Laboratorio di Calcolo, Prova d'esame del 15/09/2025

Tutti i canali, Anno accademico 2024-25

Nome: _____ Cognome: _____
Matricola: _____ □ Ritirata/o

Lo scopo di questa prova d'esame è di scrivere un programma in C e uno script in python seguendo la traccia riportata di seguito. Si tenga presente

1. Il tempo a disposizione è di 3 ore. Sono ammessi libri di testo, prontuari, appunti. Non si può parlare con nessuno, utilizzare cellulari/tablet/laptop, pena l'annullamento del compito.
2. Il programma va scritto e salvato esclusivamente sul computer del laboratorio, a cui si deve accedere utilizzando come username **studente** e come password **informatica**
3. **Tutti i file vanno salvati in una cartella chiamata** `LCESAME_NOME_COGNOME` **nella home directory**, dove `NOME` e `COGNOME` indicano rispettivamente il tuo nome e cognome. Ad esempio lo studente *Nicolò Maria De Rossi Salò* deve creare una cartella chiamata `LCESAME_NICOLOMARIA_DEROSSISALO` contenente tutti i file specificati nel testo. **Tutto ciò che non si trova all'interno della cartella suddetta non verrà valutato.**
4. **È necessario consegnare il presente foglio indicando nome, cognome e numero di matricola** (vedi sopra), barrando la casella **"Ritirato/a"** se ci si vuole ritirare, ovvero se non si vuole che la presente prova venga valutata.

Consideriamo una variabile casuale discreta che può assumere valori **interi** $n = 0, \ldots, N-1$, distribuiti secondo una legge di probabilità $p(n)$.
Un metodo molto usato per generare numeri interi che seguano una distribuzione discreta arbitraria $p(n)$ è Metropolis-Hastings, che si basa sulle cosiddette catene di Markov. L'idea è costruire una sequenza di valori n_k in cui ogni elemento dipende solo dal precedente, in modo che, dopo un certo numero di passi, la sequenza riproduca la distribuzione desiderata.
Una possibile realizzazione del metodo che permette di generare una sequenza di `NSAMPLE` numeri interi compresi nell'intervallo $[0, N-1]$ secondo una $p(n)$ assegnata, richiede questa serie ordinata di operazioni:

1. Si sceglie un valore iniziale n_{init} in modo casuale e uniforme nell'intervallo $[0, N-1]$.
2. Si costruisce una sequenza di `NSAMPLE` numeri interi a partire da n_{init}. Dato il generico elemento n_k (con $0 \leq k <$ `NSAMPLE`):
   (a) si propone un nuovo valore n_{tmp}, estratto in modo uniforme tra 0 e $N-1$;
   (b) si calcola il rapporto
   $$A = \min\left(1, \frac{p(n_{\text{tmp}})}{p(n_k)}\right);$$
   (c) si estrae un numero reale y uniforme in $[0, 1]$.
   - Se $y \leq A$, si accetta la proposta e si pone $n_{k+1} = n_{\text{tmp}}$.

- Se $y > A$, si la proposta e si pone $n_{k+1} = n_k$.

In questo modo si costruisce una sequenza di valori n_k. **NOTA BENE:** si considera come primo elemento utile quello prodotto dopo la prima iterazione, n_0, e non n_{init}.

Con una scelta adeguata di `NSAMPLE`, la sequenza finale di numeri avrà una distribuzione prossima a $p(n)$, come desiderato.

► **Esercizio in C:** Scrivere un programma `NOME_COGNOME.c` che rispetti le seguenti specifiche:

1. Definire, tramite direttive `#define`: (i) `NSAMPLE` = 10000 (numero di elementi della sequenza), (ii) `N` = 10 (numero di possibili valori della variabile casuale),
2. Dichiarare due array, di tipo e dimensioni opportune: `p[]` per memorizzare la distribuzione di probabilità, `samples[]` per i valori campionati.
3. Implementare la funzione `init_prob()`, di tipo e argomenti opportuni, che inizializza `p[]` come segue:
   - 3a. riempie `p[]` con `N` numeri casuali (tipo `double`), uniformi in $[0, 1]$;
   - 3b. calcola la somma `norm` degli elementi di `p[]`;
   - 3c. normalizza la distribuzione dividendo ogni elemento di `p[]` per `norm`.
4. Implementare la funzione `metropolis()`, di tipo e argomenti opportuni, che riempie `samples[]` con `NSAMPLE` valori generati usando l'algoritmo di Metropolis-Hastings.
5. Implementare la funzione `frequency()`, di tipo e argomenti opportuni, che riempie un array `freq[]` (anche'esso definito nel `main`, di stessa dimensione e tipo di `p[]`), in cui l'elemento i-esimo, con $0 \leq i \leq N-1$, è la frequenza relativa del valore i in `samples[]` (cioè il numero di occorrenze di i nell'array `samples[]` diviso `NSAMPLE`).
6. Dopo aver generato la sequenza con `metropolis()` e calcolato `freq[]` con `frequency()`, scrivere su file `NOME_COGNOME.dat` i valori di `p[]` e `freq[]` in tre colonne:

```
0 p(0) freq(0)
1 p(1) freq(1)
...
N-1 p(N-1) freq(N-1)
```

I valori della seconda e terza colonna devono essere stampati con 4 cifre decimali.

► **Esercizio in Python:**
Dopo aver verificato la corretta creazione del file, creare uno script python `NOME_COGNOME.py` che legga i dati contenuti nel file `NOME_COGNOME.dat`, crei un grafico che riporti sia `p(n)` sia `freq(n)` in funzione di n e lo salvi sul file `NOME_COGNOME.png`. Il grafico deve essere completo di titolo e di opportune legende per i dati e gli assi.