پروژه جستجوی خصمانه

هوش مصنوعی و سیستم‌های خبره، بهار ۹۸

۱ توضیحات اولیه

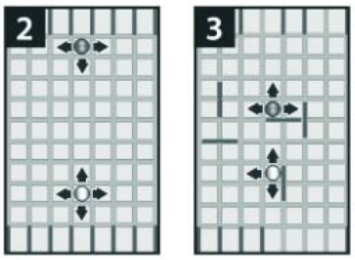
هدف این پروژه اشنایی بیشتر با الگوریتم MiniMax و پیاده‌سازی آن است. برای این کار بازی Quoridor در نظر گرفته شده است. استارتر کد برای GUI بازی آپلود شده است و شما بخش مربوط به هوش آن را تکمیل می‌کنید.

۱.۱ توضیحات و قوانین بازی

این بازی به صورت ۲ الی ۴ نفره است. هر بازیکن یک مهره و ۱۰ دیوار دارد. صفحه بازی به شکل مربع است و هر بازیکن در ابتدا در وسط یک ضلع آن قرار میگیرد.
هدف هر بازیکن این است که زود تر از بقیه خود را یکی از خانه های ضلع مقابل برساند. اولین نفری که به ضلع مقابل برسد بازی را برده است. و می‌تواند با دیوارهایی که دارد راه بازیکن‌های دیگر را سد کند.

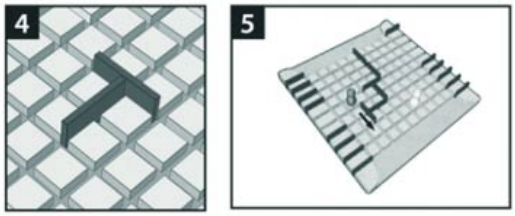
شکل ۱: تصویری از صفحه بازی

قوانین

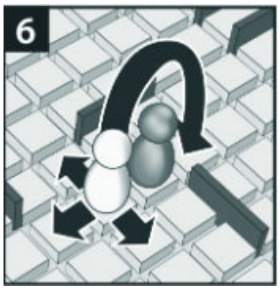


بازی به صورت نوبتی است و در هر نوبت بازیکن حق دارد یا یک حرکت انجام دهد یا دیوار در صفحه قرار دهد اگر دیوارهای بازیکن تمام شده باشد بازیکن باید حرکت کند.در هر نوبت حرکت مهره بازیکن می‌تواند فقط یک حرکت افقی یا عمودی ـ جلو یا عقب برود. مهره‌ها باید از دور دیوارها عبور کنند.

قراردادن دیوارها باید به صورتی باشد که هر دو مربع را بپوشاند و همچنین بعد از قرار دادن دیوار صفحه بازی همچنان باید همبند باقی بماند (بین هر دو نقطه صفحه مسیری باقی بماند) در غیر این صورت آن هرکت مجاز نیست.



وقتی دو مهره در حالتی مقابل هم قرار می گیرند که هیچ مانعی بین آنها نباشد مهرهی که نوبتش رسید می تواند از روی مهره دیگر بپرد و بدین طریق یک خانه اضافی پیش می رود. اگر مانعی در پشت مهره حریف قرار داشته باشد، بازیکن باید مهره خود را به سمت راست یا چپ مهره حریف ببرد.



۲ پیاده‌سازی

۱.۲ نوشتن Brain

ابتدا فایل‌های آپلود شده را دانلود کنید. سپس زیر کلاسی از کلاس `martijn.quoridor.brains.Brain` بسازید. برای مثال

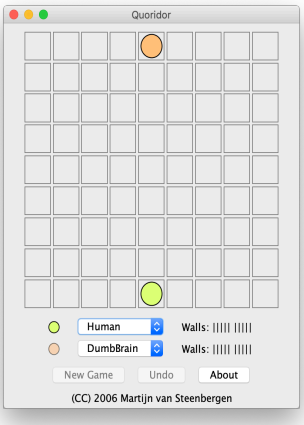
```java
// RandomBrain.java
import java.util.Collections;
import java.util.LinkedList;
import java.util.List;
import java.util.Set;
import martijn.quoridor.brains.Brain;
import martijn.quoridor.model.Board;
import martijn.quoridor.model.Jump;
import martijn.quoridor.model.Move;
import martijn.quoridor.model.Position;
/** RandomBrain always moves the player's pawn to a random position. */
public class RandomBrain extends Brain {
    @Override
    public Move getMove(Board board) {
        Set<Position> positions = board.getTurn().getJumpPositions();
        List<Position> list = new LinkedList<Position>(positions);
        Collections.shuffle(list);
        return new Jump(list.get(0));
    }
}
```

۲.۲ اضافه کردن Brain به بازی

برای اضافه کردن Brain به بازی باید کلاس خود را به DefaultBrainFactory اضافه کنید. یا پیاده سازی خودتان را از BrainFactory انجام بدید. برای استفاده از DefaultBrainFactory می‌توانید به سادگی کلاس خودتان را در تابع addBrain اضافه کنید. به‌طور مثال کد زیر برای اضافه کردن کلاس نوشته شده در بالا به بازی است.

```
package martijn.quoridor.brains;
import java.util.List;
/** ... */
public class DefaultBrainFactory implements BrainFactory {
    public void addBrains(List<Brain> brains) {
        brains.add(new RandomBrain());
    }
}
```

۳ اجرا بازی



برای اجرا بازی کلاس QuoridorApplication را اجرا کنید به صورت پیشفرض این کد بازی را با DefaultBrainFactory اجرا می‌کند. از لیست‌های نمایش داده شده اسم کلاس Brain ای را که نوشته‌اید انتخاب کنید. می‌توانید خودتان به در مقابل آن بازی کنید یا هوش‌های مختلف خودتان را با هم به رقابت بگذارید.

۴ نحوه تحویل

برای تحویل پروژه همه پیاده سازی ها و توابع مورد نیازتان در یک زیر کلاس از کلاس Brain پیاده کنید و همان یک فایل java. که باید برای اضافه و تست کردن هوش شما کافی باشد را در Quera آپلود کنید.

۱.۴ نمره دهی

هوش شما باید پیاده‌سازی از الگوریتم MiniMax با هرس AlphaBeta باشد. توجه داشته باشید که کد شما مدت زمان محدودی برای هر حرکت خود دارد مثلا ۵ ثانیه برای همین درخت MiniMax را نمی‌توان تا عمق زیادی ساخت. به همین دلیل قسمت مهم‌تر کار شما نوشتن تابع ارزیابی[۱] مناسبی است.

نمره اضافی

در پایان مسابقه‌ای بین کد ارسالی شما برگزار خواهد شد. و نفرات برتر در این مسابقه نمره بیشتر از این پروژه دریافت خواهند کرد. جزئیات بیشتر در مورد نحوه برگذاری مسابقه بعدا اطلاع رسانی خواهد شد.

[۱] Evaluation Function