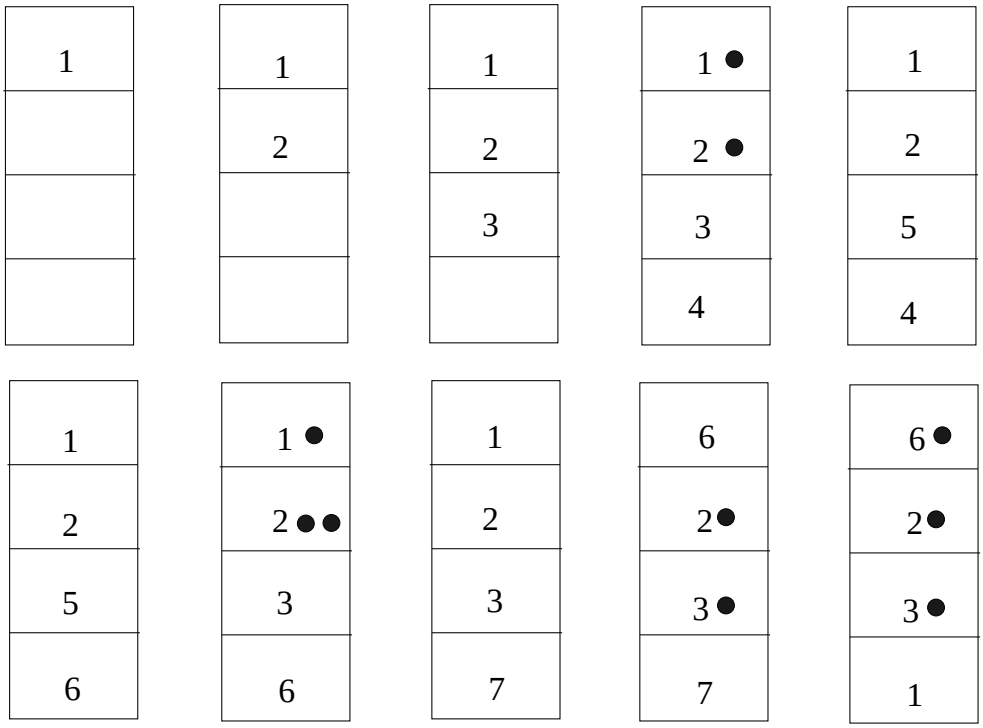
- **Esercizio 1**
  Facendo riferimento ad un ambiente di gestione della memoria virtuale con paginazione su richiesta, si consideri un processo caratterizzato dalla seguente stringa di riferimenti a pagina:

  1 2 3 4 2 1 5 6 2 1 2 3 7 6 3 2 1 2 3 6

  – Si illustri il comportamento dell'algoritmo LRU nel caso che al processo siano assegnati 4 blocchi fisici. Si calcoli il numero dei page–fault, supponendo un regime iniziale di paginazione pura.

  – Come supporto alla trasformazione degli indirizzi viene usata una tabella di registri associativi. L'accesso alla memoria centrale richiede 200 nanosecondi, la ricerca nella tabella associativa richiede 20 nanosecondi. Si calcoli il tempo medio di accesso ai dati, supponendo una hit–ratio del 75%.

  Soluzione
  Il comportamento dell'algoritmo LRU è il seguente:



  La figura evidenzia (tramite l'evoluzione del contenuto dei 4 blocchi fisici messi a disposizione del processo) che si sono verificati 10 page–fault (i pallini accanto al numero di pagina segnalano una richiesta della pagina che non provoca page–fault).
  Il tempo medio di accesso ai dati viene calcolato mediante la formula:

$$\begin{aligned} T_{medio} &= 0.75 \times (T_{ta} + T_{mc}) + 0.25 \times (T_{ta} + 2\,T_{mc}) \\ &= 0.75 \times 220 + 0.25 \times 420 = 270\,nsec \end{aligned}$$

  Si ha quindi un rallentamento del 35% nel tempo di accesso in memoria (da 200 a 270 nanosecondi).

- **Esercizio 2**

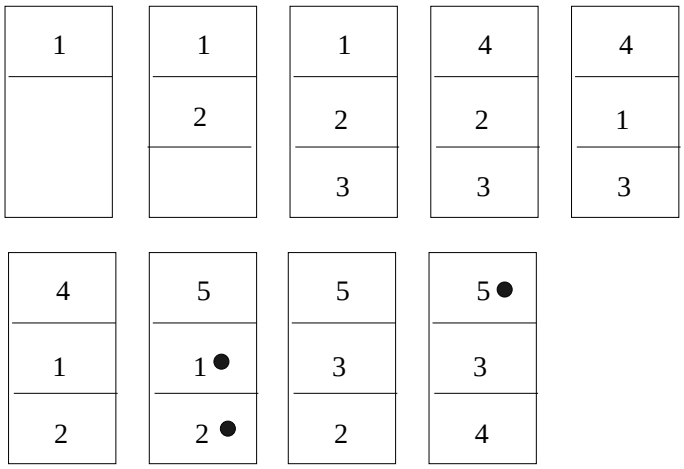
  Relativamente alla stringa di riferimenti alle pagine

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

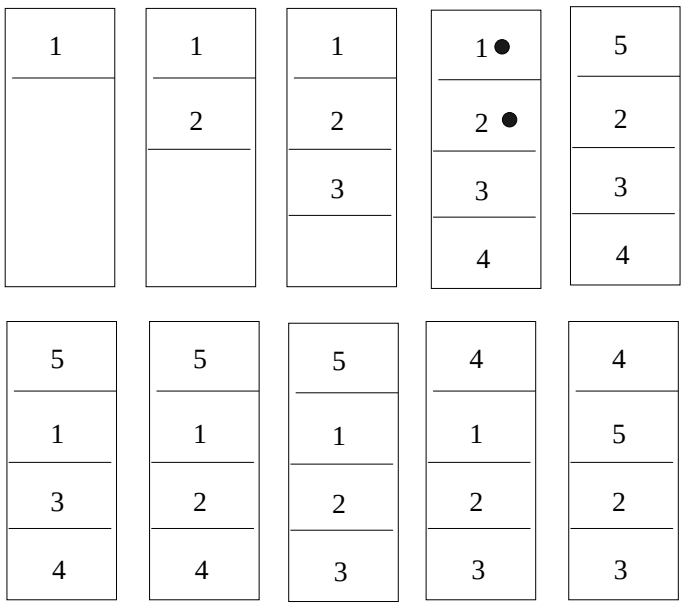
  si stabilisca quanti page–fault hanno luogo utilizzando una strategia di sostituzione FIFO (con paginazione pura), avendo a disposizione un numero di pagine fisiche rispettivamente pari a 3 e 4. Che cosa si è verificato?

  Soluzione

  Il comportamento dell'algoritmo FIFO con 3 frame è il seguente:



  Si hanno quindi 9 page–fault. Nel caso, invece, di 4 frame la situazione è la seguente:



  con 10 page–fault. Si evidenzia, in questo esempio, l'*anomalia di Belady*, per cui il numero di page–fault aumenta all'aumentare dei frame messi a disposizione per il processo.

- **Esercizio 3**

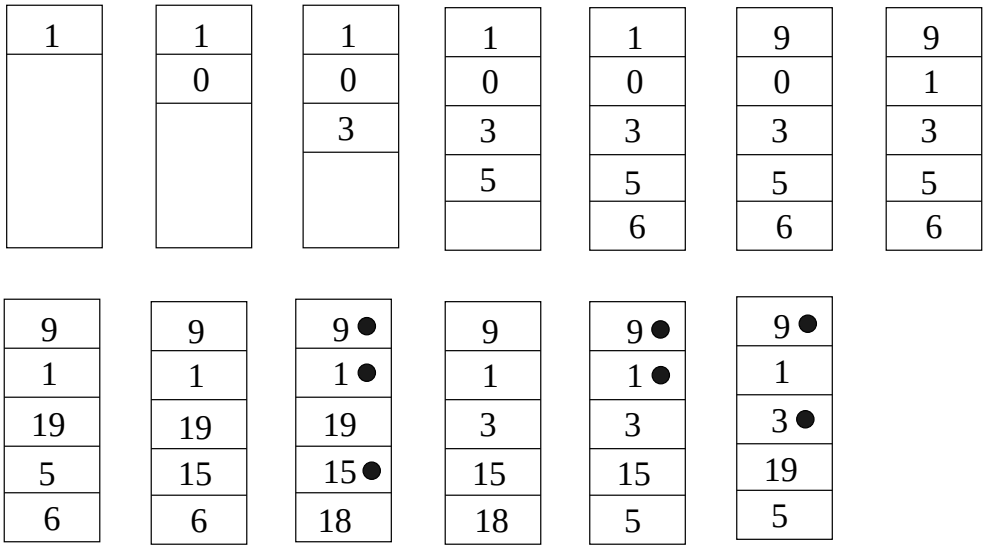
  Facendo riferimento ad un ambiente di gestione della memoria virtuale con paginazione su richiesta, si consideri un processo caratterizzato dalla seguente stringa di riferimenti a pagina:

  1 0 3 5 6 9 1 19 15 18 9 15 1 3 5 1 9 19 9 3.

  Si illustri il comportamento dell'algoritmo LRU nel caso che al processo siano assegnati 5 blocchi fisici. Si calcoli il numero di page–fault, supponendo un regime iniziale di paginazione pura.

  Soluzione

  Il comportamento dell'algoritmo LRU è il seguente:



  La figura evidenzia (tramite l'evoluzione del contenuto dei 5 blocchi fisici messi a disposizione del processo) che si sono verificati 13 page–fault (i pallini accanto al numero di pagina segnalano una richiesta della pagina che non provoca page–fault).

- **Esercizio 4**

  Si consideri il seguente frammento di programma in linguaggio C:

```
{
  ...
  int a[1000], b[1000];

  for (i=0; i<999; i++)
      a[i]=a[i]+b[i];
  ...
}
```

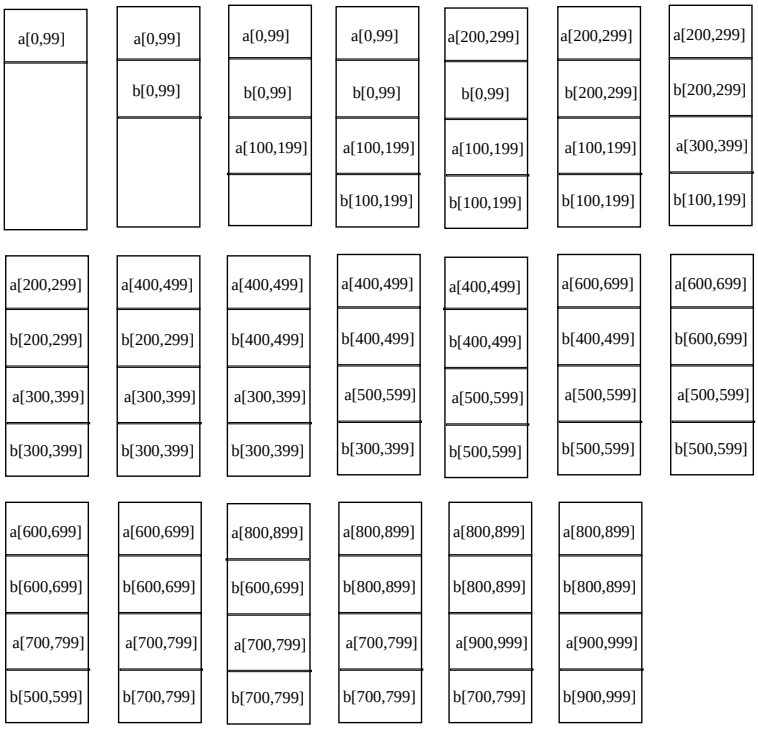
  Si supponga che un intero occupi una parola (2 byte) e si calcoli il numero di page–fault generati dal programma in ciascuno dei seguenti casi:

  1. Si possono allocare, in ogni momento, due pagine contenenti 100 parole per ognuno dei vettori a e b.
  2. Si possono allocare, in ogni momento, tre pagine contenenti 100 parole per ognuno dei vettori a e b.
  3. Si possono allocare, in ogni momento, due pagine contenenti 500 parole per ognuno dei vettori a e b.

Si utilizzi la strategia FIFO. Si assuma, infine, che il codice e la variabile `i` siano collocati in un'altra pagina e che nessun accesso a tali entità provochi un page–fault. La memoria principale è inizialmente vuota.

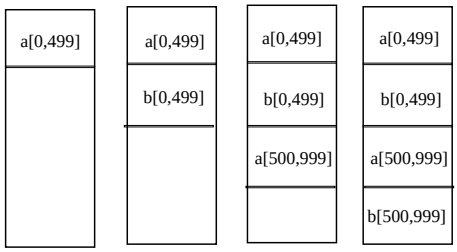
Soluzione

Nel caso in cui si abbiano due pagine contenenti 100 parole per ognuno dei vettori a e b, in regime di paginazione pura, si ha la situazione illustrata nella figura seguente:



con 20 page–fault. Situazione del tutto analoga si verifica quando si hanno, in ogni momento, tre pagine contenenti 100 parole per i vettori a e b.

Nel caso invece di pagine contenenti 500 parole, la situazione è la seguente:



con soli 4 page–fault per la sezione dati del programma sopra descritto.

- **Esercizio 5**

  Si consideri il seguente frammento di programma in linguaggio C:

```
{
  ...
  int a[100][100], b[100], c[100];

  for (i=0; i<99; i++)
      c[i]=0;
      for (j=0; j<99; j++)
          c[i]=c[i]+a[i][j]*b[j];
  ...
}
```

  Si supponga che un intero occupi una parola (2 byte) e si calcoli il numero di page–fault generati dal programma in ciascuno dei seguenti casi:

  1. La matrice `a` è memorizzata per colonne e si possono allocare, in ogni momento, tre pagine contenenti 100 parole.
  2. La matrice `a` è memorizzata per righe e si possono allocare, in ogni momento, tre pagine contenenti 100 parole.

  Si utilizzi la strategia OPT (si sostituisce la pagina che non verrà utilizzata per il periodo di tempo più lungo). Si assuma, infine, che il codice e le variabili `i` e `j` siano collocati in un'altra pagina e che nessun accesso a tali entità provochi un page–fault. La memoria principale è inizialmente vuota.

  Soluzione
  Nel primo caso, si eseguono 2 page–fault per portare in memoria i vettori `b` e `c` che restano residenti per tutta l'esecuzione. Invece, ogni volta che si esegue l'operazione di prodotto, viene sostituito il vettore colonna relativo alla matrice `a`. Di conseguenza, per calcolare il primo elemento del vettore `c` si fanno 102 page–fault, per calcolare il secondo elemento 100 page–fault, ..., per calcolare il centesimo ancora 100 page–fault, per un totale di 10002 page–fault. Se, viceversa, la matrice `a` è memorizzata per righe sono sufficienti 3 page–fault per calcolare il primo elemento di `c` e un solo page–fault per calcolare i successivi, per un totale di soli 102 page–fault.

- **Esercizio 6**

  Un file è composto da record di 512 byte ed è allocato su un disco caratterizzato da blocchi anch'essi di 512 byte. Si considerino le tre strategie di memorizzazione del file, contigua, concatenata e indicizzata, e si supponga che le informazioni che riguardano il file, in ognuno dei tre casi, siano già in memoria centrale. L'ultimo record letto dal file è il record 3, il prossimo record da leggere è il record 8. Per ognuno dei tre casi, si calcoli quanti accessi a disco sono necessari per la lettura del record 8 e si motivi la risposta.

  Soluzione
  Notiamo innanzitutto che, date le dimensione di record e blocco sopra specificate, ogni record del file occupa esattamente un blocco del disco. Pertanto...

  – nel caso di allocazione contigua si avrà un unico accesso alla memoria di massa che permetterà di reperire direttamente il record 8;
  – nel caso di allocazione concatenata occorreranno cinque accessi alla memoria, dato che i record che separano il terzo dall'ottavo devono comunque essere acceduti sequenzialmente;
  – nel caso di allocazione indicizzata occorreranno due accessi alla memoria, il primo al blocco indice, per reperire il puntatore al blocco di dati richiesto, e il secondo al blocco di dati.

- **Esercizio 7**
  Si consideri un file formato da 70 record e le sue possibili allocazioni su disco di tipo sequenziale, con puntatori e con tabella indice. In ognuno di questi casi i record del file sono memorizzati uno per blocco. Le informazioni che riguardano il file sono già in memoria centrale e la tabella indice è contenuta in un unico blocco. Si dica quanti accessi al disco (letture e/o scritture) sono necessari, in ognuna di queste situazioni, per cancellare

  1. il primo record
  2. il 40–esimo record
  3. l'ultimo record

  Soluzione

  - Nel caso di allocazione sequenziale l'operazione di cancellazione ha un costo indipendente dalla posizione del record nel file. Un unico accesso alla memoria è dunque necessario per accedere uno qualsiasi dei record del file.
  - Nel caso di allocazione concatenata, l'unica modalità di accesso ai record del file è quella sequenziale. Pertanto per cancellare il primo record occorre un unico accesso alla memoria, per cancellare il quarantesimo record occorrono quaranta accessi alla memoria e, infine, per accedere l'ultimo record ne occorrono settanta.
  - Nel caso di allocazione con tabella con indice con le caratteristiche sopra indicate, per reperire qualsiasi record del file occorrono un accesso al blocco contenente la tabella, più un accesso al blocco contenente il record richiesto. Di conseguenza si hanno due accessi alla memoria di massa per ciascun record.

- **Esercizio 8**
  Si consideri un file formato da 100 record e le sue possibili allocazioni su disco: di tipo contiguo, concatenato, e con tabella indice. In ognuno di questi casi i record del file sono memorizzati 5 per blocco (tutto l'indice è invece memorizzato in un solo blocco). Si supponga che le informazioni che riguardano il file siano, comunque, già presenti in memoria centrale.

  1. Si dica quanti accessi a disco sono necessari, in ognuna di queste situazioni, per aggiungere un record (costruito in memoria centrale) all'inizio, a metà ed alla fine del file (n.b. nel caso di allocazione contigua, si supponga che il blocco contiguo all'ultimo blocco del file sia libero).
  2. Si dica quanti accessi a disco sono necessari, in ognuna di queste situazioni, per leggere il record numero 18.

  Soluzione

  - Nel caso di allocazione contigua occorrono rispettivamente quarantuno, ventuno ed un accesso alla memoria per inserire un nuovo record all'inizio, alla metà ed alla fine del file. Il motivo di ciò risiede nel fatto che per posizionare correttamente un record in una locazione che non sia l'ultima, occorre spostare tutti i record che dovranno seguirlo (a partire dall'ultimo), e lo spostamento richiede un'operazione di lettura ed una di scrittura dei relativi blocchi. Viceversa per leggere un qualsiasi record (e quindi anche il numero 18) occorre un unico accesso alla memoria.
  - Nel caso di allocazione concatenata il costo dell'inserimento di un nuovo record è proporzionale alla posizione di quel record nel file (l'inserimento vero e proprio richiede solo un costo costante dovuto all'aggiornamento di due puntatori). Quindi si ha un accesso alla memoria per inserire un record all'inizio del file, undici accessi per l'inserimento nel mezzo del file, ventuno per l'inserimento in coda. Per reperire il diciottesimo record occorrono invece quattro accessi alla memoria di massa $\left(\left\lceil \frac{18}{5} \right\rceil\right)$.
  - Nel caso di allocazione indicizzata si hanno infine tre accessi alla memoria per l'inserimento di un record in una qualsiasi posizione (due per la lettura e l'aggiornamento dell'indice ed uno per l'inserimento vero e proprio), mentre occorrono due accessi (uno all'indice ed uno al blocco di dati) per reperire qualsiasi record (quindi anche il diciottesimo).

- **Esercizio 9**

  Si consideri un file formato da 80 blocchi, allocato su disco in modo sequenziale. Il blocco precedente a quelli occupati dal file è occupato, mentre sono liberi i due blocchi successivi. Si dica quante operazioni di I/O su disco sono necessarie per aggiungere un blocco all'inizio del file.

  Soluzione

  Per inserire un blocco all'inizio del file occorre scorrere tutti i blocchi attualmente allocati di una posizione, partendo dall'ultimo. Si deve cioè leggere l'ultimo blocco e riscriverlo nel primo blocco libero alla fine del file, leggere il penultimo blocco e riscriverlo sull'ultimo e così via. Una volta completata questa operazione occorrerà scrivere il nuovo blocco sul primo blocco del file. Conseguentemente il numero totale di operazioni di I/O su memoria di massa è 161 (corrispondente a 80 operazioni di lettura dei blocchi esistenti, più 81 operazioni di scrittura dei vecchi blocchi nelle nuove posizioni e del nuovo blocco inserito).

- **Esercizio 10**

  Gli *inode* di UNIX impiegano indirizzamento di blocchi diretto e indirizzamento indiretto singolo, doppio e triplo per localizzare i blocchi di un file fisico. Si supponga che ciascun indirizzo occupi quattro byte. Si calcoli la dimensione del più grande file che può essere gestito con questo schema di indirizzamento, per dimensioni dei blocchi di 512, 1024 e 4096 byte.

  Soluzione

  Nella seguente tabella sono riportate, in dipendenza della dimensione dei blocchi, e quindi del conseguente numero di puntatori contenuto in ciascun blocco, le dimensioni massime dei file indirizzabili. Nelle colonne dalla terza alla sesta, i numeri riportati rappresentano dimensioni in byte.

dim. blocco	no. puntatori per blocco	diretti	1–indir.	2–indir.	3–indir.	max dim. file
512	128	6144	65536	8388608	1073741824	1.01 Gbyte
1024	256	12288	262144	67108864	17179869184	16.06 Gbyite
4096	1024	49152	4194304	4294967296	4398046511104	410 Gbyte

- **Esercizio 11**

  Un floppy disk ha 40 cilindri. Un'operazione di ricerca richiede 6 $msec$ per lo spostamento tra un cilindro e l'altro, la latenza rotazionale media è di 10 $msec$ ed il tempo di trasferimento è di 25 $msec$ per blocco.

  a) Quanto tempo è necessario per leggere un file costituito da 20 blocchi e memorizzato in modo tale che blocchi logicamente contigui nel file distino mediamente 13 cilindri l'uno dall'altro sul dischetto?

  b) Quanto tempo è necessario per leggere un file con 100 blocchi mediamente distanti 2 cilindri?

  Soluzione

  Indipendentemente dal numero dei cilindri del disco...

  a) Il tempo medio per accedere e quindi trasferire in memoria principale un blocco del file è

  $$[(6 \times 13) + 10 + 25] msec,$$

  corrispondente alla somma del tempo medio di ricerca, tempo medio di latenza e tempo di trasferimento effettivo. Conseguentemente, il tempo totale necessario a trasferire tutto il file è

  $$20 \times [(6 \times 13) + 10 + 25] = 4080\, msec \approx 4\, sec$$

  .

  b) Il tempo totale per la lettura del file è

  $$100 \times [(6 \times 2) + 10 + 25] = 4700\, msec = 4.7\, sec$$

  .

- **Esercizio 12**

  Al driver di un disco arrivano, nell'ordine, richieste per i cilindri 10, 22, 20, 2, 40, 6 e 38. Uno spostamento da una traccia a quella adiacente richiede 6 $msec$. Si stabilisca quanto tempo è necessario per servire le richieste con:

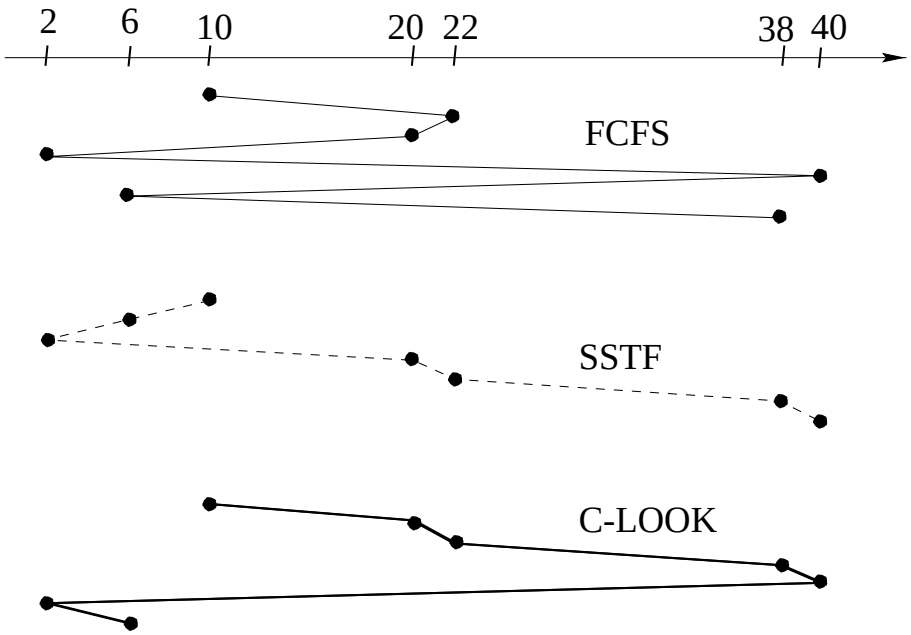
  – la politica FCFS,

  – la politica SSTF,

  – la politica C–LOOK (crescente).

  Si assuma, per tutti i casi, che il braccio si trovi inizialmente posizionato sul cilindro 10.

  Soluzione



$$
\begin{aligned}
t_{FCFS} &= 6 \times (12 + 2 + 18 + 38 + 34 + 32) msec = 816\, msec \\
t_{SSTF} &= 6 \times (4 + 4 + 18 + 2 + 16 + 2) msec = 276\, msec \\
t_{C-LOOK} &= 6 \times (10 + 2 + 16 + 2 + 38 + 4) msec = 432\, msec
\end{aligned}
$$

Si noti come, così come ci si aspetta, la tecnica migliore risulta SSTF, che però è effettivamente applicabile solo nel caso in cui vi sia una lista bufferizzata di richieste di accesso al disco.

- **Esercizio 13**

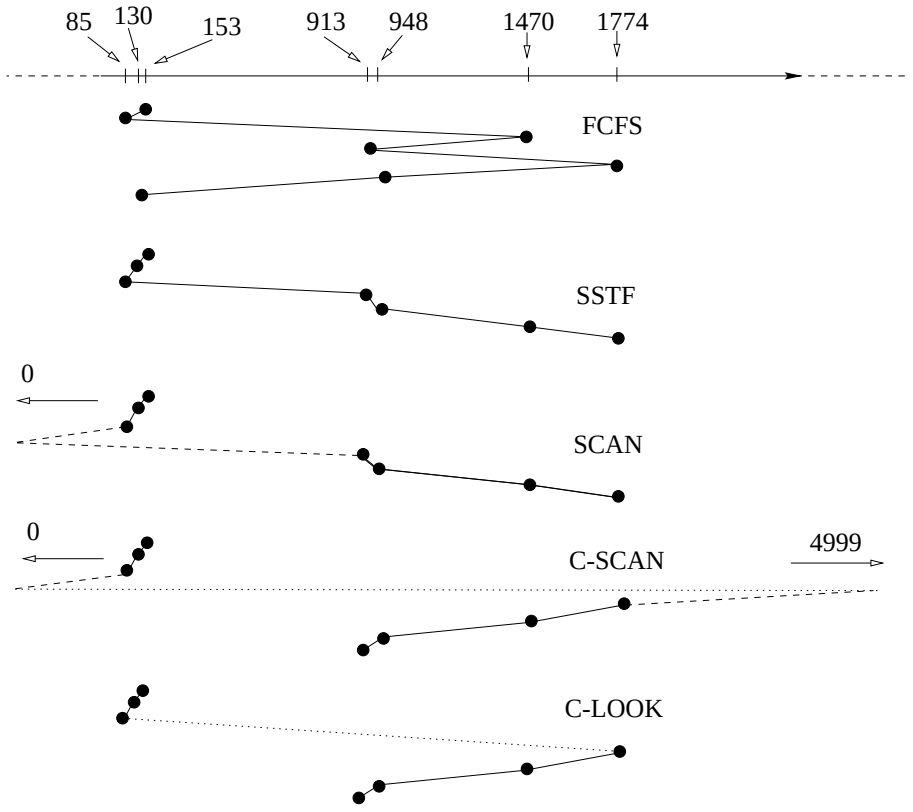
  Un disco possiede 5000 cilindri, numerati da 0 a 4999. Il driver del disco sta attualmente servendo una richiesta al cilindro 153. La coda di richieste in attesa, in ordine FIFO, è:

  $$85, 1470, 913, 1774, 948, 130.$$

  A partire dalla posizione corrente, qual è la distanza totale (indicata in cilindri) che deve percorrere il braccio del disco per soddisfare tutte le richieste in attesa, per ciascuno dei seguenti algoritmi di scheduling?

  1. FCFS
  2. SSTF
  3. SCAN (descrescente)
  4. C–SCAN (decrescente)
  5. C–LOOK (decrescente)

  Soluzione



$$
\begin{aligned}
dist(\text{FCFS}) &= (68 + 1385 + 557 + 861 + 826 + 818) = 4515 \text{ cilindri} \\
dist(\text{SSTF}) &= (23 + 45 + 828 + 35 + 522 + 304) = 1757 \text{ cilindri} \\
dist(\text{SCAN}) &= (23 + 45 + 85 + 913 + 35 + 522 + 304) = 1927 \text{ cilindri} \\
dist(\text{C–SCAN}) &= (23 + 45 + 85 + 5000 + 3226 + 304 + 522 + 35) = 9240 \text{ cilindri} \\
dist(\text{C–LOOK}) &= (23 + 45 + 1689 + 304 + 522 + 35) = 2618 \text{ cilindri}
\end{aligned}
$$

Esercizi da svolgere

1. Se si usa un algoritmo di rimpiazzamento FIFO con quattro pagine fisiche e otto pagine virtuali, quanti fault di pagina si avranno con la stringa di riferimenti

   0 1 7 2 3 2 7 1 0 3

   se le 4 pagine fisiche sono inizialmente vuote? Ripetere l'esercizio per l'algoritmo LRU.

2. Supponendo che lo spazio degli indirizzi virtuali sia costituito da otto pagine, mentre la memoria fisica può allocarne effettivamente quattro, si stabilisca quanti page–fault hanno luogo utilizzando una strategia di sostituzione LRU, relativamente alla stringa di riferimenti alle pagine

   0 2 1 3 5 4 6 3 7 4 7 3 3 5 5 3 1 1 1 7 2 3 4 1

   Si illustri inoltre l'anomalia di Belady, per un programma con cinque pagine virtuali, numerate da 0 a 4, per la stringa di riferimenti

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

   relativamente ai due casi di memoria fisica dotata, rispettivamente, di tre e quattro pagine.

3. Il problema di scegliere adeguatamente la dimensione delle pagine in un sistema di paginazione su richiesta è critico per quanto riguarda l'efficienza della gestione della memoria virtuale. Si supponga di avere a disposizione una data memoria fisica (la scelta della dimensione è libera). Si descriva:

   (a) un esempio in cui raddoppiare la dimensione delle pagine produce una riduzione dei page fault;

   (b) un esempio in cui dimezzare la dimensione delle pagine produce un analogo risultato.

4. Si consideri un file formato da 80 record e le sue possibili allocazioni su disco di tipo contiguo, concatenato, e con tabella indice e si supponga che le informazioni che riguardano il file siano già in memoria centrale. In ognuno di questi casi i record del file sono memorizzati 8 per blocco (tutto l'indice è invece memorizzato in un solo blocco). Si dica quanti accessi a disco sono necessari, in ognuna di queste situazioni, per leggere il primo record e i record in posizione 37 e 53.

5. Al driver di un disco arrivano, nell'ordine, richieste per i cilindri 38, 23, 59, 3, 63, 22, 2, 11 e 45.
   Si utilizzino:

   - la politica FCFS,
   - la politica SSTF,
   - la politica SCAN (si supponga che il braccio si muova verso cilindri con numeri progressivi decrescenti),

   e si stabilisca la distanza totale (in cilindri) percorsa dal braccio del disco per soddisfare tutte le richieste in attesa. Si assuma, per tutti i casi, che il braccio si trovi inizialmente posizionato sul cilindro 15.