TECNICHE DI PROGRAMMAZIONE

11/06/19

Simulazione d'esame: Soluzione lab 12 (seconda parte).

Come prima cosa, abbiamo capito che i vertici sono collegati tra di loro in modo forte (ogni vertice è collegato con tutti gli altri). Non siamo quindi costretti a calcolare il cammino minimo tra tutti i vertici che hanno gli agenti liberi e il vertice in cui è avvenuto il crimine perché vale la disuguaglianza triangolare e quindi il cammino diretto sarà sempre il più breve.

Partiamo quindi dalla classe Evento. Abbiamo 3 tipi di evento:

1. L’evento criminoso ( quello che andiamo a simulare).
2. L’evento “agente arriva sul posto”.
3. L’evento “crimine terminato” -> l’agente si libera.

Nella classe Evento andiamo subito a definire i 3 tipi possibili mediante “enum”. Dopodiché, questa classe avrà sicuramente un campo “tipoEvento” per disciminare i vari tipi di evento. Avremo un campo su cui si farà l’odrinamento (nel nostro caso LocalDateTime). Come ultima cosa, suppongo di avere un riferimento al crimine considerato quindi mi creo un campo “Event” (la classe che modellava le righe della tabella).

Mi faccio generare i vari Getter/Setter, il costruttore e il Comparable in quanto questi eventi dovranno essere ordinabili ( dobbiamo inserirli in una priority queue). Il criterio in base al quale ordiniamo gli eventi è la data!



Inoltre, nel simulatore ci siamo dimenticati di creare un riferimento al grafo per recuperare le distanze dai distretti e la PriorityQueue.

Ora andiamo a compilare i due metodi principali; init() e run().   
Nell’init andremo a ricevere i vari parametri (N, anno, mese, giorno e grafo) e a settare tutti gli attributi del Simulatore. Avremo anche le inizializzazioni di tutte le strutture dati che utilizzeremo. Impostiamo la variabile “malGestiti” uguale a 0.

Bisogna creare la mappa di “agenti”. Ricordiamo che abbiamo scelto di avere una mappa in cui le chiavi sono i distretti e i valori sono il numero di agenti liberi attualmente in quel distretto.

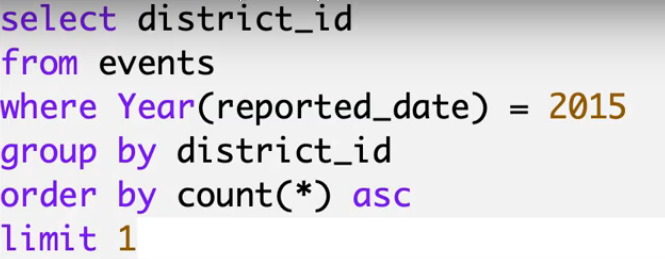


Sempre nel metodo “init” andiamo quindi a creare questa mappa e a inizializzarla con i valori iniziali. Per far questo, attraverso un “for”, impostiamo i valori di ogni distretto uguale a 0.

Immagine che contiene screenshot

Descrizione generata automaticamente

Dobbiamo scegliere ora dove si trova la centrale ( cioè dobbiamo trovare il distretto che ha minore criminalità). Ci serve una query in più (devo farmelo dire dal DAO). Creiamo quindi il DAO e da esso ci facciamo dire qual è il distretto a minor criminalità mediante il metodo “getDistrettoMin”. La qurey che dovrà contenere questo metodo è la seguente:



Il comando “limit 1” ci permette di prendere solamente il primo valore tra tutte le soluzioni. Così facendo, otteniamo il distretto con minor criminalità.

Mi faccio creare il metodo del DAO e inserisco questa query. Aggiungo try-catch come sempre e ritorno con un “if” il distretto corrispondente.

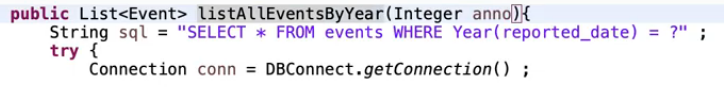
Immagine che contiene screenshot

Descrizione generata automaticamente

Torniamo nella classe Simulatore e continuiamo con il metodo “init” in cui eravamo arrivati al punto di cercare il distretto a minor criminalità per creare la centrale e inserire tutti agli agenti (fa sempre parte dell’inizializzazione dei valori iniziali). Con il metodo “getDistrettoMin” abbiamo quindi trovato il distretto e ci inseriamo gli agenti.

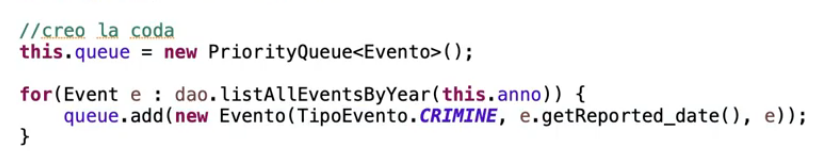


Il metodo “init” dovrà contenere anche la creazione della coda. Per quanto riguarda invece gli eventi iniziali, possiamo inserire i crimini avvenuti nell’anno che stiamo analizzando. Abbiamo già un metodo “listAllEvents” nel DAO che ci restituisce tutti gli eventi ma non in base all’anno. Basta modificarlo aggiungendo come parametro l’anno. Cambiamo il nome al metodo in “listAlleEventByYear”.



Ricordarsi di settare il valore del punto di domanda.

Posso quindi usare questo metodo per settare gli eventi iniziali. Scorro quindi gli eventi della tabella e aggiungo alla coda un nuovo evento in cui il tipo è “CRIMINE” e a cui passo tutto l’evento.



Posso passare al metodo “run()”. In questo metodo avremo come sempre il ciclo while che estrarrà gli eventi fino a quando ce ne sono.

L’esercizio chiedeva di modellare anche il mese e il giorno quindi dobbiamo andare a modificare il metodo del DAO e specificare anche questi due valori nella query.

Immagine che contiene screenshot

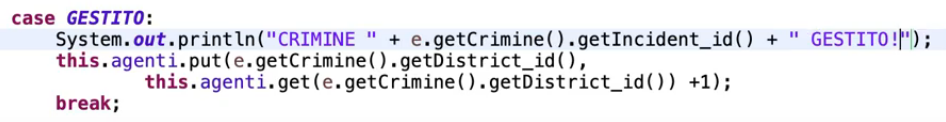
Descrizione generata automaticamente

Nel ciclo while ci sarà il solito “switch-case” e i 3 casi:

Immagine che contiene screenshot

Descrizione generata automaticamente

Partiamo dal caso “Gestito” nel quale dobbiamo liberare l’agente. Incrementiamo di 1 il numero di agenti liberi nel distretto dove c’è stato il crimine (usiamo la mappa). Stampiano infine il risultato.



Per il caso “Arriva\_Agente” mi definisco una variabile durata in cui suppongo di avere una funzione che mi dia una durata rispetto al tipo di evento. Andiamo a definire il metodo “getDuration” (privato) sempre all’interno del Simulatore. Questo metodo deve randomizzare il numero di ore che impiegherà l’agente per risolvere il crimine.

Immagine che contiene screenshot

Descrizione generata automaticamente

Una volta trovata la durata, posso schedulare l’evento “Gestito”. La data sarà uguale a quella di arrivo dell’agente più la durata della gestione del crimine.

Dopo averlo schedulato, devo solo più controllare se il crimine è mal gestito. Un crimine è mal gestito se un agente arriva sul posto 15 minuti dopo la sua segnalazione. Possiamo usare una funzione che ci offre LocalDateTime. Se la data di arrivo dell’agente è dopo la data del crimine più 15 minuti, allora è mal gestito.

Immagine che contiene screenshot

Descrizione generata automaticamente

Passiamo infine all’evento “Crimine”. Qui dobbiamo selezionare l’agente più vicino e settarlo a “occupato”. Abbiamo però due casi limite:

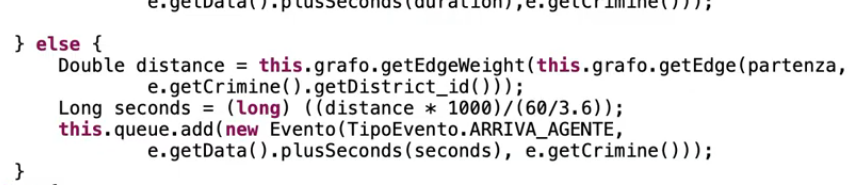
1. Agente libero non presente (sono tutti occupati).
2. Agente libero nello stesso distretto del crimine; ci mette 0 minuti per arrivare sulla scena del crimine ed è quindi inutile schedulare l’evento “Arriva\_Agente”.

Definiamo una variabile “partenza” che sarebbe il distretto di partenza dell’agente e lo impostiamo uguale a “null”. Poi impostiamo che il valore di questa variabile sia uguale ad una cerca funzione “cercaAgente” ( la creiamo dopo). Se questa variabile è diversa da “null”, vuol dire che c’è un agente libero, altrimentei non c’è nessuno agente libero ( malGestiti++). Nel caso in cui l’agente sia libero, gestisco l’evento in cui il distretto sia uguale a quello in cui è avvenuto il crimine. Se così fosse, è come se ci trovassimo nell’evento “Arriva\_Agente”, in cui l’agente è arrivato sul posto. Posso copiare quindi il codice da quell’evento senza considerare il caso in cui l’agente sia arrivato dopo 15 minuti (non serve). Per adesso il codice è il seguente:

Immagine che contiene screenshot

Descrizione generata automaticamente

Se il distretto è un altro lontano, dobbiamo calcolare la distanza tra i due distretti e vedere quanto ci mette l’agente ad arrivare sapendo che ha una velocità di 60km/h. La distanza la possiamo recuperare dal grafo attraverso il peso dell’arco. Per trovare i secondi mi basta fare spazio fratto tempo (meglio lavorare con i secondi). Una volta trovati, creo un nuovo evento “Arriva\_Agente” in cui sommo questi secondi trovati.



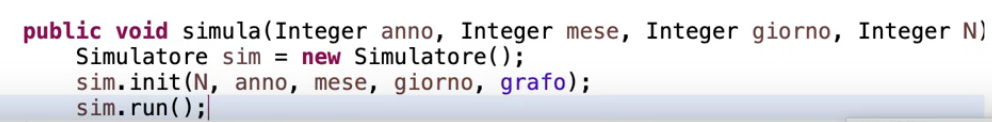
Come ultima cosa, dobbiamo creare il metodo “cercaAgente” definito in precedenza. Lo andiamo a creare esternamente sempre nel Simulatore. Per trovare l’agente più vicino usiamo la tecnica di creare una variabile “distanza” con un valore estremamente alto. Successivamente, scorriamo i valori e ogni volta che troviamo un valore più piccolo lo sostituiamo. Dobbiamo però considerare il caso limite in cui c’è un agente libero nel distretto del crimine. In questo caso non serve farsi dare la distanza ma basta imporre la distanza uguale a 0.

Immagine che contiene screenshot

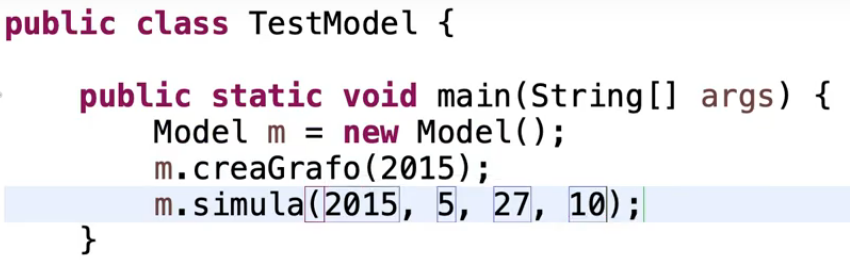
Descrizione generata automaticamente

Il metodo deve poi restituire il distretto. Abbiamo terminato con la classe Simulatore.

Definiamo ora nel modello il metodo “simula” a cui ci passiamo i parametri (anno, mese, giorno e N). Al suo interno ci creiamo il Simulatore e chiamiamo i metodi “init” e “run”.



Come ultima cosa, andiamo nel TestModel e proviamo il nostro Simulatore con dei valori a caso.



FINE.