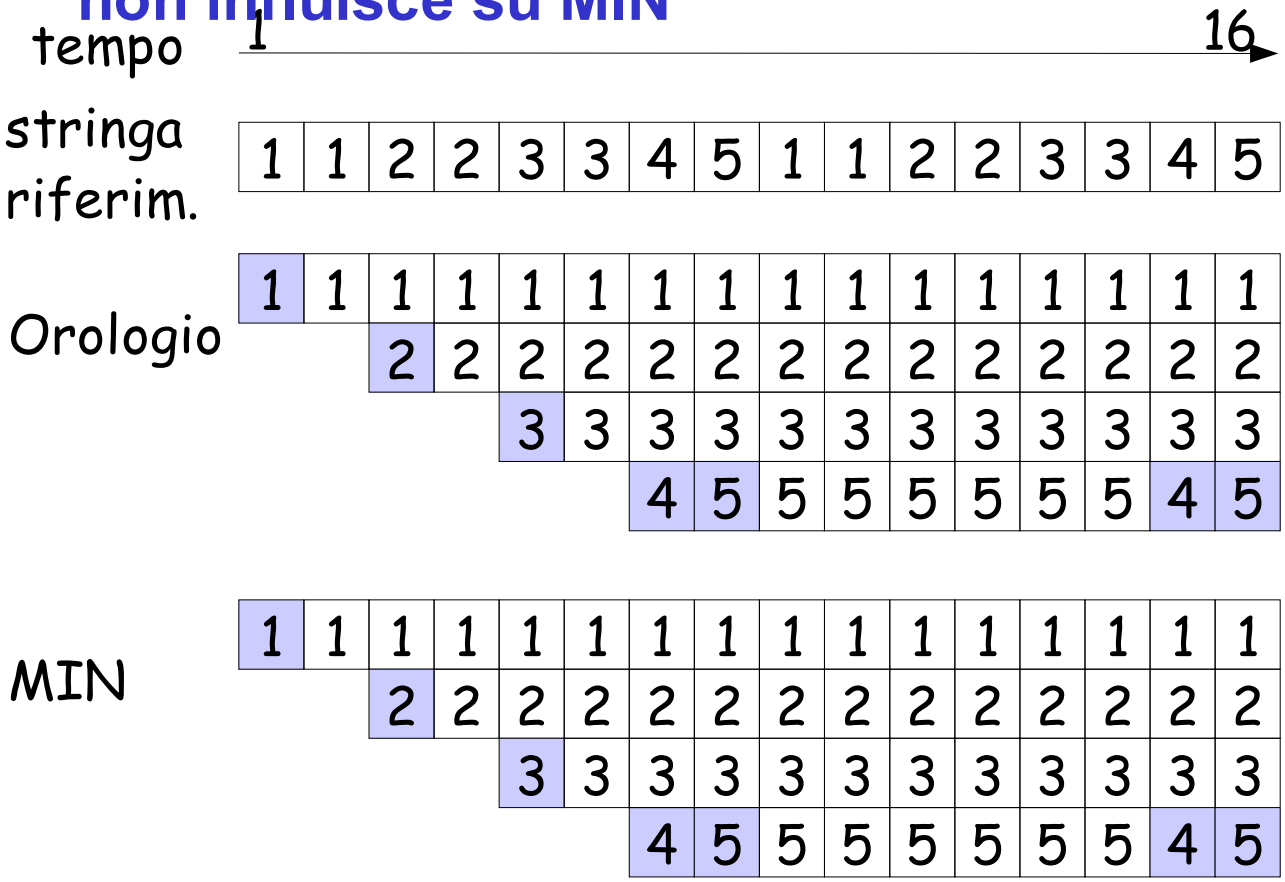
Esercizio 3 - Parte B - Soluzione

- **MIN è l'algoritmo ottimale. Questo significa che basta prendere una stringa che causi molti riferimenti sull'algoritmo dell'orologio**



Esercizio 3 - Parte A - Soluzione

- **Utilizziamo il doppio riferimento alle pagine per "forzare" la pagina da sostituire nell'orologio. Il doppio riferimento non influisce su MIN**



Esercizio 4

- Sia dato uno scheduler preemptive di tipo "Shortest-Remaining-Time-First" (ovvero uno scheduler che approssima SJF) con valore alfa=1/2 e valore iniziale T_0=0 per tutti i processi. Sia data la storia esecutiva dei seguenti processi:

P1: 2ms CPU, 5 ms I/O un.2, 2ms CPU, 5 ms I/O un 1, 3 ms CPU

P2: 4ms CPU, 5 ms I/O un.1, 4ms CPU, 5 ms I/O un 2, 3 ms CPU

P3: 5ms CPU, 5 ms I/O un.1, 6ms CPU, 5 ms I/O un 2, 3 ms CPU

P4: 3ms CPU, 5 ms I/O un.2, 3ms CPU, 5 ms I/O un 1, 4 ms CPU

mostrare il diagramma di Gannt e calcolare il tempo totale di esecuzione.