Übungsblatt 4

Abgabe: 14.06.2024

Auf diesem Übungsblatt macht für Aufgabe 1 die Person die Implementierung, die in einem Telefonbuch zuletzt gelistet würde. Die andere Person macht die Tests.

Aufgabe 1 Vier Gewinnt als Brettspiel (100 %)

Bei *Vier in einer Reihe* setzen zwei Spielende abwechselnd Steine auf ein quadratisches Feld. Wer zuerst vier Steine in einer Reihe gelegt hat (waagerecht, senkrecht oder diagonal), hat gewonnen.

	O	O	X	
O	X	X	X	X
	O	X	X	
	O	X	O	
		O		

Vervollständigt die vorgegebene Klasse *FourInARow* so, dass eine Spieler:in gegen den Computer spielen kann. Das Spielfeld ist dabei ein zweidimensionales Array aus Werten des Aufzählungstyps *Player* (leer, Mensch, Computer). Die Spieler:in macht ihre Züge mit der Methode *humanMove(int, int)*. Der Computer macht die seinen in der Methode *computerMove* und soll dazu den *Minimax*- bzw. *NegaMax*-Ansatz mit Tiefensuche verwenden. Beide Methoden antworten mit einer Konstanten des Aufzählungstyps *Result* (Spiel geht weiter, Mensch hat gewonnen, Computer hat gewonnen, Unentschieden). Die bereitgestellte Klasse *Move* kann benutzt werden, um während der Tiefensuche Züge zusammen mit ihrer Bewertung zu repräsentieren. Um das Testen zu erleichtern, sollte die Implementierung auch funktionieren, wenn ein schon teilweise belegtes Feld an den Konstruktor übergeben wird.

Bei der Tiefensuche gibt es zwei Abbruchkriterien, wobei in beiden Situationen eine Bewertung zurückgeliefert werden muss:

- Wenn ein Zug zum Sieg einer Spieler:in führt, also vier in einer Reihe erreicht werden, sollte die Bewertung umgekehrt proportional zu der Anzahl der Züge sein, die noch bis zum Sieg gebraucht werden, d.h. je schneller der Sieg, desto höher die Bewertung.
- Wenn die maximale Tiefe der aktuellen Suche erreicht wurde, ohne dass jemand gewonnen hat, ist die Bewertung 0. Alternativ könnt ihr euch natürlich auch eine bessere Bewertungsmethode ausdenken. Beachtet, dass die maximale Tiefe durch die Anzahl der noch möglichen Züge beschränkt ist, d.h. wenn das Brett voll ist, ist die maximale Tiefe auch erreicht.

Über die bereitgestellte Klasse *Main* kann das Spiel ausprobiert werden. Die dort eingegebenen Zeilen- und Spaltennummern des Zugs der Spieler:in sind 1-basiert, also 1–5 in dem Beispiel oben. Intern wird aber mit 0-basierten Koordinaten gearbeitet (also gültigen Array-Indizes).

Tests: Es soll nur die Klasse *FourInARow* getestet werden. Vervollständigt dazu die in der Testklasse beschriebene Methode *asField*, die es erleichtert, Felder zum Testen zu erzeugen. Es bietet sich auch an, zur Überprüfung des Ergebnisses von Zügen die in der Klasse *FourInARow* bereitgestellte Methode *toString* zu nutzen.

Aufgabe 2 Bonusaufgabe (10 %)

Lest euch an, was *Alpha-Beta-Pruning* ist, und baut dieses in Aufgabe 1 ein. Kann dadurch die Suchtiefe erhöht werden? Für diese Erweiterung brauchen keine zusätzlichen Tests implementiert zu werden.