Automi cellulari

Progetto d'esame del corso di "Programmazione e Laboratorio"

1 Introduzione

Scopo del progetto è realizzare un programma che consenta l'esplorazione del comportamento di semplici *automi cellulari*. Nelle sezioni seguenti, sarà definita la classe di automi cellulari sotto investigazione, saranno specificate le funzionalità del programma che servirà alla loro esplorazione (con particolare riferimento al formato dell'input/output a cui tale programma dovrà strettamente attenersi) e saranno indicate le modalità di consegna del progetto.

2 Automi cellulari

Un *automa cellulare* è costituito da una collezione di celle identiche che interagiscono l'una con l'altra. Le celle sono disposte regolarmente in una, o più dimensioni (ad esempio secondo una segmento, un quadrato, o un cubo). Gli elementi descrittivi del comportamento di un automa cellulare sono, per ogni cella,

- lo *stato*, in cui essa si trova ad ogni istante discreto di tempo,
- l'insieme delle *celle vicine* ad essa,
- la *dinamica* che ne descrive l'evoluzione di stato, in funzione del suo stato e di quello delle celle vicine.

In questo progetto l'attenzione è ristretta alla famiglia di automi cellulari, definita da Stephen Wolfram, per cui:

- le celle sono in numero finito, disposte lungo un segmento di lunghezza data L (numerate da 0 a $L-1$),
- lo stato $s_t(i)$ dell'i-esima cella al tempo $t \geq 0$ può essere 0, o 1,
- l'insieme delle celle vicine all'i-esima cella (per $0 < i < L-1$) è dato dalle celle di posizione $i-1$ e $i+1$,
- la dinamica è descritta da un numero intero ad otto bit $b_7 b_6 \ldots b_0$.

La dinamica può essere compresa grazie alle tabelle seguenti

$s_t(i-1)$	$s_t(i)$	$s_t(i+1)$	$s_{t+1}(i)$
0	0	0	b_0
0	0	1	b_1
0	1	0	b_2
0	1	1	b_3

$s_t(i-1)$	$s_t(i)$	$s_t(i+1)$	$s_{t+1}(i)$
1	0	0	b_4
1	0	1	b_5
1	1	0	b_6
1	1	1	b_7

che indicano come lo stato $s_{t+1}(i)$ in cui la cella i-esima (per $0 < i < L-1$) evolve al tempo $t+1$ è dato da uno degli otto bit della dinamica in dipendenza dalle otto possibili combinazioni al tempo t dei tre bit $s_t(i-1), s_t(i), s_t(i+1)$ corrispondenti al suo stato e a quello delle celle vicine. Si assume che lo stato delle celle di posto 0 e $L-1$ resti immutato.

Ad esempio, la dinamica data dal numero 54 (00110110 in binario) corrisponde alle tabelle

$s_t(i-1)$	$s_t(i)$	$s_t(i+1)$	$s_{t+1}(i)$
0	0	0	0
0	**0**	**1**	**1**
0	1	0	1
0	1	1	0

$s_t(i-1)$	$s_t(i)$	$s_t(i+1)$	$s_{t+1}(i)$
1	0	0	1
1	0	1	1
1	1	0	0
1	1	1	0

Dato un segmento con $L = 13$, se al tempo $t = 0$ l'unica cella con stato 1 è quella centrale, l'evoluzione temporale (fino a $t = 6$) dell'automa è descritta nella seguente tabella

t	$s_t(0)$	$s_t(1)$	$s_t(2)$	$s_t(3)$	$s_t(4)$	$s_t(5)$	$s_t(6)$	$s_t(7)$	$s_t(8)$	$s_t(9)$	$s_t(10)$	$s_t(11)$	$s_t(12)$
0	0	0	0	0	0	0	1	0	0	0	0	0	0
1	0	0	0	0	0	1	1	1	0	0	0	0	0
2	0	0	0	0	1	0	0	0	1	0	0	0	0
3	0	0	0	1	1	1	0	1	1	1	0	0	0
4	0	0	1	0	0	0	1	0	**0**	**0**	**1**	0	0
5	0	1	1	1	0	1	1	1	0	**1**	1	1	0
6	0	0	0	0	1	0	0	0	1	0	0	0	0

Prendiamo ad esempio lo stato della cella $i = 9$ al tempo $t = 5$, che è $s_5(9) = 1$ (in grassetto sottolineato, nella tabella dell'evoluzione). Tale valore è determinato dalla dinamica a partire dal valore dello stato della cella e delle vicine al tempo 4, ossia da $s_4(8), s_4(9), s_4(10)$: tre bit che valgono $0, 0, 1$ (in grassetto, nella tabella dell'evoluzione). La dinamica indica infatti che per $0, 0, 1$ (in grassetto, nella prima tabella della dinamica) al tempo $t = 4$, lo stato al tempo $t + 1 = 5$ deve essere 1 (in grassetto sottolineato, nella prima tabella della dinamica).

3 Il programma

Il programma deve simulare il comportamento di un automa cellulare di dimensione fissata e soggetto all'evoluzione secondo varie dinamiche.

Il programma ha come primo argomento sulla linea di comando un numero intero N (in decimale, compreso tra 1 e 10 000) che determina la dimensione dell'automa $L = 2N + 1$. Lo stato iniziale è pari a 0 per tutte le celle, tranne per quella di posizione centrale, che ha stato 1. Il programma legge quindi da `stdin` una sequenza (separata da "a capo" `\n`) di coppie di numeri interi (separati da uno spazio) che indicano rispettivamente la durata (in decimale, compresa tra 0 e 10 000) di un intervallo di tempo e la dinamica da seguire in quell'intervallo (in decimale, compresa tra 0 e 255).

A fronte di tali indicazioni, il programma emette su `stdout` l'evoluzione temporale degli stati dell'automa. Più precisamente, il carattere i-esimo della riga t-esima deve essere un punto (`.`), oppure un asterisco (`*`), a seconda che lo stato $s_t(i)$ dell'i-esima cella al tempo t sia 0, o 1 (i posti e le righe sono numerati da 0). Le righe sono separate tra loro da un solo carattere "a capo" e punti e asterischi su ogni riga sono contigui (ossia: non separati da alcuno spazio).

Ad esempio, se l'argomento sulla linea di comando è 6 e lo `stdin` è dato dal contenuto del riquadro di sinistra

```
2 54
1 255
3 24
```

```
......*......
.....***.....
....*...*....
.**********.
.*..........
..*.........
...*........
```

il programma deve produrre su `stdout` il contenuto del riquadro di destra, che corrisponde allo stato iniziale (sulla prima riga), all'evoluzione per 2 passi con la dinamica 54 (seconda e terza riga), 1 passo con la dinamica 255 (quarta riga) e quindi 3 passi con la dinamica 24 (righe restanti).

Si ribadisce nuovamente che il formato dell'input/output deve attenersi strettamente a quanto specificato in questa sezione: non deve essere letto null'altro che numeri interi (separati da spazi e "a capo") e scritto null'altro che asterischi e punti (separati da "a capo"). **In mancanza di completa e stretta aderenza alle indicazioni precedenti il progetto non sarà ritenuto valido.**

4 Modalità di consegna

La consegna del progetto avviene tramite le risorse di calcolo del laboratorio SILab. Lo studente che avesse sviluppato il progetto utilizzando risorse diverse (ad esempio il proprio computer di casa), dovrà trasferirne i file, prima della consegna, nella sua home del SILab ed accertarsi che il suo programma compili e funzioni correttamente in tale ambiente.

Programmi consegnati diversamente da come specificato in questa sezione, o che non compilano ed eseguono correttamente nell'ambiente di calcolo del SILab non saranno ritenuti validi.

Più precisamente, deve essere possibile compilare il programma invocando il comando `gcc -lm -Wall *.c` in una directory che contenga tutti e soli i file sorgenti relativi al progetto. In particolare, se il programma si basa su più di un file sorgente, tra quelli con estensione `.c` deve esisterne uno e uno solo che contenga una funzione di nome `main`. Una volta compilato il programma, la sua esecuzione (su input di formato specificato) non deve produrre errori.

Lo studente deve consegnare il progetto invocando il comando `/users/ms000123/consegna` messo a disposizione dal docente. Il comando va invocato passando come parametri i nomi di tutti e soli i file sorgenti necessari alla compilazione del programma.

Assumendo ad esempio che tali file siano contenuti nella directory corrente e abbiano nome `automa.c`, `lib.c` e `lib.h`, la corretta invocazione è

```
$ /users/ms000123/consegna automa.c lib.c lib.h
```

l'output del comando indica se la consegna è andata a buon fine, oppure se non è avvenuta (a causa, ad esempio, di un errore di compilazione).

Il comando `/users/ms000123/verifica` messo a disposizione dal docente può essere invocato, senza argomenti, per verificare l'esito della consegna.

Lo studente è libero di effettuare più volte la consegna, **in caso di consegne multiple, verrà considerata valida solo l'ultima consegna in ordine temporale**. Poiché il processo di consegna potrebbe non andare a buon fine al primo tentativo (a causa di errori vari), si consiglia vivamente di provare a consegnare una versione preliminare del progetto appena possibile, per non trovarsi a dover risolvere problemi inaspettati in prossimità della scadenza.

Si ricorda che chi intende sostenere l'esame nell'appello del 14 giugno 2006 deve effettuare la consegna entro e non oltre il 7 giugno 2006; similmente, chi intende sostenere l'esame nell'appello del 5 luglio 2006 deve effettuare la consegna entro e non oltre il 28 giugno 2006. Fa fede la data riportata dal comando `/users/ms000123/verifica`.

Una volta consegnata la versione definitiva del progetto lo studente deve *stampare una copia di tutti i file sorgenti che porterà con se all'esame orale*.