# APPELLO SCRITTO DI PROGRAMMAZIONE 1
## CORSO DI LAUREA IN MATEMATICA
## UNIVERSITÀ DEGLI STUDI DI MILANO
### 2014–2015
### 9.VII.2015

VINCENZO MARRA

INDICE


| | |
|---|---|
| Preliminari | 2 |
| Esercizio 1 | 3 |
| Lettura della funzione $f$ da file. | 3 |
| Punti: 6. | 3 |
| Esercizio 2 | 3 |
| Permutazioni e lunghezza delle orbite. | 3 |
| Punti: 8. | 4 |
| Esercizio 3 | 4 |
| Generazione casuale della distribuzione $\mu$. | 4 |
| Punti: 9. | 4 |
| Esercizio 4 | 5 |
| Valore atteso di $f$ e misura delle orbite. | 5 |
| Punti: 7. | 5 |
| *Istruzioni per la consegna* | 6 |


**Avvertenza.** Il tema d'esame richiede lo sviluppo di un singolo programma che computi alcuni dati relativi a una endofunzione $f\colon X \to X$ memorizzata in un file; si veda sotto il paragrafo Preliminari per i dettagli. La funzione è acquisita da uno dei file `f1`, `f2`, ecc., allegati al tema d'esame. Svilupperete varie funzioni ausiliarie nel risolvere gli esercizi. Integrerete poi tutte le funzioni ausiliarie in un unico programma finale, che consisterà di un unico file sorgente. (Il tema d'esame richiede esplicitamente l'implementazione di alcune funzioni ausiliarie. È comunque sempre ammissibile aggiungere altre funzioni ausiliarie qualora lo riteniate opportuno per una migliore implementazione della vostra soluzione.) Si raccomanda di leggere interamente il tema d'esame con attenzione prima di cominciare a scrivere le soluzioni dei singoli esercizi, in modo da avere chiaro l'obiettivo finale. La raccomandazione si applica a maggior ragione ai Preliminari.

Ultima revisione: 7 ottobre 2015.

## Preliminari

Considereremo in questo tema d'esame un numero naturale $n \geq 1$, l'insieme

$$X := \{0, 1, 2, \ldots, n-1\},$$

e una funzione

$$f \colon X \to X.$$

Denotiamo con $[0,1] \subseteq \mathbb{R}$ l'intervallo unitario chiuso dei numeri reali compresi fra 0 e 1. Una *distribuzione di probabilità* su $X$ è una funzione

$$\mu \colon X \to [0,1]$$

che soddisfi la condizione di normalizzazione:

$$\sum_{i \in X} \mu(i) = 1.$$

Scriviamo $2^X$ per l'insieme delle parti di $X$ — cioè, $2^X := \{S \mid S \subseteq X\}$. La distribuzione $\mu$ dà luogo a una *misura di probabilità* sull'insieme $X$, ossia ad una funzione

$$M_\mu \colon 2^X \to [0,1]$$

definita in questo modo per ogni $S \subseteq X$:

$$(\star) \qquad M_\mu(S) := \sum_{i \in S} \mu(i).$$

Data la distribuzione di probabilità $\mu$, si definisce inoltre il *valore atteso* di $f$ rispetto a $\mu$ in questo modo:

$$\mathbb{E}\,(f, \mu) := \sum_{i \in X} \mu(i) f(i).$$

La funzione $f$ induce una dinamica sull'insieme $X$ tramite iterazione della sua applicazione: dato l'elemento $i \in X$, si può considerare l'azione di $f$ su $i$ iterata $k$ volte, $f \circ f \circ \cdots \circ f(i) := f(f(\cdots f(i) \cdots))$. Se la funzione $f$ è una permutazione, $X$ is partiziona sotto l'azione di $f$ nella famiglia disgiunta delle orbite dei suoi elementi. Ecco la definizione precisa di alcune di queste nozioni, necessarie alla risoluzione del tema.

La funzione $f \colon X \to X$ è una *permutazione* se è biiettiva, ossia iniettiva e suriettiva. Nota Bene: poiché $X$ ha cardinalità finita, $f$ è una biiezione se e solo se $f$ è una suriezione se e solo se $f$ è una iniezione.

Nel caso in cui $f$ sia una permutazione, diremo che l'*orbita* di un elemento $i \in X$ sotto l'azione di $f$ è l'insieme

$$\mathcal{O}_i := \{j \in X \mid \text{ Esiste } k \geq 1 \text{ tale che } \underbrace{f \circ f \circ \cdots \circ f}_{k \text{ volte}}(i) = j\}.$$

Nota Bene: l'orbita di $i \in X$ si può enumerare facilmente applicando ripetutamente $f$ ad $i$ fino a quando non valga $f \circ f \circ \cdots \circ f(i) = i$.

Infine, se l'insieme $X$ è dotato di una distribuzione di probabilità, si può considerare la misura dell'orbita $\mathcal{O}_i$ secondo la $(\star)$, e cioè la somma dei valori $\mu(j)$ al variare di $j$ nell'orbita $\mathcal{O}_i$ di $i$.

## Esercizio 1

**Lettura della funzione $f$ da file.**

**Punti:** 6.

Scrivete una funzione `main` che accetti in ingresso dalla riga di comando il nome di un file. Se non vi sono argomenti sulla riga di comando, il programma termina con un messaggio d'errore appropriato. Se vi è almeno un argomento, il programma tenta di aprire il file dal nome corrispondente in modalità di lettura. Se l'apertura non riesce, il programma termina con un messaggio d'errore appropriato.

Se invece l'apertura avviene correttamente, il programma assume che il file sia composto esattamente nel modo seguente: la prima riga contiene un numero intero $n > 0$ seguito dal ritorno a capo. Poi, il file contiene $n$ righe di dati, ciascuna terminata dal ritorno a capo, e ciascuna contenente esattamente due interi `i` ed `fi` nel formato:

```
i     fi
```

Si intende che la riga qui sopra specifica il valore `fi` della funzione $f$ applicata al numero `i`. Per esempio, un file composto così:

```
3
2     0
0     1
1     1
```

descrive la funzione $f\colon \{0,1,2\} \to \{0,1,2\}$ tale che $f(2) = 0$, $f(0) = 1$ ed $f(1) = 1$. (Si noti quindi che la prima colonna di dati, che consiste degli elementi dell'insieme dominio $X = \{0, 1, \ldots, n-1\}$ (assieme al primo valore $n$ che dà la cardinalità di $X$), non è necessariamente ordinata.) Nel leggere i dati dal file, assicuratevi di trattare i caratteri `'\n'` in modo opportuno.

Il programma dichiara un array di interi di nome `f` e di dimensione appropriata, e lo inizializza leggendo i dati relativi alla funzione $f$ dal file. Dopo la lettura, il valore di tipo `int`

```
f[i]
```

deve coincidere con il numero intero

$$f(i).$$

Fatto ciò, il programma visualizza la funzione acquisita e termina. Si veda la Fig. 1 per un esempio di formattazione dei dati in uscita. Per testare la vostra risoluzione a questo e ai successivi esercizi, usate i file `f1`, `f2`, ecc. allegati al tema d'esame.

## Esercizio 2

**Permutazioni e lunghezza delle orbite.**

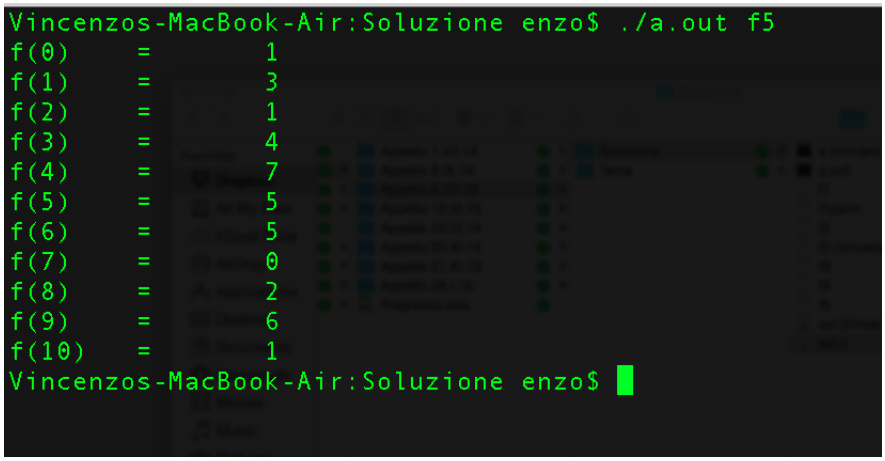


FIGURA 1. Esempio di esecuzione (Esercizio 1).

**Punti:** 8.

Scrivete una funzione di prototipo

```
int isPerm(int *f, int n)
```

che restituisca 0 se la funzione `f` in argomento non è una permutazione, e un valore diverso da 0 altrimenti. Il parametro `n`, come in tutto il resto del tema d'esame, è la dimensione dell'array di interi `f`.

Scrivete poi una seconda funzione di prototipo

```
int lungorb(int *f, int i);
```

che restituisca la lunghezza (ossia la cardinalità) dell'orbita dell'elemento `i` di $X$ sotto l'azione di $f$. La funzione `lungorb` assume che la funzione `f` in argomento sia una permutazione.

Ampliate la funzione `main` del vostro programma in modo che essa comunichi all'utente, prima di terminare, se la funzione $f$ letta dal file specificato come argomento dalla riga di comando sia una permutazione oppure no. In caso affermativo, il programma visualizza anche l'elenco delle lunghezze delle orbite di ciascun elemento di $X$.

## ESERCIZIO 3

**Generazione casuale della distribuzione $\mu$.**

**Punti:** 9.

Scrivete una funzione di prototipo

```
void genera_m(double *m, int n)
```

che generi in modo casuale una distribuzione di probabilità `m` sull'insieme di valori `0,1,...,n-1`. Nella funzione `main`, la distribuzione sarà memorizzata in un array di `double` di dimensione `n`. Con riferimento alle notazioni dei Preliminari, ciascun

valore `m[i]` dell'array rappresenta il valore $\mu(i)$ della distribuzione di probabiità $\mu$ su $X$.

Si assuma quindi nell'implementare la funzione che `m` sia un puntatore a una zona di memoria atta a contenere un array di `n` double. La funzione deve inizializzare tale array con valori generati casualmente che costituiscano una distribuzione di probabilità, ossia che siano non negativi e assommino a 1. Ecco adesso dei suggerimenti su come generare numeri pseudo-casuali.

La chiamata

```
rand();
```

alla funzione di libreria `rand()` restituisce un valore *intero* casuale compreso fra 0 e la costante `RAND_MAX`. Prima di invocare la funzione `rand`, è necessario eseguire la chiamata

```
srand(time(NULL));
```

per inizializzare il generatore di numeri pseudo-casuali utilizzato da `rand`. Basta fare una sola chiamata a `srand` in tutto il programma: poi potrete chiamare `rand` arbitrariamente molte volte, senza dover richiamare `srand`. Converrà quindi invocare `srand` come qui sopra *una sola volta* nel `main`. Tutte le funzioni e le costanti di libreria citate in questo capoverso sono definite nei due file di intestazione `stdlib.h` e `time.h`: vi basterà quindi includerli nel vostro programma per poter generare numeri casuali interi nel modo appena spiegato.

Progettate adesso un semplice algoritmo per generare gli $n$ numeri `double` "casuali" $\mu(0), \ldots, \mu(n-1)$ soggetti alla condizione $\sum_{i=0}^{n-1} \mu(i) = 1$. Ecco qualche suggerimento per una possibile risoluzione.

Dichiarate e implementate innanzitutto una funzione

```
double casuale(double min)
```

che restituisca un numero `double` casuale fra `min` e 1, dove si assume $0 \leq$ `min` $\leq 1$. Per esempio, se `min` $= 0$ allora la funzione restituisce un valore `double` casuale in $[0, 1]$. Abbiamo appena visto come generare valori *interi* casuali fra 0 e `RAND_MAX`. Quindi vi basta dividere per...

Per il caso generale in cui $0 \leq$ `min` $\leq 1$ occorre rinormalizzare linearmente i valori in [0,1] a valori in [`min`, 1].

Con la funzione ausiliaria `causale` a disposizione, non è difficile implementare `genera_m`. Pensate per esempio alla suddivisione dell'intervallo $[0, 1]$ in sottointervalli, ciascuno dei quali avente lunghezza $m_i$ da assegnare a `m[i]`...

Ampliate la vostra funzione `main` in modo che il programma visualizzi, prima di terminare, la distribuzione di probabilità $\mu$.

## Esercizio 4

**Valore atteso di $f$ e misura delle orbite.**

**Punti:** 7.

Scrivete una funzione di prototipo

```
double valatt(int *f, double *m, int n)
```

che restituisca il valore atteso della funzione `f` secondo la distribuzione di probabilità `m`.

Scrivete poi una seconda funzione di prototipo

```
double misorb(int *f, double *m, int i, int n)
```

che restituisca la misura dell'orbita dell'elemento `i` di `{0,1,...,n-1}` sotto l'azione di `f`, dove la misura è quella indottta dalla distribuzione `m`. La funzione `misorb` assume cha la funzione `f` in argomento sia una permutazione.

Ampliate la vostra funzione `main` in modo che il programma visualizzi, prima di terminare, il valore atteso della funzione $f$ rispetto a $\mu$. Nei soli casi in cui $f$ sia una permutazione, il programma visualizza anche la misura dell'orbita di ciascun elemento di $X$ secondo la misura $M_\mu$ indotta su $X$ da $\mu$. Si vedano le Fig. 2 e 3 per degli esempi di esecuzione.

*Istruzioni per la consegna*

- Funzioni da consegnare (in un unico file sorgente).

  `main()`
  `isPerm()`
  `lungorb()`
  `casuale()`
  `genera_m()`
  `valatt()`
  `misorb()`
  Tutte le eventuali altre funzioni ausiliarie
  che abbiate scritto.

- Sito per la consegna:

  `https://upload.mat.unimi.it`

- Autenticarsi con le proprie credenziali di posta elettronica d'ateneo.
- Consegnare solo il codice sorgente (file `*.c`), non l'eseguibile.

(V. Marra) Dipartimento di Matematica *Federigo Enriques*, Università degli Studi di Milano, via Cesare Saldini, 50, I-20133 Milano
*E-mail address*: vincenzo.marra@unimi.it

```
Vincenzos-MacBook-Air:Soluzione enzo$ ./a.out f1
f(0)    =       1
f(1)    =       3
f(2)    =       1
f(3)    =       4
f(4)    =       7
f(5)    =       5
f(6)    =       5
f(7)    =       0
f(8)    =       2
f(9)    =       6
f(10)   =       1
Permutazione: No.
Distribuzione di probabilita':
m(0)    =       0.774397
m(1)    =       0.0668137
m(2)    =       0.0798563
m(3)    =       0.0299561
m(4)    =       0.0247938
m(5)    =       0.0087425
m(6)    =       0.000644654
m(7)    =       0.0106523
m(8)    =       0.0021466
m(9)    =       0.00140491
m(10)   =       0.00059171
Massa totale: 1
Valore atteso di f rispetto ad m: 1.40833
Vincenzos-MacBook-Air:Soluzione enzo$
```

FIGURA 2. Primo esempio di esecuzione del programma completo. (Qui $f$ non è una permutazione.)

```
Vincenzos-MacBook-Air:Soluzione enzo$ ./a.out f4
f(0)    =       1
f(1)    =       2
f(2)    =       3
f(3)    =       4
f(4)    =       5
f(5)    =       0
Permutazione: Si'.
Lunghezza orbita 0: 6
Lunghezza orbita 1: 6
Lunghezza orbita 2: 6
Lunghezza orbita 3: 6
Lunghezza orbita 4: 6
Lunghezza orbita 5: 6
Distribuzione di probabilita':
m(0)    =       0.775047
m(1)    =       0.0480934
m(2)    =       0.0385819
m(3)    =       0.0618759
m(4)    =       0.0554616
m(5)    =       0.0209402
Massa totale: 1
Valore atteso di f rispetto ad m: 1.51179
Misura orbita 0: 1
Misura orbita 1: 1
Misura orbita 2: 1
Misura orbita 3: 1
```

FIGURA 3. Secondo esempio di esecuzione del programma completo. (Qui $f$ è una permutazione.)