Opdracht 1, Programmeertechnieken

Jenny Vermeltfoort, s3787494, groep 48

31 maart 2025

1 Uitleg

1.1 Optimale score.

De implementatie bepaald de eindscore door alle leaves van de actie-toestand boom te bekijken. Voor elke stand bij de leaves word de score berekend, bij Brian als initiele speler: (grooste_cluster_brian - grootste_cluster_robin). Door voor elke zet de score te vergelijken is het mogelijk om de optimale score te bepalen en daarmee de optimale zet.

De implementatie maakt gebruik van doeZet/unDoeZet. Een kopie maken van het bord is onnodig, aangezien de staat van het bord niet wordt veranderd wanneer er een doeZet vervolgd met een unDoeZet uitgevoerd word. Er wordt namelijk in de hand van elke speler bijgehouden op welke positie elke unieke steen komt te zitten. Bijvoorbeeld wanneer steen (1,2) op de tweede positie uit de hand van Brian wordt gehaald bij een doeZet, dan komt deze later ook weer terug op de twee positie bij een unDoeZet.

De toestand-actie boom eerder afkappen lijkt niet mogelijk. Vandaar dat mijn implementatie de volledige boom afloopt. Een aantal dingen die zijn overwogen:

1. Een hoogste score bijhouden voor elke node en wanneer deze score niet meer gehaald kan worden de node in de boom overslaan. Dit is wanneer de hoeveelheid te maken standen plus de huidige grootste cluster minder is dan deze perfecte score. Deze beredenering is echter incorrect, aangezien met de laatste zet vier clusters met elkaar verbonden kunnen worden. Een voorbeeld is wanneer vier clusters van grootte twee met elkaar worden verbonden met de laatste zet, dan gaat de grooste cluster voor de speler plots van twee naar acht. Dit is dus alleen te bepalen door de volledige boom af te lopen.
2. Bestaat er symmetrie in de boom? Nee, aangezien elke zet die gemaakt kan worden produceerd een tot een uniek bord, elke stand van het bord bij de leaves van de toestand-actie boom is dus uniek.
3. Voor elke speler een hash map bijhouden die de staat van het bord en de staat van de hand van de speler hash'd en mapped aan de te behalen scoren; voor elke bezochte node deze hash map vullen met deze score en met de hash map kijken of de hash voor de huidige stand van het bord al eens bepaald is. Zo ja, hoeft de hele tak van node tot leave niet opnieuw bepaald te worden; je kan namelijk gewoon de hash map uitlezen. Dit zou theoretisch mogelijk zijn, echter wordt deze hash map ontzettend groot, deze heeft namelijk dezelfde grootte als de hoeveelheid leaves van de toestand actie boom.
4. Een maal de optionele score berekenen door de hele boom af te lopen en vervolgens deze score cachen. Wanneer de optionele score opnieuw wordt berekend voor nodes verder dan de gecachete nodes, dan

kunnen alle leaves met deze score vergeleken worden. Wanneer deze hoogste score behaald kan worden met de huidige tak van de boom dan hoeft de rest van de toestand-actie boom niet meer worden afgelopen; je hebt namelijk al de zetten gevonden om de optionele score te behalen. Dit is echter niet geimplementeert in mijn implementatie.

2 Spellen

In de volgende subsecties zijn de resultaten van de implementatie te zien. In de resultaten is dat de Monte Carlo implementatie vergelijkbare resultaten bied als het berkenen van de optimale score. Dit is interrestant, aangezien de Monte Carlo methode altijd maar 100 * k0 of 100 * k1 aantal standen bekijkt. Dit is dus in vergelijking met de optimale score bereking aanzienlijk sneller, bij spel1 een factor 80k sneller.

De grootste cluster implementatie is presteerd echer ondermaats, maar is daarintegen wel ontzettend snel.

2.1 Spel 1

Resultaten voor spel1.txt

```
    Optimale score is: 0
    Een optimale zet is: (1,2)
    We hebben hiervoor 16777216 standen bekeken.
    Dit kostte 6361731 clock ticks, ofwel 6.36173 seconden.


Score grootste cluster tegen optimaal is: -3


Score Monte Carlo tegen optimaal is: 0
```

2.2 Spel 2

Resultaten voor spel2.txt

```
    Optimale score is: 2
    Een optimale zet is: (2,0)
    We hebben hiervoor 93312 standen bekeken.
    Dit kostte 19915 clock ticks, ofwel 0.019915 seconden.


Score grootste cluster tegen optimaal is: 1


Score Monte Carlo tegen optimaal is: 2
```

3 Toestand-actie boom

De hoeveelheid leaves van de toestand-actie boom is te berekenen met de volgende formule:

$$l = \frac{k_0 * \lceil p/2 \rceil * k_1 * \lfloor p/2 \rfloor * k_0! * k_1!}{n - f(k_0, k_1, p)} \tag{1}$$

De factor $\frac{1}{n-m}$ is een correctie voor wanneer het spel eerder stopt. Dit gebeurd wanneer een van de spelers geen stenen meer heeft, dit komt voor wanneer het verschil $|k_0 - k_1| >= 3$.

Waarbij n de hoeveelheid te spelen stenen.

$$n = hoogte * breedte - alopbord \tag{2}$$

De variable p is de hoeveelheid te leggen stenen van begin toestand tot eind toestand.

$$p = n - k_0 - k_1 \tag{3}$$

De variable m is de hoeveelheid standen. Hierbij floor en ceil omdat Brian een steen meer krijg als Robin wanneer p oneven is. En $+1$ in de twee lijn, omdat brian een zet meer krijgt wanneer Robin's hand groter is.

$$f(k_0, k_1, p) = \begin{cases} 2k_0 + \lceil \frac{p}{2} \rceil \cdot 2 & \text{als } k_0 < k_1 \\ 2k_1 + \lfloor \frac{p}{2} \rfloor \cdot 2 + 1 & \text{als } k_0 > k_1 \\ k_0 + k_1 + p & \text{als } k_0 = k_1 \end{cases} \tag{4}$$

4 Grootste Cluster

5 Experimenten

5.1 Experiment 1: GrootsteCluster vs Optimaal

```
config 0: De gemiddelde score van Brian is: -0.99
Dit kostte 200511 clock ticks, ofwel 0.200511 seconden.

config 1: De gemiddelde score van Brian is: -1.69
Dit kostte 3563727 clock ticks, ofwel 3.56373 seconden.

config 2: De gemiddelde score van Brian is: 0.69
Dit kostte 100200 clock ticks, ofwel 0.1002 seconden.

config 3: De gemiddelde score van Brian is: -0.18
Dit kostte 1677776 clock ticks, ofwel 1.67778 seconden.
```

5.2 Experiment 2: MonteCarlo vs Optimaal

```
config 0: De gemiddelde score van Brian is: -0.37
Dit kostte 259286 clock ticks, ofwel 0.259286 seconden.

config 1: De gemiddelde score van Brian is: -1.03
Dit kostte 3694570 clock ticks, ofwel 3.69457 seconden.

config 2: De gemiddelde score van Brian is: 1.19
Dit kostte 146323 clock ticks, ofwel 0.146323 seconden.

config 3: De gemiddelde score van Brian is: 0.58
Dit kostte 1805194 clock ticks, ofwel 1.80519 seconden.
```

5.3 Experiment 3: GrootsteCluster vs MonteCarlo

```
De gemiddelde score van brian is: -1.77
```