Autore: Galasso Francesco, Tomassoni Lorenzo

RELAZIONE PROGETTO D’ESAME

CORSO DI PROGRAMMAZIONE

Bologna, 2022/23

# INDICE:

1. Scopo del progetto
2. Descrizione generale e scelte progettuali
3. Calcolo probabilistico
4. Istruzioni su modalità di compilazione, testing ed esecuzione
5. Interpretazione e analisi dei risultati in output
6. Strategia di testing

# SCOPO DEL PROGETTO

Lo scopo della **prima parte** del progetto è quello di simulare la diffusione di un’epidemia tra la popolazione, seguendo il modello teorico SIR.

La **seconda parte** del progetto invece si basa sulla rappresentazione grafica della diffusione epidemica in una popolazione in movimento all’interno di una griglia toroidale 40x40 e con la possibilità di avere individui vaccinati più resistenti al contagio.

# DESCRIZIONE GENERALE E SCELTE IMPLEMENTATIVE

Il modello SIR è il più semplice modello teorico utilizzato per descrivere il comportamento di un’epidemia che si diffonde nella popolazione. Ogni individuo rientra in uno dei seguenti gruppi:

* + - Suscettibili (S)
    - Infetti (I)
    - Rimossi (R)

Secondo questo modello, è possibile un trasferimento di gruppo per un individuo solo nella modalità:

𝑆𝑢𝑠𝑐𝑒𝑡𝑡𝑖𝑏𝑖𝑙𝑒 → 𝐼𝑛𝑓𝑒𝑡𝑡𝑜 → 𝑅𝑖𝑚𝑜𝑠𝑠𝑜

Il trasferimento è regolato da tre equazioni differenziali:

Il parametro 𝛽, il cui valore è compreso tra 0 e 1, rappresenta il tasso di infettività dell’epidemia, cioè la probabilità che un infetto contagi un suscettibile; mentre il parametro 𝛾, compreso tra 0 (escluso) e 1, rappresenta il tasso di recupero ed è l’inverso del tempo medio di malattia

Discretizzando le equazioni per ∆𝑡 = 1 giorno, si ottiene:

Rispettando sempre la condizione: 𝑆 + 𝐼 + 𝑅 = N *(Popolazione)* 𝑆, 𝐼, 𝑅 ∈ N

* **Prima parte:**

Per la simulazione, si prendono in input i parametri che determinano la diffusione dell’epidemia.

Altri parametri sono il tasso di infettività e il tasso di rimozione, che compongono la *struct* “Parameters”.

Il numero dei suscettibili, degli infetti e dei rimossi costituiscono i valori degli elementi della *classe* “SIR”.

Infine, il numero di giorni di studio, maggiore di 0 ed intero, indica per quanti giorni verrà monitorata l’evoluzione della popolazione, a meno che il numero totale di infetti non diventi 0, perché ciò comporta l’interruzione della diffusione dell’epidemia.

Verificata la correttezza degli input tramite le funzioni “insert\_parameter()” per i parametri razionali (,) e “insert\_people()” per i parametri interi (suscettibili, infetti, rimossi e la durata in giorni) si può osservare il risultato della diffusione epidemica.

Tramite una funzione “print\_SIR()” vengono stampati su standard output il numero dei suscettibili, degli infetti e dei rimossi ed eventualmente il numero del giorno in cui la simulazione è stata interrotta.

I numeri di suscettibili, infetti e rimossi possono inoltre essere visualizzati in un file “results.txt”, che verrà creato alla fine della simulazione.

Visto l’apparente contrasto tra il carattere razionale delle equazioni e la necessità di garantire che il totale dei componenti dei tre gruppi sia sempre intero (in quanto numero totale di individui), si è scelto di approssimare ad intero il numero degli individui per ogni gruppo tramite la funzione “static\_cast<int>()” e poi separare la parte decimale da quella intera sottraendo al numero degli individui di ogni gruppo il numero approssimato precedentemente. A seguire vengono sottratte dal numero totale di persone le approssimazioni e ciò che si ottiene lo si confronta con gli scarti decimali e avviene una distribuzione di elementi nei vari gruppi dando la precedenza al gruppo con scarto maggiore. Vengono quindi approssimati ad interi i valori dei suscettibili, degli infetti e dei rimossi in modo che la loro somma sia costante ed intera.

* **Seconda** **parte:**

Nella seconda parte del progetto abbiamo scelto di visualizzare graficamente la diffusione dell’epidemia tramite una griglia 40x40 toroidale in cui ogni casella è a sua volta 3x3 in modo da migliorare qualitativamente la visualizzazione da parte dell’utente nel caso avvenga una sovrapposizione tra due o più elementi.

Per la parte grafica abbiamo usato *SFML*: un’interfaccia di programmazione che facilita lo sviluppo di giochi e applicazioni multimediali. È composto da cinque moduli: sistema, finestra, grafica, audio e rete.

Abbiamo scelto di assegnare i seguenti colori agli individui della popolazione:

Colore verde: suscettibile

Colore rosso: infetto

Colore blu: vaccinato

Per avere una rappresentazione quantitativa e non solo qualitativa dell’andamento dell’epidemia sono presenti i numeri in continuo aggiornamento dei 3 gruppi (S, I, R) nella barra bianca sotto la griglia.

Ci siamo affidati ad un modello di popolazione dinamico, basato sullo spostamento randomico in una casella, assegnando continuamente ad ogni elemento le due componenti della quantità di moto comprese tra -1 e 1.

Gli assi sono stati orientati in modo tale che uno spostamento verso destra abbia la componente della quantità di moto orizzontale positiva e uno spostamento verso l’alto abbia la componente verticale positiva.

Per la simulazione, si prendono in input i parametri che determinano la diffusione dell’epidemia.

Il numero dei suscettibili e degli infetti è interno all’oggetto definito dalla *classe* *“*Population” che contiene un vettore composto da elementi definiti dalla *classe “*Person*”* con attributi: la posizione nella griglia, le componenti della quantità di moto e, da notare, il *bool* “is\_vaccinated”, cruciale per l’assegnazione del vaccino. “Population” contiene anche i parametri razionali e ,da inserire in input, appartenenti alla *enum class* “Status” che sono legati rispettivamente al tasso di infezione e rimozione. Agli individui viene poi assegnata una posizione che segue la distribuzione di probabilità uniforme nell’intervallo da 0 a 1600.

Viene poi chiesto all’utente se vuole visualizzare o meno la griglia in modo da avere un riferimento spaziale sulla posizione dei singoli individui.

A differenza della prima parte del progetto abbiamo aggiunto la possibilità di vaccinare, e quindi rafforzare la difesa immunitaria dei suscettibili, per combattere la diffusione dell’epidemia.

La vaccinazione, se resa possibile dall’utente, inizierà non appena una certa percentuale di rimossi decisa in input sarà raggiunta. Abbiamo scelto di vaccinare, ad ogni ciclo, l’1% della popolazione.

Verificata la correttezza degli input tramite le funzioni “insert\_parameter()” per i parametri razionali (,) e “insert\_people()” per i parametri interi (suscettibili, infetti, rimossi e la durata in giorni) si può osservare il risultato della diffusione epidemica.

La diffusione epidemica a questo punto rispetta la seguente logica:

se un infetto entra nella stessa casella di un suscettibile avverrà l’infezione secondo una probabilità dipendente da .

Un individuo infetto ha una probabilità di essere rimosso dipendente da .

Se un suscettibile è vaccinato avrà una probabilità minore di diventare infetto.

­­

# CALCOLO PROBABILISTICO

# Nel paragrafo precedente, riguardo alla seconda parte, si è parlato di probabilità influenzata dai parametri β, γ e dal vaccino. Ora analizzeremo nel dettaglio come è gestita la componente aleatoria della simulazione.

# Con l’ausilio dell’*header* <random> della *standard library*, ci siamo serviti della distribuzione di probabilità uniforme all’interno di un intervallo fissato e abbiamo simulato i seguenti processi stocastici:

# Random walk:

# Il random walk o passeggiata aleatoria è il processo per il quale lo spostamento degli individui dipende solo da un numero generato in un intervallo da -1 a 1 assegnato alla componente della quantità di moto. Tale spostamento è simulato nella funzione “evolve()”, il cuore della simulazione.

# Infezione e rimozione:

# L’infezione, causata dalla condivisione della posizione di un suscettibile con un infetto, avviene all’interno della funzione “infection()” nella quale è generato ad ogni ciclo un numero reale nell’intervallo da 0 a 1.

# Se tale numero generato è minore di β allora l’individuo suscettibile a contatto con l’infetto diventa tale.

# La rimozione, causata dallo stato infetto dell’individuo, è anch’essa confrontata con un numero reale compreso tra 0 e 1 e avviene lo stesso procedimento descritto per l’infezione, questa volta però se tale numero è minore di γ, l’individuo viene rimosso dalla popolazione ed eliminato dal vettore contenente le persone.

# Vaccino

# Il vaccino gioca un ruolo importante nel contrasto della diffusione dell’epidemia perché sia nel caso dell’infezione che nel caso della rimozione moltiplica il numero generato per 1.3 incrementandolo così del 30% e rendendo più difficile la condizione di infezione o rimozione.

# Il vaccino viene distribuito ad ogni ciclo all’1% della popolazione suscettibile.

# ISTRUZIONI SU MODALITA’ COMPILAZIONE, TESTING ED ESECUZIONE

* **Prima parte:**

Per compilare il programma, inserire su terminale il seguente comando:

g++ -Wall -Wextra -g -fsanitize=address sir.hpp

Seguito da:

g++ -Wall -Wextra -g -fsanitize=address first\_part.cpp sir.cpp -o sir

Per eseguirlo:

./sir

Per compilare i test, inserire su terminale il comando:

g++ -Wall -Wextra -g -fsanitize=address sir.hpp

Seguito da:

g++ -Wall -Wextra -g -fsanitize=address sir.test.cpp sir.cpp -o test

Per testare il programma:

./test

* **Seconda parte:**

Per compilare il programma, inserire su terminale il seguente comando:

# INTERPRETAZIONE E ANALISI DEI RISULTATI

Come previsto, lunghi tempi di malattia incidono in modo significativo sull’andamento dell’epidemia, perché permettono una maggiore diffusione anche se il tasso di infettività è contenuto.

# STRATEGIA DI TESTING

* **Prima parte:**

Per verificare il corretto funzionamento del codice, sono stati scritti diversi test, raccolti nel file

𝑠𝑖𝑟. 𝑡𝑒𝑠𝑡. 𝑐𝑝𝑝.

I test consistono nel verificare: la corretta costruzione del modello SIR, se i metodi che leggono ed impostano i valori degli attributi facciano il loro compito, e infine, che l’operatore “==” uguaglianza definito per due oggetti SIR restituisca il valore booleano aspettato.