NUMERI CASUALI E SIMULAZIONE

NUMERI CASUALI

Usati in:

✔ statistica

✔ programmi di simulazione

....

Strumenti:

- tabelle di numeri casuali
- generatori hardware
- generatori software

DESCRIZIONE DEL PROBLEMA

Un programma che usa numeri casuali richiede un generatore software

☞ una macchina virtuale software è deterministica, quindi non può generare numeri veramente casuali

È possibile ideare algoritmi per la generazione di numeri *apparentemente casuali*

In questo caso otteniamo un generatore di numeri *pseudo-casuali*

REQUISITI PER UN PROGRAMMA GENERATORE

L'obiettivo teorico è generare *serie infinite* di numeri *statisticamente indipendenti* all'interno di un dato intervallo, ad esempio (0,1), intervallo aperto

Il generatore deve essere:

- ❑ efficiente - programmi di simulazione possono doverlo attivare milioni di volte
- ❑ ripetibile - la stessa sequenza deve poter essere rigenerata a piacere

Nel 1951 D.H. Lehmer ha proposto un algoritmo parametrico che, *per una opportuna scelta dei parametri*, ha superato numerosi test empirici di *casualità*

Generatore di Lehmer

Siano dati:

i) modulo: m intero, primo, *grande*

ii) moltiplicatore: a intero, $1 < a < m$

iii) funzione $f(z)$: $z_{n+1} = a * z_n \mod m$

iv) seme: z_1 intero, $1 \leq z_1 \leq m-1$

v) normalizzazione $u_n = z_n / m$

- poichè m è primo, la funzione non produce mai 0 per $1 \leq z \leq m-1$, quindi la sequenza non collassa mai a 0 (diversamente sarebbe possibile a1 * m1 * z1 * m2 mod m1 * m2 = 0)
- la normalizzazione v) non influenza l'apparente casualità della sequenza
- la sequenza non ha, ovviamente, nulla di casuale, ma per una opportuna scelta di a ed m non è distinguibile da una vera sequenza casuale
- fissati a ed m, risulta fissata la lunghezza del periodo p $(p<=m)$, tale che $z_p = z_1$
- definiamo *sequenza a periodo completo* una permutazione dei numeri 1,...,m-1
- esistono coppie di valori a ed m che generano *sequenze a periodo completo*
- il seme influenza soltanto il punto di partenza della sequenza, ma non l'ordine
- i valori di u saranno $1/m, 2/m, ..., (m-1)/m,$ con media → 1/2 e σ → $1/\sqrt{12}$

Esempio: **`f(z) = 6z mod 13 ...1,6,10,8,9,2,12,7,3,5,4,11,1...`**

UNA SOLUZIONE EFFICACE

- ☞ occorre trovare coppie di parametri *a* ed *m* a periodo completo con periodo sufficientemente lungo: per $m = 2^{31}-1$ esistono 534 milioni di moltiplicatori adatti
- ☞ occorre implementare la funzione f(z) in modo efficiente e coerente con le possibilità di rappresentazione numerica di una specifica macchina
- ❑ $a=16807$ ed $m = 2^{31}-1$ generano sequenze a periodo completo che hanno superato numerosi test statistici di casualità, ma, per una implementazione diretta della funzione di generazione, richiedono l'utilizzo di interi a 46 bit per contenere il massimo valore del prodotto $a * z$
- ❑ ogni sequenza deve essere inizializzata attribuendo un valore al seme
- ❑ la libreria `stdlib.h` del linguaggio C mette a disposizione la funzione

  `rand()`

  che produce un numero intero pseudo-casuale compreso fra `0` e `RAND_MAX` (costante pre-definita anch’essa in `stdlib.h`)

ESEMPIO DI UTILIZZO DI `rand()`

```
#include <stdlib.h>
#include <stdio.h>
#define N 10
int main(){
 int i,r;
 printf(“%d numeri a caso fra 0 e %d \n”,N,RAND_MAX);
 for (i=0;i<N;i++){ /* genera N numeri casuali */
  r = rand();
  printf(“%d\n”,r);
 }
 return 0;
}
```

- si noti che successive esecuzioni di questo programma producono sempre lo stesso numero, perche all’avvio del programma il *seme* viene sempre re-inizializzato a 1

- se si vogliono generare sequenze diverse a ogni esecuzione è possibile utilizzare la funzione `srand()` per assegnare un valore al seme
- si può riottenere la stessa sequenza, per riprodurre uno stesso esperimento, riassegnando lo stesso valore

```
#include <stdlib.h>
#include <stdio.h>
#define N 10
int main(){
 int i,r;
 unsigned int s; /* seme */
 printf("Inserire un numero per inizializzare la sequenza: ");
 scanf("%d",&s);
 srand(s);
 printf("%d numeri a caso fra 0 e %d \n",N,RAND_MAX);
 for (i=0;i<N;i++){ /* genera N numeri casuali */
  r = rand(); printf("%d\n",r);
 }
 return 0;
}
```

Generazione di un numero intero con distribuzione uniforme fra 0 e 1

☞ è sufficiente dividere il risultato di random per RAND_MAX

```
double randR(){
  return (double)rand()/RAND_MAX;
}
```

Generazione di uno fra n eventi con probabilità discreta p_i

☞ sia $P_k = \sum_{i=1}^{k} p_i, 1 \le k \le n$, probabilità cumulativa, $P_n = 1$

❑ si genera l'evento e_i se $P_{i-1} < randR \le P_i$ (P_0=0)

❑ la probabilità dell'evento e_i è pari all'ampiezza dell'intervallo $P_i - P_{i-1} = p_i$

Distribuzione normale (cfr. teorema del limite centrale)

- ❑ È noto che una distribuzione normale è il limite della somma di variabili casuali con distribuzione uniforme

- ☞ date DN variabili indipendenti con distribuzione uniforme, media m e deviazione standard σ, la funzione $z = (x_1+...+x_{DN}-DN*m)/\sigma *sqrt(12)$ approssima una variabile con distribuzione normale, valore medio 0 e deviazione standard 1

- ☞ la somma di 12 valori ottenuti da RandR approssima una distribuzione normale con media 6 e $\sigma=1$

Osservazione: **`randNorm`** produce valori nell'intervallo m ± 6 σ