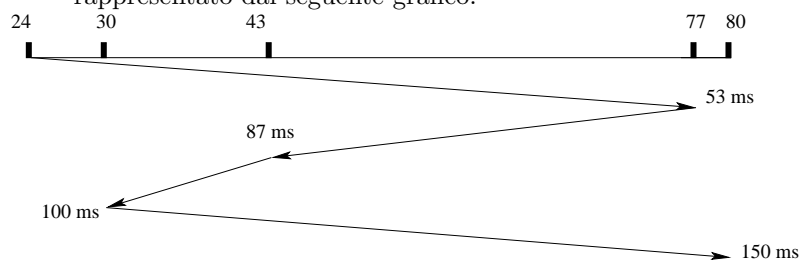


1. Si consideri un disco gestito con politica LOOK. Inizialmente la testina è posizionata sul cilindro 24, ascendente; lo spostamento ad una traccia adiacente richiede 1 ms. Al driver di tale disco arrivano richieste per i cilindri 77, 43, 30, 80, rispettivamente agli istanti 0 ms, 20 ms, 50 ms, 55 ms. Si trascuri il tempo di latenza.
 - (a) In quale ordine vengono servite le richieste?
 - (b) Il tempo di attesa di una richiesta è il tempo che intercorre dal momento in cui è sottoposta al driver a quando viene effettivamente servita. Qual è il tempo di attesa medio per le quattro richieste in oggetto?
 - (c) Se invece della politica LOOK, si considera la politica SSTF, in quale ordine vengono servite le richieste? Qual è il tempo di attesa medio in questo caso?

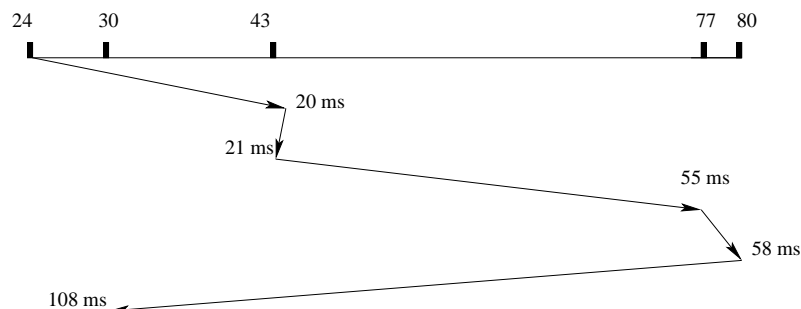
Soluzione:

- (a) l'ordine in cui vengono servite le richieste con la politica LOOK è rappresentato dal seguente grafico:



All'istante 0, la testina inizia a muoversi alla velocità di 1 traccia/ms verso il cilindro 77. Dopo 20ms, quando arriva la richiesta per il cilindro 43, la testina si trova già oltre, sul cilindro 44, la direzione è ascendente e quindi non viene servita. Stesso discorso per l'istante 50, quando arriva la richiesta per il cilindro 30. La testina continua il movimento verso il cilindro 77, ove giunge all'istante $77 - 24 = 53$ ms. Dopo aver servito questa richiesta, ci sono le due richieste a 30 e 43 in sospenso; la testina inverte quindi la direzione, verso la traccia 30. Quando, dopo 2ms, arriva la richiesta per il cilindro 80, è troppo tardi: la testina continua a scendere fino al cilindro 43 (che viene raggiunto dopo $77 - 43 = 34$ ms, ossia all'istante $53 + 34 = 87$ ms) e poi al cilindro 30 (che viene raggiunto dopo altri $43 - 30 = 13$ ms, ossia all'istante $87 + 13 = 100$ ms). Infine, viene invertita nuovamente la direzione per raggiungere il cilindro 80 dopo $80 - 30 = 50$ ms, ossia all'istante $100 + 50 = 150$ ms. Quindi l'ordine è: 77, 43, 30, 80.

- (b) I tempi di attesa per le quattro richieste sono rispettivamente: per il cilindro 77: $53 - 0 = 53$ ms; per il cilindro 43: $87 - 20 = 67$ ms; per il cilindro 30: $100 - 50 = 50$ ms; per il cilindro 80: $150 - 55 = 95$ ms. La media è $\frac{53 + 67 + 50 + 95}{4} = 66,25$ ms.
- (c) Nel caso della politica SSTF l'ordine in cui vengono servite le richieste è rappresentato dal seguente grafico:



All'istante 0 la testina inizia a muoversi verso il cilindro 77; dopo 20ms, quando arriva la richiesta per il cilindro 43, la testina si trova sul cilindro 44 ed inverte la direzione (siccome il cilindro 43 è più vicino del cilindro 77). Quindi dopo 21ms viene servita la richiesta per il cilindro 43. La testina riprende poi a muoversi verso il cilindro 77: quando, dopo altri 29ms, arriva la richiesta per il cilindro 30, la testina si trova sul cilindro 72 e quindi prosegue verso il cilindro 77 (in quanto più vicino del cilindro 30). Quindi la richiesta per il cilindro 77 viene servita all'istante 55ms. A questo punto arriva anche la richiesta per il cilindro 80 che viene quindi servita all'istante 58ms. Infine la testina inverte la direzione per andare a servire l'ultima richiesta rimasta, ovvero, quella per il cilindro 30 (richiesta servita all'istante 108ms). Quindi l'ordine è: 43, 77, 80, 30. I tempi di attesa per le quattro richieste sono rispettivamente: per il cilindro 43: $21-20=1$ ms; per il cilindro 77: $55-0=55$ ms; per il cilindro 80: $58-55=3$ ms; per il cilindro 30: $108-50 = 58$ ms. La media è $\frac{1+55+3+58}{4} = 29,25$ ms.

2. Un computer ha quattro frame, i cui istanti di caricamento, di ultimo riferimento e i reference bit sono riportati nella seguente tabella:

Frame	Caric.	Rifer.	R
2	135	287	1
1	240	250	1
0	169	253	0
3	203	266	1

Dire quale pagina verrebbe liberata dall'algoritmo FIFO, da LRU e da CLOCK (in questo caso si supponga che l'ultimo frame controllato sia il 3).

Soluzione: L'algoritmo FIFO, in base ai dati della tabella, avrà la coda 2, 0, 3, 1 e quindi sceglierà la pagina nel frame 2. LRU invece avrà la pila 2, 3, 0, 1, dove la pagina del frame 1 sarà quella riferita meno recentemente e quindi quella liberata dall'algoritmo. Infine CLOCK avrà la stessa coda di FIFO, ovvero, 2, 0, 3, 1 ma la gestirà in modo circolare, dando una seconda chance alle pagine con reference bit impostato a 1. Quindi (siccome viene detto che l'ultimo frame controllato è stato il 3) viene controllato dapprima il frame 1: siccome il reference bit è 1, quest'ultimo viene impostato a 0 e la pagina del frame risparmiata. Il controllo prosegue quindi con il frame 2 che, avendo il reference bit impostato a 1, viene saltato (impostando il reference bit a 0). Si giunge così al frame 0 che, avendo il reference bit

impostato a 0, viene scelto dall'algoritmo e la pagina che contiene viene scelta per la sostituzione.

- Supponendo di avere un sistema con quattro frame e otto pagine, adottando una politica di rimpiazzamento FIFO, quanti page fault si verificheranno con la reference string seguente?

0 1 7 2 3 2 7 1 0 3

(Si assuma che i quattro frame siano inizialmente vuoti.) Ripetere il problema per LRU.

Soluzione: inizialmente FIFO caricherà nei quattro frame le pagine 0, 1, 7, 2 provocando quattro page fault. Poi scaricherà la pagina 0 e caricherà la pagina 3 (1 page fault) originando la coda 1, 7, 2, 3. I successivi riferimenti alle pagine 2, 7, 1 non provocheranno ulteriori page fault, lasciando la coda invariata. Il riferimento a 0 invece provocherà 1 page fault scaricando la pagina 1 e generando la coda 7, 2, 3, 0. Infine, l'ultimo riferimento alla pagina 3 non induce page fault, in quanto già presente in memoria. Quindi in totale si hanno 6 page fault.

Simuliamo ora il funzionamento di LRU sulla reference string data:

	0	1	7	2	3	2	7	1	0	3
		0	1	7	2	3	2	7	1	0
			0	1	7	7	3	2	7	1
				0	1	1	1	3	2	7
					0	0	0	0	3	2

P P P P P P P

Abbiamo quindi 7 page fault per LRU.

- Si consideri un sistema con un controller con I/O guidato da interrupt, ma senza DMA, su bus PCI (che trasporta parole di 4 byte a 66,6 MHz), con un buffer interno di 4 Kbyte. Se la gestione di ogni interrupt costa $2 \mu\text{sec}$ e un accesso in RAM (a parole di 4 byte) costa 10 nsec, quanto si impiega per gestire l'input di 4 Kbyte? Qual è la banda massima (in MB/sec) di input sostenibile da questo sistema?

Soluzione: Per leggere una parola di 4 byte (32 bit) dal controller alla CPU ci si mette $1/(66,6 * 10^6) = 15 \text{ nsec}$; per trasferire tale parola alla RAM servono altri 10 nsec. Quindi per trasferire $4KB=1024$ parole servono $(10 + 15) * 1024 = 25600 \text{ nsec} = 25,6 \mu\text{sec}$. A questo bisogna aggiungere $2 \mu\text{sec}$ per l'interrupt, per un totale di $27,6 \mu\text{sec}$. La banda massima $4KB/27,6 \mu\text{sec} = 144927KB/sec = 141,5 \text{ Mbyte/sec}$.

- Si consideri un sistema con memoria paginata ad un livello, la cui page table sia mantenuta in memoria principale. Il tempo di accesso alla memoria principale sia $t = 50 \text{ ns}$.

(a) Qual è il tempo effettivo di accesso alla memoria?

- (b) Aggiungendo un TLB, con tempo di accesso $\epsilon = 1ns$, quale hit rate dobbiamo avere per un degrado delle prestazioni del 5% rispetto a t ?
- (c) E con una paginazione a due livelli?

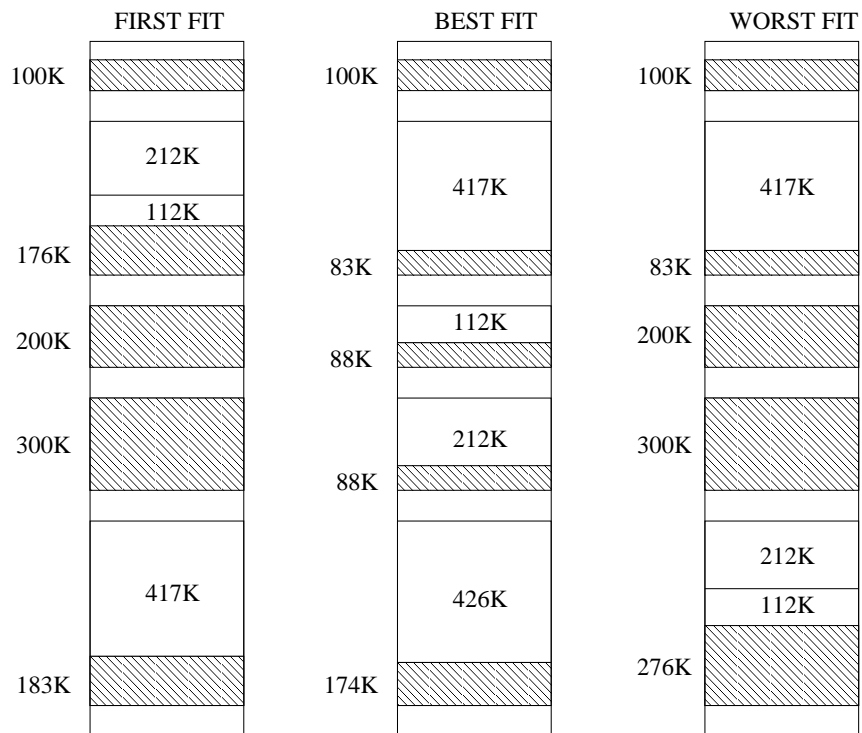
Soluzione:

- (a) $EAT = 2t = 100ns$ (50ns per recuperare il frame number, 50ns per l'indirizzo reale)
- (b) Un degrado del 5% significa che $EAT = 1,05 * 50 = 52,5ns$. Ricordando che $EAT = \epsilon + \alpha t + (1 - \alpha)(2t)$, abbiamo che $EAT = \epsilon + 2t - \alpha t$, e quindi $\alpha = \frac{2t + \epsilon - EAT}{t} = \frac{100 + 1 - 52,5}{50} = 0,97$, ossia un hit rate del 97%.
- (c) $EAT = 3t = 150ns$ (50ns per l'accesso alla page table esterna, 50ns per l'accesso alla page table interna e 50ns per l'accesso al dato in memoria). Un degrado del 5% significa che $EAT = 1,05 * 50 = 52,5ns$. Ricordando che $EAT = \epsilon + \alpha t + (1 - \alpha)(3t)$, abbiamo che $EAT = \epsilon + 3t - \alpha 2t$, e quindi $\alpha = \frac{3t + \epsilon - EAT}{2t} = \frac{150 + 1 - 52,5}{100} = 0,985$, ossia un hit rate del 98,5%.
6. Si consideri un sistema con quattro processi allocati in memoria principale. Tenendo conto del fatto che tali processi sono in attesa di eventi di I/O per metà del loro tempo, calcolare la percentuale di tempo di CPU "spreco".

Soluzione: sia $p = 0,5$ la probabilità che un processo sia in attesa di eventi di I/O. Supponendo che i quattro processi in esecuzione siano indipendenti, la probabilità che siano tutti contemporaneamente in attesa di eventi di I/O sarà il prodotto delle singole probabilità. Tale evento implica ovviamente lo "spreco" del tempo di CPU, in quanto in tale occasione la CPU non può essere allocata a nessun altro processo (essendoci per ipotesi soltanto quattro processi in esecuzione). Quindi lo spreco ammonta a $(0,5)^4 = 0,0625$, ovvero, a circa il 6% del tempo totale.

7. Date le zone di memoria libera di 100K, 500K, 200K, 300K e 600K (in quest'ordine), descrivere come gli algoritmi First-fit, Best-fit e Worst-fit allocherebbero processi delle dimensioni di 212K, 417K, 112K e 426K (in quest'ordine). Quale dei tre algoritmi utilizza la memoria nel modo più efficiente?

Soluzione: i tre algoritmi allocheranno i processi dando origine alle configurazioni di memoria seguenti (le zone tratteggiate rappresentano i "buchi" residui in seguito alle allocazioni operate):



Sia il First-Fit che il Worst-Fit non riescono ad allocare il quarto processo (dimensione: 426K). Quindi l'algoritmo che in questo caso gestisce meglio le richieste è il Best-Fit.

8. Si consideri uno spazio di indirizzamento logico di otto pagine di 1024 parole ognuna, mappate su una memoria fisica di 32 frame.

- Da quanti bit è costituito l'indirizzo logico?
- Da quanti bit è costituito l'indirizzo fisico?

Soluzione: siccome $8 = 2^3$, $1024 = 2^{10}$ e $32 = 2^5$, si ha:

- indirizzo logico: 3 (n. di pagina) + 10 (offset nella pagina) = 13 bit,
- indirizzo fisico: 5 (n. del frame) + 10 (offset nel frame) = 15 bit.

9. Si consideri la seguente *segment table*:

Segmento	Base	Lunghezza
0	219	600
1	2300	14
2	90	100
3	1327	580
4	1952	96

Quali sono gli indirizzi fisici corrispondenti ai seguenti indirizzi logici?

- 0, 430
- 1, 10
- 2, 500

- 3, 400
- 4, 112

Soluzione:

- 0, 430 corrisponde all'indirizzo fisico $219+430=649$ (indirizzo valido dato che $430 < 600$);
- 1, 10 corrisponde all'indirizzo fisico $2300+10=2310$ (indirizzo valido dato che $10 < 14$);
- 2, 500 non corrisponde ad un indirizzo fisico valido dato che $500 > 100$;
- 3, 400 corrisponde all'indirizzo fisico $1327+400=1727$ (indirizzo valido dato che $400 < 580$);
- 4, 112 non corrisponde ad un indirizzo valido dato che $112 > 96$.

10. Si consideri un processo con text size = 2M, data size = 500K, stack size = 200K, process control block = 5K. La trap al kernel e ritorno impiega $1\mu\text{sec}$, e la CPU copia una parola di 4 byte, tra due locazioni di memoria, in 10 nsec.

- Si dia una stima del tempo impiegato da una `fork()`, nel caso in cui il sistema operativo adotti la condivisione del codice ma non il copy-on-write.
- Come sopra, ma con copy-on-write.
- Per ottimizzare la memoria, conviene usare il copy-on-write sui segmenti o sulle pagine?

Soluzione:

- La quantità di memoria da copiare $500+200+5=705\text{K}$, ossia 180.480 parole, per complessivi 1,8 msec, a cui bisogna aggiungere $1\mu\text{sec}$ per la system call (che non cambia sostanzialmente il valore).
- In questo caso, si tratta solo di copiare i 5K del PCB, quindi $1280*10 = 12800$ nsec, a cui si aggiunge $1\mu\text{sec}$ per la system call, per un totale di 13,8 μsec .
- Sulle pagine, così basta allocare memoria soltanto per le pagine modificate e non gli interi segmenti. Questo può tuttavia causare più fault, e quindi un maggiore overhead per il meccanismo di trap al kernel.

11. (a) Si descriva brevemente il concetto di *working set* $WS(t, \Delta)$, all'istante t con intervallo Δ .
- (b) Si consideri la seguente stringa di riferimenti (partendo con $t = 0$):

2 6 5 7 7 7 7 5 1 6 4

Cosa è $WS(10, 8)$, ossia dopo l'ultimo accesso?

- Nel precedente esempio, quanti page fault ci sono complessivamente con $\Delta = 4$ (supponendo che in ogni istante si mantenga in memoria esattamente il solo working set)?

Soluzione:

- (a) Il working set è un'approssimazione della località del processo, ossia è l'insieme di pagine "attualmente" riferite. In generale $WS(t, \Delta)$ = insieme delle pagine riferite negli accessi $[(t - \Delta + 1), t]$.
- (b) $WS(10, 8) = \{1, 4, 5, 6, 7\}$
- (c) 8 page fault. Basta fare la matrice seguente, facendo attenzione a togliere le pagine man mano che escono dal working set, e segnando fault quando bisogna (ri)mettercele:

2	6	5	7	7	7	7	5	1	6	4
2	6	5	7	7	7	7	5	1	6	4
		2	6	5	5		7	5	1	6
			2	6	6			7	5	1
			2					7	5	
p	p	p	p				p	p	p	p

12. Si consideri un file system Unix-like (UFS o EXT2) con blocchi da 4K, su un disco con $t_{seek} = 10ms$, a 7200 RPM. In tale file system, sia stato aperto un file i cui blocchi siano sulla stessa traccia del rispettivo inode.
- (a) Quanto si impiega per accedere direttamente alla posizione 10000 del file?
- (b) e alla posizione 100000?

Soluzione:

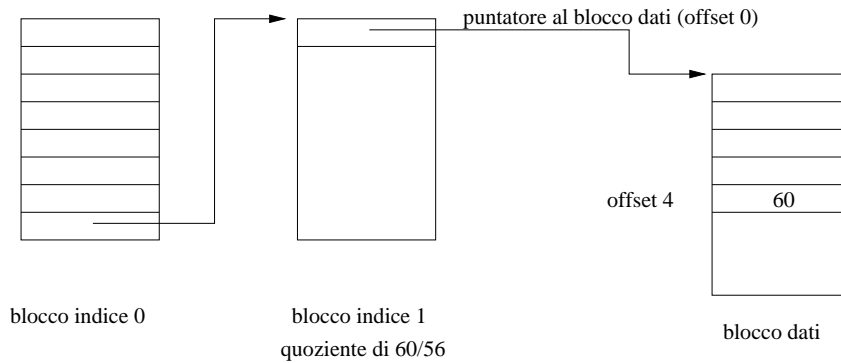
- (a) La posizione 10000 cade nel terzo blocco, che è uno dei blocchi diretti. Per cui basta 1 accesso al disco (l'inode è già stato caricato in memoria al momento dell'apertura), che costa $t_{seek} + t_{latenza} = 10 + 60/(2 * 7, 2) = 14, 17msec$.
- (b) La posizione 100000 cade in uno dei primi indiretti, per cui è necessario accedere 2 volte al disco (una volta anche per il blocco indiretto), dove però il tempo di seek si conta una volta sola perché i blocchi sono sulla stessa traccia. In totale $10 + 4, 17 + 4, 17 = 18, 34msec$.
13. Si consideri un'implementazione di un filesystem con allocazione indicizzata con indice concatenato in cui:
- la dimensione di un blocco sia di 8 byte;
 - un indirizzo richieda un byte per la sua memorizzazione.

Si calcoli l'indirizzo fisico corrispondente all'indirizzo logico 60 per un dato file.

Soluzione: prima di tutto bisogna determinare il blocco indice da consultare:

$$60 / (8 \times 7) = 60 / 56$$

il quoziente 1 rappresenta il blocco indice corretto, mentre il resto 4 viene ulteriormente diviso per la dimensione del blocco (8) fornendo il quoziente 0 (offset all'interno del blocco indice 1) ed il resto 4 che rappresenta l'offset all'interno del blocco dati (puntato, per i calcoli appena fatti, dal primo byte del blocco indice 1):



14. Al driver di un disco arrivano, nell'ordine, richieste per i cilindri 10, 22, 24, 4, 39, 6, 40. Supponendo che la testina si trovi inizialmente sul cilindro 20, e che il tempo necessario per spostare la testina di 1 cilindro sia di 6ms, quanto tempo viene speso per lo spostamento della testina per servire tutte le richieste con la politica FCFS, con la politica SSTF, e con la politica LOOK (o dell'ascensore; supponete che la direzione iniziale sia ascendente)?

Soluzione: Per FCFS: lo spostamento totale è dato dalla somma delle differenze tra il cilindro attuale e il successivo, partendo dal cilindro 20. $(20 - 10) + (22 - 10) + (24 - 22) + (24 - 4) + (39 - 4) + (39 - 6) + (40 - 6) = 10 + 12 + 2 + 20 + 35 + 33 + 34 = 146$ cilindri = $876ms$

Per SSTF: lo spostamento totale è dato dalla somma delle differenze tra il cilindro attuale e il più vicino tra quelli rimanenti, partendo dal cilindro 20. $(22 - 20) + (24 - 22) + (24 - 10) + (10 - 6) + (6 - 4) + (39 - 4) + (40 - 39) = 2 + 2 + 14 + 4 + 2 + 35 + 1 = 60$ cilindri = $360ms$

Per LOOK: lo spostamento totale è dato dalla distanza tra il cilindro attuale (20) e il massimo nella direzione attuale (40), più la distanza da tale massimo al cilindro minimo (4): $(40 - 20) + (40 - 4) = 56$ cilindri = $336ms$

15. Si consideri un processo che generi la seguente stringa di riferimenti alle pagine virtuali:

0 1 2 0 1 4 5 0 2 3

- (a) Se il processo ha 3 frame, gestiti con LRU, quanti page fault vengono generati?
 (b) Qual è il numero minimo di frame necessario per minimizzare i page fault?

Soluzione:

- (a) Si generano 8 page fault:

	0	1	2	0	1	4	5	0	2	3
		0	1	2	0	1	4	5	0	2
			0	1	2	0	1	4	5	0
						2	0	1	4	5
							2	2	1	4
										1

P P P P P P P P

- (b) Il minimo è 6 page fault (perché il processo accede a 6 pagine). Per determinare il numero minimo di frame per avere solo 6 page fault, si può procedere empiricamente, ripetendo la simulazione aumentando via via il numero di frame disponibili e fermandosi non appena si riscontrano solo 6 page fault sulla reference string data. Oppure si può sfruttare la *distance string*, che nel caso in questione risulta essere la seguente:

$\infty \infty \infty 3 3 \infty \infty 4 5 \infty$

Si ricorda che la distance string rappresenta la distanza fra la posizione di una pagina nel modello e la prima posizione, ovvero, quella nella prima riga della matrice (contando anche la casella di partenza) nel momento in cui la pagina stessa viene riferita. Se una pagina non è presente nella matrice, allora la sua distanza, quando viene riferita è ∞ . Ad esempio nella simulazione della parte a) dell'esercizio, quando viene riferita inizialmente la pagina 0, la sua distanza è ∞ (dato che ancora non era presente nella matrice). Invece il secondo riferimento alla pagina 0 ha come distanza 3, in quanto la posizione precedentemente occupata dalla pagina 0 nella matrice si trova nella terza riga.

Indichiamo ora con C_i il numero di volte che il numero i compare nella distance string; nel caso in questione abbiamo: $C_1 = 0, C_2 = 0, C_3 = 2, C_4 = 1, C_5 = 1, C_\infty = 6$. Indicando poi con m il numero di frame e con n il numero più grande che compare nella distance string, indichiamo con $F_m = \sum_{k=m+1}^n C_k + C_\infty$ il numero di page fault che si verificano con m frame e con la reference string data. L'intuizione è la seguente: se ho a disposizione m frame i page fault saranno provocati dai riferimenti a pagine che "distano" almeno $m + 1$ dal top della matrice e dal numero di ∞ (ovvero da riferimenti a pagine non ancora presenti nel modello). Nel nostro caso abbiamo: $F_1 = 10, F_2 = 10, F_3 = 8, F_4 = 7, F_5 = 6$, quindi il numero minimo di frame che minimizza i page fault è 5.