Rapport chocola

Wij hebben gekozen voor een rule based oplossing omdat het dan mogelijk is om bijna altijd een winst te behalen. We hebben de volgende websites bestudeerd:

- https://en.m.wikipedia.org/wiki/Nim#Proof_of_the_winning_formula
- https://www.geeksforgeeks.org/combinatorial-game-theory-set-2-game-nim/

bestudeerd en het idee geprobeerd te implementeren. Het chocolade spel gebruikt 2 verschillende algoritmes. Een voor als hij begint en één voor als hij als 2e begint. Als je als eerste begint kun je in het standaard 20 X 5 spel altijd winnen door een bepaalde volgorden aan zetten te doen. Dit zal later worden behandeld. Als het algoritme als 2e aan de beurt is, is er nog een goede kans dat het algoritme kan winnen. Er zit namelijk maar één zwakte punt in. Dit is de manier hoe hij zelf ook wint in zijn eerste situatie. CHE staat voor Chocolade eerste speler. Dit is terug te vinden in de code.

Als eerste zet

Het logaritme bestaat uit een aantal regels uniek genummerd die voor elke situatie waar winst mogelijk is een zet te leveren. Mocht het spel al verloren zijn dan geeft het algoritme het op en neemt hij het laatste stukje. Het algoritme speelt in deze versie veel ‘agressiever’ dan wanneer het als 2e begint.

Rule CHE01

```
if (Bord[1, 1])
{
    if (MaxX % 2 == 0 && MaxY % 2 == 0)
    {
        if (MaxX > MaxY)
        {
            if ((MaxX + MaxY) % 2 == 0)
            {
                zet = new Point(MaxX, 0);
            }
            else
            {
                zet = new Point(MaxX - 1, 0);
            }
        }
        else
        {
            if ((MaxX + MaxY) % 2 == 0)
            {
                zet = new Point(0, MaxY);
            }
            else
            {
                zet = new Point(0, MaxY - 1);
            }
        }
    }
    else
    {
        zet = new Point(1, 1);
    }
}
```

CHE01 bestaat uit 2 delen. Deze delen zijn dat als vakje 1,1 nog aanwezig is en de zijden zijn beide oneven dan kan vakje 1,1 veilig weg gehaald worden voor een winst situatie. Als er een situatie is

ontstaan waarin de zijde op een even situatie zouden uitkomen dan wordt dit eerst gecorrigeerd. Deze regel kan worden overschreven door betere zetten met een hogere rang.

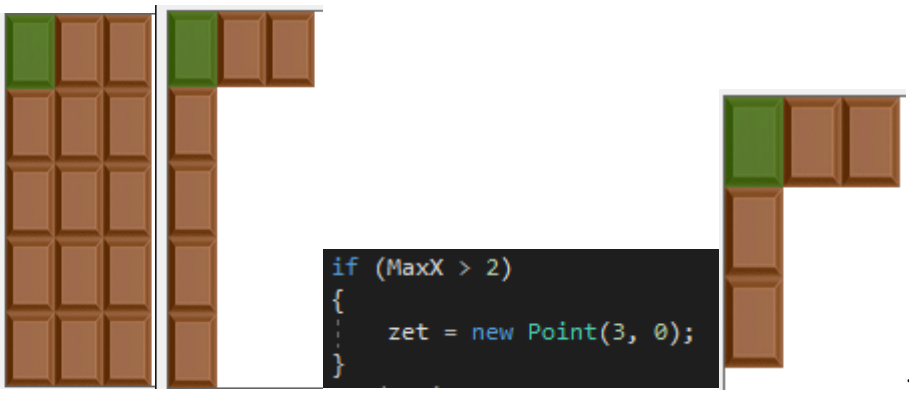
```
else
{
    if (MaxX > MaxY)
    {
        if ((MaxX + MaxY) % 2 == 0)
        {
            zet = new Point(MaxX - 1, 0);
        }
        else
        {
            zet = new Point(MaxX, 0);
        }
    }
    else
    {
        if ((MaxX + MaxY) % 2 == 0)
        {
            zet = new Point(0, MaxY - 1);
        }
        else
        {
            zet = new Point(0, MaxY);
        }
    }
}
```

Deel 2 van rule CHE01 doet het omgekeerde van de eerste. Om een even situatie voor te tegenstander neer te zetten. Op deze manier blijft je in een winst positie

Rule CHE02

```
if (MaxX == 2 && MaxY == 4)
{
    if (Bord[2, 2])
    {
        zet = new Point(2, 2);
    }
    else if (Bord[1, 1])
    {
        if (Bord[2, 1])
        {
            zet = new Point(1, 2);
        }
        if (Bord[3, 1])
        {
            zet = new Point(1, 3);
        }
        else if (Bord[4, 1])
        {
            zet = new Point(1, 4);
        }
        else if (Bord[1, 2])
        {
            zet = new Point(2, 1);
        }
        else
        {
            zet = new Point(1, 1);
        }
    }
}
```

Rule CHE02 probeert een optimale stand te genereren waarin winst bijna gegarandeerd is. Dit zal de chocolade reep verkleinen naar een vak van 3x5. Uit deze situatie door slim stukken weg te halen kan er een even situatie worden gemaakt voor de regenstander. In het volgende voorbeeld laten we zien wat er we bedoelen met een even situatie. Als deze L op het bord staat en de slimme speler heeft de beurt dan wint hij door de zet 0,3 te doen en een definitieve even situatie te maken.



```
if (MaxX > 2)
{
    zet = new Point(3, 0);
}
```

<- laagste even situatie

Rule CHE03

```
if (MaxY == 3 && MaxX == 2 && Bord[3, 0])
{
    zet = new Point(0, 3);
}
else if (MaxX == 3 && MaxY == 2 && Bord[0, 3])
{
    zet = new Point(3, 0);
}
```

Deze regel lost de situatie van 3x4 op. Door de situatie naar een naar de laatste even situatie te zetten. Met deze laatste even situatie is verlies niet meer mogelijk. Daarom probeert ons algoritme zo vaak mogelijk naar deze situatie te komen.

Rule CH#04

```
if (MaxX == 4 && MaxY == 4)
{
    if (Bord[1, 1])
    {
        zet = new Point(1, 1);
    }
    else
    {
        zet = new Point(0, 4);
    }
}
```

Deze regel lost een andere edge case op die 01 en 02 niet kunnen oplossen. De 5x5 waarin 1,1 weg is. Deze zet wordt niet opgepakt door 01 en wordt daarom hier geforceerd. Mocht 1,1 nog niet weg zijn op dit punt zal deze eerst worden verwijdert.

Rule CHE05

```
if (Bord[1, 0] && MaxY > 1 && MaxX == 1)
{
    zet = new Point(0, 2);
}
else if (Bord[0, 1] && MaxX > 1 && MaxY == 1)
{
    zet = new Point(2, 0);
}
```

Deze zet lost de case op om naar de 2x2 situatie te komen waarin 1,1 al weg is. Deze situatie is het zelfde als de laagste even situatie.

Rule CHE06

```
// Winning move vertical
if (Bord[1, 0] && !Bord[0, 1])
{
    zet = new Point(0, 1);
}
// Winning move horizontal
if (Bord[0, 1] && !Bord[1, 0])
{
    zet = new Point(1, 0);
}
```

Als ik kan winnen dan zal ik winnen. Deze regel haalt het laatste blokje weg om de winst te behalen.

De soorten spelers

Als het algoritme als eerste begint zal deze de regels CHE01 en CHE02 gebruiken. 02 heeft hierin de prioriteit omdat dit de beste situatie veroorzaakt voor de beginnende speler. Uit alle spellen die wij hiermee gespeeld hebben is het een enkele keer gelukt dat de random speler won. Wij hebben dit niet kunnen her creëren om deze zet volgorde te vinden en te voorkomen. Uit meerdere testen hebben wij dit ook niet kunnen vinden. Deze tactiek is niet optimaal maar heeft een hoog overwinningen percentage en weinig mogelijkheden om dit te ‘counteren’.

Als de speler als 2^{e} gaat gebruikt hij alle regels en speelt zo defensief mogelijk. Mocht hij de situatie van regel CHE02 kunnen creëren zal hij dit doen en hiermee verder gaan. Mochten deze strategie gebruikt worden tegen deze speler moet er 1 specifieks zet worden gedaan om hiermee te winnen.

Aftermath

Het algoritme is niet optimaal en niet onverslaanbaar. Maar het is niet zo simpel als een random speler. Er is zeker ruimte voor verbetering in de CHE02 strategie om het zwakte punt te vinden en te elimineren dit zal allen veel tijd kosten om dit te vinden.

Spel data van 100 spellen tegen een random speler waarvan 50 het algoritme begon en 50 de random speler.