

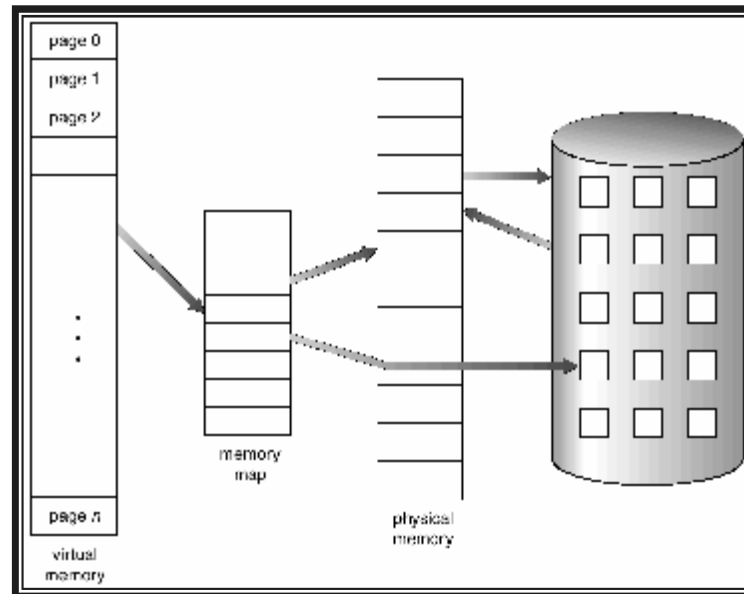
Memoria Virtuale

- Background
- Paginazione su richiesta
- Sostituzione delle pagine
- Algoritmi di sostituzione delle pagine
- Allocazione dei frame
- Thrashing

Background

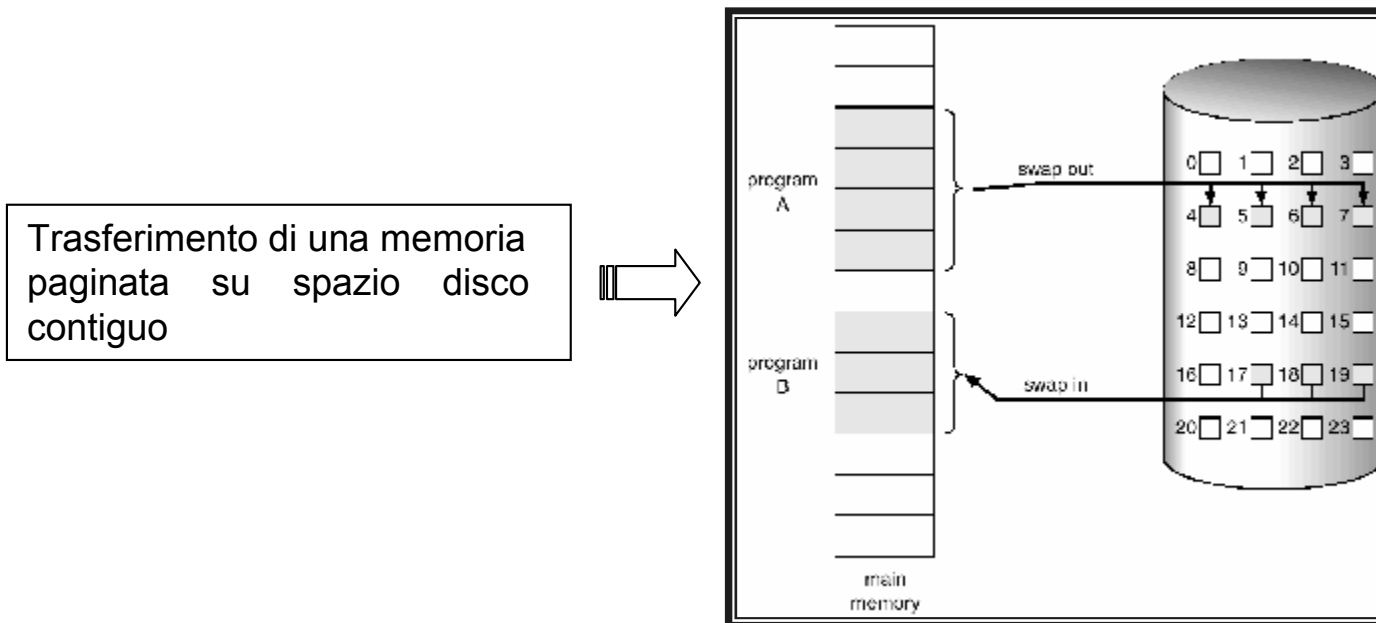
- **Memoria virtuale** — separazione della memoria logica dell'utente dalla memoria fisica.
 - ◆ Solo parti del programma devono trovarsi in memoria per l'esecuzione.
 - ◆ Lo spazio logico degli indirizzi può essere più grande dello spazio fisico.
 - ◆ Porzioni di spazio fisico possono essere condivise da più processi.
- La memoria virtuale può essere implementata per mezzo di:
 - ◆ **Paginazione su richiesta**
 - ◆ **Segmentazione su richiesta**

Esempio di spazio degli indirizzi virtuali più esteso dello spazio degli indirizzi fisici.



Paginazione su richiesta

- *Swapper pigro*: porta una pagina in memoria solo quando è necessaria:
 - ◆ Occorrono meno operazioni di I/O
 - ◆ Viene impiegata meno memoria
 - ◆ Si ha una risposta più veloce
 - ◆ Si possono gestire più utenti
- Richiesta di una pagina \Rightarrow si fa un riferimento alla pagina
 - ◆ Riferimento non valido \Rightarrow abort
 - ◆ Pagina non in memoria \Rightarrow trasferimento in memoria



Bit di validità

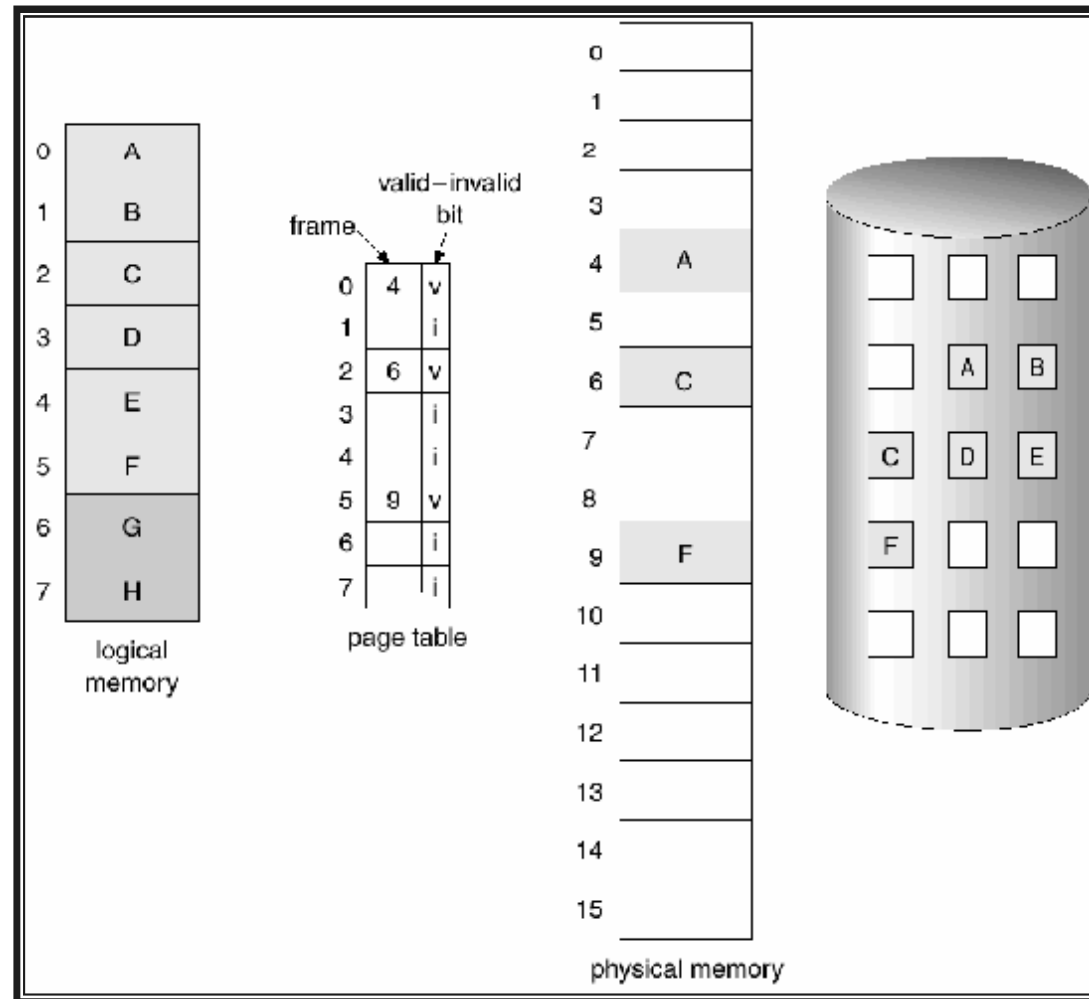
- Un bit di validità viene associato a ciascun elemento della tabella delle pagine (1 \Rightarrow in memoria, 0 \Rightarrow non in memoria).
- Inizialmente il bit di validità viene posto a 0 per tutte le pagine.
- Esempio di una configurazione della tabella delle pagine.

# Frame	Bit di validità
	1
	1
	1
	1
	0
⋮	
	0
	0

Tabella delle pagine

- In fase di traduzione degli indirizzi, se il bit di validità vale 0 \Rightarrow si ha un ***page fault***.

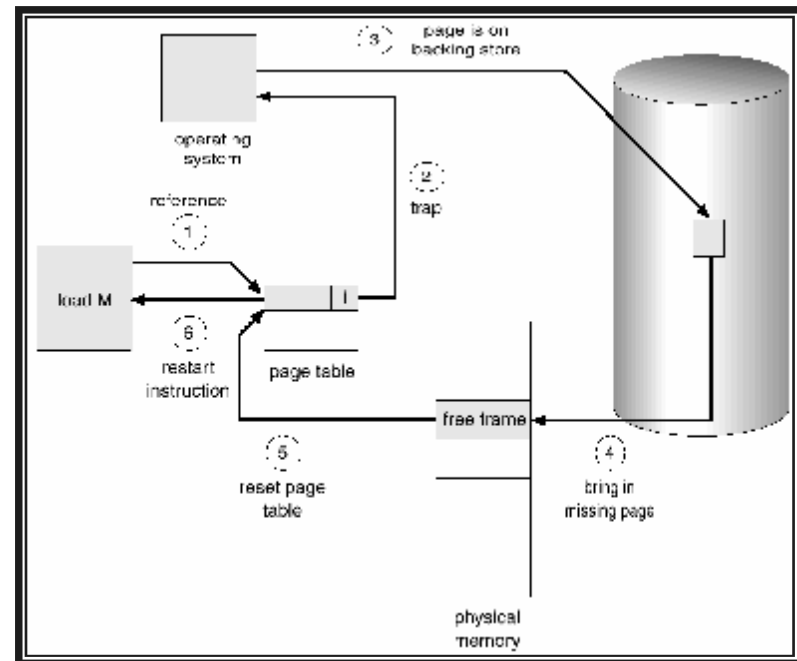
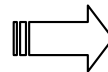
Istantanea di una tabella delle pagine con pagine non allocate in memoria principale



Page fault

- Se non ci sono mai stati riferimenti ad una pagina, il primo riferimento causa un **trap** al SO \Rightarrow page fault.
- Il SO consulta una tabella per decidere se si tratta di...
 - ◆ riferimento non valido \Rightarrow abort;
 - ◆ pagina non in memoria.
- Seleziona un frame vuoto.
- Sposta la pagina nel frame.
- Aggiorna le tabelle (bit di validità = 1).
- Viene riavviata l'istruzione che era stata interrotta.

Gestione del page fault

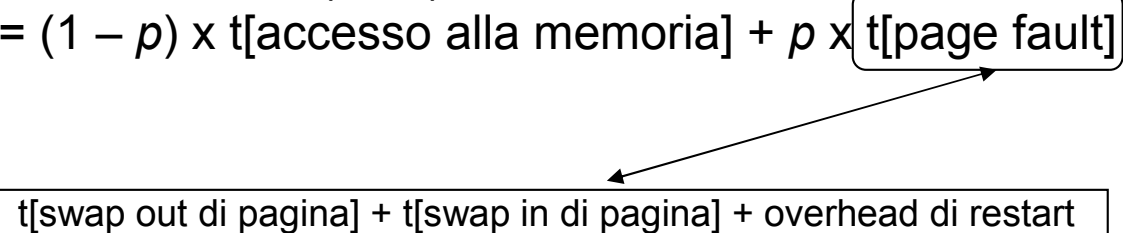


Cosa accade quando non ci sono frame liberi ?

- *Sostituzione di pagina* — si trova una pagina in memoria che non venga utilizzata attualmente, e si sposta sul disco.
 - ◆ Scelta di un algoritmo di selezione;
 - ◆ Prestazioni: è richiesto un metodo che produca il minimo numero di page fault.
- La stessa pagina può essere riportata in memoria più volte.
- *Page Fault Rate*: $0 \leq p \leq 1.0$
 - ◆ se $p = 0$ non si hanno page fault;
 - ◆ se $p = 1$, ciascun riferimento è un fault.
- Tempo medio di accesso (EAT):

$$\text{EAT} = (1 - p) \times t[\text{accesso alla memoria}] + p \times t[\text{page fault}]$$

$t[\text{swap out di pagina}] + t[\text{swap in di pagina}] + \text{overhead di restart}$



Esempio di paginazione su richiesta

- Tempo di accesso alla memoria = 1 μsec
- Il 50% delle volte che una pagina viene rimpiazzata ha subito delle modifiche e deve essere sottoposta a swap out.
- Tempo di swap per pagina = 10 msec = 10000 μsec

$$\begin{aligned} \text{EAT} &= (1 - p) \times 1 + p \times (15000) \\ &\approx 1 + 15000p \quad (\text{in } \mu\text{sec}) \end{aligned}$$

Nota: Se un accesso su 1000 causa un page fault, EAT = 16 μsec \Rightarrow con la paginazione su richiesta l'accesso in memoria viene rallentato di un fattore 16.
Se si desidera un rallentamento inferiore al 10%:

$$0.1 > 15000p \Rightarrow p < 0.0000067$$

cioè può essere permesso meno di un page fault ogni 150000 accessi in memoria.

Creazione di processi

- La memoria virtuale fornisce altri benefici durante la creazione dei processi:
 - ◆ *Copia su scrittura*
 - ◆ *File mappati in memoria*

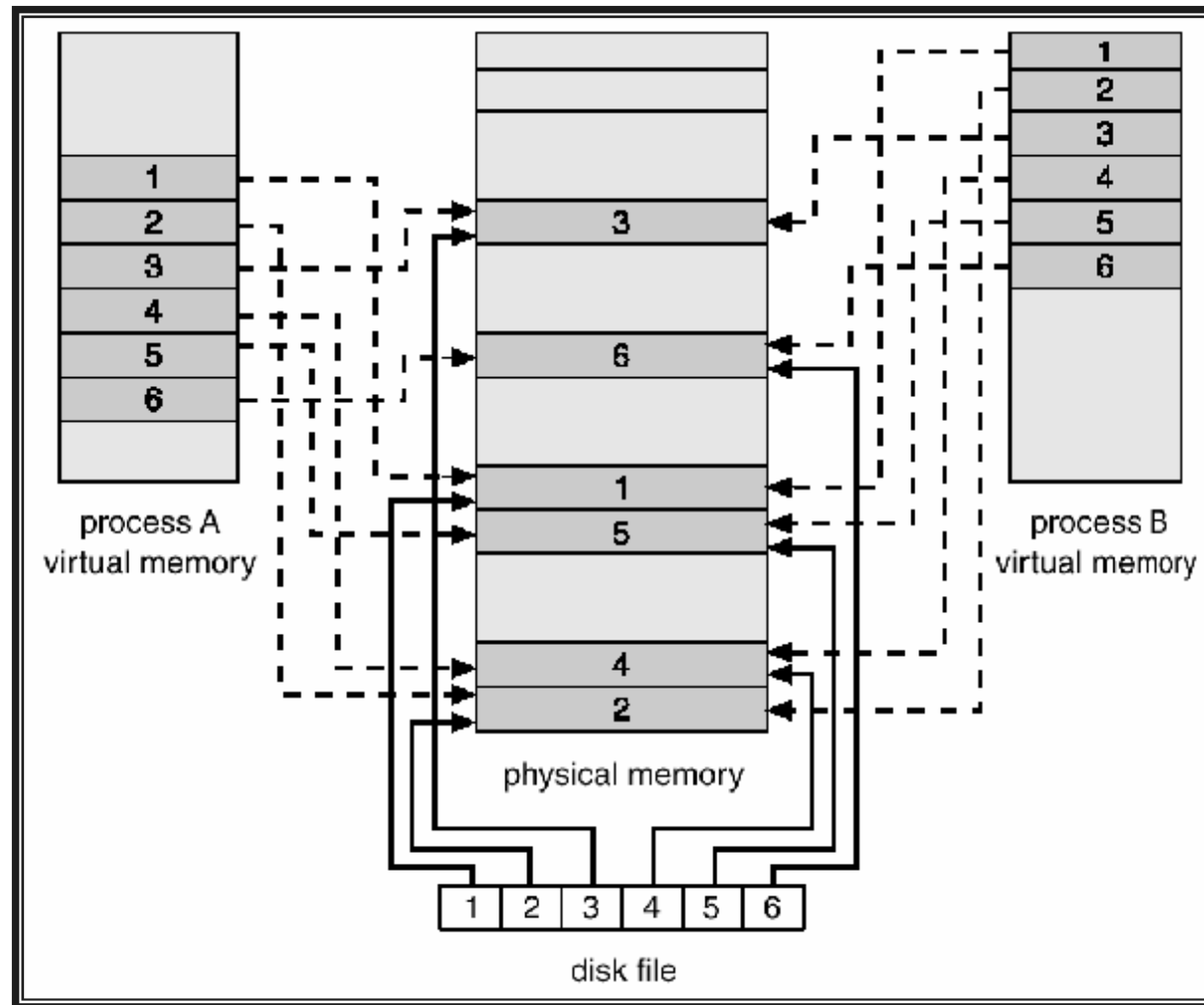
Copia su scrittura

- La copia su scrittura permette sia al processo genitore, sia al figlio, di condividere *inizialmente* pagine di memoria.
- Se uno dei processi cerca di modificare una pagina, solo allora la pagina è copiata.
- La copia su scrittura permette una creazione dei processi, in quanto solo le pagine modificate vengono copiate.
- Le pagine libere sono allocate da un pool di pagine azzerate.
- Alcuni sistemi UNIX mettono a disposizione la chiamata a **vfork()**.

File mappati in memoria

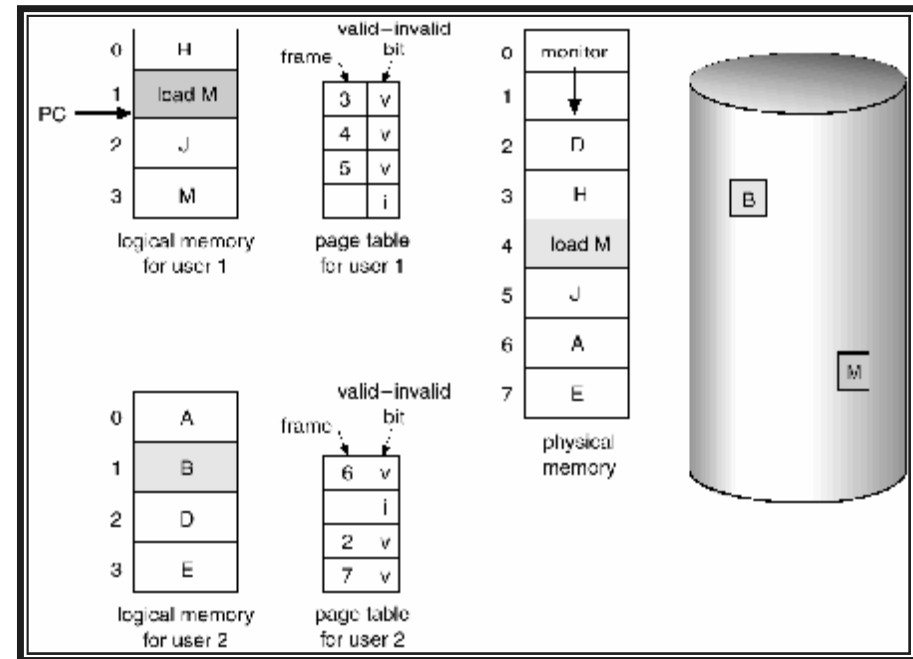
- L'I/O su file mappati in memoria permette di trattare le operazioni di I/O come ordinari accessi alla memoria, *mappando* un blocco del disco su una pagina della memoria.
- Un file è letto inizialmente usando la paginazione su richiesta; una parte del file della dimensione di una pagina è prelevata dal disco e memorizzata su una pagina della memoria.
- Le successive operazioni di lettura e scrittura su file vengono trattate come accessi alla memoria.
- Tale operazione semplifica l'accesso al file, trattando tali operazioni come I/O attraverso la memoria, piuttosto che chiamate a **read()** e **write()**.
- Permette anche a più processi di accedere allo stesso file, mediante la condivisione delle pagine della memoria.

File mappati in memoria



Sostituzione delle pagine

- La sovra-allocazione della memoria si verifica quando è richiesta più memoria di quella effettivamente disponibile.
- Si previene la sovra-allocazione delle pagine modificando le routine di servizio, includendo la sostituzione delle pagine (oppure terminando i processi).
- Si impiega un bit di modifica (*dirty*) per ridurre il sovraccarico dei trasferimenti di pagine: solo le pagine modificate vengono riscritte sul disco.
- La sostituzione delle pagine completa la separazione fra memoria logica e memoria fisica — una grande memoria virtuale può essere fornita ad un sistema con poca memoria fisica.

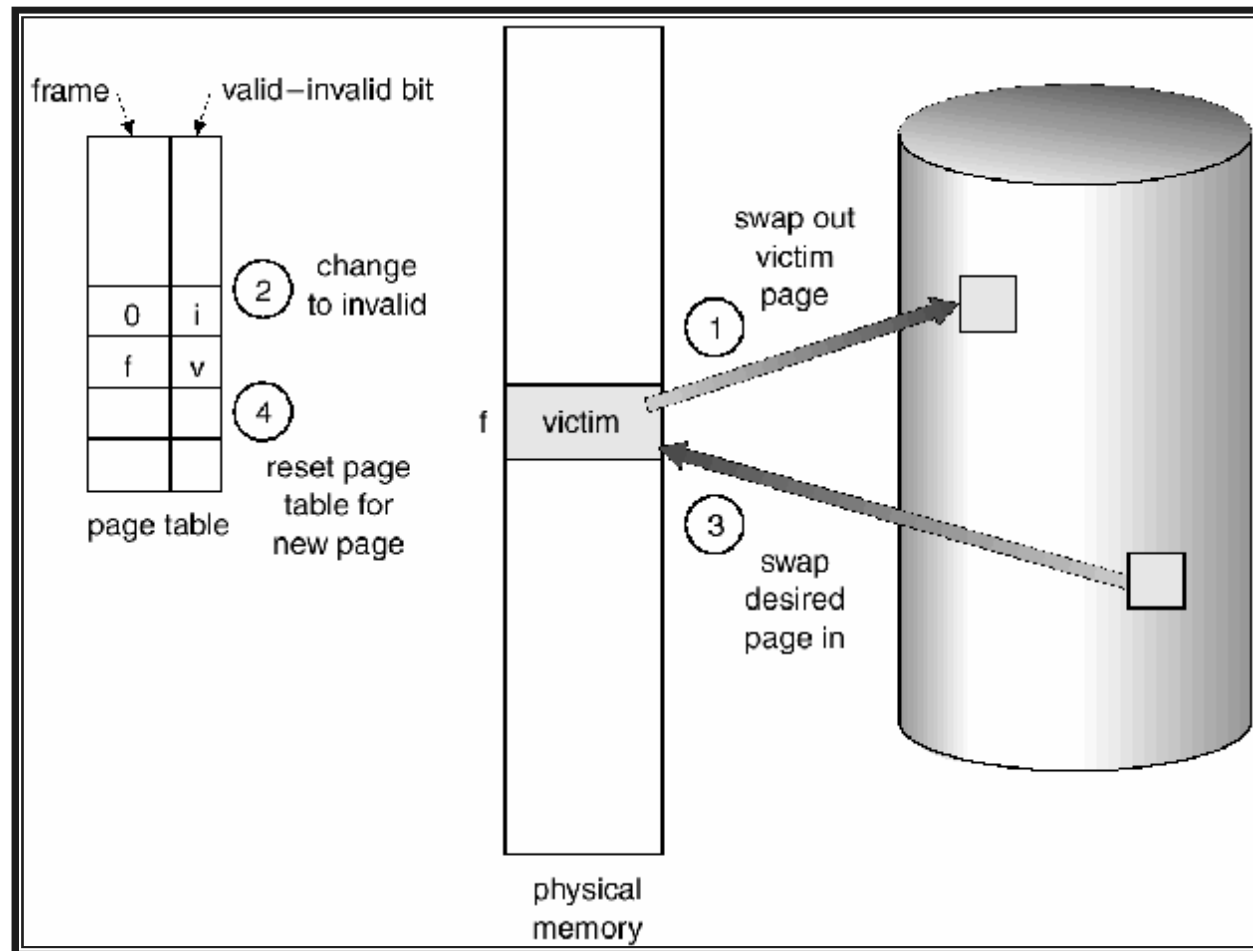


Necessità di trasferimento di una pagina

Sostituzione delle pagine

1. Individuazione della pagina richiesta su disco.
2. Individuazione di un frame libero:
 - A. Se esiste un frame libero, viene utilizzato.
 - B. Altrimenti viene utilizzato un algoritmo di sostituzione per selezionare un frame *vittima*.
 - C. La pagina vittima viene scritta sul disco; le tabelle delle pagine e dei frame vengono modificate conformemente.
3. Lettura della pagina richiesta nel frame appena liberato; modifica delle tabelle delle pagine e dei frame.
4. Riavvio del processo utente.

Sostituzione delle pagine



Algoritmi di sostituzione delle pagine

- È richiesta la minimizzazione della frequenza di page fault.
- Si valutano gli algoritmi eseguendoli su una particolare stringa di riferimenti a memoria (*reference string*) e contando il numero di page fault su tale stringa.
- In tutti gli esempi seguenti, la reference string è

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

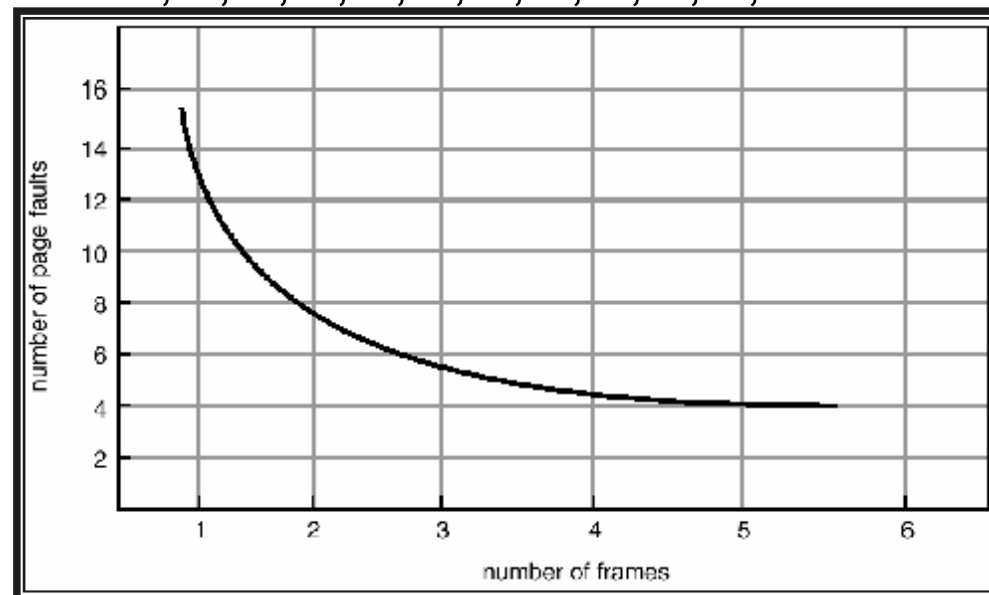


Grafico del numero di page fault rispetto al numero di frame

Algoritmo First-In-First-Out (FIFO)

- Stringa dei riferimenti: 1, 2, 3, 4, 1, 2, 5, 1, 2, 3, 4, 5
- 3 frame (3 pagine per ciascun processo possono trovarsi in memoria contemporaneamente).

1	1	4	5	
2	2	1	3	9 page fault
3	3	2	4	

- 4 frame

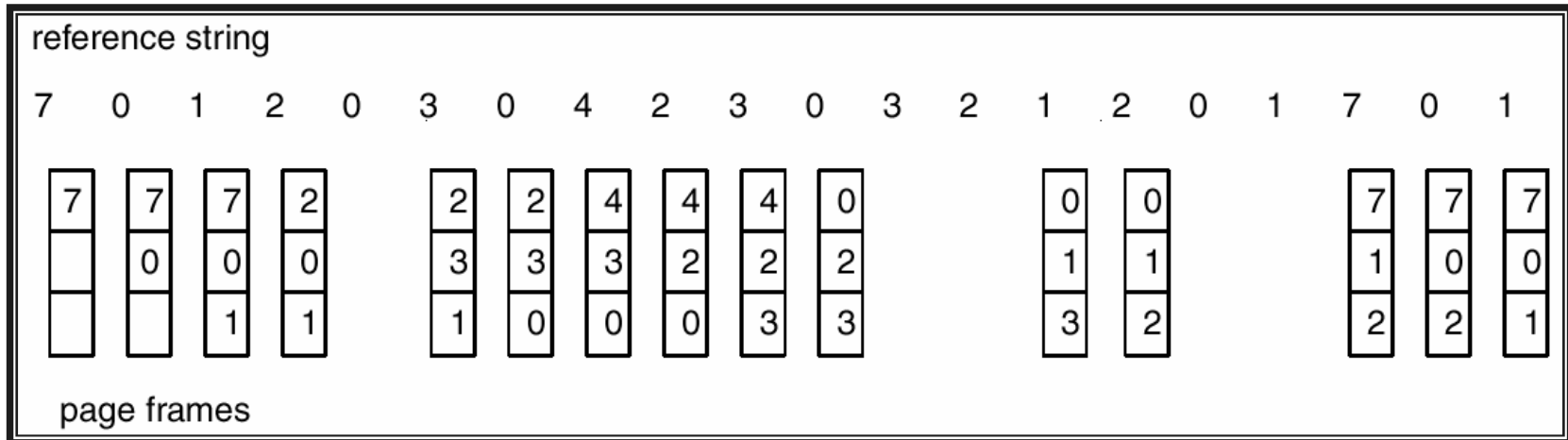
1	1	5	4	
2	2	1	5	10 page fault
3	3	2		
4	4	3		

- Algoritmo FIFO — **Anomalia di Belady**
 - ◆ più frame \Rightarrow più page fault

Sostituzione delle pagine FIFO

ESEMPIO

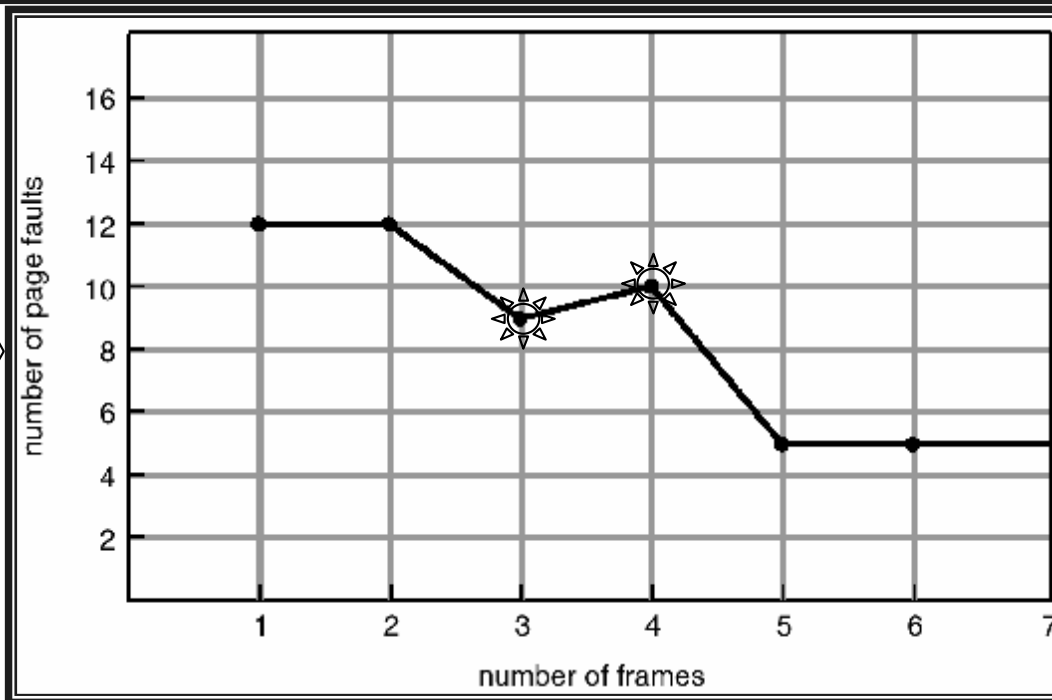
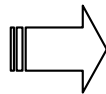
15
fault



Curva dei page fault per sostituzione FIFO



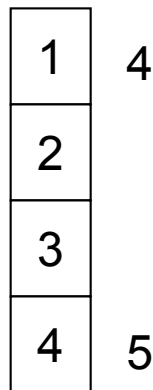
Anomalia di Belady



Algoritmo ottimo

- Sostituire la pagina che non verrà usata per il periodo di tempo più lungo.
- Esempio: 4 frame, con stringa dei riferimenti

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

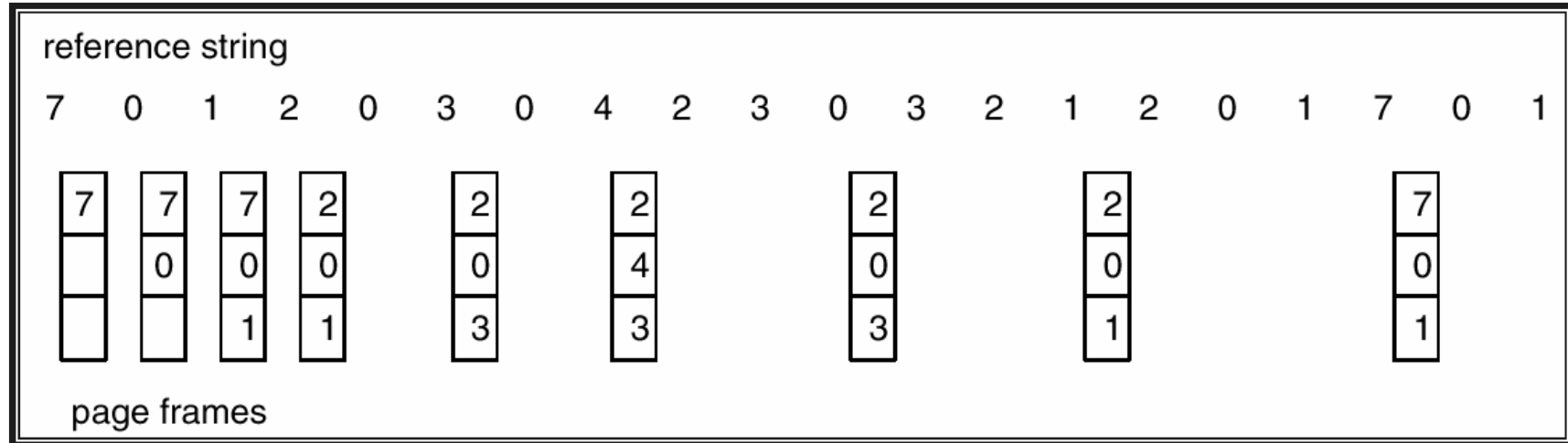


6 page faults

- Come si può conoscere l'identità della pagina?
- Di solo interesse teorico: viene impiegato per misurare le prestazioni (comparative) degli algoritmi con valenza pratica.

Sostituzione delle pagine ottima

ESEMPIO



9
fault

Algoritmo Least-Recently-Used (LRU)

■ Stringa dei riferimenti:

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

1	5
2	
3	5 4
4	3

■ Implementazione con contatore

- ◆ Ciascuna pagina ha un contatore; ogni volta che si fa riferimento alla pagina si copia l'orologio nel contatore.
- ◆ Quando si deve rimuovere una pagina, si analizzano i contatori per scegliere quale pagina cambiare.

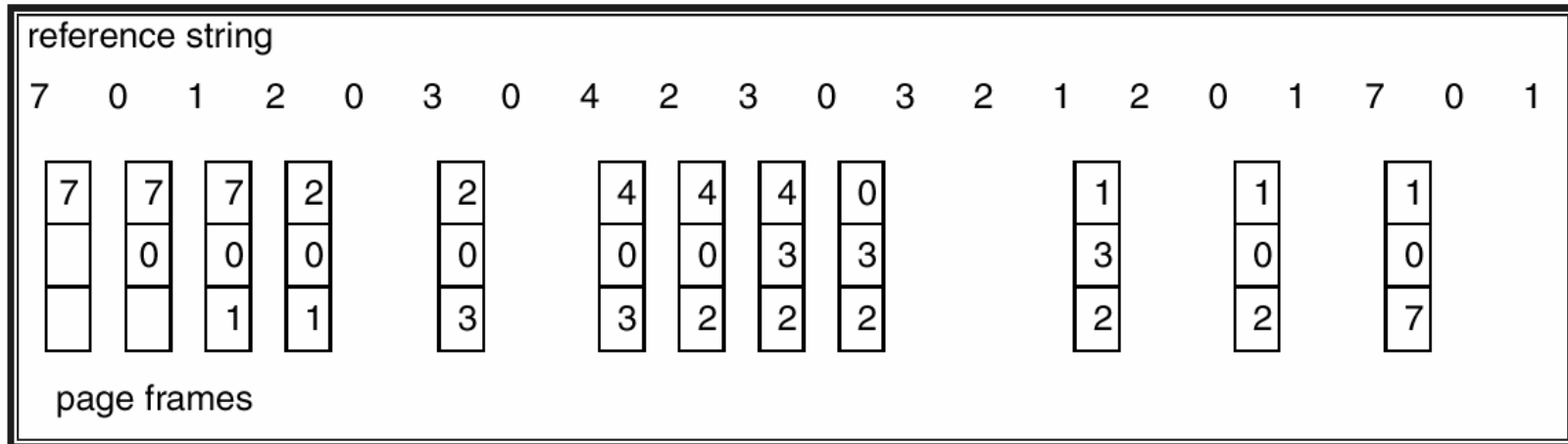
■ Implementazione con stack — si tiene uno stack di numeri di pagina in forma di lista doppiamente concatenata:

- ◆ Pagina referenziata:
 - ✓ Si sposta in cima allo stack
 - ✓ È necessario al più aggiornare 6 puntatori
- ◆ Non è necessario fare ricerche per la scelta.

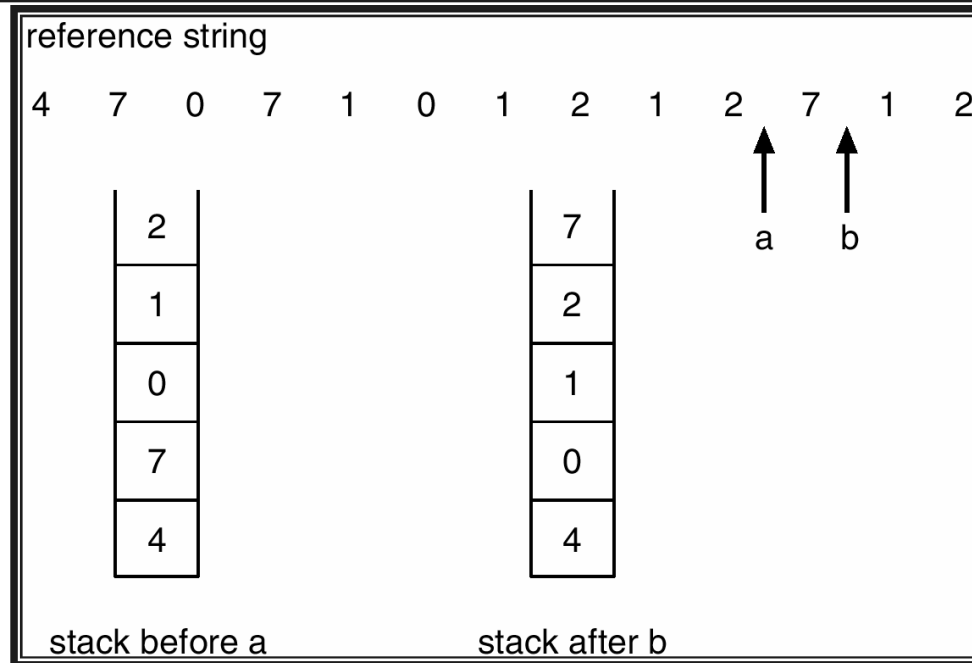
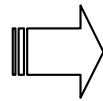
Sostituzione delle pagine LRU

ESEMPIO

12
fault



Utilizzo dello stack per registrare i riferimenti alle pagine più recenti



Algoritmi di approssimazione a LRU

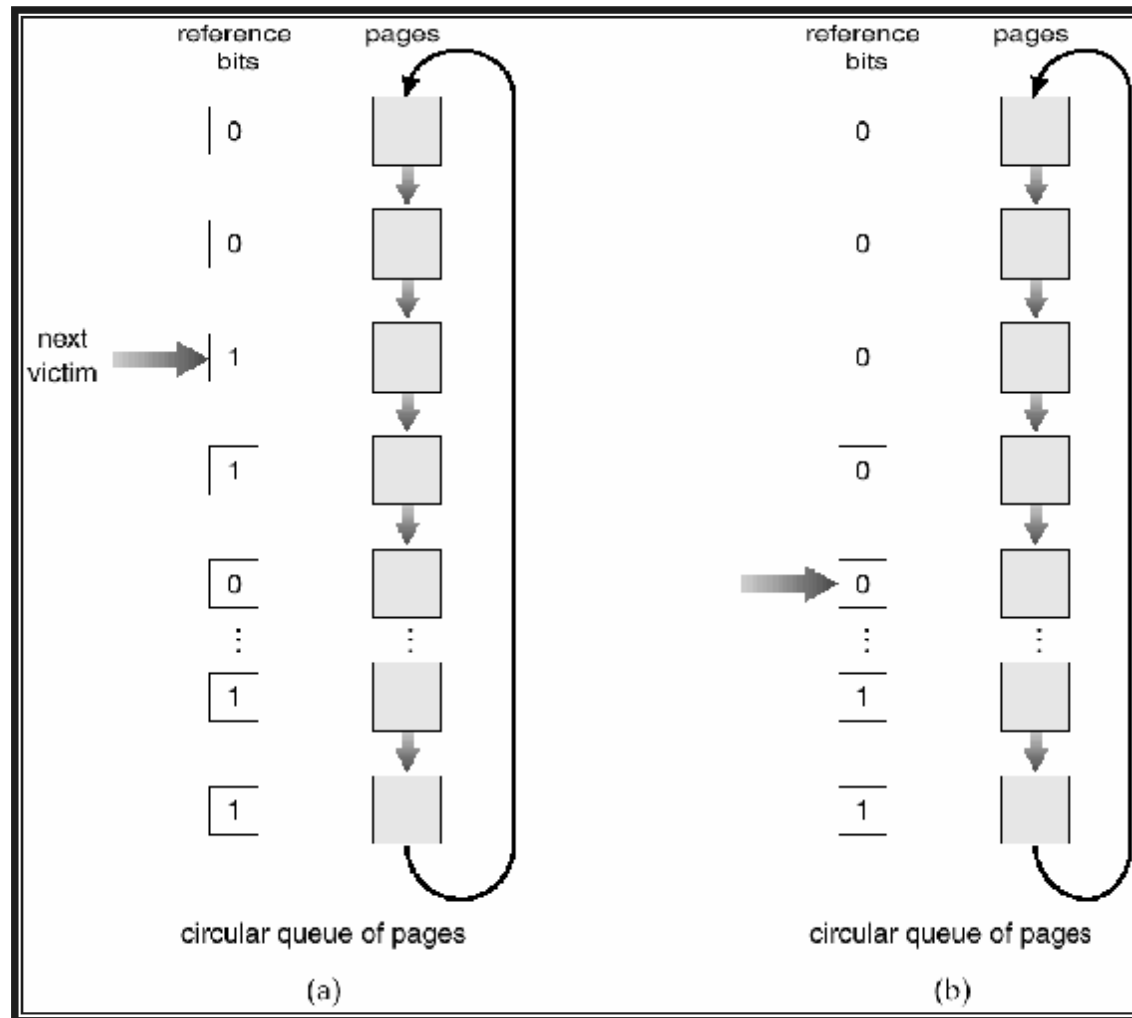
■ Bit di riferimento

- ◆ A ciascuna pagina si associa un bit, inizialmente = 0.
- ◆ Quando si fa riferimento alla pagina si pone il bit a 1.
- ◆ Si rimpiazza la pagina il cui bit=0 (se esiste). Tuttavia non si conosce l'ordine di accesso alle pagine.

■ Seconda Chance

- ◆ È necessario un bit di riferimento.
- ◆ Quando la pagina riceve una seconda chance il bit di riferimento viene azzerato ed il tempo di arrivo aggiornato al tempo attuale.
- ◆ Se la pagina da rimpiazzare (in ordine di clock) ha il bit di riferimento = 1, allora:
 - ✓ Si pone il bit di riferimento a 0.
 - ✓ Si lascia la pagina in memoria.
 - ✓ Si rimpiazza la pagina successiva (in ordine di clock), in base alle stesse regole.

Algoritmo di sostituzione delle pagine Seconda Chance



Algoritmi con conteggio

- Si tiene un contatore del numero di riferimenti che sono stati fatti a ciascuna pagina.
- Algoritmo LFU (*Least Frequently Used*): si rimpiazza la pagina col valore più basso del contatore.
- Algoritmo MFU (*Most Frequently Used*): è basato sulla assunzione che la pagina col valore più basso del contatore è stata spostata recentemente in memoria e quindi deve ancora essere impiegata.
- L'implementazione è molto costosa; le prestazioni (rispetto all'algoritmo ottimo) scadenti.

Allocazione dei frame

- Ciascun processo richiede un numero minimo di pagine.
- Esempio: IBM 370 — l'istruzione MVC (da memoria a memoria) può utilizzare fino a 6 frame:
 - ◆ L'istruzione occupa 6 byte, quindi può dividersi su 2 pagine.
 - ◆ 2 pagine per gestire **from**.
 - ◆ 2 pagine per gestire **to**.
- Si hanno due schemi principali di allocazione.
 - ◆ Allocazione fissa.
 - ◆ Allocazione con priorità.

Allocazione fissa

- Stessa allocazione per ogni processo — Esempio: se si hanno a disposizione 100 frame e 5 processi, si assegnano 20 pagine a ciascun processo.
- Allocazione proporzionale — Si allocano frame sulla base della dimensione del processo.

— s_i = size of process p_i

— $S = \sum s_i$

— m = total number of frames

— a_i = allocation for $p_i = \frac{s_i}{S} \times m$

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

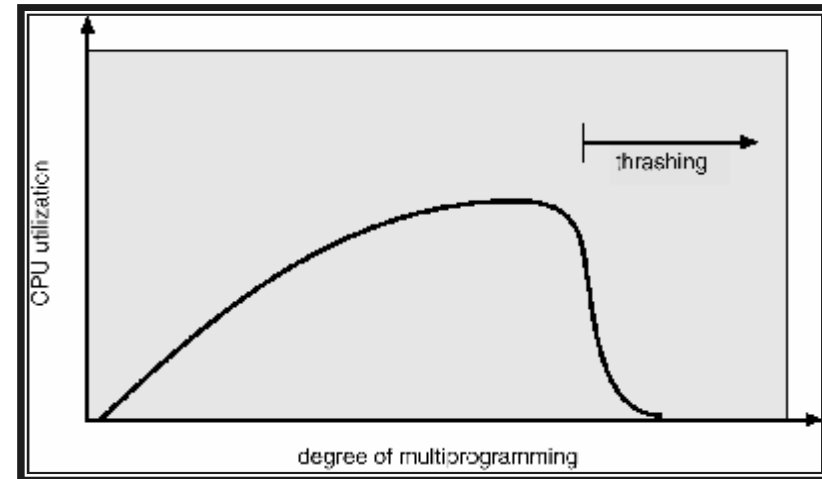
- SI impiega uno schema di allocazione proporzionale impiegando delle priorità piuttosto che la dimensione.
- Se il processo P_i genera un page fault,
 - ◆ Si seleziona per la sostituzione uno dei suoi frame;
 - ◆ Si seleziona per la sostituzione un frame di un processo con numero di priorità più basso.

Allocazione globale e locale

- Sostituzione **globale** — il processo seleziona il frame da sostituire dall'insieme di tutti i frame; il processo può selezionare un frame di un diverso processo.
 - ⇒ Un processo non può controllare la propria frequenza di page fault.
- Sostituzione **locale** — ciascun processo seleziona i frame solo dal proprio insieme di frame allocati.
 - ⇒ Non rende disponibili a processi che ne facciano richiesta pagine di altri processi scarsamente utilizzate.
- La sostituzione globale garantisce maggior throughput.
 - ⇒ Più utilizzata nei SO più diffusi.

Thrashing

- Se un processo non ha abbastanza pagine, la frequenza di page fault è molto alta:
 - ◆ basso impiego della CPU;
 - ◆ il SO crede di dover aumentare il grado di multiprogrammazione;
 - ◆ si aggiunge un altro processo al sistema!
- **Thrashing** \equiv un processo è costantemente occupato a spostare pagine dal disco alla memoria e viceversa.
- Perché la paginazione funziona?
Modello di località:
 - ◆ Una località è un insieme di pagine che vengono usate attivamente insieme.
 - ◆ Il processo passa da una località ad un'altra.
 - ◆ Le località possono essere sovrapposte.
- Perché avviene il thrashing?
dim. località > dim. memoria (disponibile per il processo).

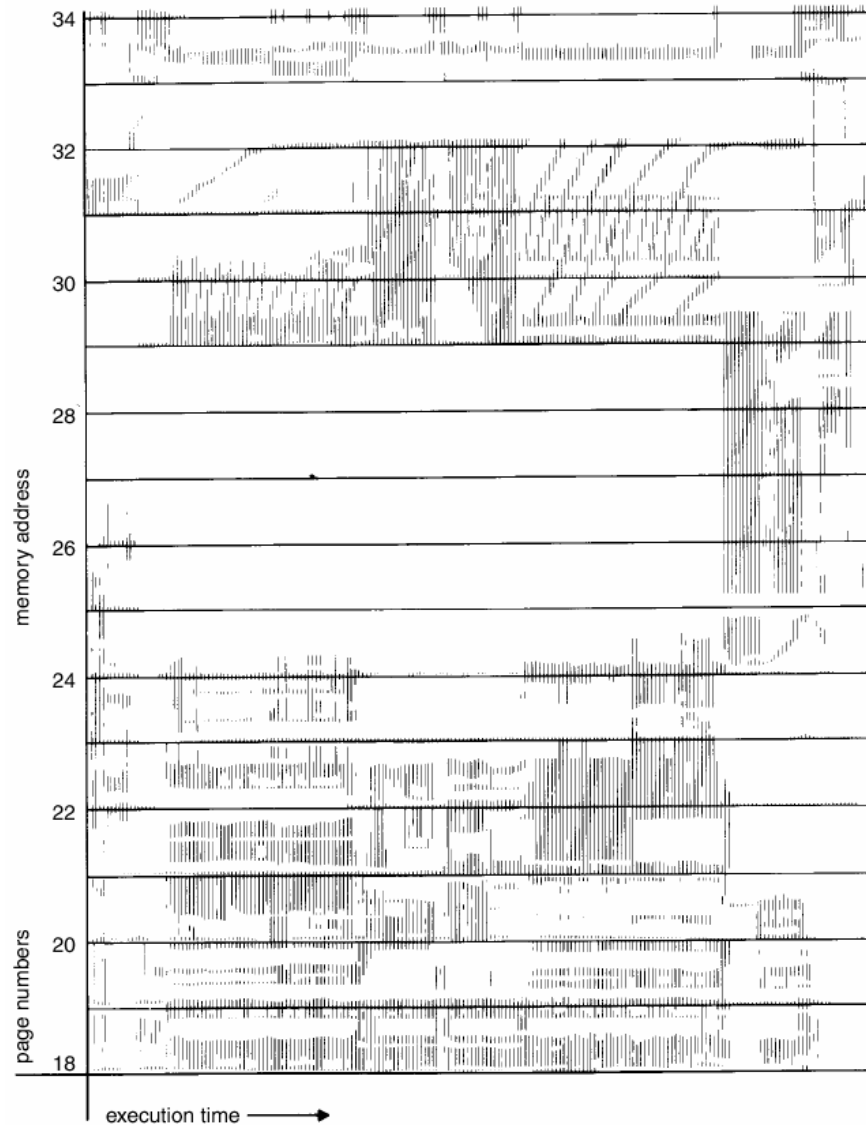


Località in una sequenza di riferimenti a memoria

- Le località hanno una connotazione spazio-temporale.

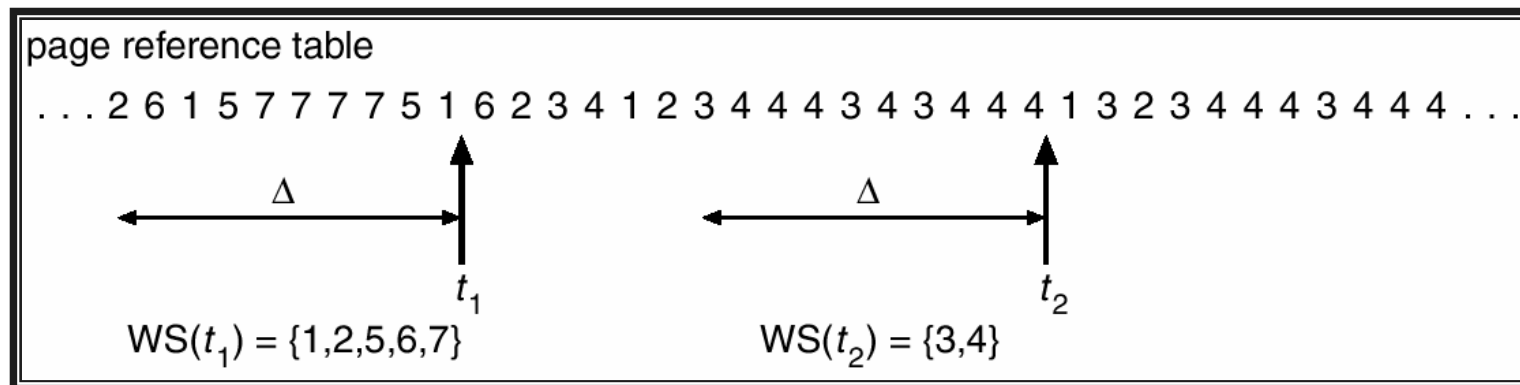
Esempio

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



Modello Working-Set

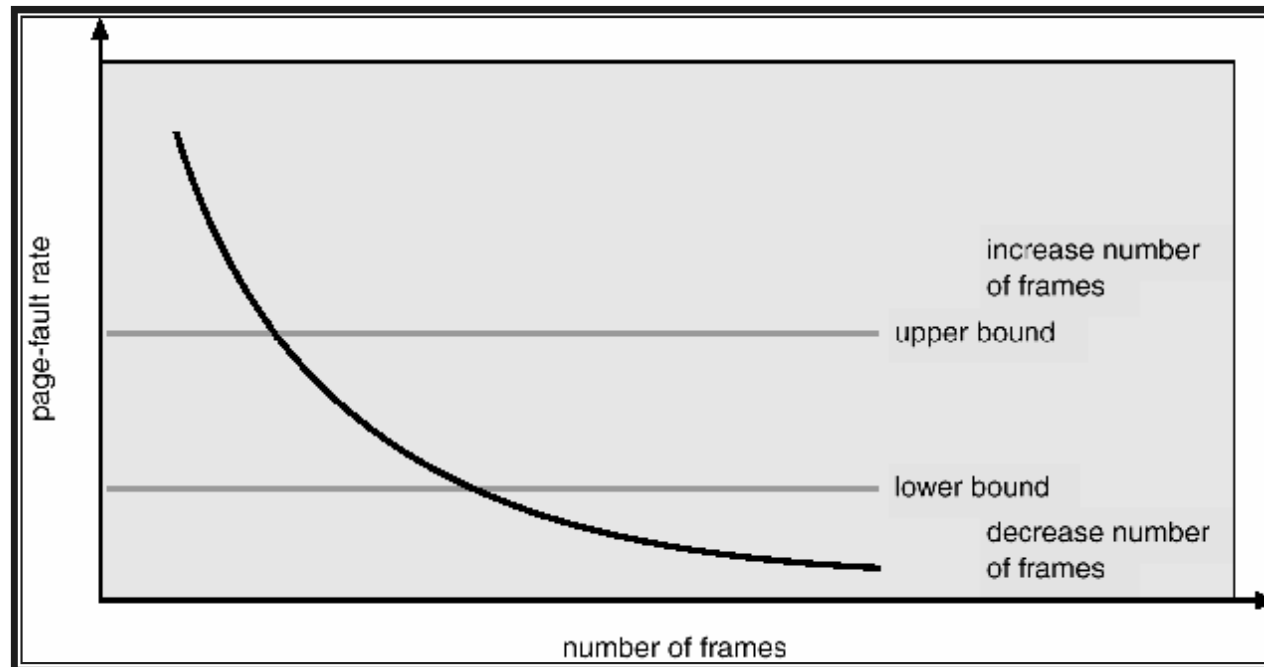
- $\Delta \equiv$ finestra di working-set \equiv un numero fisso di riferimenti a pagina; esempio: 10.000 istruzioni.
- WSS_i (working-set del processo P_i) = numero di pagine referenziate nel più recente Δ (varia col tempo):
 - ◆ se Δ è troppo piccolo non comprende tutta la località.
 - ◆ se Δ è troppo grande comprenderà più località.
 - ◆ se $\Delta = \infty \Rightarrow$ comprende l'intero programma.
- $D = \sum WSS_i \equiv$ numero totale di frame richiesti
- se $D > m$ (numero totale dei frame) \Rightarrow Thrashing
- Politica: se $D > m$, occorre sospendere un processo.



Modello Working-Set

- Problema: la finestra del working-set è una finestra in movimento, con riferimenti che entrano ed escono dal working-set.
- Si approssima con un interrupt del timer e un bit di riferimento
- Esempio: $\Delta = 10.000$
 - ◆ Il timer emette un interrupt ogni 5000 unità di tempo.
 - ◆ Si tengono in memoria 2 bit per ogni pagina.
 - ◆ Quando si ha un interrupt del timer, si copiano i valori di tutti i bit di riferimento e si pongono a 0.
 - ◆ Se uno dei bit in memoria è 1 \Rightarrow pagina nel working-set.
- Questo approccio non è completamente accurato.
- Miglioramento: 10 bit e interrupt ogni 1000 unità di tempo.

Frequenza di page fault



- Si stabilisce una frequenza di page fault “accettabile”.
 - ◆ Se la frequenza effettiva è troppo bassa, il processo rilascia dei frame.
 - ◆ Se la frequenza è troppo elevata, il processo acquisisce nuovi frame.

Altre considerazioni

- **Prepaginazione:** Si portano in memoria tutte le pagine richieste in una volta sola. Ad esempio si può memorizzare il working-set al momento della sospensione per I/O per poi riprendere tutte le pagine che gli appartengono.
- Criteri per la determinazione della **dimensione delle pagine:**
 - ◆ frammentazione
 - ◆ località
 - ✓ ⇒ pagine di piccole dimensioni
 - ◆ dimensione della tabella delle pagine
 - ◆ sovraccarico (overhead) di I/O
 - ✓ ⇒ pagine di grandi dimensioni

Altre considerazioni

■ Struttura dei programmi

◆ `A[][] = new int[1024][1024];`

◆ Ciascuna riga viene memorizzata in una pagina

◆ Programma 1

```
for (j = 0; j < A.length; j++)  
    for (i = 0; i < A.length; i++)  
        A[i,j] = 0;
```

1024 x 1024 page fault

◆ Programma 2

```
for (i = 0; i < A.length; i++)  
    for (j = 0; j < A.length; j++)  
        A[i,j] = 0;
```

1024 page fault

Altre considerazioni

- **I/O Interlock** — Talvolta, occorre permettere ad alcune pagine di rimanere *bloccate* in memoria.
- Esempio: Le pagine utilizzate per copiare un file da un device di I/O devono essere bloccate, affinché non possano essere selezionate come vittime da un algoritmo di sostituzione.

