Quoridor projekt

Projektet går ut på att göra en enkel "AI" spelare för spelet Quoridor. Det finns ett ramverk som sköter spelandet och ni behöver bara bygga själva algoritmen som gör dragen. Vi avslutar projektet med en turnering så får vi se om någon har lyckats göra ett bra spel.

Spelet

Spelet är ett brädspel med 9 x 9 rutor där du ska flytta en spelpjäs från ena sidan till den andra, din motspelare försöker göra samma sak fast från andra hållet. Du kan antingen flytta din pjäs eller placera en vägg mellan rutorna. Väggarna är två rutor långa. Viktigt är att man inte får stänga in motståndaren utan han måste alltid ha möjlighet att komma till målet. Det finns ett Quoridorspel i spellabbet som ni kan prova, annars finns mycket på webben. Trots de enkla reglerna så är det inte uppenbart hur man ska göra en duktig AI-spelare – eller hur man spelar själv för att vinna.
Observera att våra regler avviker från bordsspelet!

Er AI-spelare har en metod ”`Drag SökNästaDrag(SpelBräde bräde)`” som anropas när det är er tur. Spelbräde innehåller en beskrivning av ställningen i spelet och ni ska använda den informationen för att bygga en graf som beskriver läget, dvs. noderna är rutorna på spelplanen och det finns en båge om det går att flytta mellan rutorna. Man måste ta hänsyn till att det inte ska finnas bågar om det är en vägg emellan däremot tillåts att flytta till en plats där motståndaren står (inte tillåtet i det riktiga spelet). För den som är intresserad av att köra med de riktiga reglerna så finns det möjlighet att göra det också.

Krav på er spelare:

- Ni ska bygga en graf som beskriver läget.
- Den får inte göra ogiltiga drag – observera att detta kräver att om ni sätter ut en vägg så bör ni kontrollera att motspelaren fortfarande kan komma fram.
- Den måste vinna om motspelaren "tillåter" det, och då måste den gå direkt på målet (kräver i praktiken Dijkstras algoritm eller breddenförst sökning).

För att spela ett spel så behövs tre program, en server och två spelare. I den kod ni får så finns det ett serverprogram, en AI spelare och en ”manuell” spelare. Sen finns det ett färdigt Visual Studio projekt där ni ska implementera en klass som är AI-spelaren.
För att spela en omgång så ska ni starta en server och två spelare. Om ni vill spela mot en annan students AI-spelare så kan han köra sin spelare på sin dator (kräver att man sätter rätt IP adresser etc.)

Inlämning och rapport:
Ni ska även skriva en liten rapport som i första hand ska beskriva vilka algoritmer ni använt och analyserar algoritmerna. Lämna in programkod, plus en pdf-fil som beskriver lösningen (vilka algoritmer och eventuella källor för programkod ni använt, och annat som kan vara intressant) samt analyserar tids- och utrymmeskomplexitet tillräckligt noggrant för att kunna fungera som bevis. Det

går bra med O-uttryck för att ange komplexiteten. Det går också bra att hänvisa till kända komplexitetsuttryck för komponenter (algoritmer och datastrukturer) som beskrivs i boken. Upprepa alltså inte analyser som finns i boken, utan hänvisa till resultaten och sätt in dem i sammanhanget. Observera att tiden det tar att bygga upp grafen också ingår!
I analysen så ska N vara antalet rutor på ert spelbräde, så tanken är att ni ska bygga en algoritm som inte bara spelar bra men också går snabbt om man ökar storleken på spelbrädet.

Turnering:
Vi kommer naturligtvis avsluta med en turnering. Närmare detaljer kommer senare.

Quoridor regler, den enkla versionen

- Varje spelare har en pjäs och 10 väggar, väggarna är två rutor långa.
- Spelaren startar mitt på ena sidan. Motståndaren startar på motstående sida. Den som först når andra sidan vinner.
- Ett drag består av att flytta din pjäs ett steg (upp, ner, vänster, höger) eller placera ut en vägg.
- Väggarna måste vara helt på spelplanen och de kan inte flyttas.
- Det måste alltid finnas en väg för motspelaren till målet
- Du **kan** ställa dig på samma ruta som motspelaren

Spelsystemet:

På Canvas så finns en zipfil som innehåller ett Visual Studio 2017 projekt där du skriver in din kod. Där finns även exekverbara filer för:

1. Server: Servern spelar enligt de enklaste reglerna, inom kort kommer en option där man kan välja regler.
2. En manuell spelare (klicka på ruta dit du vill flytta eller tryck ned vänstra musknappen och "dra" dit väggen skall gå).
3. En mycket enkel AIspelar som bara går närmaste vägen till målet. Om det finns flera vägar så slumpas det olika varje dag.

Teknisk beskrivning!

Ni ska implementera SökNästaDrag () metoden, denna returnerar vad man vill göra för sorts drag till servern.

```
class Agent {
    public Agent() { }
    public Drag SökNästaDrag(SpelBräde bräde) {
        throw new System.NotImplementedException();
    }
    public Drag GörOmDrag(SpelBräde bräde, Drag drag) {
        //Om draget ni försökte göra var felaktigt så kommer ni hit
        System.Diagnostics.Debugger.Break();    //Brytpunkt
        return SökNästaDrag(bräde);
    }
}
```

De drag ni kan göra är flytta spelare eller placera en vägg enligt följande datastruktur:

```
enum Typ { Flytta, Horisontell, Vertikal }
struct Drag {
    public Typ typ;
    public Point point;
}
```

För förklaring av var väggarna hamnar se nedan.
Spel data:

```
//Brädet är numrerat från nedre vänstra hörnet (0,0) till övre högra hörnet (8,8)
//Spelaren går alltid uppåt dvs (-,8) är målraden.

public enum Typ { Flytta, Horisontell, Vertikal }
public struct Drag {
    public Typ typ;
    public Point point;
}
public class SpelBräde {
    public const int N = 9;
    public bool avanceradeRegler;       //Normalt false
    public List<Spelare> spelare;       //Spelaren är alltid först i listan.

    //Horisontella väggar, om (x,y) är satt
    // så kan spelaren inte gå från (x,y) till (x,y+1)
    public bool[,] horisontellaVäggar = new bool[N, N - 1];  //9*8

    //Vertikala väggar, om (x,y) är satt
    // så kan spelaren inte gå från (x,y) till (x+1,y)
    public bool[,] vertikalaVäggar = new bool[N - 1, N]; //8*9

    #region Överkurs
    //Om man vill hindra väggarna från att korsas.
    //Horisontella långa väggar, om (x,y) är satt
    // så kan spelaren inte gå från(x, y) till(x, y+1)
    //          och inte från (x+1, y) till (x+1, y+1)
    public bool[,] horisontellaLångaVäggar = new bool[N - 1, N - 1]; //8*8

    //Vertikala långa väggar, om (x,y) är satt
    // så kan spelaren inte gå från (x,y) till (x+1,y)
    //          och inte från (x, y+1) till (x+1, y+1)
    public bool[,] vertikalaLångaVäggar = new bool[N - 1, N - 1]; //8*8
    #endregion Överkurs
    public SpelBräde() {    }
}
public enum Färg { Röd, Blå };  //Röd börjar alltid
public class Spelare {
    public Point position;  //Platsen spelaren eller väggen ska placeras
    public Färg färg;
    public int antalVäggar; //Antal väggar kvar att placera ut
}
```

Quoridor regler, den korrekta versionen (överkurs)

- Varje spelare har en pjäs och 10 väggar, väggarna är två rutor långa.
- Spelaren startar mitt på sin egen sida. Den som först når motståndarens sida vinner.
- Spelaren startar mitt på ena sidan. Motståndaren startar på motstående sida. Den som först når andra sidan vinner.
- Ett drag består av att flytta din pjäs ett steg (upp, ner, vänster, höger) eller placera ut en vägg.
- Väggarna måste vara helt på spelplanen och de kan inte flyttas.
- Väggarna kan inte korsa varandra.
- Det måste alltid finnas en väg för motspelaren till målet (bortse från din egen pjäs när du kollar det).
- Du kan inte ställa dig på samma ruta som motspelaren men du kan hoppa över den:
    1. Om det går så kan du hoppa rakt fram över den.
    2. Om det är en vägg i vägen så kan du välja mellan att hoppa till vänster eller höger om den (om det inte är väggar i vägen).
    3. Om det är väggar på alla tre sidorna om motståndaren så kan du inte hoppa över den.