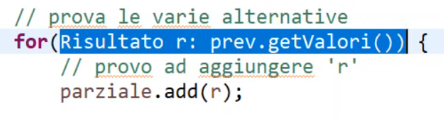
TECNICHE DI PROGRAMMAZIONE

02/04/19

ESERCIZIO SULLA RICORSIONE (Quadrato Magico).

Riprendiamo un attimo l’esercizio del totocalcio. Il nostro obbiettivo ero quello di costruire una schedina a partire da un pronostico. Quindi il pronostico è lo spazio del problema che vogliamo esplorare, che ci riassume quali sono tutte le alternative possibili da esplorare, mentre la schedina sono le soluzioni che noi stiamo trovando all’interno di questo spazio.

Noi abbiamo deciso che ogni livello della ricorsione mette a posto una partita, un risultato, una riga successiva della schedina.



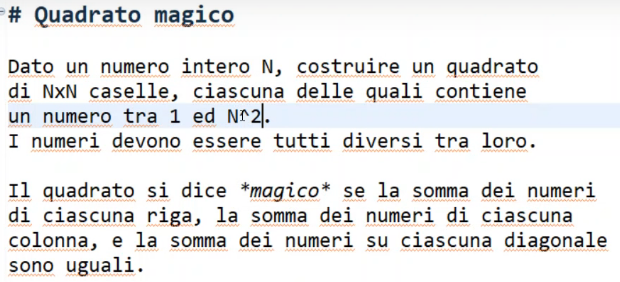
Attraverso queste righe di codice noi andiamo a vedere quali sono tutte le possibili alternative di valori da poter inserire in quel livello.

Il problema principale di questi problemi è il significato che noi diamo ad ogni livello e alla soluzione parziale che noi stiamo costruendo. Concettualmente, invece, è lo stesso meccanismo.

Abbiamo modificato il codice del programma “TotoCalcio” in modo da poter fare la visualizzazione non con un println ma attraverso una lista. Andare a vedere il programma aggiornato per vedere i cambiamenti (niente di particolarmente difficile).

Analizziamo ora il problema del QUADRATO MAGICO.

L’obiettivo di questo programma è il seguente:



Quindi se per esempio viene dato come numero il 5, noi dobbiamo creare un quadrato di lato 5 (con 25 spazietti), all’interno del quale inseriamo i numeri da 1 a 25 e tali che la somma di ciascuna riga, colonna, diagonale sia uguale.

Per sapere quale è il valore della somma delle diverse righe, basta conoscere appunto il valore del lato (N).

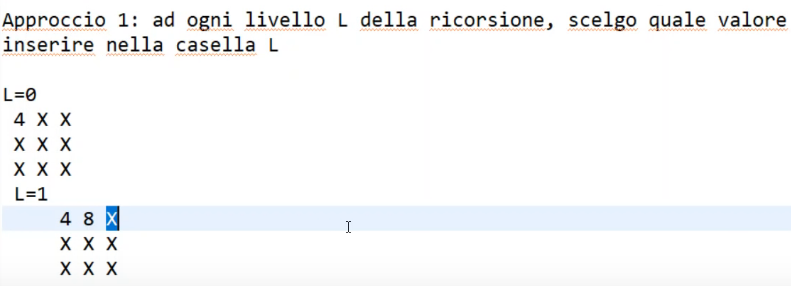
Nel caso in cui il lato sia per esempio 3, il valore della somma delle righe è:

La somma dei valori da 1 a 9 (dove 9 sarebbe N^2) diviso N che è il numero di righe (o colonne). Nel nostro

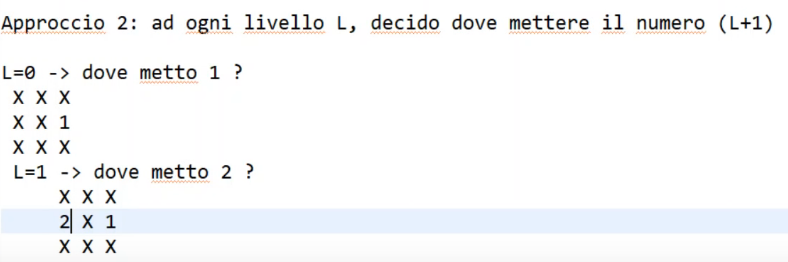
Caso sarebbe quindi 45/3=15.



Il nostro obiettivo è quello quindi di riempire una matrice 3x3 (in generale NxN), inizialmente vuota. Per fare ciò possiamo usare 2 diversi approcci:

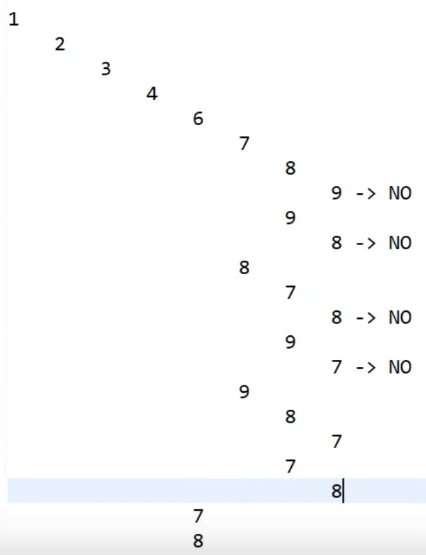


Arrivati al livello N^2 avremo riempito tutta la nostra matrice di numeri diversi tra loro. Da quel punto dobbiamo ancora controllare che la somma di tutte le righe, colonne e diagonali sia uguale.



Entrambi gli approcci ci permettono solamente di riempire la matrice, non di avere un quadrato magico. Inoltre, non c’è un metodo migliore dell’altro.

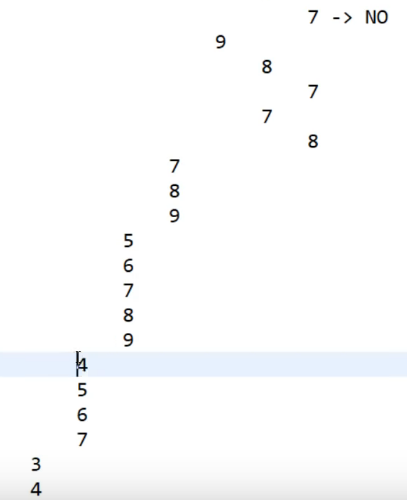
Noi abbiamo deciso di usare il secondo metodo, inserendo per ogni livello il numero N+1 in ordine. Successivamente il professore ha dunque spiegato come procederebbe la ricorsione in questo caso.



Incominciamo con la procedura di lvl 0 in cui inseriamo il primo numero (1) nella prima casella; al livello 1 inseriamo il 2 nella seconda casella e così via.

Arrivati al livello 8 inseriamo l’ultimo numero (9) ed arriviamo al caso terminale, ma ci accorgiamo che il quadrato non è magico. Quindi arrivati a questo punto, la procedura lvl 8 ha finito, l’unico tentativo che aveva a disposizione l’ha provato a fare ma senza buono esito. Torniamo quindi indietro alla procedura 7 che ha inserito il numero 8 ma poteva anche inserire il numero 9. Proviamo quindi a inserire il numero 9 e, nella procedura 8 inserisco il numero 8. Non va di nuovo bene quindi andiamo indietro fino alla procedura lvl 6 che proverà prima a mettere l’8 e poi il 9. Andiamo avanti così fino a quando non otteniamo il quadrato magico.

Adesso, concettualmente, scrivere un programma così sembra molto complicato ma per risolvere questo problema basta che ci focalizziamo cosa deve fare ogni colonna, per esempio la colonna con il 3 (vedi immagine sotto dove c’è il cursore).



Alla fine, proseguendo il metodo descritto sopra, ci accorgiamo che ogni colonna deve solo inserire i numeri dopo il suo e basta. Nel caso del 3, sa che deve fare 3 e poi 4 e poi 5 ecc…

Cosa succede tra quando ha messo 3 e quando ha messo 4 non gli interessa, non è più un problema suo ma della chiamata ricorsiva successiva. Quindi ogni chiamata ricorsiva vede solo una colonna dell’immagine di sopra, però se andiamo a vedere lo sviluppo nel tempo, vediamo che si costruisce per righe.

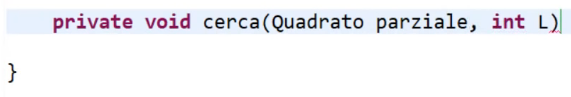
Un’altra domanda che ci dobbiamo fare e chiederci se noi vogliamo trovare tutti i quadrati magici di lato 3 (perché ce ne sono più di uno), o solo 1. Nel secondo caso, dobbiamo già pensare ad un modo per interrompere la serie di chiamate nel caso in cui ne avessimo trovato uno.

Ci sono dei casi in cui possiamo già capire che una soluzione parziale non va bene. Per esempio, dopo avere inserito i primi 3 numeri, se la loro somma non fa 15 (nel nostro caso) sappiamo già che non è un quadrato magico.

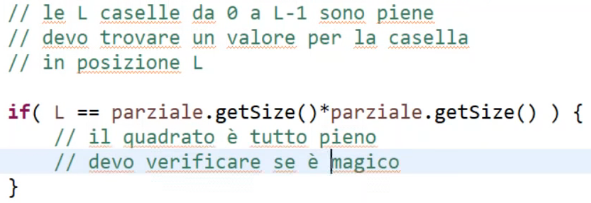
Passiamo ora ad Eclipse e apriamo la classe Quadrato. Notiamo che sono già state inserite una dimensione per inserire il lato del quadrato e la lista (mappa) dei numeri interi inseriti. Lista e non array perché quando definisco un array io ho già fisso il numero di caselle e quindi ho il problema di capire quali caselle sono piene e quali sono vuote. Con una lista invece parto con nessuna casella e ne aggiungo ogni volta che mi serve. Quindi questa classe ha solo la dimensione e la mappa con i relativi getter e setter.

Creiamo quindi una nuova classe chiamata “RicercaQuadrato” sempre nello stesso package che conterrà l’implementazione ricorsiva dell’approccio che abbiamo visto prima. Di solito ci servono 2 metodi:

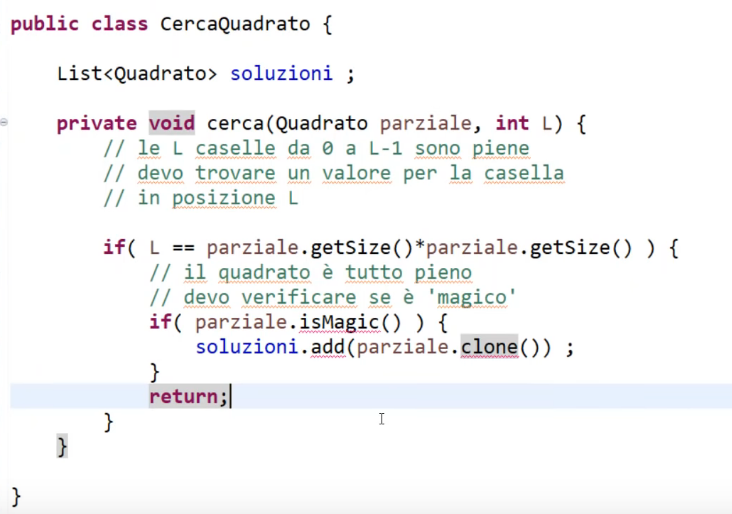
1. Interfaccia pubblica alla ricerca, per preparare i dati che ci servono, creare le collection di appoggio e poi formattare il risultato.
2. Metodo Ricorsivo che appunto lavora sulla ricorsione. Noi partiamo da questa.



Questo sarà quindi il nostro metodo ricorsivo che avrà quindi un Quadrato parziale e un livello. Se sono al livello 0 vuol dire che ho 0 caselle già messe e devo mettere un valore nella casella di numero 0.



Scriviamo sotto forma di commento ciò che dovrà fare il nostro metodo e gestiamo già l’if nel caso in cui il quadrato sia già tutto pieno. In tal caso devo controllare se sia magico mediante un metodo della .isMagic() della classe Quadrato che si occuperà di fare tutte le somme.

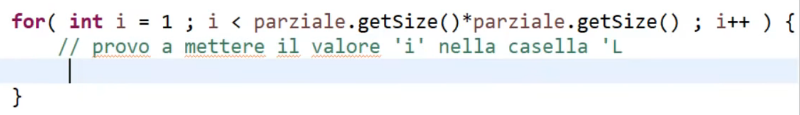


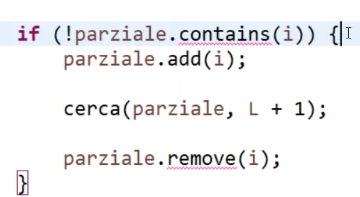
Logicamente il metodo isMagic() dovrà essere implementato nella classe Quadrato successivamente.

Abbiamo creato inoltre nella classe cercaQuadrato, una lista “soluzioni” di Quadrato per permettere di inserire il quadrato parziale.

Infine, abbiamo creato un ‘for’ che va da 1 al numero totale di caselle del quadrato che ci permette di inserire il valore ‘i’ nella casella del quadrato. Quindi, al livello 0, provo a mettere 1, poi a mettere 2, poi 3 e così via (sempre nella stessa casella. Ovviamente dopo avere provato a mettere un valore nella casella dovrò toglierlo. Dopo che ho messo il valore 1, per esempio, nella casella, ho finito di lavorare ricorsivamente, perché ci penseranno gli altri livelli a riempire le altre caselle. Quindi l’ultima cosa da fare è richiamare il codice ‘cerca’ per ricorsione.

Attenzione ad inserire il controllo che ci permette di gestire l’inserimento di duplicati. Prima di aggiungere un numero controllo che non ci sia ancora con un semplice ‘if’.



 Codice dentro al for.

FINE.