

# Memoria virtuale

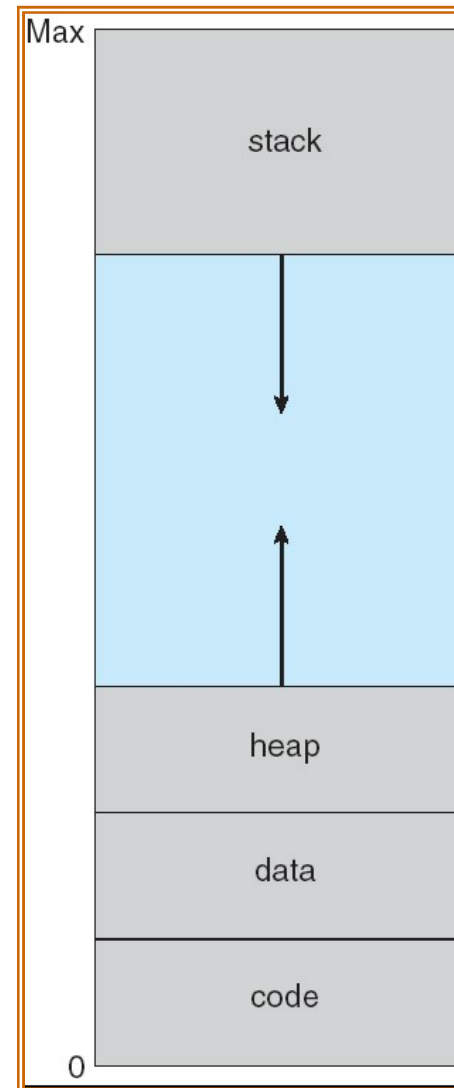
Capitolo 9 - Silberschatz

# Processo in memoria

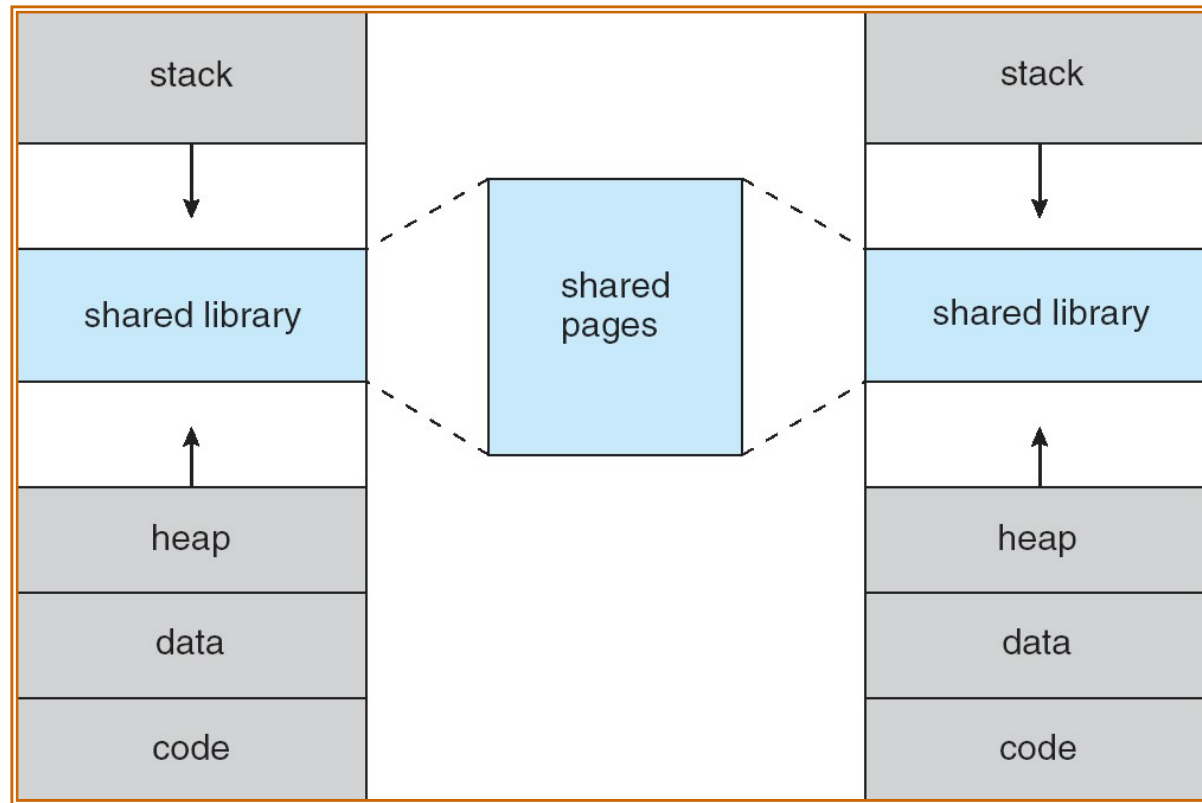
- Istruzioni in memoria prima di essere eseguite. Condizione necessaria e ragionevole ... ma le dimensioni del programma devono essere correlate a quelle della memoria
- Spesso, i programmi contengono:
  - Codice per condizioni d'errore eseguite raramente
  - Tabelle e array sovradimensionati
  - Opzioni e caratteristiche utili raramente
- Soluzione: caricamento dinamico
  - si carica una procedura, tenuta in memoria secondaria, solo quando è richiamata durante l'esecuzione

# Spazio di indirizzamento virtuale

- Lo spazio tra l'heap e lo stack è **spazio di indirizzamento virtuale** del processo, ma richiede pagine fisiche realmente esistenti solo nel caso che l'heap o lo stack crescano



# Libreria condivisa usando la memoria virtuale

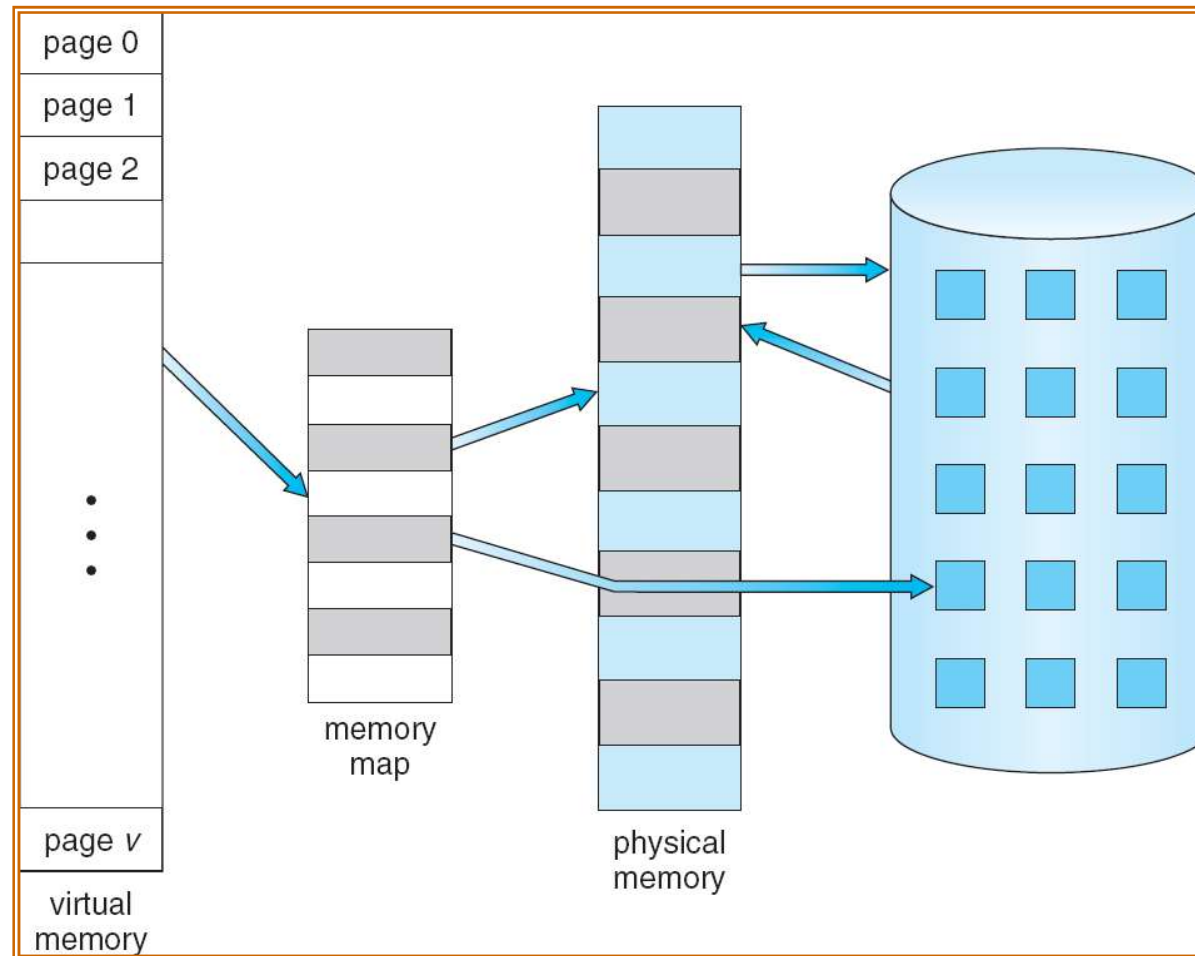


Le librerie sono condivisibili da parte di più processi associando l'oggetto allo spazio degli indirizzi virtuali; ciascun processo vede le librerie come se facessero parte del proprio spazio degli indirizzi

# Memoria virtuale

- Completa la separazione della memoria logica dalla memoria fisica.
- Solo **una parte** del programma necessita di essere in memoria per l'esecuzione.
- Lo spazio di indirizzamento logico può quindi essere **più grande** dello spazio di indirizzamento fisico.
- Gli spazi di indirizzamento virtuali possono essere **condivisi** da diversi processi.
- Maggiore **efficienza** nella creazione dei processi.

# Memoria virtuale più grande della memoria fisica



# Memoria virtuale

- La memoria virtuale può essere implementata attraverso:
  - Paginazione su richiesta (demand paging).
  - Segmentazione su richiesta (demand segmentation).

# Paginazione su richiesta

- Introdurre una pagina in memoria solo se necessario
  - Meno I/O
  - È necessaria meno memoria.
  - Più utenti.
- La pagina è necessaria → riferimento ad essa:
  - riferimento non valido → termine del processo.
  - non in memoria → portare in memoria.



# Bit di validità

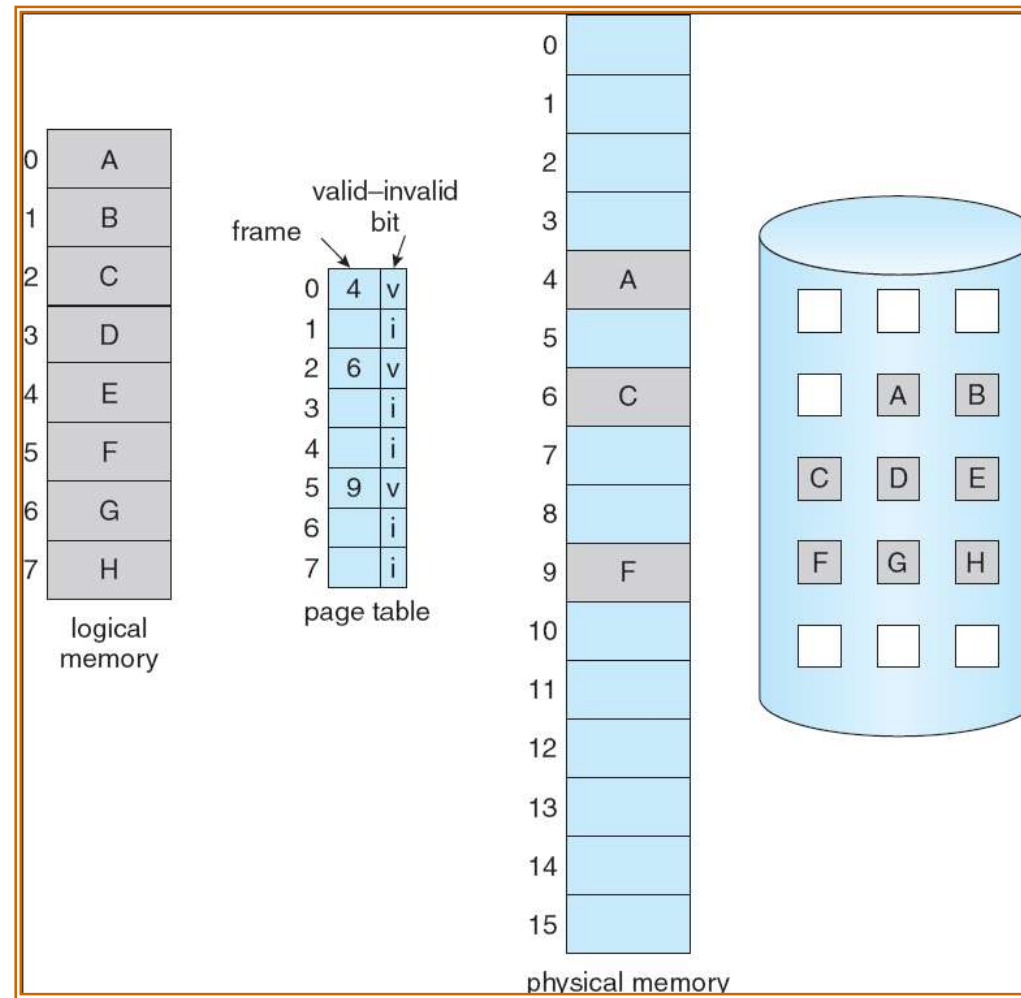
- Ad ogni elemento della tabella delle pagine è associato un bit  
(**v**  $\Rightarrow$  in-memoria, **i**  $\Rightarrow$  non-in-memoria)
- Inizialmente il bit valido-non valido è impostato a **i** per tutti gli elementi
- Esempio di un'istantanea della tabella delle pagine:

| Frame # | bit valido-non valido |
|---------|-----------------------|
| 4       | <b>v</b>              |
|         | <b>i</b>              |
|         | <b>i</b>              |
|         | <b>i</b>              |
|         | <b>i</b>              |
| ⋮       |                       |
|         | <b>i</b>              |
|         | <b>i</b>              |

page table

- Durante la traduzione dell'indirizzo, se il bit è **i** nell'elemento della tabella delle pagine  $\rightarrow$  mancanza di pagina **i**  $\Rightarrow$  fault.

# Memoria e disco



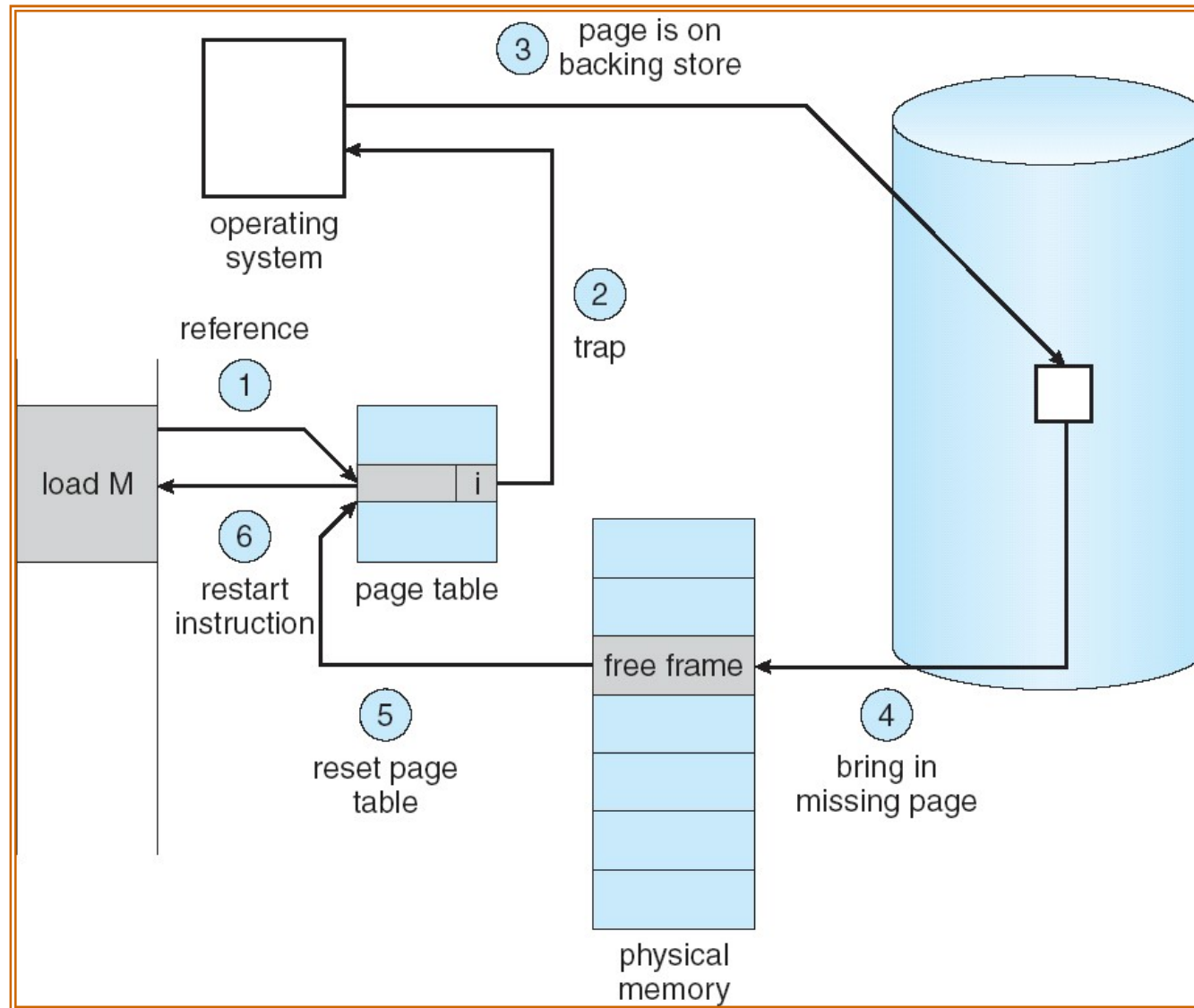
# Mancanza di pagina

- Quando avviene un riferimento ad una pagina non presente in memoria (cioè una pagina contrassegnata come non valida), si provoca una trap al sistema operativo:

## page fault

- Il sistema op. esamina una tabella interna per decidere:
  - Riferimento non valido (non ci si riferisce alla mem del processo) → termine del processo.
  - Riferimento valido. Pagina non in memoria.
- Cercare un frame libero.
- Spostare la pagina nel frame (marcando come occupato il frame utilizzato).
- Modificare la tabella della pagine, validità del bit = **v**.
- Riavviare l'istruzione che ha causato il page fault

# Passi per gestire un Page Fault



# Istruzioni parzialmente eseguite

- Riavvia l'istruzione
  - Add (a,b,c)
    - Se eseguendo tale istruzione manca la pagina dove collocare il risultato c, si carica la pagina dove deve essere conservato c e poi viene rieseguita tutta l'istruzione (caricamento di a, con accesso alla sua pagina - caricamento di b, con accesso alla sua pagina - somma – memorizzazione in c)
- MVC (move character)
  - Muove una sequenza di byte da una locazione ad un'altra. Se durante tale movimento si scopre che una delle 2 sequenze esce dal confine di una pagina, si carica la pagina e riparte l'istruzione

# Gestione di pagine fault: passi

1. Segnale di eccezione al SO
2. Salvataggio registri e stato
3. Verifica se interruzione dovuta a mancanza pagina
4. Determinazione locazione su disco
5. Lettura da disco – attese: coda/latenza/posizionamento
6. Durante l'attesa: allocazione CPU ad altro processo
7. Interrupt dal controller disco per I/O completato
8. Aggiornamento tabelle
9. Attesa CPU nuovamente allocata
10. Salvataggio registri e stato dell'altro processo (se è stato fatto il passo 6.)
11. Recupero registri utente, stato e ripresa

# Prestazione della paginazione su richiesta

- Probabilità di page-fault  $0 \leq p \leq 1$ 
  - se  $p = 0$  non ci sono mancanze di pagine
  - se  $p = 1$ , ogni riferimento è una mancanza di pagina
- Tempo di accesso effettivo (EAT)

$$\begin{aligned} \text{EAT} = & (1 - p) \times \text{accesso alla memoria} \\ & + p (\text{page fault overhead} \\ & \quad + \text{lettura della pagina} \\ & \quad + \text{overhead ripresa}) \end{aligned}$$

# Esempio di paginazione su richiesta

- Tempo di accesso alla memoria = 200 nanosecondi
- Tempo medio di gestione di un page fault = 8 millisecondi = 8,000,000 nanosecondi

$$\begin{aligned} \text{EAT} &= (1 - p) * (2 * 200) + p * 8,000,000 \\ &= (2 * 200) + p * 7,999,600 \end{aligned}$$

- Se un accesso su 1000 causa un page fault, allora

$$\text{EAT} = 8199,8 \text{ nanosecondi}$$

il tempo di accesso effettivo è 40 volte superiore  
al tempo di accesso alla memoria centrale!!

$10^3 \text{ nanosec} = 1 \text{ microsec}$     $10^6 \text{ nanosec} = 1 \text{ millisecc}$     $10^9 \text{ nanosec} = 1 \text{ sec}$

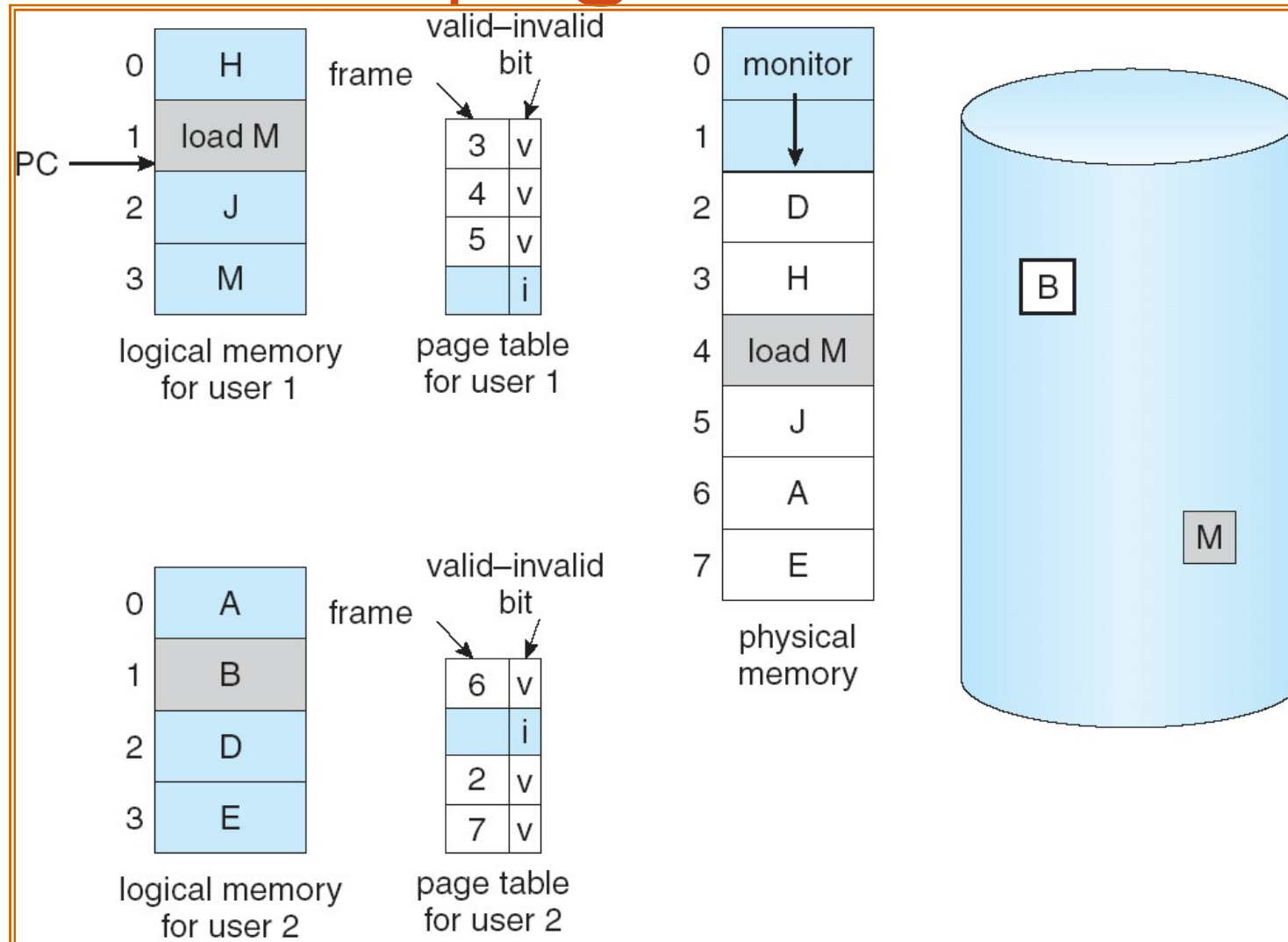


# Sovrallocazione

Cosa succede se non ci sono frame liberi?

- Sostituzione di pagina – trovare alcune pagine in memoria, ma non veramente in uso, e spostarle sul disco.
- algoritmo di sostituzione per scegliere frame vittima
- prestazione – l'algoritmo deve minimizzare il numero di page fault
- La stessa pagina può essere portata in memoria più volte.

# Necessità di sostituzione pagina

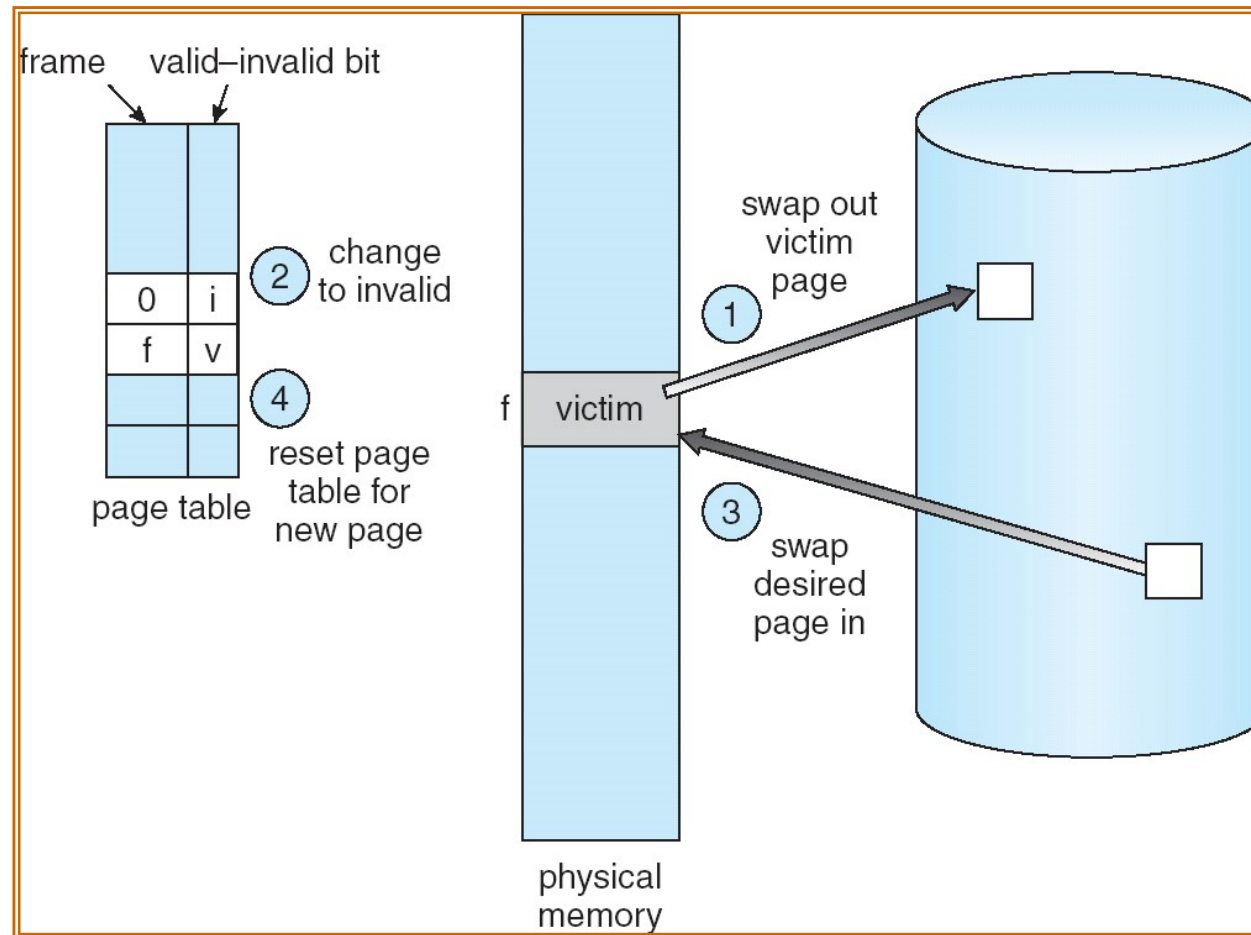


# Gestione del page fault con sostituzione di pagina

Modifica della procedura di gestione di page fault per includere la sostituzione di pagina.

1. Individuare la posizione della pagina desiderata sul disco.
2. Trovare un frame libero:
  - se c'è un frame libero, usarlo;
  - se non c'è nessun frame libero, usare l'algoritmo di sostituzione delle pagine per selezionare un frame vittima
3. Caricare la pagina desiderata nel frame libero; aggiornare le tabelle dei frame e delle pagine.
4. Riprendere il processo

# Sostituzione della pagina



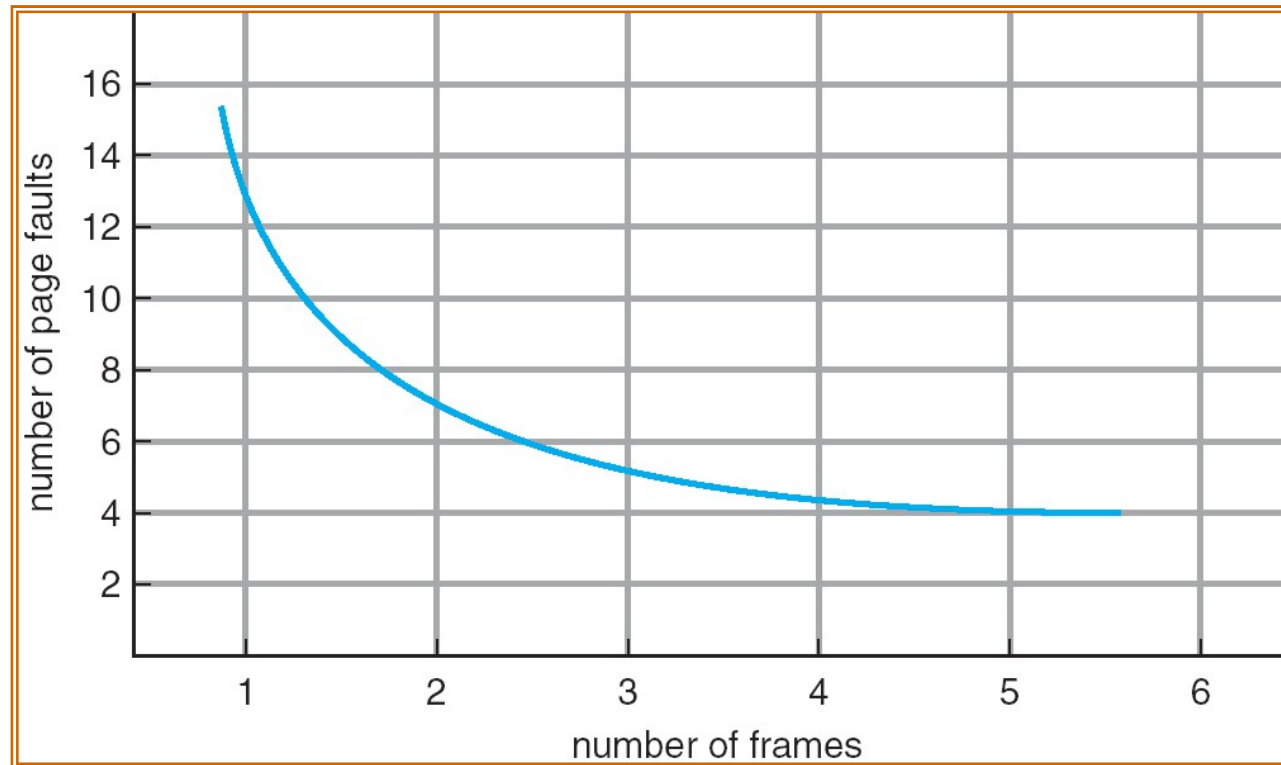
# Sostituzione di pagina

- Se non esiste alcun frame libero c'è bisogno di **due** trasferimenti di pagine
- Soluzione: *bit di modifica* (modify bit) – solo le pagine modificate sono scritte su disco.
- Questa tecnica vale anche per le pagine a sola lettura.
- La sostituzione della pagina completa la separazione fra memoria logica e memoria fisica.

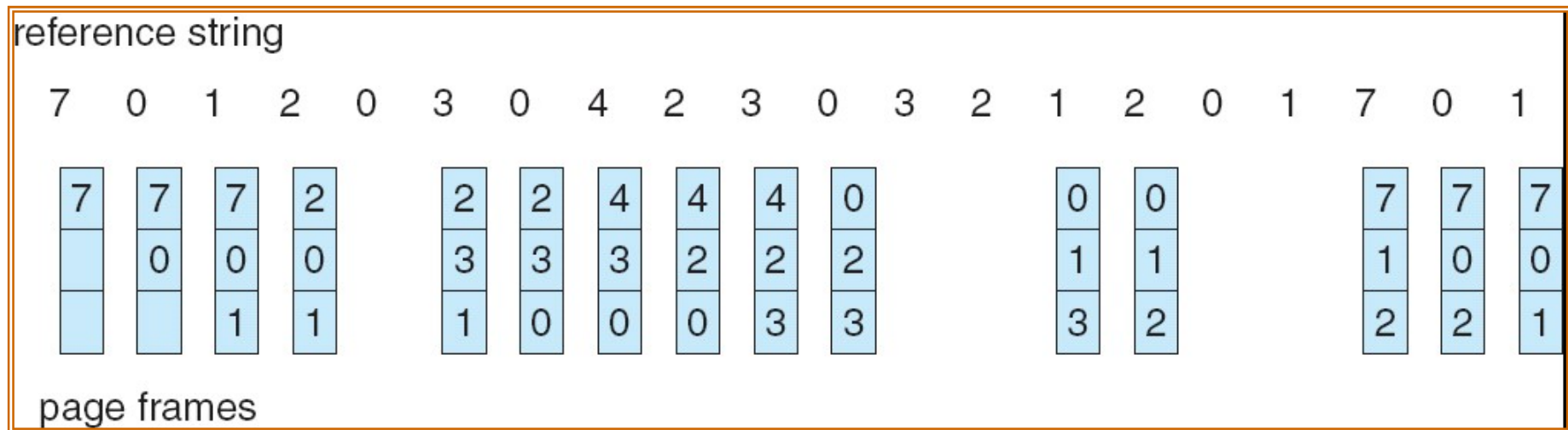
# Algoritmo di sostituzione delle pagine

- Si desidera il più basso tasso possibile di page fault
- Un algoritmo viene valutato facendolo operare su una particolare stringa di riferimenti alla memoria e calcolando il numero di page fault
- La stringa di riferimanto che useremo è:  
**7, 0, 1, 2, 0, 3, 0, 4, 2, 3, 0, 3, 2, 1, 2, 0, 1, 7, 0, 1.**
- Numero dei frame disponibili è essenziale per la valutazione

# Mancanza di pagina e numero di frame



# Sostituzione First-In-First-Out (FIFO) della pagina





# Algoritmo FIFO

- Stringa di riferimenti: 1, 2, 3, 4, 1, 2, 5, 1, 2, 3, 4, 5
- 3 frames (3 pagine possono essere in memoria per ogni processo)

|  |   |   |   |   |   |   |   |   |   |   |   |
|--|---|---|---|---|---|---|---|---|---|---|---|
|  | 1 | 1 | 1 | 4 | 4 | 4 | 5 | 5 | 5 | 5 | 5 |
|  |   | 2 | 2 | 2 | 1 | 1 | 1 | 1 | 1 | 3 | 3 |
|  |   |   | 3 | 3 | 3 | 2 | 2 | 2 | 2 | 2 | 4 |

9 page faults

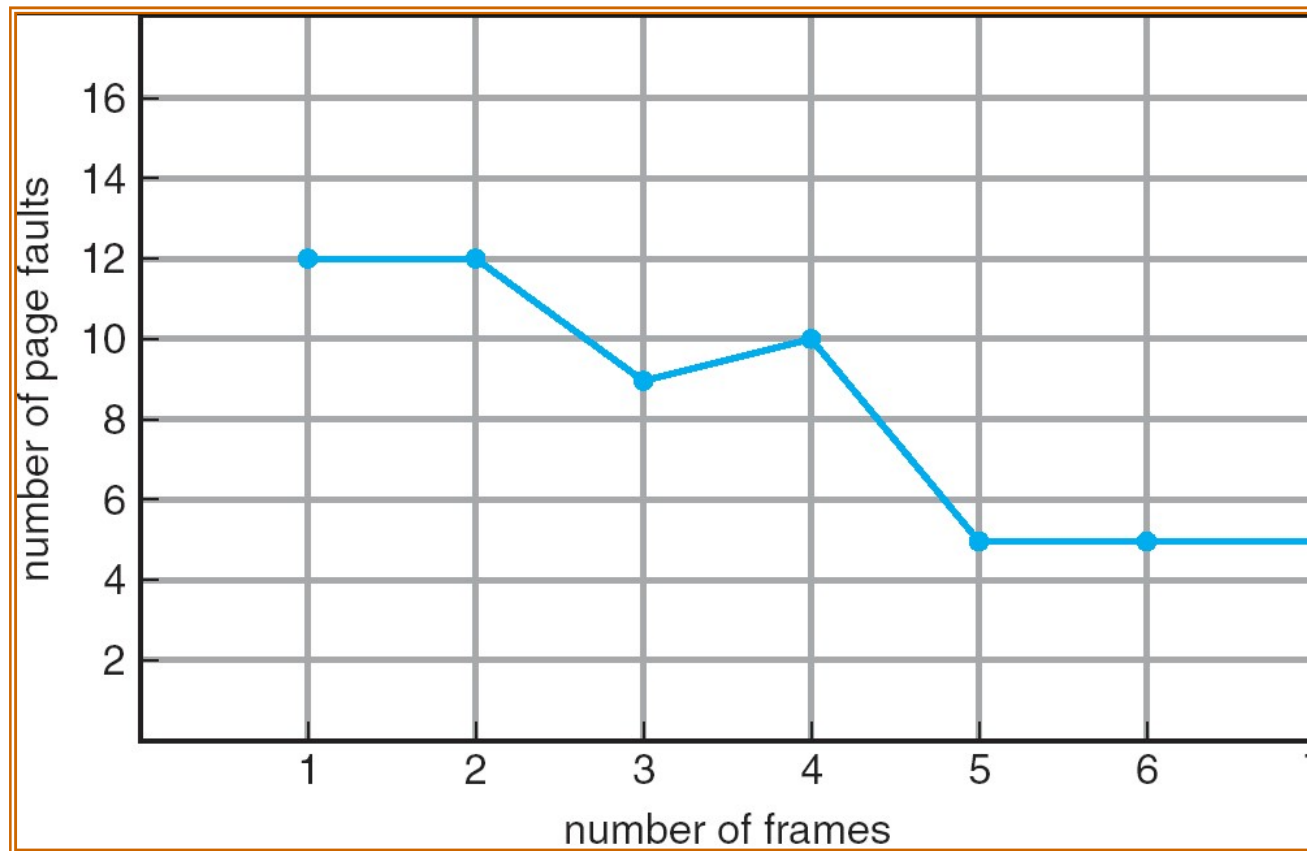
- 4 frames

|  |   |   |   |   |   |   |   |   |   |   |   |
|--|---|---|---|---|---|---|---|---|---|---|---|
|  | 1 | 1 | 1 | 1 | 1 | 5 | 5 | 5 | 5 | 4 | 4 |
|  |   | 2 | 2 | 2 | 2 | 2 | 2 | 1 | 1 | 1 | 5 |
|  |   |   | 3 | 3 | 3 | 3 | 3 | 3 | 2 | 2 | 2 |
|  |   |   |   | 4 | 4 | 4 | 4 | 4 | 4 | 3 | 3 |

10 page faults

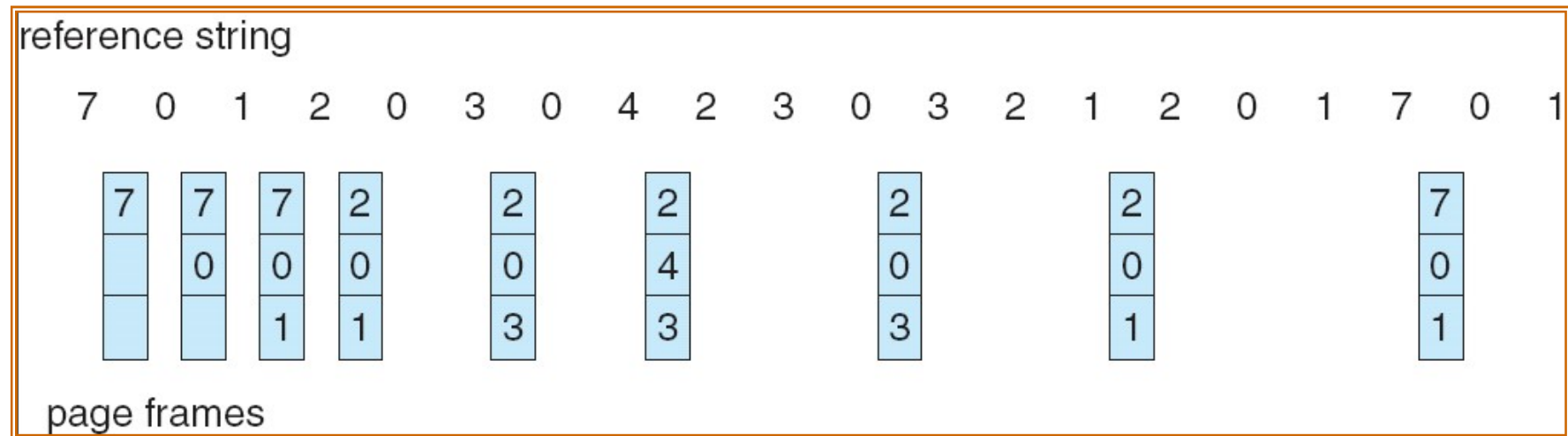
Sostituzione FIFO -  
*Belady's Anomaly*: più  
 frames  $\Rightarrow$   
 più page faults

# FIFO: l'anomalia di Belady



# Sostituzione ottimale delle pagine

Sostituisce la pagina che **non si userà** per il periodo di tempo più lungo.



# Algoritmo ottimale

- Riprendere la pagina che non sarà usata per un lungo periodo di tempo
- Esempio: 4 frames

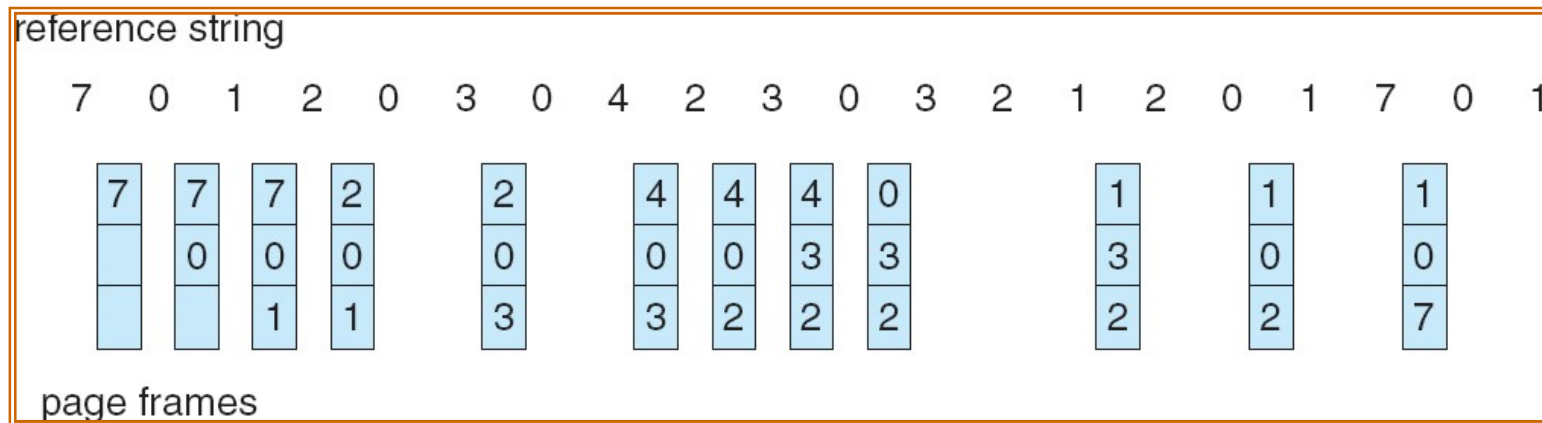
|  |  | 1 | 2 | 3 | 4 | 1 | 2 | 5 | 1 | 2 | 3 | 4 | 5 |
|--|--|---|---|---|---|---|---|---|---|---|---|---|---|
|  |  | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 4 | 4 |   |
|  |  |   | 2 | 2 | 2 | 2 | 2 | 2 | 2 | 2 | 2 | 2 |   |
|  |  |   |   | 3 | 3 | 3 | 3 | 3 | 3 | 3 | 3 | 3 |   |
|  |  |   |   |   | 4 | 4 | 4 | 5 | 5 | 5 | 5 | 5 |   |

6 page faults

- Questo algoritmo richiede la conoscenza della sequenza di riferimenti futuri. Come avere questa informazione?
- Usato per misurare quanto sono buone le prestazioni di un algoritmo

# Algoritmo LRU (Last Recently Used)

Sostituisce la pagina che **non è stata usata** per il periodo di tempo più lungo.



# Algoritmo Least Recently Used (LRU)

- Esempio: 4 frames

1, 2, 3, 4, 1, 2, **5**, 1, 2, **3**, **4**, **5**

|   |          |          |          |          |
|---|----------|----------|----------|----------|
| 1 | 1        | 1        | 1        | <b>5</b> |
| 2 | 2        | 2        | 2        | 2        |
| 3 | <b>5</b> | 5        | <b>4</b> | 4        |
| 4 | 4        | <b>3</b> | 3        | 3        |

# Implementare LRU

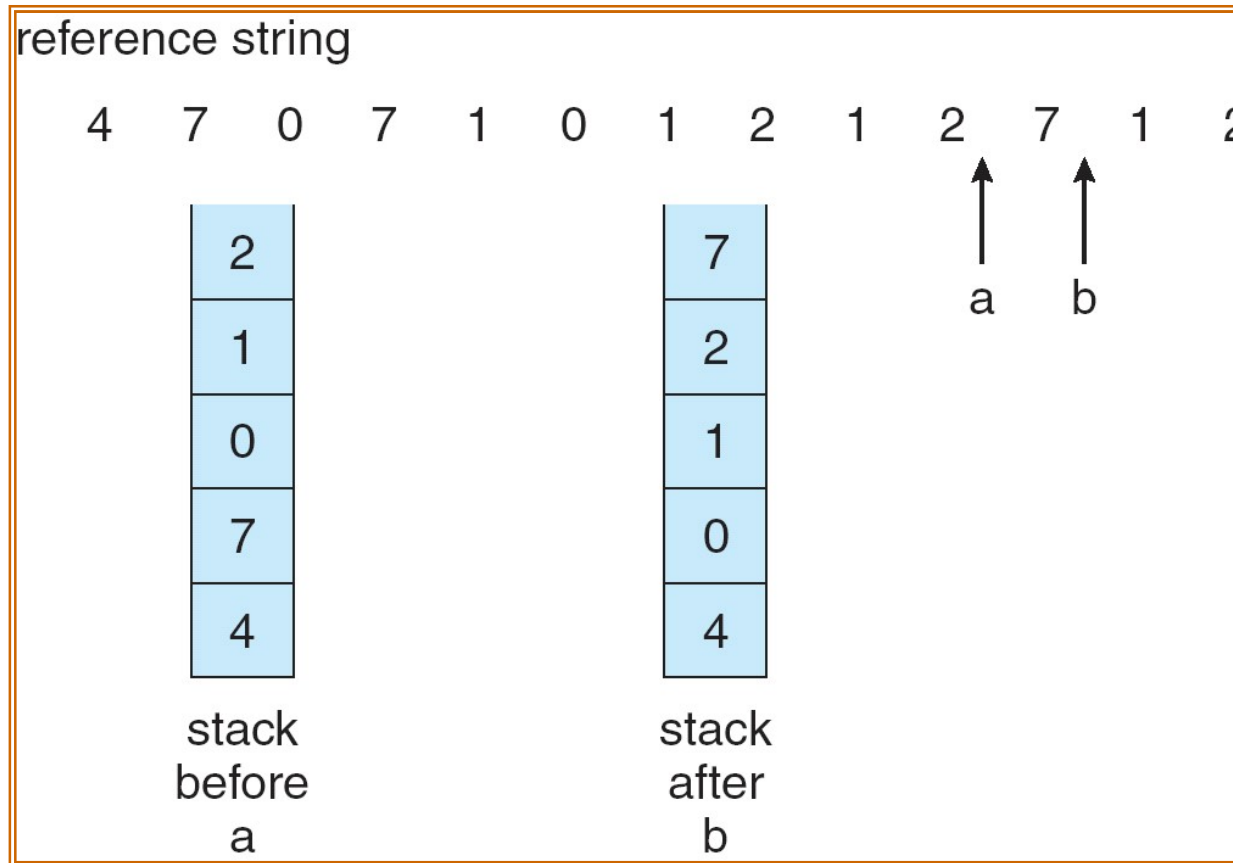
- Problema: determinare un ordine per i frame secondo il momento dell'ultimo uso fatto
- Implementazione del **contatore**.
  - Si assegna alla CPU un contatore che si incrementa ad ogni riferimento alla memoria.
  - Si assegna ad ogni entry della tabella delle pagine un ulteriore campo.
  - Ogni volta che si fa riferimento ad una pagina viene copiato il valore del contatore della CPU nel campo corrispondente per quella pagina nella tabella delle pagine.
  - Quando una pagina deve essere sostituita si guardano i valori dei contatori.
    - Si sceglie la **pagina con il valore del contatore più basso**

# Implementare LRU

- Implementazione dello **stack**:
  - mantenere uno stack dei numeri di pagina in una lista a doppio collegamento
  - Riferimento ad una pagina:
    - se è una pagina non presente nei frame la si mette in cima allo stack;
    - ogni volta che si fa riferimento ad una pagina già presente nello stack bisogna prelevare tale riferimento dalla sua posizione e metterla in cima
    - nessuna ricerca per la sostituzione: basta **prelevare la pagina dal fondo**;
  - richiede di cambiare al più 6 puntatori.



# Uso di uno stack per registrare i riferimenti alle pagine usate più di recente



# Approssimazione dell' algoritmo LRU

- Sia l'algoritmo ottimale che quello LRU non sono soggetti all'anomalia di Belady.
- Ma sono pochi i sistemi in grado di avere una architettura(contatori, stack) per la gestione di LRU.
  - Molti sistemi tentano delle approssimazioni.
- Architettura: bit di riferimento (da aggiungere all'entry della tabella delle pagine).
  - Ad ogni pagina è associato un bit, inizialmente = 0.
  - Quando la pagina è referenziata il bit è impostato a 1.
  - Rimpiazzare la pagina che è a 0 (se ne esiste una).

# Algoritmo della seconda chance

Miglioramento FIFO.

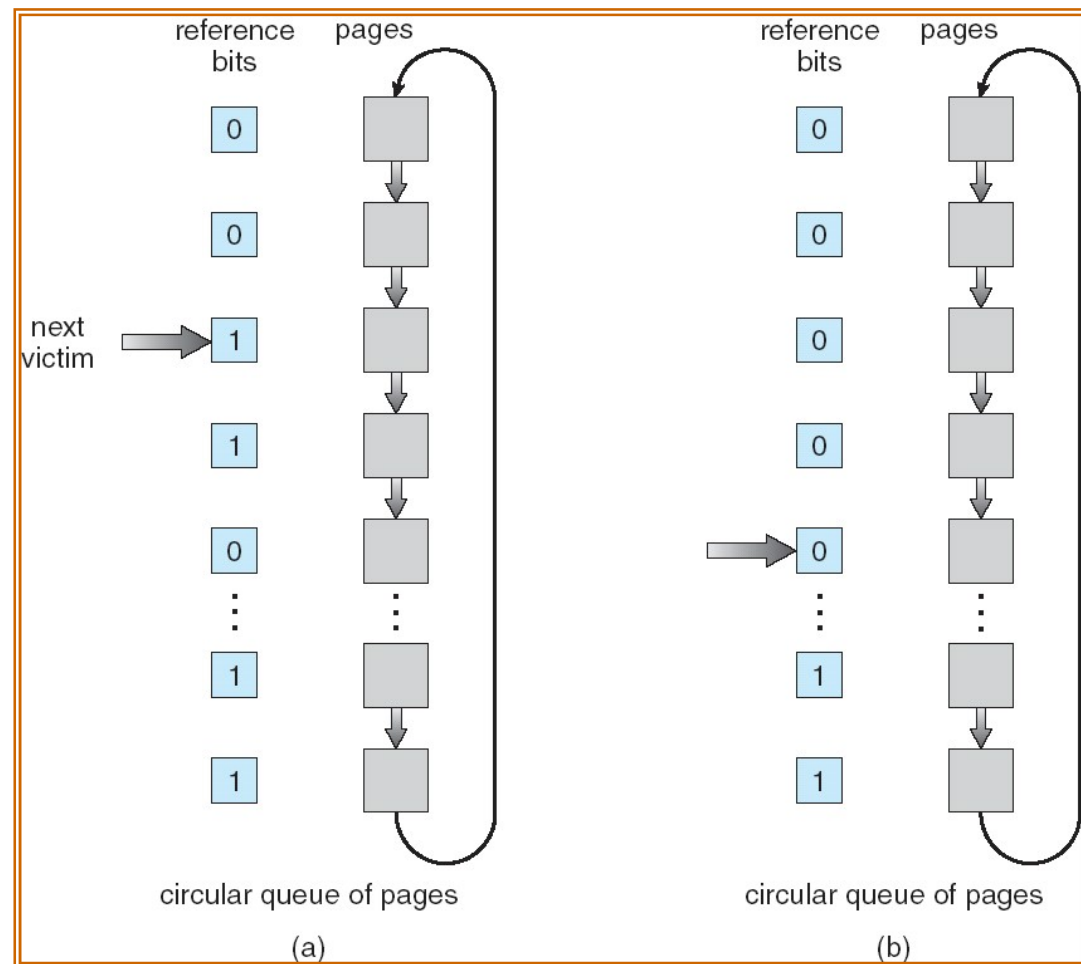
- Le pagine sono disposte in una lista circolare
- Quando occorre selezionare una pagina vittima inizia la scansione della lista:
  - Si sceglie la pagina presente in memoria centrale da più tempo, come in FIFO.
  - Si controlla il bit di riferimento di tale pagina.
  - se una pagina ha il bit di riferimento a 1 lo si pone a 0 e si passa alla successiva (la pagina rimane in memoria – le si dà una seconda chance);
  - altrimenti si seleziona per essere sostituita.

# Algoritmo della seconda chance

- Se ad una pagina è data una “seconda chance”, essa non sarà sostituita:
  - fino a quando tutte le altre pagine non siano state sostituite, oppure
  - fino a quando tutte le altre pagine non abbiano avuto la loro seconda chance.
- Se una pagina è acceduta ciclicamente con frequenza sufficientemente elevata, ottiene sempre una seconda chance e non viene mai sostituita.

# Algoritmo della Seconda Chance

- Un puntatore indica quale è la testa della fifo, cioè quale è la prossima pagina candidata ad essere una vittima.
- Nel caso peggiore tutti i bit di riferimento sono a 1 e quindi dopo aver scorso la lista e messo tutti i bit di riferimento a 0, la vittima è proprio la prima pagina esaminata; si tratta cioè di un normale fifo



# Altre approssimazioni

- Raffinamento dell'algoritmo della seconda chance  
(bit riferimento, bit modifica)
  - (0,0) non usato di recente, non modificato
  - (0,1) non usato di recente, modificato
  - (1,0) usato di recente, non modificato
  - (1,1) usato di recente, modificato
    - Si sceglie la pagina con la coppia di più piccolo valore
- Più bit di riferimento (e.g., tenuti in un registro a 8 bit). Periodicamente, allo scadere di un intervallo temporale, il registro viene shiftato a destra di un bit, e il bit di riferimento della pagina viene copiato nel bit più significativo del registro.
  - La pagina con il più piccolo intero viene scelta

# Algoritmi di conteggio

- Tenere un contatore del numero di riferimenti che sono stati fatti ad ogni pagina.
- **Algoritmo LFU (Least Frequently Used)**: sostituisce la pagina con il più basso conteggio.
- **Algoritmo MFU (Most Frequently Used)**: sostituisce la pagina con il conteggio più alto.
  - è basato sulla assunzione che la pagina col valore più basso è stata spostata recentemente in memoria e quindi deve ancora essere impiegata

# Bufferizzazione delle pagine

- Pool di frame liberi per soddisfare le richieste velocemente.
- Quando si sceglie una pagina vittima, la si trasferisce nel pool dei frame liberi.
  - Il processo può essere riattivato rapidamente, senza attendere la fine dell'operazione di salvataggio del frame vittima sulla memoria di massa
- Pagine modificate del pool dei frame liberi sono scritte sul disco periodicamente in background.
- Ricerca nel pool dei frame liberi in memoria di una pagina precedentemente sostituita e nuovamente necessaria. Probabilmente ancora in memoria e non sovrascritta se il frame non è stato riallocato.
  - Non è necessaria nessuna operazione di I/O



# Allocazione dei frame

- Ogni processo ha bisogno di un numero **minimo** di pagine.
- Esempio: IBM 370 – 6 pagine per gestire l'istruzione MVC:
  - L'istruzione richiede 6 byte, che possono estendersi su 2 pagine.
  - 2 pagine per gestire il “from” dell'istruzione.
  - 2 pagine per gestire il “to” dell'istruzione.
- Due principali schemi di allocazione.
  - Allocazione fissa.
  - Allocazione a priorità.

# Allocazione fissa

- **Allocazione uniforme** – avendo  $m$  frame ed  $n$  processi, si assegnano  $m/n$  frame a ciascun processo (Es: se 100 frame e 5 processi, ognuno prende 20 frame).
- **Allocazione proporzionale** – si assegna la memoria disponibile ad ogni processo in base alle dimensioni di quest'ultimo.

$s_i$  = size del processo  $p_i$

$$S = \sum s_i$$

$m$  = numero totale di frames

$$a_i = \text{numero di frame allocati per } p_i = \frac{s_i}{S} \times m$$

Esempio

$$m = 64$$

$$s_1 = 10$$

$$s_2 = 127$$

$$a_1 = \frac{10}{137} \times 64 \approx 5$$

$$a_2 = \frac{127}{137} \times 64 \approx 59$$

# Allocazione a priorità

- Usare uno schema di allocazione proporzionale basato sui valori delle priorità piuttosto che delle dimensioni.
- Combinazione dimensioni e priorità.

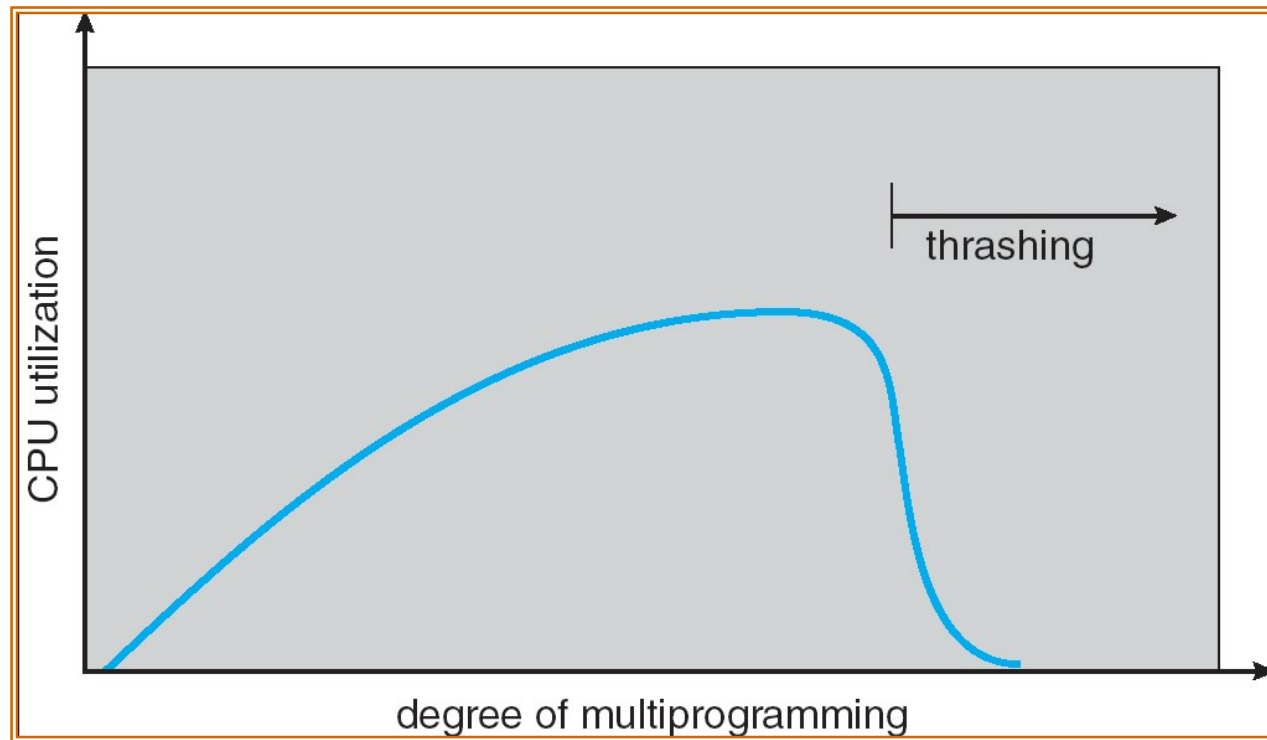
# Sostituzione globale e locale

- **Sostituzione locale** – vengono considerati solo i frame allocati al processo.
- **Sostituzione globale** – permette di selezionare un frame di sostituzione a partire dall'insieme di tutti i frame, anche se quel frame è correntemente allocato a qualche altro processo, i.e., un processo può prendere un frame da un altro processo.
  - Un processo non può controllare la propria frequenza di page fault
  - Il tempo di esecuzione di ciascun processo può variare in modo significativo

# Thrashing

- Se un processo non ha abbastanza pagine, il tasso di page fault aumenta. Questo comporta:
  - riduzione utilizzo della CPU;
  - il sistema operativo ritiene che sia necessario aumentare il livello di multiprogrammazione;
  - un altro processo aggiunto al sistema.
- **Thrashing**  $\equiv$  si spende più tempo nella paginazione che nella esecuzione dei processi.

# Thrashing



# Paginazione e Thrashing

- Per evitare il thrashing occorrerebbe sapere quali, oltre che quante pagine, servono.
- Modello di località:
  - Una località è un insieme di pagine che vengono accedute insieme - che sono contemporaneamente in uso attivo
  - il processo si muove da una località all'altra,
  - le località possono sovrapporsi.
- Perché si verifica il thrashing?

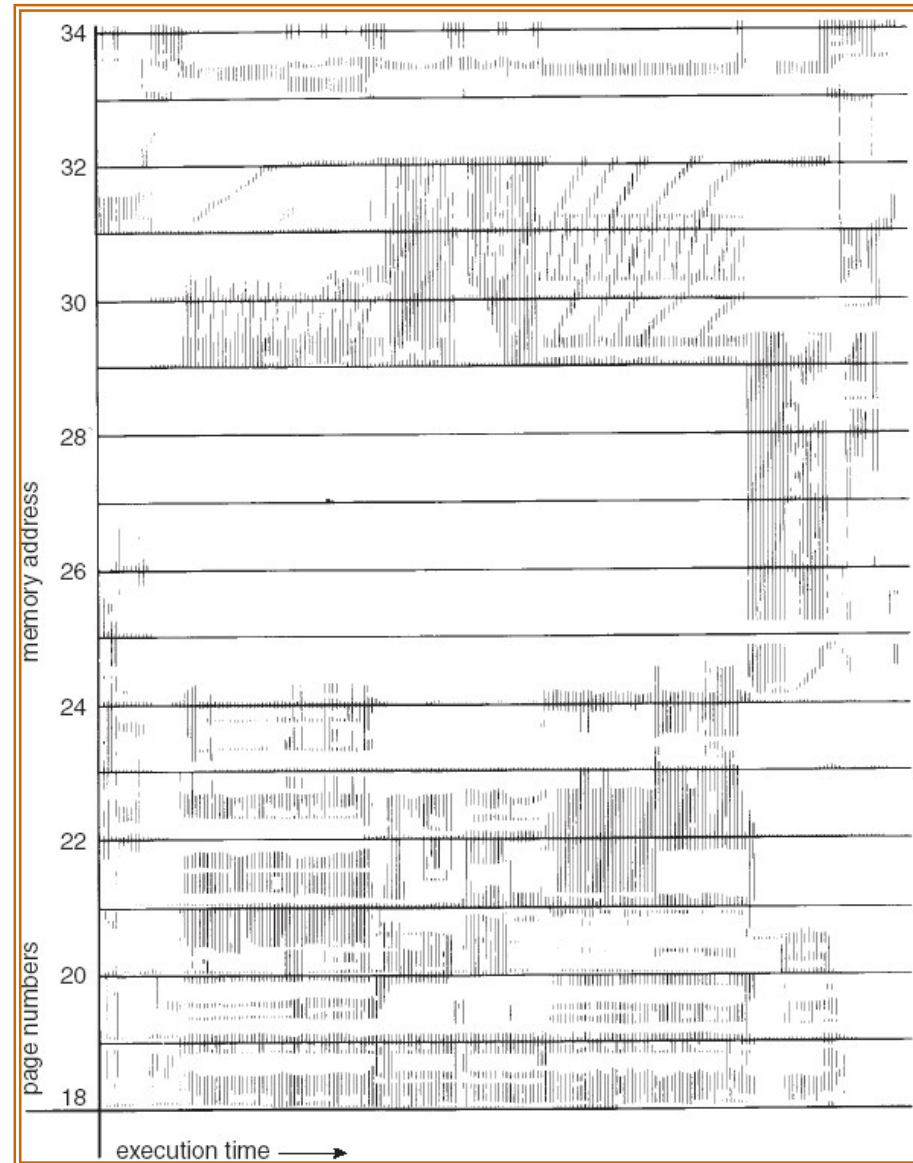
dimensione della località > numero di frame assegnati.

# Località in una sequenza di riferimenti alla memoria

## Esempio

- Quando viene invocato un sottoprogramma, si definisce una nuova località: vengono fatti riferimenti alle sue istruzioni, alle sue variabili locali ed a un sottoinsieme delle variabili globali
- Quando il sottoprogramma termina, il processo lascia la località corrispondente
- Le località sono definite dalla struttura del programma e dei dati

Si noti che il modello di località è il principio su cui si basa il caching





# Modello del working-set

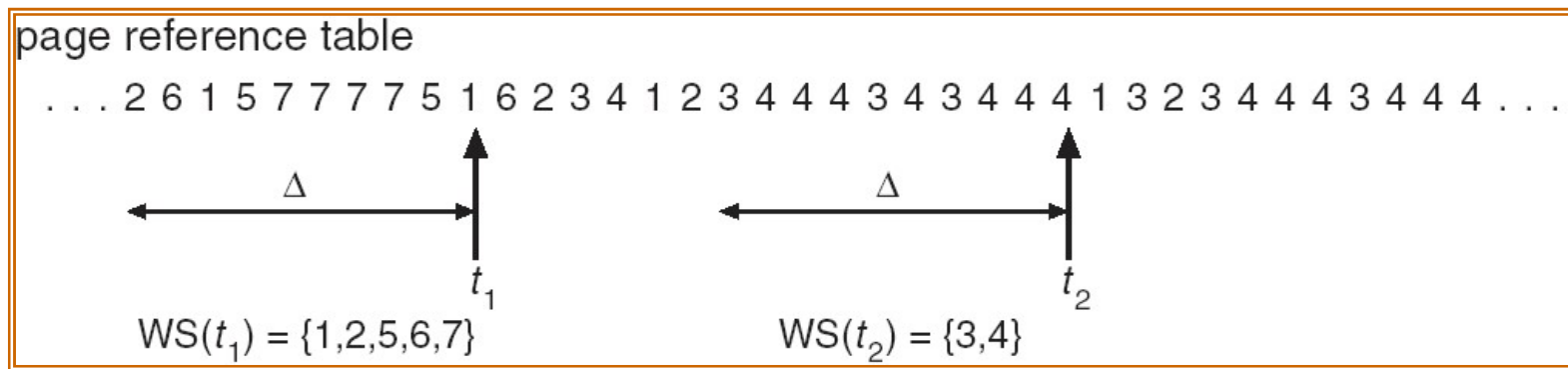
- $\Delta \equiv$  finestra di working-set  $\equiv$  un numero fisso di riferimenti di pagina  
Esempio: 10,000 istruzioni
- $WSS_i$  (working set del processo  $P_i$ ) = numero totale di pagine cui  $P_i$  si riferisce nel più recente  $\Delta$  (varia nel tempo)
  - se  $\Delta$  è troppo piccolo non comprenderà l'intera località
  - se  $\Delta$  è troppo grande può sovrapporre parecchie località
  - se  $\Delta = \infty \Rightarrow$  il working set è l'insieme delle pagine toccate durante l'esecuzione del processo

$D = \sum WSS_i \equiv$  richiesta globale dei frame

$m$  = numero di frame

- Se  $D > m \Rightarrow$  Thrashing
- Se  $D > m$ , allora occorre sospendere uno dei processi

# Working-set model



- Man mano che la finestra scorre può aumentare o diminuire il numero di frame dati al processo.
- Nell'esempio in  $t_1+1$  la pagina 2 scompare, quindi il frame ad essa allocato viene considerato libero (può eventualmente essere usato da un altro processo il cui working set invece cresce)

# Modello del working-set

Scelto  $\Delta$

- Il SO controlla il working set di ogni processo e gli assegna un numero di frame in base alle dimensioni del working set
  - Se, dopo aver sistemato tutti i processi, rimangono ancora frame liberi allora si può far iniziare un altro processo
  - Se la somma delle dimensioni dei working set supera il numero di frame disponibili allora il SO individua un processo da sospendere (portando le sue pagine su disco)
- Il problema è tener traccia delle pagine presenti nel working set di un processo man mano che la finestra scorre

# Mantenere traccia del Working Set

**Problema:** la finestra del working-set è “mobile”, con riferimenti che entrano ed escono dal working-set

- Si può approssimare il modello del working set con interrupt a intervalli fissi di tempo generati da un temporizzatore e un bit di riferimento

- **Esempio:**  $\Delta = 10,000$

- Interrupt ogni 5000 riferimenti
- Tenere in memoria 2 bit per ogni pagina
- Ogni volta che si riceve l'interrupt del temporizzatore si copiano i valori del bit di riferimento di ogni pagina e poi si azzerano
- Se uno dei bit è uguale a 1  $\Rightarrow$  la pagina è nel working set

- Perché non è del tutto preciso?

- se un bit è a 1 sappiamo che si è fatto riferimento a tale pagina ma non sappiamo se ciò è avvenuto pochi riferimenti prima o 5000 riferimenti prima

- Incremento = 10 bit e interrupt ogni 1000 riferimenti (il costo per servire tali segnali di interruzione più frequenti aumenta in modo corrispondente)

# Frequenza di Page-Fault

- Approccio più diretto e intuitivo rispetto all'utilizzo del working-set
- Stabilire un tasso “accettabile” di page-fault
  - Se il tasso attuale è troppo basso, il processo può avere troppi frame
  - Se il tasso attuale è troppo alto, il processo ha bisogno di più frame

