Laboratorio di Calcolo, Esercitazione 7, 28-29 novembre 2024

Canale Pet-Z, Docenti: Shahram Rahatlou, Sibilla Di Pace

Cammino aleatorio unidimensionale: Lo scopo di questa esercitazione è di studiare il cammino aleatorio utilizzando numeri casuali tramite le funzioni `srand48()` es `lrand48()` ed immagazzinando i dati in array.

Il cammino aleatorio (random walk) unidimensionale descrive il moto una particella lungo una retta a partire da una posizione iniziale x_0. Ad ogni passo, la particella ha la stessa probabilità di avanzare (di una cella) a destra o a sinistra. La simulazione finisce dopo N_{step} passi.

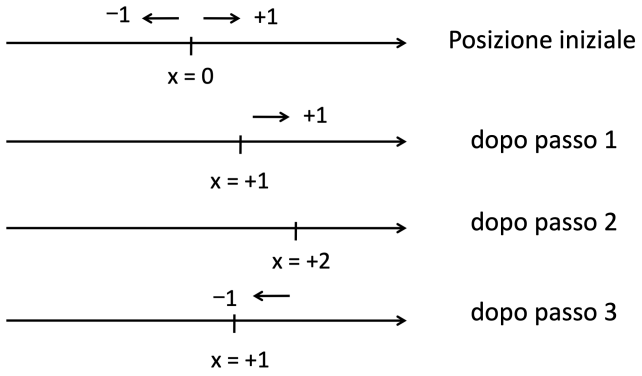

Figura 1: Esempio del cammino aleatorio dopo 3 passi

In questa prova si vogliono studiare alcune caratteristiche del cammino aleatorio al variare del numero di passi N_{step} per $N_{\text{exp}} = 10000$ simulazioni. Ciascun cammino finisce dopo N_{step} passi in una posizione finale `posFinal` nell'intervallo $[-N_{\text{step}}, +N_{\text{step}}]$.

► Cartella di lavoro

Fare login sulla postazione utilizzando le credenziali user-id `studente` e password `informatica`. Creare una cartella `LCSR7` nella *home directory* con il comando `mkdir` in cui scriverete i programmi di oggi. Tutti i file di codice sorgente in C e in python dovranno trovarsi in questa cartella per essere visualizzati. **Le cartelle create sulla scrivania (Desktop) o in altre sotto-cartelle non verranno copiate né valutate.**

Prima parte

Creare un programma `walk-NNN.c`, dove `NNN` è il vostro numero di gruppo, ad esempio `098`, nella cartella `LCSR7` utilizzando l'editor di testo `emacs`, per eseguire le seguenti operazioni:

1. Fare $N_{\text{exp}} = 10000$ simulazioni del cammino aleatorio a partire dall'origine con $N_{\text{step}} = 20$;
2. per ciascuna simulazione calcolare
   - la posizione più lontana raggiunta durante il cammino (`maxDist`)
   - la posizione finale del cammino (`posFinal`);

   Salvare questi dati per ciascuna simulazione in array di opportuna lunghezza.

3. ogni 500 simulazioni stampare sullo schermo il numero di simulazione e la posizione finale raggiunta al termine del cammino;

4. per ciascuna posizione finale possibile, memorizzare in un array `frequenza` di lunghezza opportuna, il numero di simulazioni terminate in quella posizione;

5. Al termine delle simulazioni

   - stampare sullo schermo la media aritmetica su tutte le simulazioni della distanza della posizione finale dall'origine (`distFinMedia`)
   - stampare sullo schermo un sommario dei valori immagazzinati nell'array `frequenza` con il seguente formato

```
****************** Sommario posizioni finali ****************
posizione: -20   simulazioni:     0
posizione: -18   simulazioni:     0
posizione: -16   simulazioni:     1
posizione: -14   simulazioni:    14
posizione: -12   simulazioni:    37
posizione: -10   simulazioni:   156
posizione:  -8   simulazioni:   384
posizione:  -6   simulazioni:   759
posizione:  -4   simulazioni:  1158
posizione:  -2   simulazioni:  1606
posizione:  +0   simulazioni:  1724
posizione:  +2   simulazioni:  1667
posizione:  +4   simulazioni:  1212
posizione:  +6   simulazioni:   725
posizione:  +8   simulazioni:   377
posizione: +10   simulazioni:   125
posizione: +12   simulazioni:    46
posizione: +14   simulazioni:     6
posizione: +16   simulazioni:     3
posizione: +18   simulazioni:     0
posizione: +20   simulazioni:     0
```

6. creare un file di testo `frequenza.txt` dove ciascuna riga contiene solo due valori: la posizione finale e il relativo numero di simulazioni terminate in quella posizione finale; potete creare questo file usando `emacs` copiando i dati dal terminale, oppure scrivendo i dati sul file con le funzioni `fopen()` e `fprintf()`

7. scrivere un programma python `frequenza-NNN.py` per fare il grafico del numero delle simulazioni in funzione della posizione finale con opportuni nomi per gli assi x e y

Si ricorda che per creare l'eseguibile utilizzando la libreria matematica dovete usare il comando `gcc -Wall -o app.exe programma.c -lm` dalla riga di comando nel terminale.

Si consiglia di scrivere il programma in modo incrementale, verificando la corretta compilazione e l'esecuzione almeno dopo ciascuno dei passi indicati nel testo.

► **Seconda parte**

In questa seconda parte, vogliamo studiare l'andamento della distanza media finale in funzione del numero di passi del cammino, variando N_{step}. A tal fine, dobbiamo creare un file di dati `distfin.txt` che contenga per ciascuna riga solo due numeri

1. il numero di passi del cammino N_{step};
2. distanza media della posizione finale `distFinMedia`.

Potete ottenere questi dati modificando il vostro programma `walk-NNN.c`:

1. con un opportuno ciclo variare il numero N_{step} aumentando di un fattore 2, $N_{\text{step}} = 2, 4, 8, 16, 32, 64, 128$
2. per ciascun valore di N_{step} riptere quanto fatto nella prima parte, commentando però la stampa delle frequenze che non ci interessa;
3. al termine delle simulazioni con un valore di N_{step}, stampare sullo schermo il numero di passi e la distanza media della posizione finale;
4. creare il file `distfin.txt` come richiesto. Come sempre, potete creare questo file con `emacs` copiando i valori dallo schermo, oppure tramite la scruttura sul file dal proframma;
5. scrivere il codice python `andamento-NNN.py` per graficare l'andamento della distanza media finale `distfin.txt` in funzione del numero di passi N_{step}.

► **Nozioni utili**

1. l'inizializzazione dei numeri casuali con la funzione `srand48( seed )` va fatta una sola volta e all'inizio della funzione `main()`;
2. per N valori $\{x_1, ..., x_N\}$, la media aritmetica $\langle x \rangle$ è definita come $\langle x \rangle = \sum_{j=1}^{N} x_j / N$
3. per eseguire il codice in python, dalla riga di comando nel terminale dovete eseguire il comando `python3 nomefile.py`