Softwarepraktikum SS 2018

Assignment 5

Group - 3

Ramil Sabirov	369500	ramil.sabirov@rwth-aachen.de
Joel Choi	345575	joel.choi@rwth-aachen.de
Eric Remigius	366895	eric.remigius@rwth-aachen.de

Task 1

An dieser Stelle wollen wir auf das Assignment 3 verweisen. Dort wurde bereits unsere Zeitschätzung für die nächste Tiefe erläutert.
Kurzgefasst: Wir wählen sehr konservativ den maximalen Verzweigungsgrad des Variantenbaumes als Faktor für die nächste Tiefe. Damit gehen wir nahezu allen möglichen Timeouts aus dem Weg. Jedoch beenden wir auch des öfteren Spiele mit einem riesigen Zeitkonto, welches dann verschwendet ist.

Task 2

Wir verwenden die Suchfenster innerhalb des Deepeners (Iterative Deepening Calculator). Nach der ersten Suche mit Tiefe 1 **ohne** Fenster, also mit Grenzen $+\infty$ und $-\infty$ haben wir einen Richtwert für den Stellungswert val_1. Für weitere Tiefen t, zentrieren wir ein Suchfenster mit Breite b um den zuvor gefundenen Stellungswert herum und suchen den neuen Stellungswert in dem Intervall $[val_{t-1} - b, val_{t-1} + b]$ in der Hoffnung, dass der gesuchte Wert tatsächlich in dem gegebenen Intervall liegt und von Anfang an mehr Teilbäume abgeschnitten werden können. Leider kann es passieren, dass dies nicht der Fall ist und die Einschränkung zu stark war. Der Alpha-Beta Algorithmus liefert in dem Fall entweder einen Wert kleiner-gleich der unteren Schranke $val_t <= val_{t-1} - b$ oder einen Wert größer-gleich der oberen Schranke $val_t >= val_{t-1} + b$. Jedoch ist der berechnete Stellungswert nicht aussagekräftig, da der optimale Pfad beschnitten worden sein könnte. Also ist eine Neusuche notwendig:

- **Fall 1: untere Schranke wird unterschritten.**
  Dann liegt der tatsächliche Stellungswert unterhalb der unteren Schranke. Wenn also das Intervall $[\alpha, \beta]$ war, dann erfolgt die Neusuche im Intervall $[-\infty, \alpha]$.

- **Fall 2: obere Schranke wird überschritten.**
  Analog zum ersten Fall muss oberhalb der oberen Schranke neu gesucht werden. Also im Intervall $[\beta, +\infty]$.

In der Implementierung wird das jeweilige Fenster dem Calculator über einen Parameter gegeben von der Klasse $CalculatorConditions$. Diese enthält Vorgaben, welche der jeweilige Calculator zu erfüllen hat. Zum Beispiel gehört eine maximale Tiefe und ein Zeitlimit auch dazu.

Task 3

Um einen möglichst großen Gewinn in der Performance zu erzielen, müssen wir versuchen, die *Aspiration-Windows* möglichst klein zu halten, damit viele Zweige abgeschnitten werden.

Da unsere Bewertungsfunktion uns einen Wert zwischen 0 und 250 liefert, kann die Größe des Fensters maximal 250 groß sein. Aber wir wollen ja ein möglichst kleines Fenster haben. Deshalb haben wir zuerst auf der Standard Karte eine Binäre Suche ausgeführt, und haben festgestellt, dass erst ab einer Größe von ca. 15-20 die oben beschriebenen *Aspiration-Window Fails* auftreten.

Um diesen Wert besser ausfeilen zu können, haben wir ein Script geschrieben, dass auf fünf Karten je sieben Spiele spielt und dabei die folgenden Fenstergrößen ausprobiert:

$$50, 40, 30, 20, 15, 10, 5$$

Dabei haben wir festgestellt, dass auf verschiedenen Karten die Schwelle, ab der *Aspiration-Window Fails* auftreten, unterschiedlich ist. Somit macht es keinen Großen Sinn, die optimale Fenstergröße exakt zu bestimmen und eine Auflösung von 5 sollte genügen.

Die genauen Ergebnisse können in der Datei Aspiration-Ergebnisse.txt eingesehen werden. Hier deshalb nur die wichtigsten Eckdaten:

Map	Aspiration Window Size	Fails
standard_map.txt	20	0
	15	3
	10	20
005_g5_map2.txt	20	1
	15	9
	10	5
006_g6_question_mark.map	20	0
	15	2
	10	2
009_g7_Map_Pirate_Heist	20	6
	15	10
	10	24
011_g3_Map1.txt	20	0
	15	0
	10	0

Da ein *Aspiration-Window Fail* dazu führt, dass eine Suche erneut gestartet wird, kann es, vor allem in großen Tiefen, sehr teuer sein, weshalb wir den optimalen Wert sehr konservativ entschieden haben.

Dieser optimale Wert ist für uns: 15

Auf allen getesteten Maps führt dieser Wert dazu, dass weniger als 0,03% der Suchen fehlschlagen. Der nächst bessere Wert von 10 hätte auf einigen der Karten diese Grenze überschritten. Insgesamt denken wir also, dass es ein guter Kompromiss ist.