Memoria Virtuale

Memoria virtuale

- Introduzione
- Paginazione su richiesta
- Copiatura su scrittura
- Sostituzione delle pagine
- Allocazione di frame
- Paginazione degenere (*thrashing*)
- File mappati in memoria
- Allocazione di memoria del kernel
- Ulteriori considerazioni
- Esempi di sistemi operativi

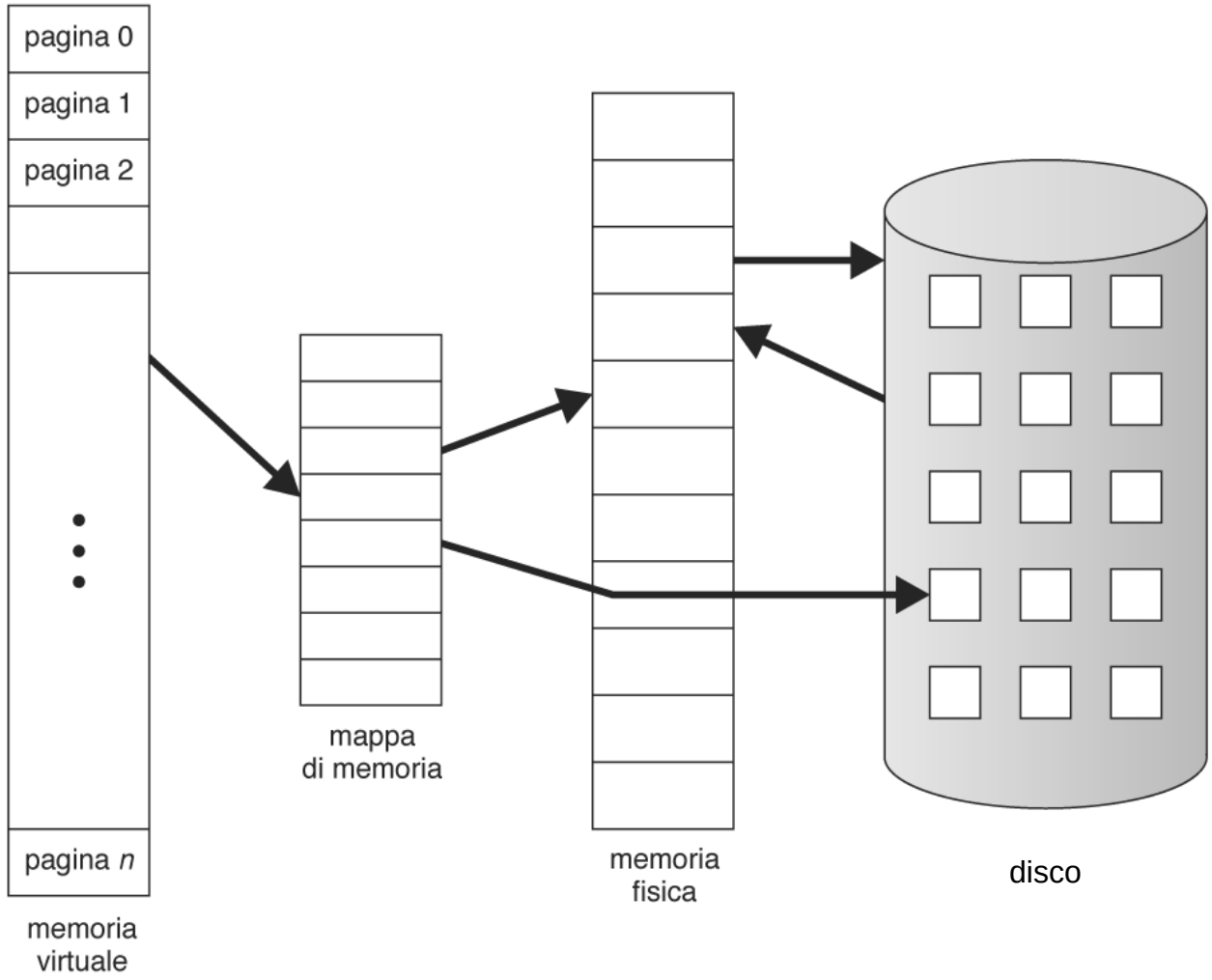
Memoria virtuale

- **Memoria Virtuale**: Tecnica che realizza la *separazione della memoria logica dalla memoria fisica*.

- **Solo una parte del programma è caricato in memoria per l'esecuzione.**
  - Lo spazio degli indirizzi logici è quindi più grande dello spazio degli indirizzi fisici.
  - Lo spazio degli indirizzi può essere diviso meglio tra più processi.
  - Permette una più efficiente creazione dei processi.

- La memoria virtuale può essere implementata tramite:
  - ***Paginazione su richiesta***
  - *Segmentazione su richiesta*

Memoria Virtuale maggiore della Memoria Fisica



Paginazione su richiesta

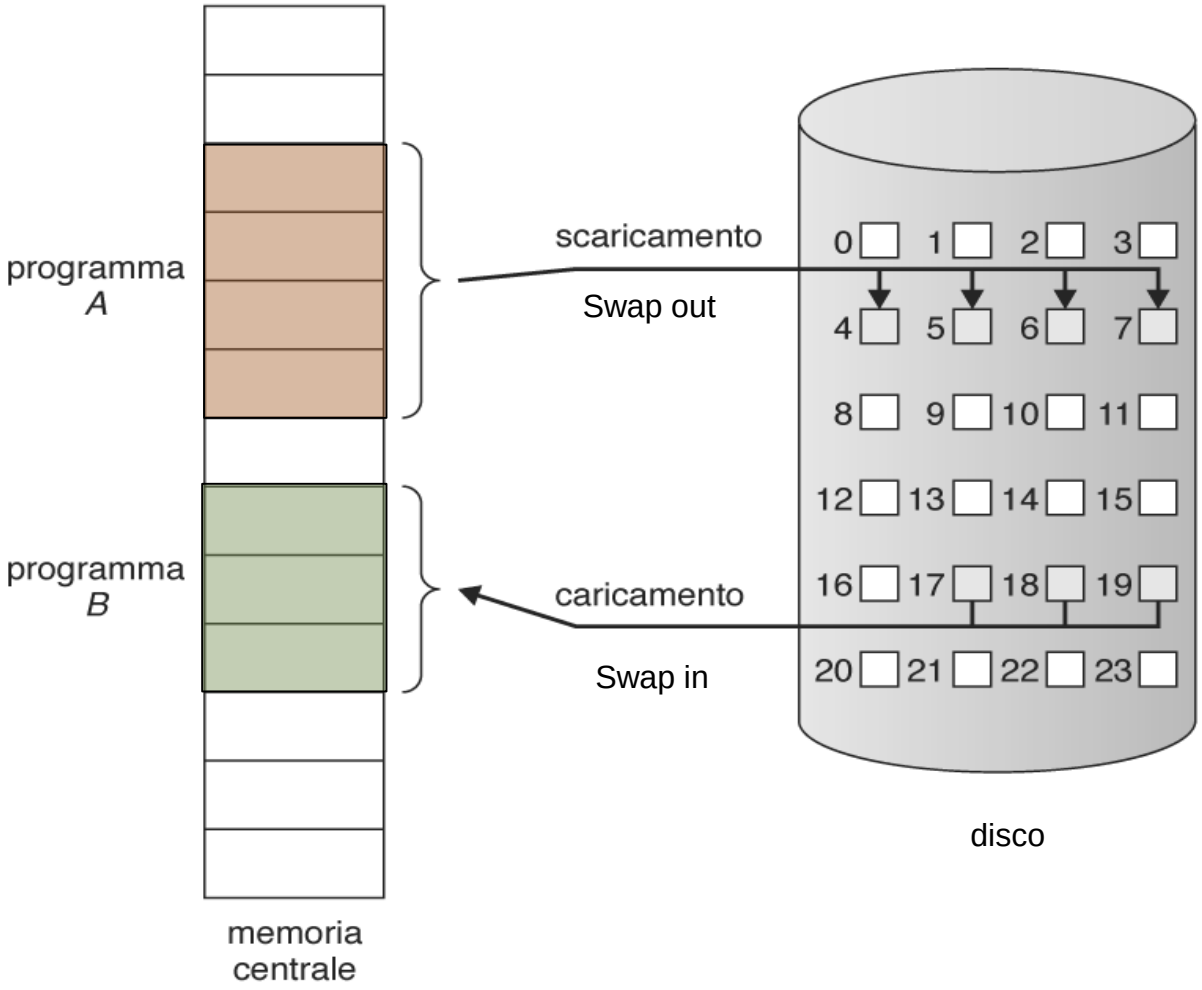
- **Nella paginazione su richiesta si porta una pagina in memoria solo quando serve**.
  - minore I/O
  - minore memoria necessaria
  - risposta più veloce
  - maggior numero di processi

- Quando serve una pagina $\Rightarrow$ riferimento ad essa
  - Se il riferimento è non valido $\Rightarrow$ ***abort***
  - Se il riferimento non è in memoria $\Rightarrow$ ***page fault***

    e trasferimento della pagina

    in memoria

Trasferimento su disco di una memoria paginata



Bit Valido/Non valido

- Ad ogni entry della tabella è associato un bit di validità (1 ⇒ in memoria, 0 ⇒ non in memoria ma su disco)
- Inizialmente il bit di validità è uguale a 0 per ogni entry.
- Esempio di tabella delle pagine:

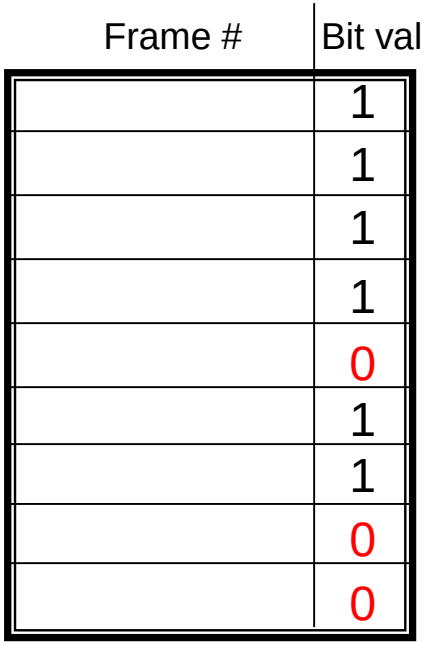
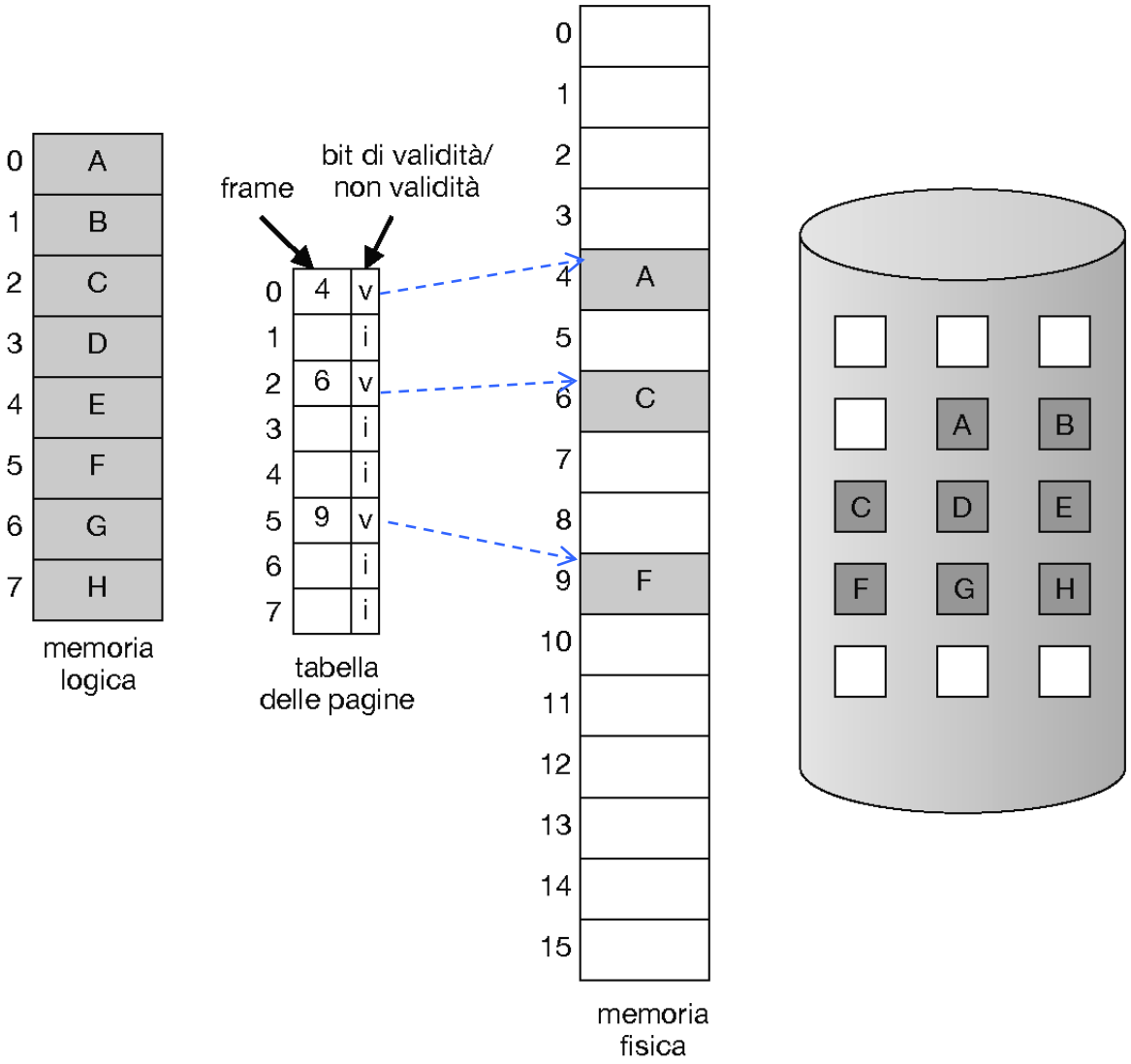


Tabella delle pagine

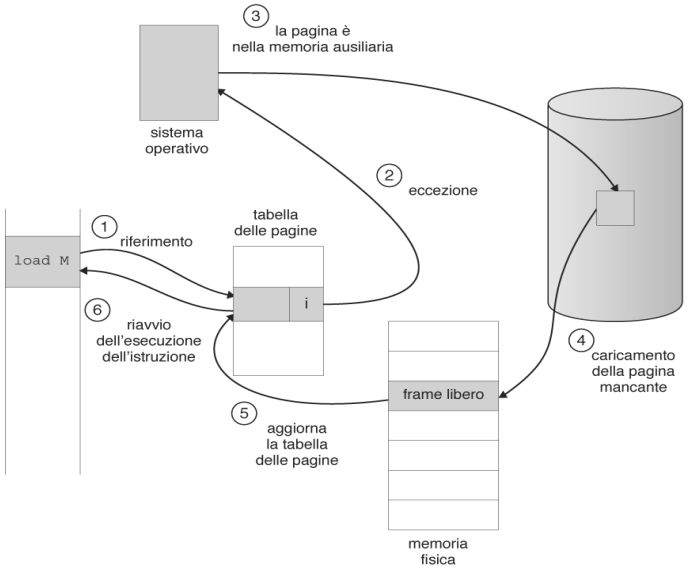
- Durante la traduzione degli indirizzi, se il bit è 0 ⇒ ***page fault*.**

Tabella delle pagine quando alcune pagine non si trovano nella memoria centrale

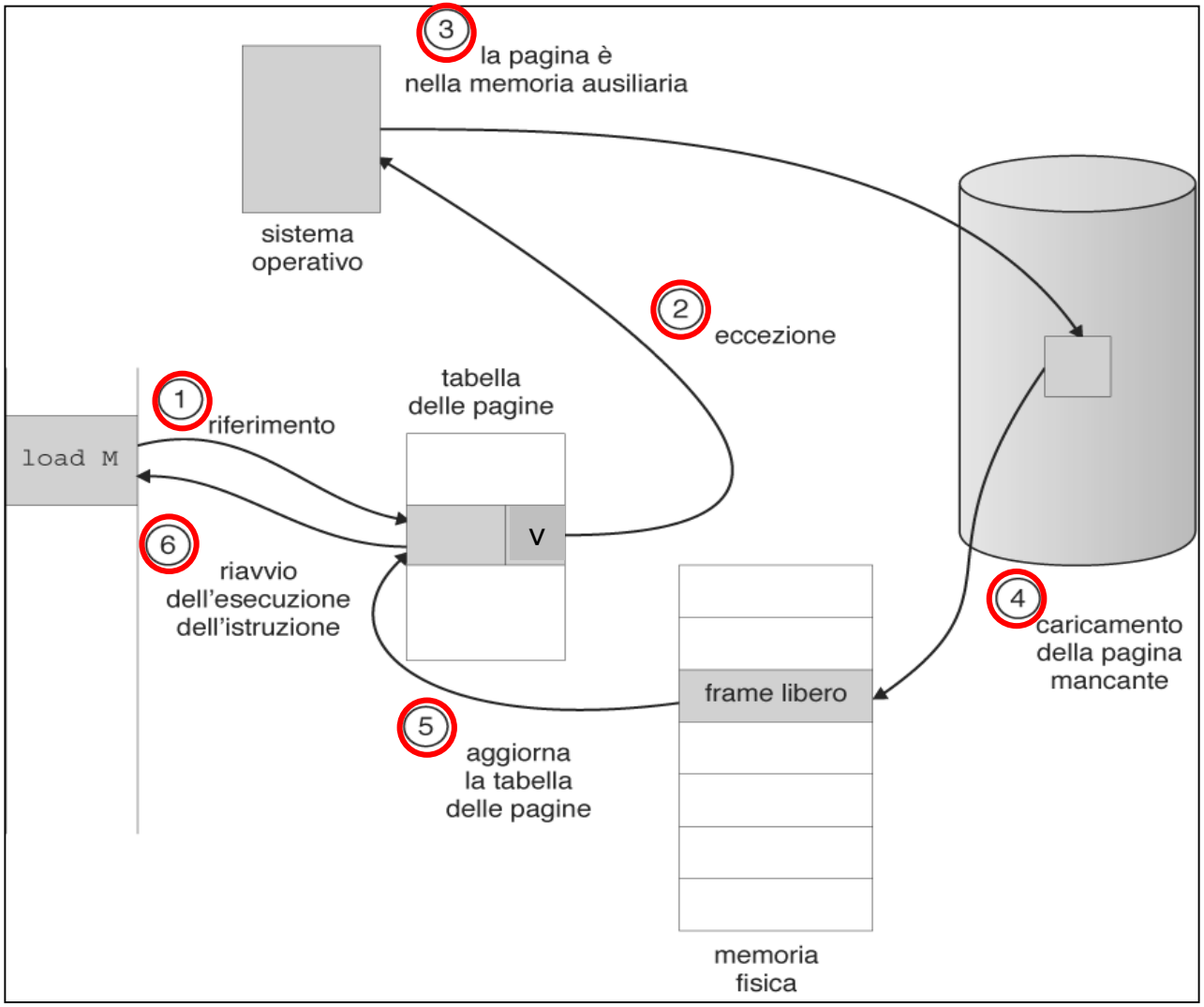


Page Fault

1. Se si tenta di accedere una pagina non in memoria ⇒ ***page fault***.
2. Il S.O. controlla in una tabella interna del processo:
   - Se il riferimento non è corretto ⇒ abort.
   - Se il riferimento è corretto occorre caricare la pagina.
3. Si trova un frame libero.
4. Si carica la pagina nel frame.
5. Si aggiorna la tabella e il bit di validazione = 1.
6. Viene riavviata l'esecuzione: la pagina diventa quella più recente.



Fasi di gestione di un page fault



Prestazioni della paginazione su richiesta

- Probabilità di page fault: $0 \leq p \leq 1$
  - se $p = 0$ nessun page fault
  - se $p = 1$ ogni accesso provoca un page fault
  - se $p = 0.5$ metà degli accessi provocano un page fault.
- Tempo di accesso effettivo (***TAE***)

$$TAE = (1 - p) \times tma + p \times tpf$$

tma = tempo di memory access

tpf = tempo di page fault

Esempio di paginazione su richiesta

- Tempo di accesso in memoria = 100 nanosecondi

- Tempo di page fault = 25 msec = 25000 μsec

$$TAE = (1 - p) \times 100 + p \times (25000000)$$
$$= 100 + (25000000-100) \times p \qquad \textit{(nanosecondi)}$$

Se p= 0.4

$$TAE = (1 - 0.4) \times 100 + 0.4 \times (25000000)$$
$$= 100 + (25000000-100) \times 0.4$$
$$= 10.000.060 \ (\textit{nanosecondi}) \rightarrow 10 \text{ msec}$$

Che cosa accade quando non c’è un frame libero ?

- **Sostituzione delle pagine**

  Occorre trovare una pagina in memoria che non è usata e si porta sul disco (*swap out*).

  - Occorre usare un algoritmo apposito.
  - Prestazioni: si vuole un algoritmo che dia il **numero minimo di page fault**.

- Alcune pagine possono essere portate in memoria (e sul disco) varie volte.

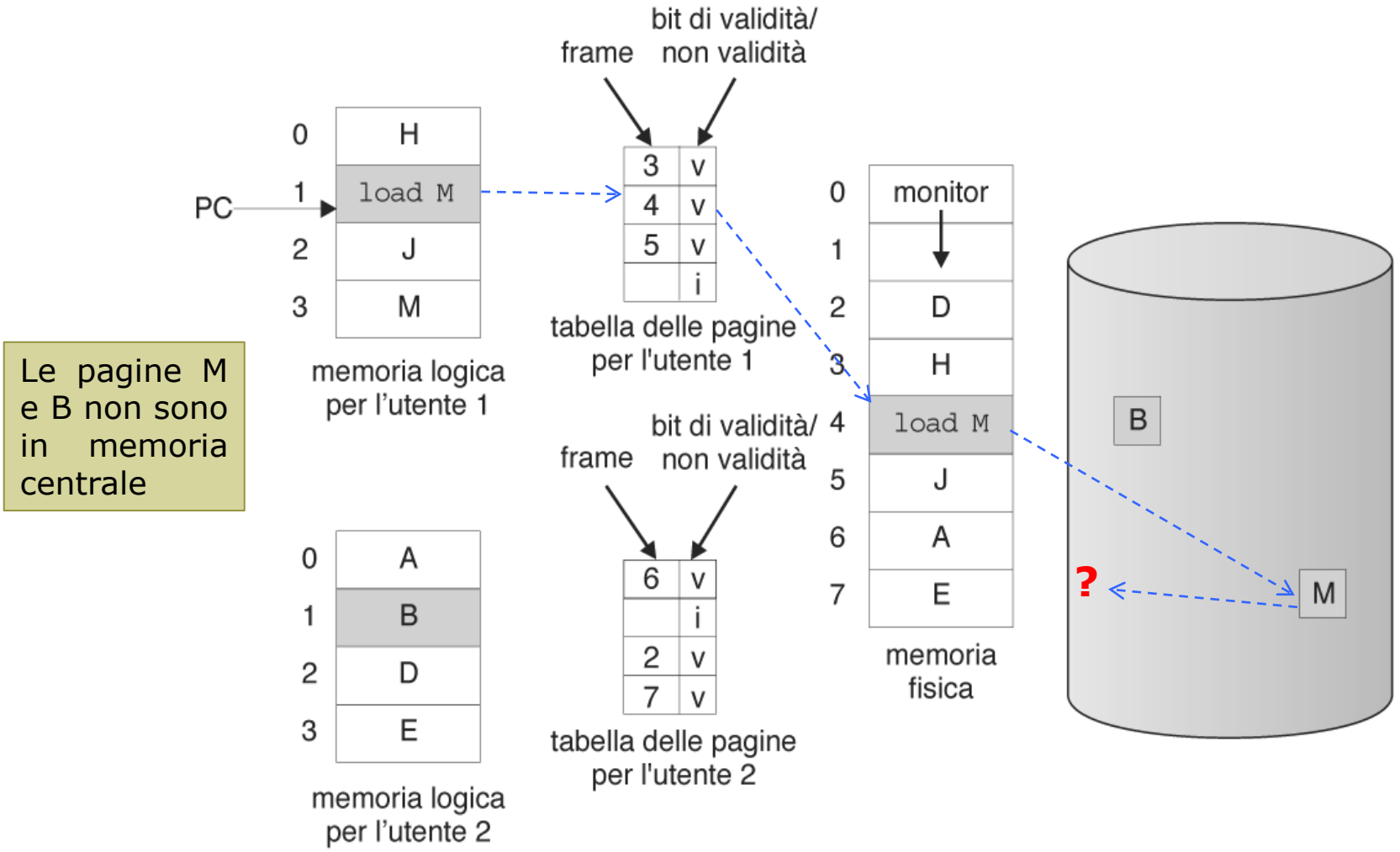
- È auspicabile evitare che una pagina che servirà a breve sia portata sul disco prima di essere richiesta nuovamente.

Sostituzione delle pagine

- Quando occorre caricare una pagina e non c'è un frame libero si può usare la **sostituzione delle pagine**.
- Il meccanismo di gestione dei page fault deve essere modificato per gestire questa possibilità.
- Uso di un *modify* (*dirty*) *bit* per ridurre il costo del trasferimento delle pagine – **solo le pagine modificate vengono (ri)scritte sul disco**.
- La ***sostituzione delle pagine*** completa la separazione tra memoria logica e memoria fisica:

  una grande memoria virtuale si può realizzare su una piccola memoria fisica.

Necessità di sostituzione di pagine

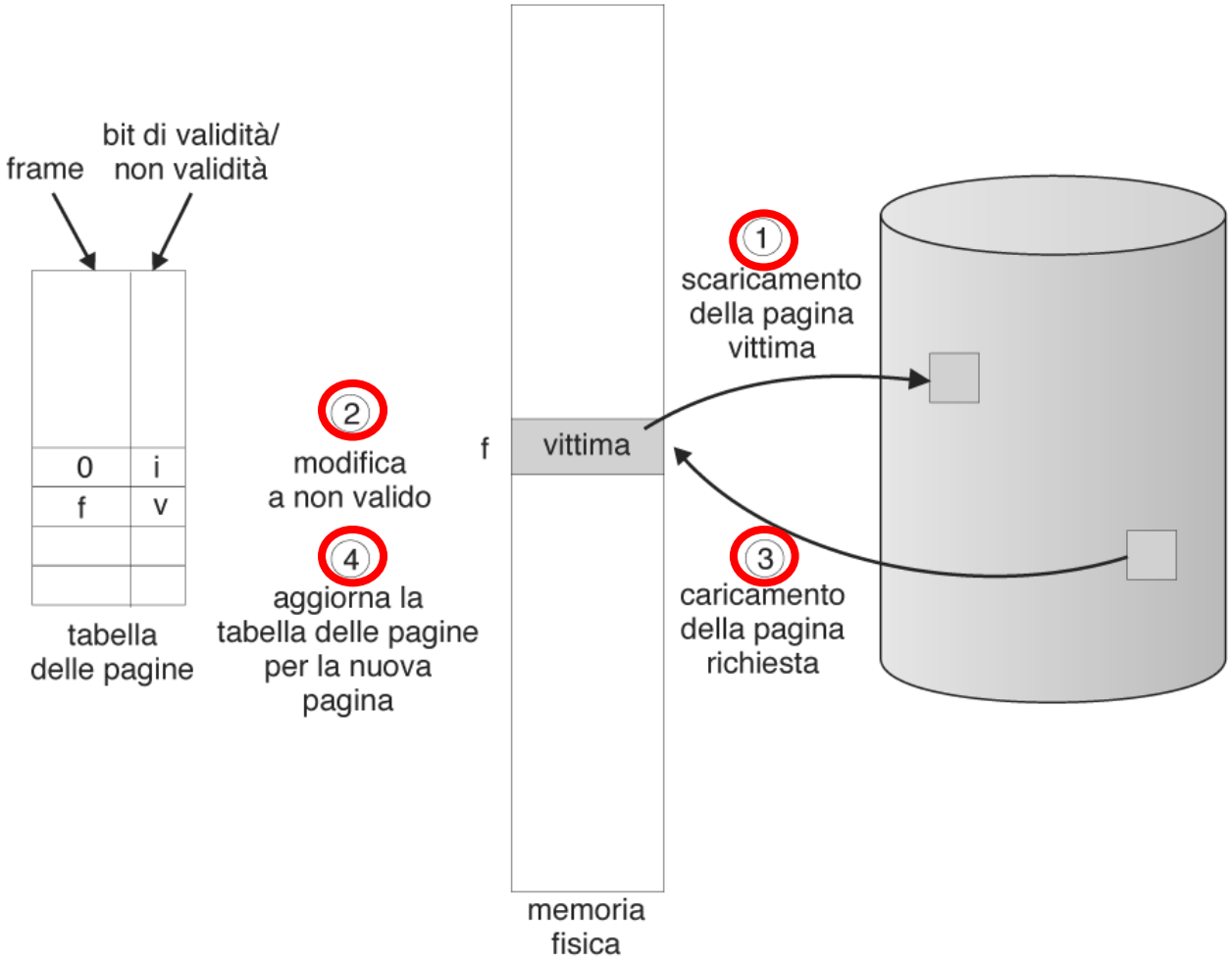


Sostituzione delle pagine

Operazioni per la sostituzione:

- Trova la locazione della pagina richiesta sul disco.
- Trova un frame libero
  - Se esiste usalo;
  - Se non c'è un frame libero seleziona un frame ***vittima*** secondo un algoritmo di sostituzione;
  - Scrivi la pagina vittima sul disco e aggiorna le tabelle.
- Leggi la pagina richiesta nel frame liberato e aggiorna le tabelle.
- Riavvia il processo.

Sostituzione di una pagina



Algoritmi di sostituzione delle pagine

- Criterio di scelta:

  L’algoritmo con la minima frequenza di page fault.

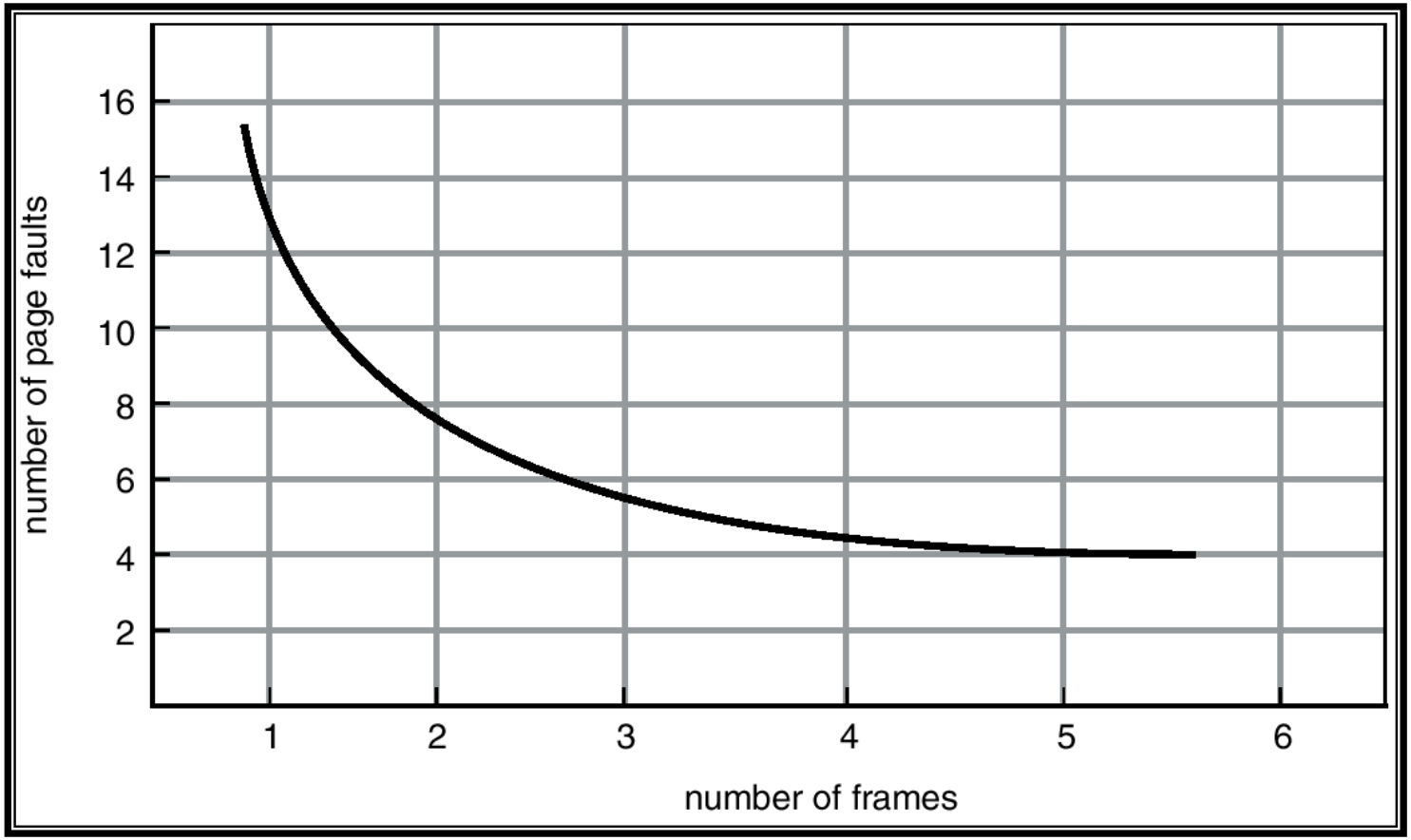
- Si valuta gli algoritmi eseguendoli su una particolare sequenza (stringa) di riferimenti alla memoria e calcolando il numero di page fault che si verificano.

- Negli esempi la stringa usata è:

  1, 2, 3, 4, 1, 2, 5, 1, 2, 3, 4, 5.

  quindi il programma è composto da 5 pagine, non tutte in memoria centrale.

Numero di page fault in funzione del numero di frame



Comportamento atteso:
all'aumentare dei frame disponibili, diminuisce il numero di page fault

Algoritmo First-In-First-Out (FIFO)

- **Si sostituisce la pagina più vecchia (in memoria da più tempo).**
- Stringa: **1, 2, 3, 4, 1, 2, 5, 1, 2, 3, 4, 5**
- 3 frame (3 pagine possono essere in memoria per processo)

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

9 page fault

- 4 frame

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

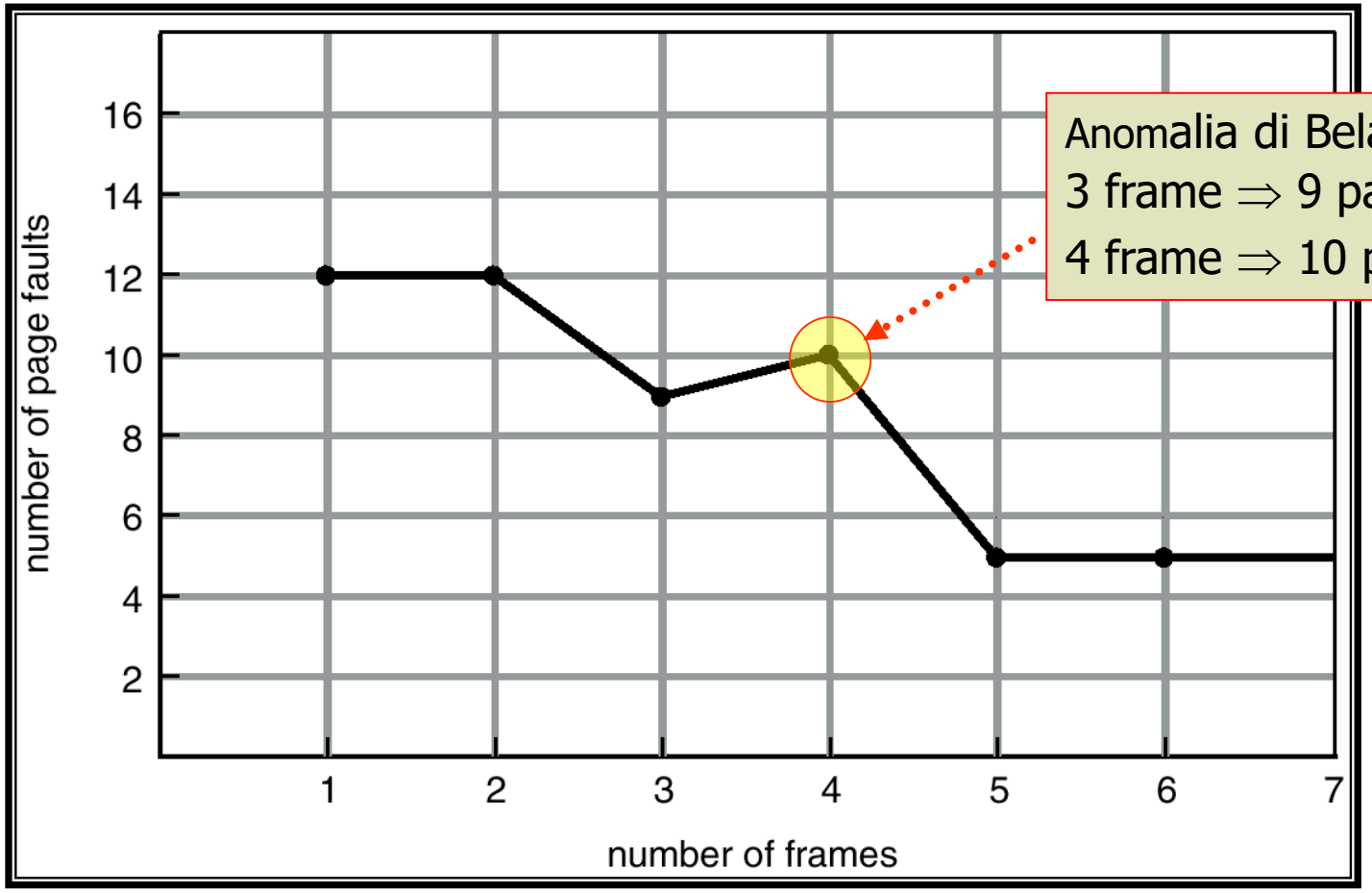
10 page fault

Anomalia di Belady

più frame ⇒ più page fault

* Si contano anche i page fault necessari a caricare i primi frame

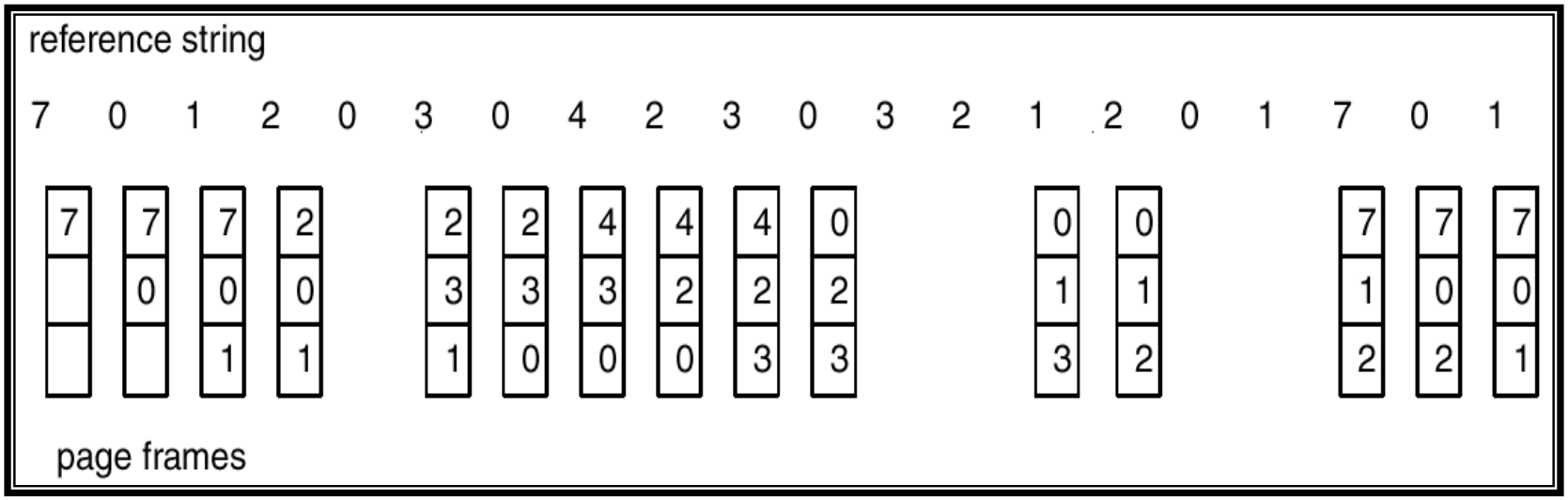
Anomalia di Belady



Stringa: **1, 2, 3, 4, 1, 2, 5, 1, 2, 3, 4, 5**

Sostituzione FIFO

Programma composto da 8 pagine (0..7) con solo 3 frame in memoria centrale.

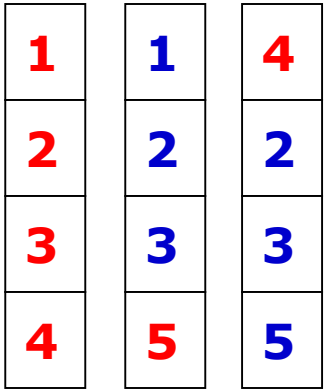


15 page fault

Algoritmo Ottimale

- **Sostituisce la pagina che non verrà usata per il periodo di tempo più lungo**.
- Esempio: 4 frame

stringa: **1, 2, 3, 4, 1, 2, 5, 1, 2, 3, 4, 5**

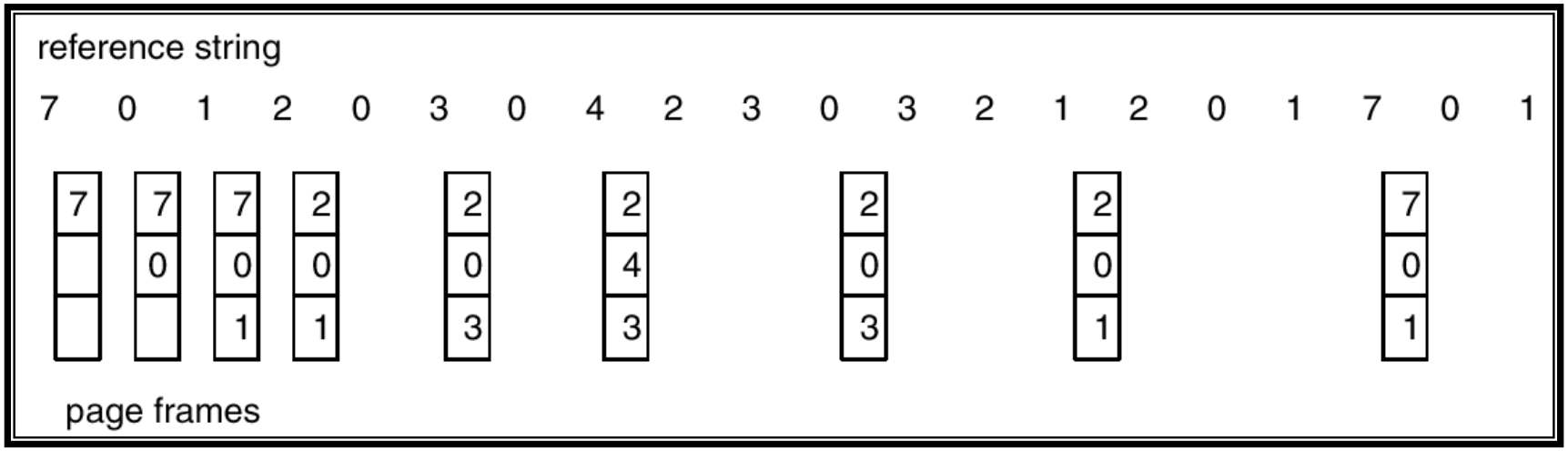


6 page fault

- Come predire il futuro (uso delle pagine) ?
- **Usato come paragone per valutare altri algoritmi**.

Algoritmo Ottimale

Programma composto da 8 pagine (0..7) con solo 3 frame in memoria centrale.

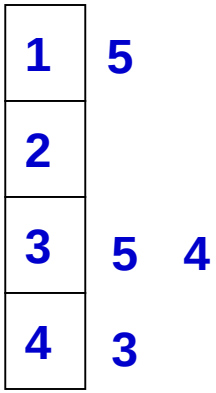


9 page fault

Algoritmo Least Recently Used (LRU)

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

- Stringa: 1, 2, 3, 4, 1, 2, 5, 1, 2, 3, 4, 5



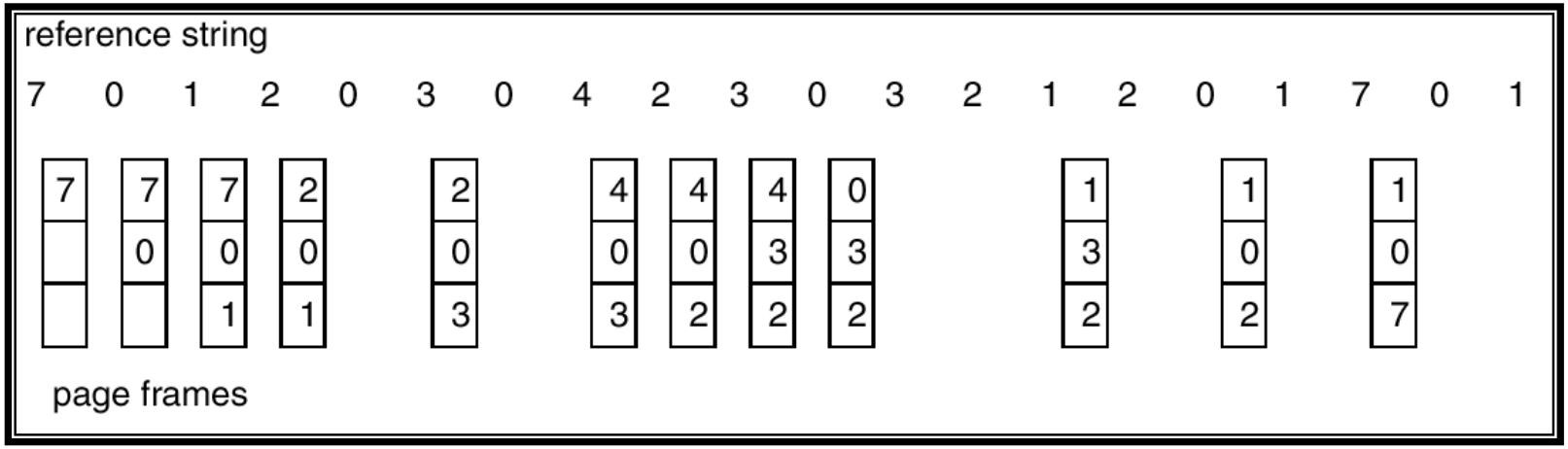
8 page fault

- Implementazione con ***contatore***

- Implementazione con **stack**

Sostituzione LRU

Programma composto da 8 pagine (0..7) con solo 3 frame in memoria centrale.



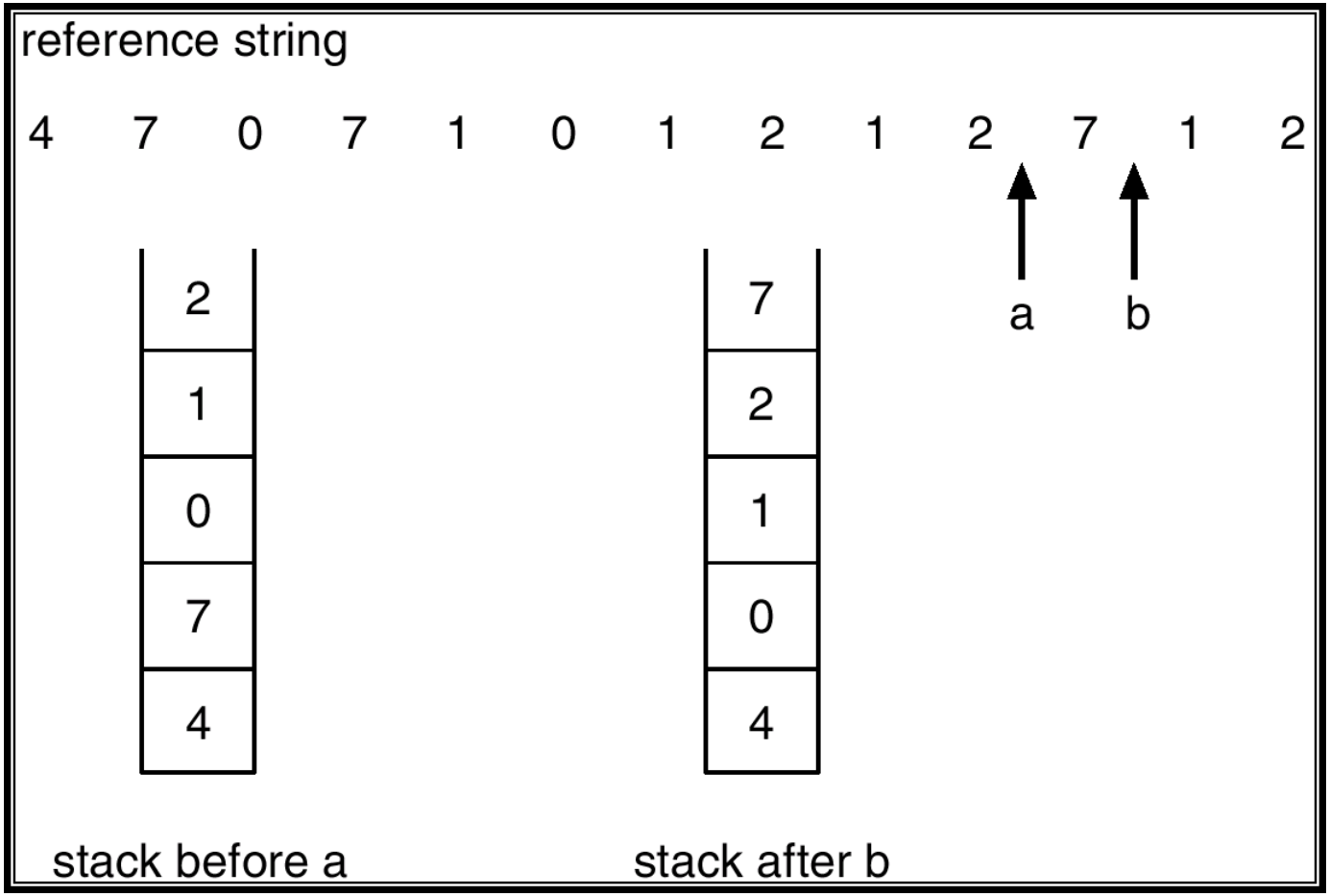
12 page fault

Sostituzione LRU

- Implementazione con ***contatore***
  - Ogni entry di pagina ha un contatore; ogni volta che la pagina viene referenziata si copia il clock nel contatore.
  - Quando occorre sostituire una pagina, occorre cercare la pagina con il valore del clock più piccolo.

- Implementazione con **stack**
  - Si mantiene uno stack con i numeri delle pagine; la pagina più recente sta in cima e la *LRU in fondo*:
  - La pagina referenziata si mette in cima:
    - spostare il numero di pagina in prima posizione
  - Non bisogna fare la ricerca, è sufficiente eliminare la pagina in fondo allo stack.

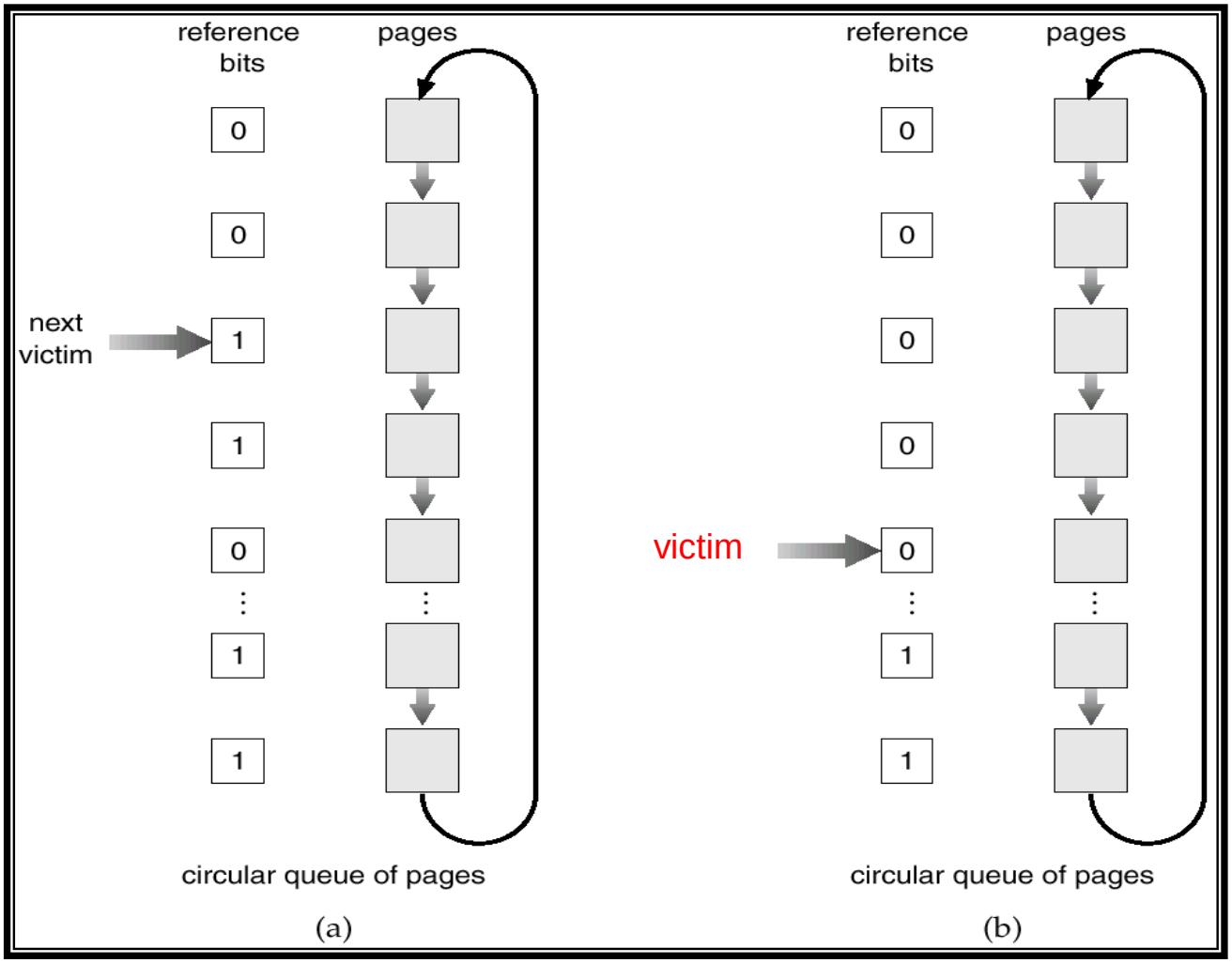
Implementazione LRU con lo stack



Algoritmi di approssimazione del LRU

- L'algoritmo LRU richiede l'uso di un supporto hardware.
- Se non è disponibile nel computer usato, si usano algoritmi approssimati basati sull'uso di un **bit di riferimento**
  - Ad ogni pagina è associato un bit, inizialmente = 0;
  - Quando la pagina è usata il bit viene modificato ad 1.
- **Algoritmo con seconda chance**
  - Si usa il bit di riferimento.
  - Si sostituisce una pagina con bit a 0 (se esiste). L'ordine di uso delle pagine non si conosce.
  - Se una pagina (in senso orario) ha il bit = 1, allora:
    - bit <-- 0.
    - Lascia la pagina in memoria.
    - Sostituisce la prossima pagina (in senso orario) con bit=0.

Algoritmo Seconda Chance (clock)



Algoritmi con conteggio

- Esistono anche alcuni algoritmi basati sul conteggio dell'uso delle pagine (poco comuni).
- Usano un contatore del numero dei riferimenti che sono stati fatti su ogni pagina.

- **Algoritmo LFU (Least Frequently Used)**: ***sostituisce le pagine con il contatore minimo***.

- **Algoritmo MFU (Most Frequently Used)**: ***sostituisce le pagine con il contatore massimo.***

  (basato sull'idea che la pagina con il contatore minimo è stata inserita da poco e non è stata usata)

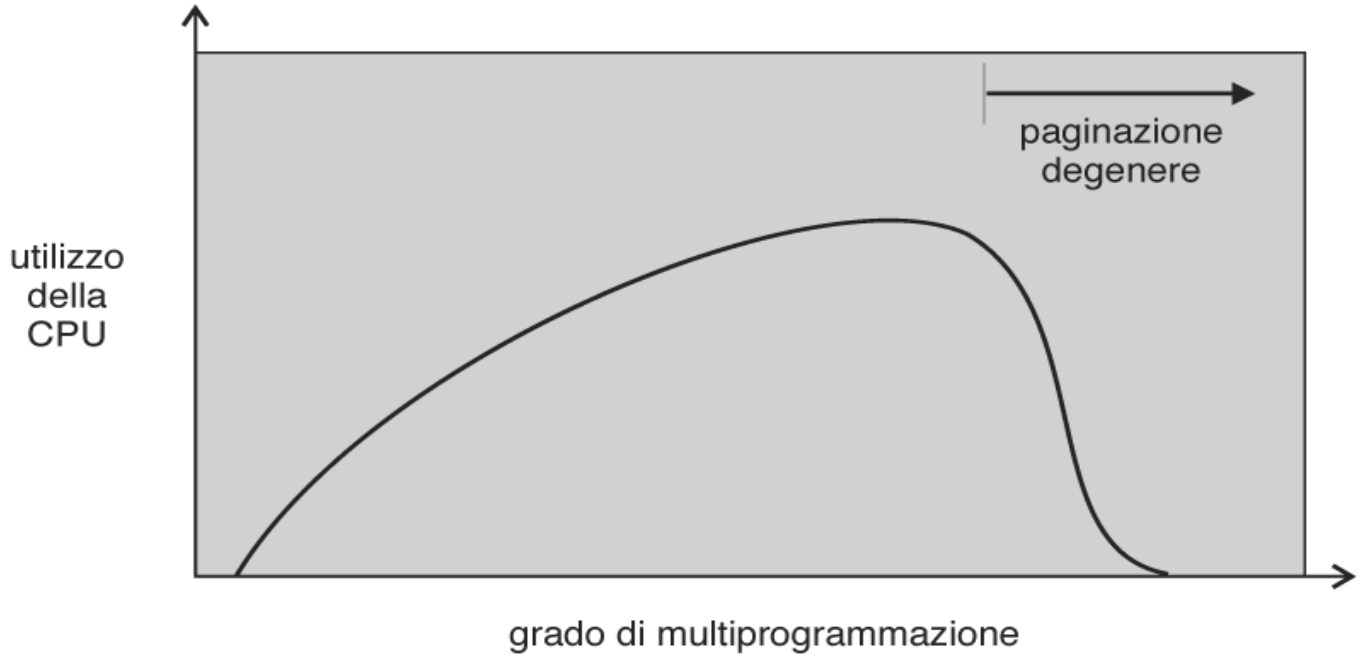
Thrashing

- La sostituzione delle pagine può essere **locale** (al processo stesso) o **globale** (tra le pagine di tutti i processi).

- Se un processo non ha abbastanza pagine in memoria il tasso di page fault è molto alto. Questo crea:
  - Bassa utilizzazione della CPU.
  - Il S.O. può credere che occorre aumentare il grado di multiprogrammazione.
  - Un nuovo processo viene inserito in memoria.

- **Thrashing** ≡ un processo è occupato principalmente nella attività di paginazione.

Paginazione degenere (*thrashing*)

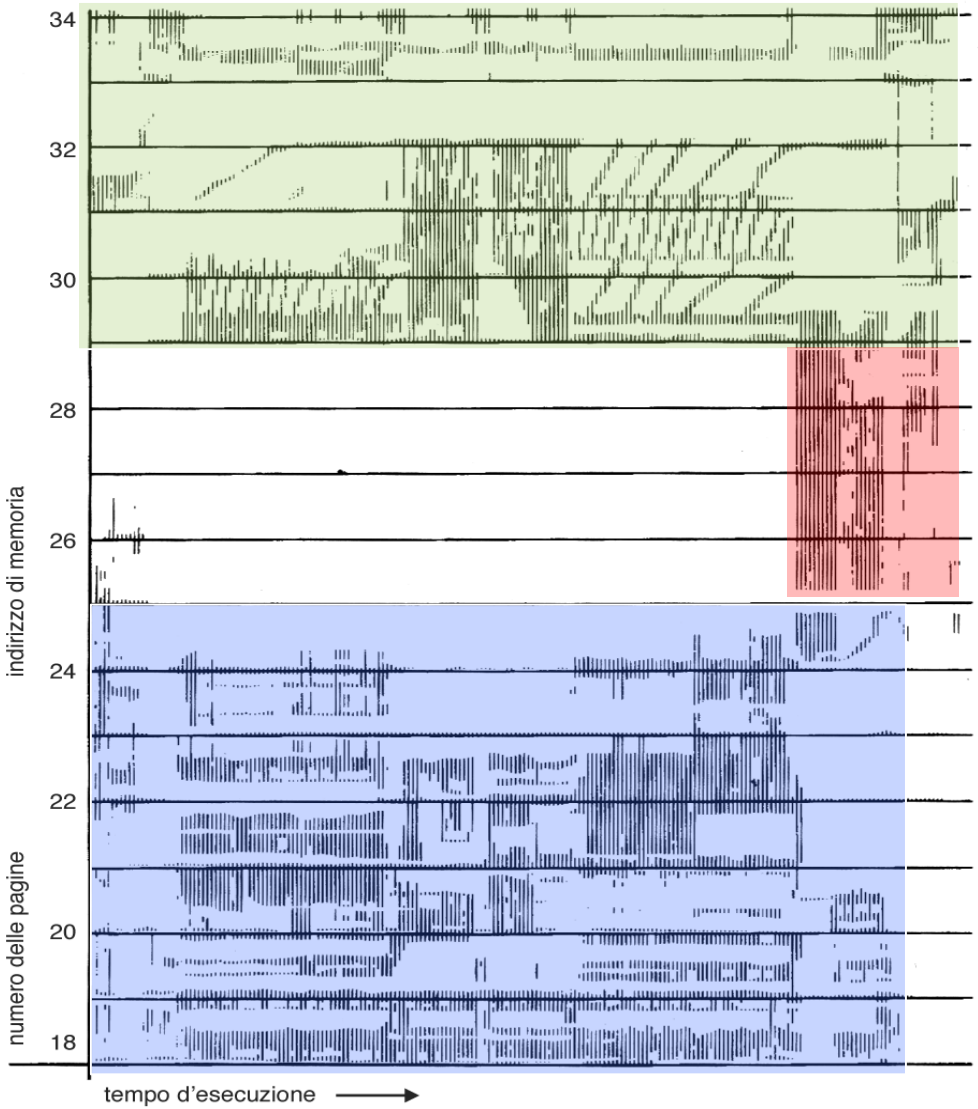


- **Modello di località** (set di pagine usate insieme)
  - Un processo di sposta da una località all'altra (da un gruppo di pagine ad un altro gruppo di pagine)
  - Le località si possono sovrapporre.

- Perché si verifica il thrashing ?

  Σ spazi di località > spazio di memoria locale

Località dei riferimenti alla memoria



Esempi di gestione di memoria virtuale

Windows XP

- Usa la **paginazione su richiesta: paging con clustering**. Il clustering mantiene le pagine vicine alla pagina che ha creato il page fault.

Solaris

- Mantiene una **lista di pagine libere** da assegnare ai processi che hanno avuto un page fault.

Windows XP

- Ai processi viene assegnato un ***Working set minimo*** e un ***Working set massimo***.
- ***Working set minimo*** è il numero delle pagine che il processo avrà in memoria (garantito)
- Gli potranno essere assegnate altre pagine fino al ***Working set massimo***.
- Quando l'ammontare di memoria libera nel sistema è minore di una data soglia viene eseguito un algoritmo di *automatic working set trimming* per aumentare la dimensione della memoria libera.
- Il *working set trimming* rimuove dai processi le pagine in eccesso rispetto al loro working set minimo.

Solaris

- **Lotsfree:** parametro di soglia per iniziare la sostituzione delle pagine.
- La sostituzione delle pagine è eseguita dal processo *Pageout*.
- *Pageout* scandisce le pagine usando un algoritmo basato sulla coda circolare (seconda chance) modificato per liberare memoria.
- **Scanrate** è la velocità con cui le pagine sono scandite. Varia tra due valori: **slowscan** e **fastscan**.
- *Pageout* è invocato più o meno frequentemente in dipendenza della memoria libera disponibile.
- Se necessario, *Pageout* esegue lo swap out di interi processi.

Scansione delle pagine in Solaris