Esame di Laboratorio di Fisica Computazionale

27 settembre 2013, ore 10.00

Il voto finale deriva dalla somma dei voti in shell scripting, `Mathematica` e `C++`.

Raggiunta la soglia di sufficienza in ciascuna delle tre parti il voto è pari a 18. La sufficienza si ottiene: in `Mathematica` ottenendo almeno 5 punti, secondo quanto indicato nel testo dei due esercizi; in `C++` risolvendo correttamente almeno i primi due punti ed impostando correttamente il terzo.

shell scripting

1. Si scriva uno script in bash che rinomina i files `vecchio1.txt, vecchio2.txt, vecchio3.txt` assegnando i nuovi nomi `nuovo1.txt, nuovo2.txt, nuovo3.txt` .

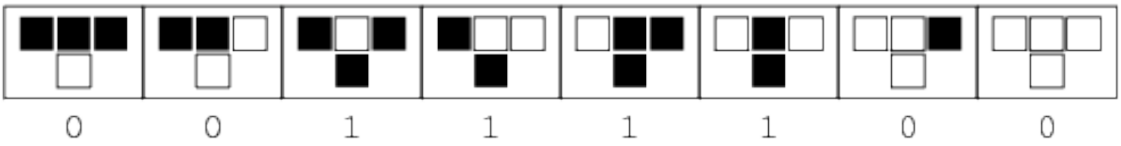
2. A partire dal file `dati.top`, si estragga l'istogramma (cioè il set di valori numerici su tre colonne) che corrisponde alla distribuzione indicata con `pt3`.

Mathematica

Automa cellulare

- Si scriva una funzione che inizializza una lista contenente 21 elementi, tutti zero con l'eccezione del quinto elemento, che deve essere posto pari a 1.
  (Si consideri l'utilizzo combinato di `ReplacePart` *e di* `Table`*).*
  (2 punti)

- Si scriva una lista di regole di sostituzione che realizzi le trasformazioni rappresentate graficamente in figura.
  (Si consideri per la singola sostituzione una regola del tipo $\{a, b, c\} \to d$*)*
  (1 punto)



- La trasformazione di una lista esistente in una nuova, secondo il meccanismo dell'automa cellulare, richiede di applicare a ciascuna terna di elementi consecutivi della lista il set di regole di sostituzione del punto precedente. Per semplicità, il primo e l'ultimo elemento della lista possono essere ricopiati da una riga all'altra, mentre si devono calcolare tutti gli elementi nuovi dal secondo al penultimo.
  Si utilizzino i comandi `Table` *per generare i nuovi elementi, dal secondo al penultimo, e poi* `Prepend` *e* `Append` *per ricopiare il primo e l'ultimo.*
  (3 punti)

- Si generi una lista di 20 righe a partire da quella iniziale. Si visualizzi la matrice risultante con i comandi `ArrayPlot` e `Show`.
  (2 punti)

Life

- Si consideri il seguente sistema di equazioni differenziali

$$\begin{cases} \frac{dx(t)}{dt} &= [a - b\,y(t)]\,x(t) \\ \frac{dy(t)}{dt} &= [c\,x(t) - d]\,y(t) \end{cases} \tag{1}$$

  Si risolvano numericamente le equazioni 1, con condizioni al contorno $x(0) = 0.2, y(0) = 0.8$ e con $a = b = c = d = 1$, nell'intervallo $t \in [0, 20]$. Si disegnino $x(t)$ e $y(t)$ in funzione del tempo. Si disegnino nel piano (x, y) le soluzioni al variare del tempo.
  (2 punti)

- Partendo dalla condizione iniziale $x(0) = 0.2, y(0) = 0.8$ e considerando degli intervalli temporali pari a $\Delta t = 0.01$, si consideri l'evoluzione del sistema definito dall'equazione 1 con $a = b = c = d = 1$; si calcolino i primi 2000 passi dell'evoluzione temporale discretizzata, usando per l'aggiornamento delle due funzioni la legge

$$\begin{cases} x(t+1) &= x(t) + [a - b\,y(t)]\,x(t)\Delta t \\ y(t+1) &= y(t) + [c\,x(t) - d]\,y(t)\Delta t \end{cases} \tag{2}$$

  Si disegni nel piano (x, y) l'evoluzione del sistema, utilizzando `ListPlot` con l'opzione `PlotJoined` $\rightarrow$ `True`.
  (3 punti)

- Si disegnino separatamente $x(t)$ e $y(t)$ in funzione del tempo, sempre utilizzando `ListPlot` con l'opzione `PlotJoined` $\rightarrow$ `True`.
  (2 punti)

C++

Si risolva l'esercizio proposto. Per facilitare la correzione, includere tutto in un unico file sorgente. Si consiglia di concentrarsi sulla pulizia del codice: pochi punti risolti in modo ordinato saranno valutati meglio di tanti punti trattati disordinatamente. La sufficienza è raggiunta risolvendo correttamente i primi due punti.

Esercizio 1

1. Si scriva una classe `Matrix` che rappresenti una matrice 2x2. Tra i membri privati si pongano le quattro componenti (reali); tra i membri pubblici si scriva un opportuno costruttore che richieda gli elementi di matrice, con valori di default impostati in modo da realizzare la matrice unità.

2. Si implementi l'operatore "$*$" (membro di `Matrix`), che restituisce il prodotto riga per colonna.

3. Si dichiari come `friend` e si implementi l'overloading dell'operatore

   ```
   ostream & operator<<
   (ostream & outstream, const Matrix & m).
   ```

   Esso dovrà stampare la generica matrice nella forma $\begin{matrix} a & b \\ c & d \end{matrix}$

4. Si scriva un `main` che istanzi una matrice identità ed una $m = \begin{pmatrix} 1 & 2 \\ 3 & 4 \end{pmatrix}$ e si controlli, stampando i risultati su `cout`, che il prodotto a sinistra (come anche quello a destra) dell'identità con M sia uguale a M.

5. Tra i membri pubblici, si scriva una funzione membro `eigenvalue`, che calcoli il modulo dell'autovalore con modulo massimo. [Hint: si può usare Mathematica per ottenere rapidamente la formula degli autovalori.]

6. Si scriva una classe `RandomMatrix` che erediti pubblicamente da `Matrix`. Il costruttore di default dovrà generare una matrice simmetrica con elementi random tra 0 e 1.

7. Generare 10 000 matrici simmetriche random e calcolare il valor medio del modulo dell'autovalore con modulo massimo. Ripetere l'esperimento moltiplicando ogni matrice a sinistra per $\sigma_3 = \begin{pmatrix} 1 & 0 \\ 0 & -1 \end{pmatrix}$.