Il cubo di Rubik (e come risolverlo)

Stefano Angeleri, Alessandro Menti, Mattia Zago

Alcune definizioni

- Considereremo un cubo 3×3
- Ogni faccia (*side*) ha un colore standard a essa associato (vedi figura)
- Ognuno dei nove pezzi di ogni faccia è detto *facelet*
- Il cubo ha 3 colonne/righe (*columns*/*rows*), 3 colonne laterali (*lateral columns*), 4 angoli (*corners*) e 8 spigoli (*edges*)

Il problema

Riarrangia il cubo (ruotando righe, colonne e/o colonne laterali) finché tutte le facelet su ogni faccia non hanno lo stesso colore.

Notazione di Singmaster

- Ogni faccia è descritta da una lettera: **F** (Front), **B** (Back), **U** (Up), **D** (Down), **L** (Left), **R** (Right)
- Ogni mossa può essere vista come una rotazione di un quarto di giro di una faccia in senso orario (N.B.: si assume che il solutore abbia la faccia di fronte a sé): **U** = ruota la faccia "Up" di un quarto di giro in senso orario
- Il simbolo $'$ indica una rotazione in senso antiorario
- Le rotazioni di righe/colonne/colonne laterali centrali sono denotate da **M** (livello fra L e R), **E** (livello fra U e D), **S** (livello fra F e B)

Notazione di Singmaster

- Per denotare le rotazioni del cubo si usano altre lettere: **X** (rotazione su R), **Y** (rotazione su U), **Z** (rotazione su F)

Il nostro modello

- Il cubo è memorizzato in un oggetto `RubikCubeModel`
- Ogni faccia è memorizzata in un array 2×2; le righe/colonne sono numerate dall'alto verso il basso e da sinistra a destra (supponendo che il solutore abbia la faccia di fronte)
- `getSide` determina la faccia che in tale momento ha il colore dato
- `getFace` recupera il colore di una facelet
- Altri metodi autoesplicativi: `get3DEdge` (per gli angoli), `get3DEdgeFacelet` (facelet di un angolo), `getCorner`, `getCornerFacelet`
- Metodi `rotate*` per ruotare il cubo
- Test standard: `isInStandardConfiguration`, `isWithSaneColors`, `isSolved`, `isCornerInPlace`, `isCornerInPlaceMaybeFlipped`, `isEdgeInPlace`, `isEdgeInPlaceMaybeFlipped`

Mosse di Singmaster

- Sono state implementate le mosse standard di Singmaster
- Ogni mossa (per motivi di astrazione) è una sottoclasse di `Move`
- Il costruttore accetta come parametri il modello del cubo (in modo che il cubo originale rimanga inalterato) e un parametro `reversed` (per sapere se la mossa è diretta o inversa)
- Per applicare una mossa, basta crearla e chiamare `perform`/`reverse`:

```
(new B(m, reversed)).perform();
```

- Ogni mossa genera un evento per comunicare i cambiamenti all'interfaccia

IDA*

Thistletwaite's algorithm

The Kociemba 2-Phase algorithm

Singmaster's algorithm