# PROGETTO SUDOKU

Classe Sudoku

La classe Sudoku estende la classe astratta BackTracking, la quale definisce in maniera generale la modalità di funzionamento dell’algoritmo del BackTracking. Come variabili d’istanza sono presenti due matrici: una di interi(9\*9) che sarà il sudoku vero è proprio, e una di booleani(9\*9) corrispondente a quella di interi. Vengono introdotte anche: una variabile per tenere conto delle soluzioni fino ad ora prodotte e un’altra per impostare il numero massimo di soluzioni. Nella matrice di booleani nelle celle corrispondenti a 0 della matrice di interi, verrà inserito false, altrimenti true e si è resa necessaria per una funzione della interfaccia grafica per bloccare le celle già assegnate. Una LinkedList di matrici conterrà le soluzioni.

* **Sudoku() :** Costruttore che inizializza la matrice tutta di interi a 0 e quindi quella di booleani a false.
* **Sudoku( int [][] mat ) :** Costruttore usato passando come parametro una matrice di interi. Costruttore utile nell’interfaccia grafica. Copia la matrice di interi e se il numero è 0 imposta false in quella di booleani nella cella corrispondente, true altrimenti.
* **boolean assegnabile( Integer[] p, Integer s) :** Override. Il metodo ha lo scopo di controllare se il numero (s) è assegnabile nella posizione p[]. Viene usato un array di due elementi per contenere le coordinate della cella della matrice: p[0] sarà la coordinata x, p[1] sarà la coordinata y. Questa assunzione varrà per tutto il codice dell’algoritmo. Il metodo ritorna false (non assegnabile) se è presente lo stesso numero o sulla stessa riga o sulla stessa colonna. Successivamente il metodo trovaMat, restituirà l’indice in alto a sinistra della sottomatrice che contiene la posizione passata come parametro. A questo punto si controlla che la sottomatrice corrispondente non contenga già il numero (s). Se tutti questi controlli danno esito positivo, il metodo ritorna true, ovvero assegnabile.
* **Integer [] trovaMat( Integer [] p ) :** Il metodo, dato una coppia (p[o]=x , p[1]=y), restituisce la posizione, sottoformadi array di due elementi, del vertice in alto a sinistra della sottomatrice in cui è contenuta la cella passata come parametro. Infatti la matrice (9\*9) del sudoku si può suddividere in 9 sottomatrici (3\*3). Metodo di utilità utilizzato in assegnabile().
* **void assegna( Integer [] p, Integer s ) :** Override. Il metodo assegna nella cella passata come parametro dell’array di due elementi, il numero s.
* **void deassegna( Integer[] ps, Integer s ) :** Override. Il metodo deassegna la cella passata come parametro nell’array di due elementi, ovvero pone la cella della matrice di interi in quella posizione a 0.
* **void scriviSoluzione(Integer [] p ) :** Override. Il metodo, che viene invocato quando si è giunti a una soluzione, copia la matrice di interi su un’altra temporanea e la inserisce nella LinkedList che contiene tutte le soluzioni. Viene incrementata la variabile numero di soluzioni.
* **List<Integer[]> puntiDiScelta() :** Override. Il metodo restituisce un ArrayList contenente tutte le possibili celle in cui si può allocare un numero nella matrice. Esso controlla che le celle non siano bloccate, attraverso la matrice di booleani.
* **Collection<Integer> scelte ( Integer [] p ) :** Override. Il metodo restituisce un ArrayList contenente tutti i possibili numeri assegnabili (1-9) nella cella passata come parametro. Fa questo controllo invocando il metodo assegnabile(), precedentemente decritto.
* **boolean esisteSoluzione( Integer [] p ) :** Override. Il metodo viene usato all’interno di tentativo() per capire quando si è giunti alla soluzione, ovvero quando si è arrivati nell’ultima cella della matrice.
* **boolean ultimaSoluzione( Integer [] p ) :** Override. Il metodo ha la funzione di bloccare il processo del backtracking quando il numero di soluzioni prodotte è maggiore del numero massimo di soluzioni, precedentemente impostato.
* **boolean imposta( int i, int j, int v ) :** Metodo usato per impostare le celle dall’interfaccia grafica. Esso controlla che nella posizione (i,j) sia assegnabile il valore v, attraverso il metodo assegnabile(). In caso contrario solleva una eccezione.
* **void risolvi() :** Override. Metodo fondamentale che da inizio alla procedura di backtracking a partire dalla prima cella (0,0), invocando il metodo tentativo(), presente nella classe BackTracking. Il metodo tentativo unisce tutti i metodi precedentemente descritti per risolvere il problema attraverso il backtracking, ovvero a tentativi.

Classe SudokuGUI

Classe creata per implementare una semplice interfaccia grafica. Sono create delle classi che corrispondono ai JFrame che si aprono interagendo con esse.

* **FinestraGUI :** Finestra principale, dalla quale attraverso la JMenuBar e quindi i JMenuItem che essa contiene si accede alle varie funzionalità. Vengono impostate le dimensioni, la visibilità e il titolo.
* **NewGame :** Finestra che viene aperta se si vuole iniziare una nuova partita. Anche in questa finestra viene impostata una JMenuBar che permette di salvare la partita corrente, impostare il numero massimo di soluzioni e di uscire. Viene inizializzato un pannello principale, impostato su BorderLayout, che conterrà tutte le componenti grafiche. Successivamente è creato un pannello che conterrà le 9 sottomatrici che formano la matrice del Sudoku(9\*9). Esso è impostato con un GridLayout(9,9). In un box sulla sinistra sono sistemati i JButton che permettono di risolvere il sudoku e di scorrere le sue soluzioni. Inoltre vengono inizializzati i JButton con il valore corrispondente alla matrice di interi del Sudoku, e sistemati nella loro sottomatrice corrispondente. A essi è associato un actionListener che permette di modificare la griglia iniziale del sudoku e quindi di aggiornare il loro valore. In questo caso è usato il metodo imposta().
* **AscoltatoreEventiAzione :** Contiene l’actionPerformed associato ai JMenuItem della FinestraGUI e NewGame.
* “Apri” : Viene inizializzato un JFileChooser, e in base al file selezionato viene invocato il metodo apri() che legge da un file una matrice di interi e la ritorna. Essa viene salvata. Se sorge un eccezione nella procedura, viene informato l’utente con un messaggio di errore.
* “Nuova Partita” : Se si è già effettuato un caricamento da file viene aperta la finestra NewGame con un Sudoku avviato con il costruttore che riceve una matrice di interi. La matrice passata come parametro è quella in precedenza salvata grazie al metodo apri(). Altrimenti, se non si è effettuato alcun caricamento da file, viene aperta la finestra NewGame, con il sudoku inizializzato normalmente, ossia con tutte le celle vuote(0).
* “Salva”: Viene aperto un JFileChooser e nel file selezionato per il salvataggio viene scritta la matrice fino ad ora impostata con l’interfaccia grafica. Questo avviene grazie al metodo salva(). Se in questo procedimento, sorge un’eccezione essa viene controllata e all’utente viene mostrato un messaggio di errore.
* “Esci” : Se si è salvato ciò che si è immesso nel sudoku, si esce normalmente, altrimenti viene mostrata una finestra di conferma, dove si avvisa l’utente che non è stato effettuato alcun salvataggio.
* “About” : Si possono selezionare una serie di JMenuItem, i quali, se cliccati, mostrano dei messaggi in cui si spiegano le varie funzionalità dell’interfaccia.