Relazione progetto

Jacopo Maria Corvino

Settembre 2023

1 Introduzione

Il progetto consiste nell’implementazione di due programmi in C++ per la simulazione di un’epidemia all’interno di una popolazione chiusa. La scelta adottata è stata quella di sviluppare un programma in cui viene implementato un modello SIR e un programma in cui viene simulato il contagio tramite automa cellulare.

2 Parte I: Modello SIR

Il primo programma è diviso in 4 file: un *header* file "*1sir.hpp*" in cui viene fornita un’interfaccia principale, un source file "*1sir.cpp*" dove vengono implementate le entità dichiarate nell’interfaccia, un file “*1sir.test.cpp*” in cui vengono eseguiti test attraverso *Doctest*, e un file “*1main.cpp*” contenente la funzione *main*.

2.1 Scelte implementative

All’interno del programma vengono utilizzate principalmente due strutture dati: una *struct* *SIR* e una class *Pandemic*. La prima è caratterizzata da cinque numeri, cioè rispettivamente le persone suscettibili *(S)*, infette *(I)*, rimosse *(R)*, totali *(N)* e vaccinate *(V)*. La seconda è caratterizzata invece da una oggetto di tipo SIR nella parte privata, e nella parte pubblica da un costruttore e da un metodo a cui vengono passati i valori di *γ* e *β*, il numero di giorni per cui si sviluppa l’epidemia e la scelta o no di vaccinazione durante quest’ultima.

All’interno della classe viene anche definita la funzione *infection* che si occupa di calcolare ogni giorno il numero di persone suscettibili, infette, rimosse, ed eventualmente vaccinate. Applicando la funzione *infection* ad un oggetto di tipo *Pandemic* si ottiene un vettore contenente tanti oggetti di tipo *SIR* quanti sono i giorni dell’epidemia osservati stabiliti, dove ogni oggetto descrive la situazione epidemiologica di un singolo giorno.

2.2 Test e gestione errori

È stata implementata la funzione *CheckImput*, utilizzata per verificare con delle *exceptions* che i parametri iniziali scelti dall’user siano validi per la simulazione. Inoltre, ci sono anche diversi *assert* che regolano l’andamento della funzione *infection*, questo per evitare risultati non verosimili o un improvviso blocco del programma.

Attraverso il file *1sir.test.cpp* è possibile testare il funzionamento delle funzioni *CheckImput* e *infection*. Vengono testati in particolare il caso in cui il *I* sia uguale a 1 ed il caso in cui *I* sia uguale a *N*, nei quali è stato scelto di far guarire direttamente gli infetti.

2.3 Compilazione ed Esecuzione

La compilazione ed esecuzione del programma si effettuano tramite i comandi:

1. g++ 1sir.cpp 1main.cpp
2. ./a.out

Sono state implementate due funzioni per l’interfaccia su terminale con l’utente: la funzione *Print* effettua la stampa dei valori contenuti nel vettore finale sotto forma di una tabella, permettendo di visualizzare per ogni giorno *(D)* i valori di *S*, *I*, *R*, *N* e *V*; la funzione *Intro* stampa delle frasi introduttive di presentazione alla simulazione, che inizierà chiedendo all’user di inserire i parametri iniziali *T* (numero di giorni), *N*, *I*, *β* e *γ*.

Immagine che contiene testo, schermata, schermo

Descrizione generata automaticamente

Figura 1: Output su terminale.

Intro, input dell’user, tabella con i risultati della simulazione

Per eseguire i test bisognerà invece compilare utilizzando i comandi:

1. g++ 1sir.cpp 1sir.test.cpp
2. ./a.out

3 Parte II: Simulazione tramite automa cellulare

Il programma comprende 1 *header* file e 3 file di implementazione. Il file “*2cell.hpp*” contiene le strutture dati utilizzate per l’implementazione del modello, le dichiarazioni delle loro funzioni membro e delle funzioni libere, in corrispondenza il file “*2cell.cpp*” contiene le definizioni di quest’ultime. Inoltre, vi sono il file di implementazione per i test “*2cell.test.cpp*” e quello per la stampa dei risultati dell’elaborazione dati su terminale “*2main.cpp*”.

3.1 Scelte implementative

Definito un enumeratore *Cell* per le tre tipologie di persone presenti sulla griglia (*s*, *i*, *r*); è stata costruita una *struct* *Point* che servirà per contenere le coordinate di una cellula nella griglia; è stato scelto un alias per il *type alias* “*grid\_t*” per abbreviare il *type* *std::vector<vector<Cell>>* di un oggetto “griglia” e infine una classe *World*.

La classe World comprende i parametri *L\_0* (lato della griglia), *S* (suscettibili), *I\_0* (infetti), *R* (rimossi), *V* (vaccinati), *Q* (quarantena), *D* (giorno), e *grid\_0* (griglia di un singolo giorno), e contiene i metodi che permettono lo svolgimento della simulazione. Quest’ultima, infatti, inizia dall’oggetto “mondo” di tipo *World* contenente i parametri iniziali, al quale viene applicato il metodo *setWorld* che si occupa di creare la griglia che rispecchia la situazione iniziale. Dopodiché entra in funzione il *loop for* che si occupa di modificare le informazioni contenute in “mondo” e di stampare su terminale con il metodo *draw\_grid*.

In particolare, all’interno del loop, il metodo *day* svolge un ruolo importante: verifica ed eventualmente effettua lo stato di quarantena e di vaccinazione; infetta cellule suscettibili basandosi sulla probabilità di infezione *β* data dall’user all’inizio del programma; aggiorna i valori dei parametri *S*, *I*, *R* e *D*; aggiorna la griglia da stampare.

Per quanto riguarda l’infezione nel caso la persona sia suscettibile, vengono conteggiate innanzitutto le persone infette che occupano le posizioni circostanti tramite la funzione *HowManyCloseInfeccted*. Il parametro *β* è stato inteso come una semplice misura della possibilità di contagio in seguito al contatto con una delle 8 persone adiacenti alla persona considerata, da cui la probabilità di contagio effettiva risulta essere *β ·* (*persone adiacenti*/8). Quindi viene estratto un numero casuale attraverso una distribuzione uniforme tra 0 e 1: se minore della probabilità di contagio, la cellula viene infettata.

In quanto la griglia è stata definita come un vettore contenente L vettori i-esimi che a loro volta contengono L oggetti j-esimi di tipo *Cell*, le funzioni *HowManyCloseInfected*, *position\_s* e *draw\_grid* si basano su un ciclo for che scorre tra i vettori i-esimi al cui interno c’è un altro ciclo for che scorre tra gli elementi j-esimi di ogni vettore. Ogni oggetto rappresenta una cellula, mentre i vettori i-esimi e gli oggetti j-esimi rappresentano righe e colonne della griglia.QQ

3.2 Test e gestione errori

La valutazione della correttezza dei valori per le variabili del modello è gestita dalla funzione *CheckImput*, la quale tramite delle *exceptions* verifica la validità dei parametri iniziali dati dall’user, e da degli *assert* contenuti nelle altre funzioni. Attraverso il file 2cell.test.cpp è possibile testare il funzionamento delle funzioni *CheckImput*, *position\_s*, *HowManyCloseInfected* e day.

3.3 Compilazione ed Output

La compilazione e l’esecuzione del programma si effettuano tramite i comandi:

1. g++ 2cell.cpp 2main.cpp
2. ./a.out

Una volta avviato il programma, l’utilizzatore viene invitato ad inserire su standard input: il lato della griglia, il numero di giorni di simulazione, il numero iniziale di infetti, *β* e *γ*.

Immagine che contiene testo, schermata, software, Carattere

Descrizione generata automaticamente

Figura 4: Output su terminale:

Intro, imput dell’user

Vengono poi stampate le griglie corrispondenti ai vari giorni. Ne segue una come esempio:

Immagine che contiene schermata, testo, Software per la grafica

Descrizione generata automaticamente

Figura 4: Output su terminale:

Griglia: "s" in bianco indica le cellule suscettibili, "r" in verde le rimosse, "i" in rosso le infette

Tabella: giorno, cellule totali, suscettibili, infette, rimosse, vaccinate, stato di quarantena (1=on, 0=off)

Infine, per eseguire i test bisognerà invece compilare ed eseguire utilizzando i comandi:

1. g++ 2cell.cpp 2cell.test.cpp
2. ./a.out