Bonus Exercise 3

Hrvoje Krizic

December 26, 2021

1 Minimax Algorithmus

Es sei ein Baum gegeben. Die Elemente dieses Baumes sind Nodes welche einen value und zwei Zeiger haben (einer zeigt nach links "left" und einer zeigt nach rechts "right". Ein Node, welchen wir "leaf" nennen, ist ein Node am Schluss des Baumes (also nur ein value und keine Pointer mehr, die auf einen nächsten Node zeigen). Die leaves sind gegeben. Es spielen zwei Spieler. Wir nennen sie Spieler "Violett" (V) und Spieler "Schwarz" (S). Es beginnt Spieler V und darf nun entscheiden, ob er links oder rechts geht. Spieler V ist clever und berechnet, bei welchem Node er einen höheren Value erreichen wird am Schluss (mit dem Gedanken, dass S immer den kleinsten Wert nehmen wird). Dann spielt S und macht das selbe, nur dass S den kleinsten Wert für V erreichen will. Wir wollen nun herausfinden, welchen Wert V erreichen wird, wenn beide perfekt spielen. Das heisst, wir beginnen bei den leaves und arbeiten uns den Baum hoch.

1.1 Rekursion

Wir führen den Minimax-Algorithmus mit Rekursion aus:

1. Base Case: Ist der Node ein "leaf", dann geben wir einfach den Wert aus.

```
if((!position->left) && (!position->right)){
    return position->value;
}
```

2. Ansonsten definieren wir zwei Variabeln (l,r), für welche wir per Rekursion für den nächsten kleineren Teilbaum links bzw. rechts den resultierenden Wert bestimmen. Wir geben letztlich je nach Farbe (S oder V) den minimalen/maximalen Wert der zwei aus. In Pseudo-Code sieht dies wie folgt aus:

```
if (nur ein node als child)
    return minimax(ab diesem child)
else
    l=minimax(ab left)
    r=minimax(ab right)
    if (colour=V)
        return max(l,r)
    else if (colour=S)
        return min(l,r)
```

2 Alpha-Beta-Suche

Die Alpha-Beta-Suche ist eine Methode, in der bestimmte Teile des Spielbaums gar nicht erst durchsucht werden, ohne dabei die Korrektheit des Ergebnisses zu gefährden. Hinter der Alpha-Beta-Suche steckt die Idee, dass man sich manche Varianten nicht mehr ansehen muss, weil sie klar schlechter sind als bisher gefundene. Beispielsweise: wenn wir wissen, dass S zwischen dem (von V gewählten) value 3 und einem value entscheiden muss, der entweder 5 oder eine andere Zahl ist (werten wir nicht aus), so wird sich ja V sicherlich für 5 entscheiden oder eine höhere Zahl, für S macht dies aber keinen unterschied. Dieser wird sich so oder so für 3 entscheiden, da 3 sicherlich kleiner ist als eine Zahl 5 oder grösser. alpha ist dabei der höchste score, den V bis jetzt hat und beta der schlechteste score, den S V geben kann.

2.1 Rekursion

Wieder verwenden wir Rekursion.

1. Base Case: Ist der Node ein "leaf", dann geben wir einfach den Wert aus.

```
if((!position->left) && (!position->right)){
    return position->value;
}
```

2. Ansonsten

```
if (colour=V){
    l=alpha_beta(ab links)
    alpha=max(alpha,l) (*)
    if (beta < alpha)
        delete_subtree
        return left_value
    else
        r=alpha_beta(ab rechts)
        return max(l,r)
}
else
    gleich wie bei V nur statt (*)
    beta=min(beta,l) und return min(l,r) am Schluss.
```