Informazioni sul documento

Nome documento	Relazione Seconda Parte del progetto PuzzleSolver
Versione documento	v.2.0.0
Data redazione	2015-01-10
Redattore	Tesser Paolo

Sommario

Lo scopo del documento è quello di fornire una presentazione della seconda parte del progetto PuzzleSolver da realizzare, descrivendo e motivando le scelte attuate in questa fase.

Indice

1 Cambiamenti e Aggiunte

In questa sezione verranno descritti i cambiamenti apportati alla precedente versione del programma per permettere all'algoritmo di risoluzione di essere parallelizzato.
In generale non vengono effettuate particolari rivisitazioni della precedente struttura, ma vengono introdotte nuove gerarchie ed estese alcune precedentemente create.
Il client rappresentato dalla classe **PuzzleSolver** andrà modificato solo nella scelta della politica di esecuzione dell'algoritmo, che da sequenziale passerà a parallelo utilizzando la nuova classe descritta nella sezione 1.2.1.

1.1 Estensioni

1.1.1 Package solver

Questo package contiene le classi che gestiscono la risoluzione del puzzle.
Viene estesa la gerarchia che comprendeva alla base l'interfaccia **SolverStrategy** e come sottotipo la classe **SolverAlgStrategy**.
In aggiunta a quanto già fatto viene estesa l'interfaccia anche con la nuova classe **SolverParStrategy** responsabile della risoluzione parallela del puzzle. TO DO (grafico)

1.2 Nuove gerarchie

1.2.1 Package solver

Questo package contiene le classi che gestiscono la risoluzione del puzzle.
TO DO (grafico)

1.2.2 Package logger

Questo package contiene solamente una classe che mi serve per la gestione di un file di log che viene usato per tracciare l'esecuzione del programma in alcuni punti.
Essendo non importante ai fini delle richieste del proponente non ne verrà fornita una rappresentazione grafica.

2 Algoritmo di risoluzione (parallelo)

L'algoritmo scelto per la risoluzione del puzzle è tipo di parallelo come richiesto dalla specifica relativa alla seconda parte del progetto.
Per arrivare alla soluzione, le strutture dati introdotte nella prima parte rimangono invariate. Si userà quindi sempre una HashMap per i tasselli del puzzle ancora disordinati e un array bidimensionale per i vari pezzi in ordine corretto.
Se nella precedente versione queste erano le uniche due strutture dati di cui l'algoritmo aveva bisogno, per la versione parallelizzata viene usato anche un array mono dimensionale di oggetti Tile che contiene in ordine casuale i valori della HashMap.
Di seguito vengono esposte le sequenze che vengo eseguite e i thread che vengono lanciati dall'algoritmo, correlate da grafici che mostrano come esso agisca sulle strutture dati utilizzate.

1. **Ricerco il primo elemento del puzzle (quello in alto a sinistra) e quello in basso a sinistra.**
   TO DO;
2. **Ordino la colonna più a sinistra (quella con i tasselli aventi id ovest uguale alla stringa VUOTO).**
   TO DO;
3. **Ordino tutte le righe.**
   TO DO.

3 Gestione dei Thread

Nella seguente sezione vengono descritte le conseguenze che possono avvenire con l'avvio dei thread necessari alla risoluzione in parallelo del puzzle.

3.1 Numero Thread

Si può sempre valutare il caso

3.2 Interferenze, Deadlock, Attesa attiva

TO DO

4 Costrutti di concorrenza

Per come è stata creata la struttura che immagazzina il puzzle e per come si è pensato di risolverlo, l'esecuzione in parallelo non necessita di particolare blocchi sull'oggetto in questione.
I costrutti che vengono utilizzate sono i seguenti.

4.1 Oggetto condiviso

TO DO

4.1.1 Synchronized

TO DO

4.2 wait()

TO DO

4.3 notifyAll()

TO DO