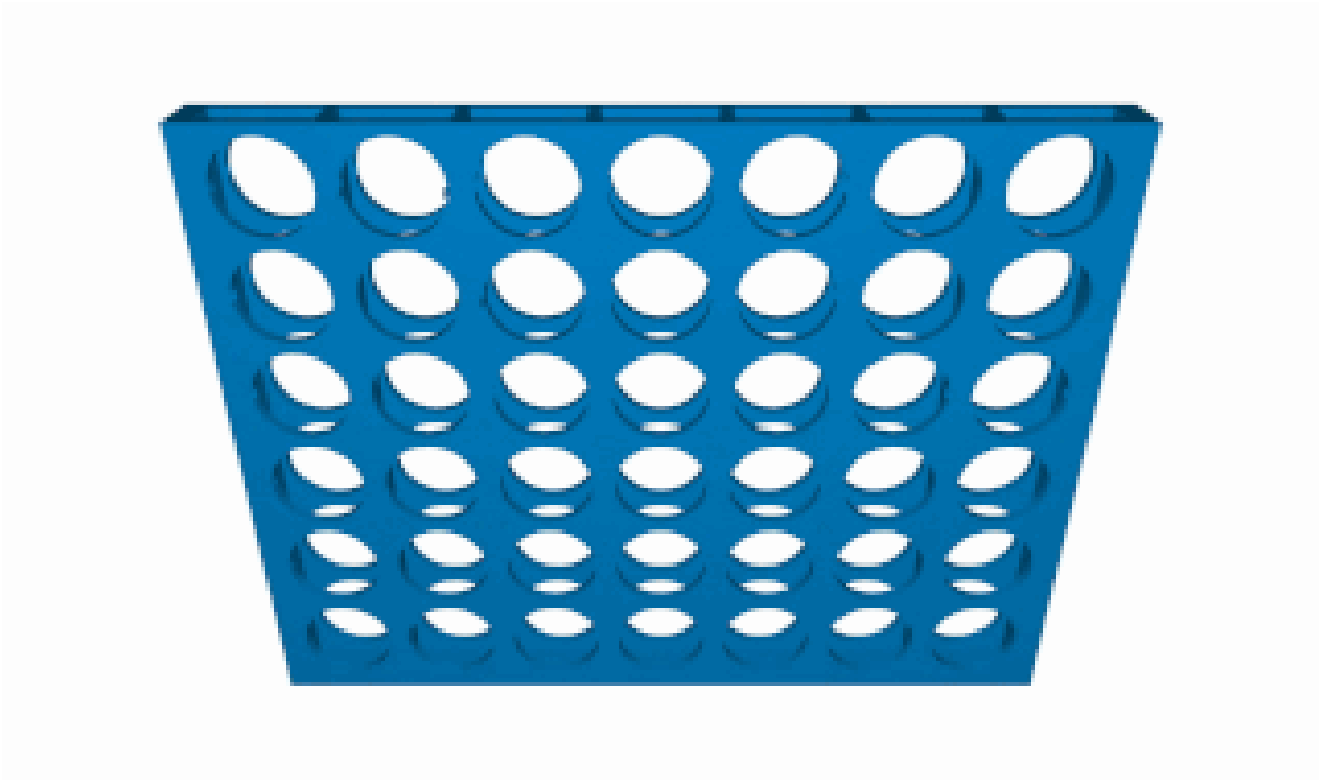
Progetto ASD2023

Pietro Di Lena

Dipartimento di Informatica – Scienza e Ingegneria
Università di Bologna

Algoritmi e Strutture di Dati
Anno Accademico 2022/2023

Forza 4 (Connect 4)



Gif animata presa da `https://en.wikipedia.org/wiki/File:Connect_Four.gif`

Connect X

- Il Connect X è una versione generalizzata del gioco Connect 4
  - La partita viene giocata su una matrice di dimensione $M \times N$
  - Per poter vincere bisogna allineare X gettoni
    - Verticalmente, diagonalmente oppure orizzontalmente
  - Il Connect 4 è un Connect X con parametri $M = 6, N = 7, X = 4$
- Lo scopo del progetto è quello di sviluppare un giocatore software in grado di giocare nel modo migliore possibile su tutte le istanze (ragionevolmente) possibili di un Connect X
  - La complessità del problema consiste nell'essere in grado di effettuare una buona scelta per una mossa di gioco rispettando un limite rigido di tempo

Vincoli da rispettare

1. Implementazione in Java di un'interfaccia Java pre-definita
   - L'interfaccia pre-definita si chiama CXPlayer (vedi avanti)

2. Vietato usare Thread e, più in generale, parallelizzazione
   - Il parallelismo non aiuta molto e complica l'implementazione
   - Rende complessa anche la fase di test dell'implementazione

3. Le implementazioni devono essere fornite come package
   - Se la classe che implementa l'interfaccia CXPlayer si chiama XXX allora il package fornito dovrà essere connectx.XXX
   - Cercate di usare nomi non troppo banali per la classe XXX (ex. Player), mi costringete a cambiarne il nome se ci sono conflitti

4. I metodi sviluppati non devono stampare nulla sullo standard output
   - E' possibile stampare sullo standard error (*System.err.println()*)
   - I miei script di analisi ignorano stampe sullo standard error

Informazioni generali

- Può essere svolto in gruppo (massimo tre persone)
- Si consegna una sola volta
  - Se necessario, saranno richieste correzioni specifiche
- Il voto del progetto resta valido per un tempo *illimitato*
  - A meno che non cambino i docenti del corso
- E' possibile consegnarlo entro Febbraio 2024
  - Oltre Febbraio 2024, attendere le nuove specifiche
- Il voto del progetto pesa per 1/3 sul voto finale
  - Media pesata tra voto dello scritto + voto del progetto

In cosa consiste la prova di progetto

- Sviluppo
  - Il linguaggio di programmazione è Java
  - Vi viene fornita l'interfaccia Java da implementare
  - Vi viene fornito tutto il software necessario per effettuare test
  - Suggerimenti più avanti
- Relazione
  - Breve relazione sulle scelte progettuali adottate
  - Suggerimenti più avanti
- Discussione orale
  - Discussione orale sul lavoro svolto
  - Non è un orale sugli argomenti del corso

Modalità di consegna e discussione

- Codici sorgenti e relazione: tramite virtuale
  - Slot di consegna a partire da fine Maggio
  - Possibile consegnare solo in periodi d'esame
  - E' sufficiente che consegni un solo componente del gruppo
  - I componenti del gruppo devono essere indicati alla consegna
- Discussione del progetto: appelli su AlmaEsami
  - Necessario iscriversi su AlmaEsami
  - Tutti i componenti del gruppo devono iscriversi su AlmaEsami
  - Discussione con tutti i componenti del gruppo insieme
  - Discussione in presenza o su TEAMS
  - La discussione deve essere fatta a ridosso della consegna
  - Per ogni slot di consegna sarà indicato anche il giorno della discussione del progetto

Valutazione

- Valutazione delle prestazioni
  - Confronto con un certo numero di giocatori di livello crescente
    - Prestazioni peggiori del Livello 0 $\Rightarrow$ progetto da sistemare
    - Prestazioni tra Livello 0 e 3 $\Rightarrow$ voto medio-basso
    - Prestazioni tra Livello 3 e 6 $\Rightarrow$ voto medio-alto
    - Prestazioni migliori del Livello 6 $\Rightarrow$ possibile lode
- Valutazione della relazione
  - Se ben scritta e dettagliata $\Rightarrow$ punti aggiuntivi
  - Se scarna $\Rightarrow$ nessun punto aggiuntivo
  - Se scritta male $\Rightarrow$ punti in meno
- Discussione orale
  - Serve per capire dettagli non chiari da relazione e codice sorgente
  - Serve per capire se tutti hanno contribuito al progetto
  - Partecipazione nulla o scarsa $\Rightarrow$ punti in meno (individualmente)

Valutazione delle prestazioni

M	N	X
4	4	4
5	4	4
6	4	4
7	4	4
4	5	4
5	5	4
6	5	4
7	5	4
4	6	4
5	6	4
6	6	4
7	6	4
4	7	4
5	7	4
6	7	4
7	7	4
5	4	5
6	4	5
7	4	5
4	5	5
5	5	5
6	5	5
7	5	5
4	6	5
5	6	5
6	6	5
7	6	5
4	7	5
5	7	5
6	7	5
7	7	5
25	25	10
50	50	15
75	75	20
100	100	30

- Lista delle 35 configurazioni che saranno testate
- Ogni implementazione viene fatta giocare sia come primo giocatore che come secondo (70 partite in totale per coppia)
- Assumendo che entrambi i giocatori abbiano una strategia ottima, il risultato della partita è influenzato dalla mossa iniziale. Esempi (Colonna indica la prima mossa giocata)
  - Per la configurazione $M=4, N=4, X=4$

Colonna	1	2	3	4
Risultato	Draw	Draw	Draw	Draw

  - Per la configurazione $M=4, N=6, X=4$

Colonna	1	2	3	4	5	6
Risultato	Win2	Win2	Win2	Win2	Win2	Win2

  - Per la configurazione $M=6, N=7, X=4$

Colonna	1	2	3	4	5	6	7
Risultato	Win2	Win2	Draw	Win1	Draw	Win2	Win2

Punteggi assegnati durante una partita

- Si utilizza il seguente schema di punteggi per ogni partita giocata:
  - Vittoria regolare o a tavolino: 3 punti
  - Patta: 1 punto
  - Sconfitta: 0 punti
- La vittoria a tavolino avviene in presenza di errori dell'avversario
  - Ad esempio: timeout, mossa non legale, ecc

Interfaccia CXPlayer

```
public interface CXPlayer {
  // Inizializza il giocatore software
  // M = numero di righe nella matrice
  // N = numero di colonne nella matrice
  // X = numero di gettoni da allineare
  // first = true se è il primo a giocare
  // timeout_in_secs = numero massimo di secondi per una mossa
  public void initPlayer(int M, int N, int X, boolean first,
                                                 int timeout_in_secs);
  // Seleziona una colonna tra quelle ancora libere
  // B = oggetto matrice di gioco
  public int selectColumn(CXBoard B);

  // Ritorna il nome del giocatore
  public String playerName();
}
```

Attenzione: il costruttore non deve prendere argomenti

Metodi pubblici della Classe CXBoard

```
// Resetta la tabella di gioco
public void reset();
// Stato della cella i,j della matrice
public CXCellState cellState(int i, int j);
// Ritorna true se la colonna col è piena
public boolean fullColumn(int col);
// Ritorna l'ultima mossa effettuata
public CXCell getLastMove();
// Ritorna lo stato del gioco (WIN1, WIN2, DRAW, OPEN)
public CXGameState gameState();
// Giocatore a cui tocca la prossima mossa
public int currentPlayer();
// Numero di celle ancora libere nella matrice
public int numOfFreeCells();
// Numero di celle già occupate nella matrice
public int numOfMarkedCells();
// Il giocatore corrente gioca sulla colonna indicata
public CXGameState markColumn(int col);
// Elimina l'ultima mossa giocata
public void unmarkColumn();
// Ritorna la lista di mosse già giocate, in ordine
public CXCell[] getMarkedCells();
// Ritorna la lista di colonne non ancora piene
public Integer[] getAvailableColumns();
// Ritorna la matrice di gioco attuale
public CXCellState[][] getBoard();
// Crea una copia dell'oggetto CXBoard
public CXBoard copy();
```

- La classe gestisce internamente i turni tra i vari giocatori
- Il metodo markColumn() assegna la mossa al giocatore corrente

Package connectx

- Pacchetto Java contente l'interfaccia grafica per giocare e testare la propria implementazione
- Nel pacchetto sono anche disponibili due implementazioni (semplici) di CXPlayer a scopo illustrativo e per effettuare test
  - Nello specifico, sono forniti i giocatori Livello 0 e Livello 1
- Da riga di comando il gioco può essere avviato nel seguente modo

java CXGame <M> <N> <X> [CXPlayer class] [CXPlayer class]

- Gli ultimi due argomenti sono opzionali:
  - Se usati entrambi parte in modalità Computer vs Computer
  - Se uno dei due è omesso parte in modalità Human vs Computer
  - Se sono omessi entrambi parte in modalità Human vs Human
- Disponibile anche tool per effettuare test senza interfaccia grafica

Suggerimenti per lo sviluppo

- Il metodo selectColumn(...) deve effettuare una scelta *ottimale* della mossa tra tutte quelle possibili
- Il numero di possibili partite giocabili cresce esponenzialmente al crescere della dimensione della matrice di gioco ed il numero di gettoni da allineare (impossibile valutarle tutte)
- L'implementazione deve cercare di trovare sempre una soluzione (mossa) accettabile in poco tempo (fissato a 10 secondi)
- L'implementazione non deve essere tarata su 10 secondi ma deve cercare di sfruttare al meglio il tempo specificato tramite il parametro *timeout_in_secs* passato al metodo initPlayer()
- Per quanto non ci siano vincoli sulle strategie da adottare per la scelta della mossa, il suggerimento è di partire dalle tecniche algoritmiche già studiate per questo tipo di problema (che vedremo)

Suggerimenti per la relazione

- Intestazione
  - Indicare il nome del progetto, nomi, cognomi e numero di matricola dei componenti del gruppo
- Descrizione del problema
  - Descrivere il problema computazionale affrontato
  - Introdurre il problema ed indicare i punti salienti che hanno portato alle scelte progettuali adottate
- Scelte progettuali
  - Descrivere ad alto livello le scelte implementative adottate per il metodo selectColumn()
  - Citare esplicitamente le strutture dati e gli algoritmi noti utilizzati (se applicabile) e sottolineare i contributi originali adottati per l'implementazione del progetto
  - Fornire un'analisi della costo computazionale (anche se molto approssimativa)

Oltre al voto

- I progetti consegnati e funzionanti saranno utilizzati per un torneo tutti-contro-tutti per determinare una classifica dei giocatori più forti
- La classifica verrà aggiornata di volta in volta
- E' solo per nostro divertimento interno, non avrà influenza sul voto anche se la classifica tenderà ad essere consistente con la valutazione
- Il nome che comparirà nella classifica è quello assegnato al giocatore tramite il metodo `playerName()`