Esercizi di preparazione al II Compito di esonero

Esercizio 1 - Debugging di un sistema software

Un programma $\mathcal{P}$ contiene inizialmente k errori ("bugs"), anche se k è incognito. La fase di "debugging" consiste nel far girare un programma di test $\mathcal{T}$ per individuare gli errori e rimuoverli. La rimozione degli errori (come haimè è noto a tutti) ne può introdurre dei nuovi.

Il programma di test $\mathcal{T}$ deve essere lanciato un numero sufficiente di volte in modo che il processo di individuazione e rimozione degli errori porti il programma $\mathcal{P}$ a comportarsi correttamente rispetto ad un insieme grande di dati di ingresso, anche se, in pratica, non si potrà mai essere sicuri di aver rimosso tutti i "bugs" presenti all'inizio.

Il processo di debugging viene modellato nel modo seguente. Ad ogni escuzione, il programma di test avrà una probabilità p_j di individuare un singolo errore se gli errori sono j, e una probabilità $1 - p_j$ di non individuare nessuno dei j errori. Sarà, ovviamente, $p_0 = 0$.

Se l'errore è individuato, esso viene corretto e rimosso; ma con una probabilità b_i ($i = 0, 1, 2$) nel processo di correzione vengono introdotti i nuovi errori (con $\sum_i b_i = 1$).

Si assuma che l'individuazione degli errori e la loro rimozione siano processi indipendenti dal passato e indipendenti fra di loro.

Sia X_n il numero di bugs presenti nel programma dopo n ripetizioni del test. Per le ipotesi fatte X_n è una DTMC con distribuzione iniziale $X_0 = k$. Disegnare il diagramma degli stati della DTMC X_n e trovarne la matrice di transizione.

Si facciano ora le seguenti ipotesi:

- $k = 10$
- $p_0 = 0 \;\; ; \;\; p_j \, = \, 0.1 \, \dfrac{(1 \, - \, \alpha) \, \alpha^{j-1}}{1 \, - \, \alpha^k} \qquad j = 1, 2, \ldots, k$
- $b_0 = 0.6 \;\; ; \;\; b_1 = 0.3 \;\; ; \;\; b_2 = 0.1$

Trovare quante esecuzioni n del programma di test $\mathcal{T}$ occorre prevedere, perchè la probabilità che ci siano più di tre errori residui nel programma $\mathcal{P}$ sia inferiore a 0.01. Calcolare i valori di n, per $\alpha = 1.2$ e $\alpha = 2.5$.

Commentare i risultati.

Esercizio 2 - Coda con rifiuto di clienti

Una coda a servitore singolo riceve in ingresso due classi di job. Le due classi arrivano in modo indipendente ciascuna secondo un processo di Poisson di parametro λ_i ($i = 1, 2$). Il tempo di servizio è distribuito esponenzialmente per entrambe le classi con lo stesso tasso μ.

La politica di accesso alla coda è definita nel modo seguente: se il numero totale di jobs nel sistema (di entrambe le classi) è maggiore o uguale a H, i job di tipo 1 sono rifiutati, mentre i job di tipo 2 possono sempre raggiungere la coda.

Descrivete il diagramma degli stati del sistema. Ricavate le probabilità stazionarie p_k ($k = 0, 1, 2, \ldots$) e il numero medio di job in coda $E[N]$, e discutete le condizioni di stabilità del sistema.

Esercizio 3 -Coda con arrivi scoraggiati

In un sistema a coda reale, i clienti possono essere indotti a non aggiungersi alla coda quando la coda tende a diventare eccessivamente lunga. Si dice, in questo caso che i nuovi arrivi tendono a essere scoraggiati dalla presenza di un numero crescente di clienti già presenti nel sistema.

Un modo possibile per tener conto di questo effetto è quello di considerare un processo di nascita/morte in cui i tempi di servizio sono esponenziali a tasso costante μ e i tempi di inter-arrivo sono esponenziali ma con tasso λ_k funzione decrescente dello stato k (dove lo stato è identificato come il numero di clienti nel sistema).

Si consideri, perciò, un processo di nascita e morte all'equilibrio caratterizzato dai seguenti parametri:

$$\lambda_k = \frac{\alpha}{k+1} \qquad k = 0, 1, \ldots$$

$$\mu_k = \mu \qquad k = 0, 1, \ldots$$

e si ricavino:

- la probabilità dello stato $k \rightarrow \ p_k \qquad (k = 0, 1, \ldots)$;
- la condizione di stabilità del sistema;
- il numero medio di clienti nel sistema.

Esercizio 4 - Efficenza di un meccanismo di paginazione

Si consideri un sistema a memoria virtuale con politica di sostituzione delle pagine di tipo *LRU*. Lo spazio di indirizzamento virtuale è di n pagine, mentre la memoria fisica contiene m page-frame. Per rendere più concreto il problema si supponga $n = 3$ e $m = 2$.

Lo stato della memoria fisica può essere caratterizzato con $m = 2$ indici che rappresentano le 2 pagine effettivamente presenti nella memoria fisica in ordine di accesso: quindi $q(i, j)$ rappresenta lo stato in cui nella memoria fisica sono presenti le pagine i $(i = 0, 1, 2, 3)$ e j $(j = 0, 1, 2, 3)$ e j è la pagina *LRU* e quindi destinata alla sostituzione in caso di un page-fault.

Si noti che con questa codifica:

- un indice $i = 0$ indica che la pagina fisica non è mai stata riempita;
- lo stato $q(i, j) \neq q(j, i)$ in quanto, pur contenendo le stesse pagine, nel primo caso la pagina j è *LRU*, nel secondo caso la pagina i è *LRU*.

Ad ogni nuova richiesta di accesso alla memoria virtuale, la probabilità di accesso alla pagina j dipende solo dalla pagina i indirizzata nell'accesso precedente. Dalla struttura del programma si possono ricavare le seguenti probabilità condizionate a un passo (indipendenti dal passo n):

$$r_{ij} = Pr\{\text{pag. indirizzata al passo } n = j \,|\, \text{pag. indirizzata al passo } (n-1) = i\}$$

L'evoluzione della memoria virtuale ad ogni accesso può essere modellata con una *DTMC* sullo spazio degli stati $q(i, j)$.

Si supponga che al passo $n = 0$ la memoria fisica sia vuota e che il primo indirizzamento sia con probabilità 1 alla pagina $i = 1$.

Si risponda alle seguenti domande:

- Disegnare il grafo dello spazio degli stati e relative transizioni;
- ricavare la matrice di transizione a 1 passo;
- discutere la classificazione degli stati;
- dati i seguenti valori numerici:

$$\begin{array}{lll} r_{11} = 0.8 & ; & r_{12} = 0.15 \quad ; \quad r_{13} = 0.05 \\ r_{21} = 0.1 & ; & r_{22} = 0.7 \quad ; \quad r_{23} = 0.2 \\ r_{31} = 0.1 & ; & r_{23} = 0.2 \quad ; \quad r_{33} = 0.7 \end{array}$$

  - ricavare il vettore delle probabilità di stato al passo n;
  - se ricorrono le condizioni, ricavare il vettore delle probabilità di stato all'equilibrio.
  - Ricavare le probabilità di fallimento di pagina *page fault*.

Esercizio 5 - Sistema parallelo: politiche di manutenzione

Si abbia un sistema a due componenti in parallelo A e B con tassi di guasto costanti denominati λ_A e λ_B, rispettivamente. Si devono studiare e confrontare possibili politiche di manutenzione sapendo che si dispone di un riparatore singolo, e che il tempo di riparazione per entrambi i componenti è distribuito con legge esponenziale con parametro μ_A per il componente A e μ_B per il componente B.

Si sono prese in considerazione le seguenti di politiche di manutenzione alternative:

a) - Riparazione al guasto singolo o in base alla lista di priorità ($A \rightarrow B$ prima A e poi B nel caso di guasti multipli) senza interruzione del servizio.

b) - Riparazione solo al guasto del sistema nel suo complesso, in base alla lista di priorità ($A \rightarrow B$) con interruzione del servizio (il sistema non funziona durante tutta la durata della riparazione).

c) - Riparazione sulla base della politica *FCFS (First Come First Served)* (cioè il primo che si guasta viene riparato per primo) senza interruzione del servizio.

Disegnare il diagramma degl stati nei 4 casi, e ricavare la disponibilità asintotica, con i seguenti valori numerici:

- $\lambda_A = 0.001 \qquad \lambda_B = 0.005.$
- $\mu_A = 1 \qquad \mu_B = 1.5$

Commentare i risultati.