

## Mini-Räuberschach – die Spielregeln:

Mini-Räuberschach ist ein Brettspiel für 2 Personen, welches die normalen Schachregeln „auf den Kopf stellt“. Es wird auf einem 6'6 Felder großen Brett mit 24 Figuren gespielt. Ein Spieler bekommt die weißen, der andere die schwarzen Figuren. Es wird abwechselnd gezogen, Weiß beginnt.

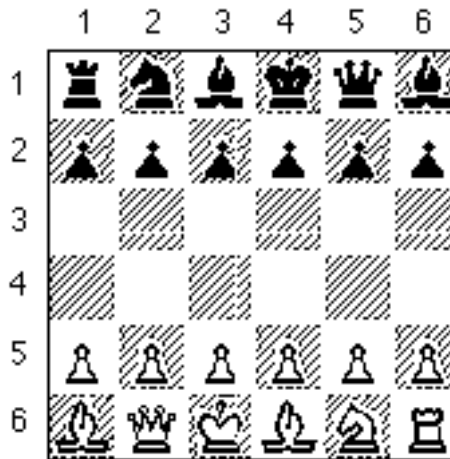


Abb. 2: Grundaufstellung

Die Felder werden wie entsprechende Positionen in einer Matrix bezeichnet, so steht z. B. auf dem Feld 62 die weiße Dame. Die Felder 11 bis 16 bezeichnen die schwarze Grundlinie, entsprechend 61 bis 66 die weiße.

Es gibt die Figurenarten Turm, Springer, Läufer, Dame, König und Bauer. Die Figurenart bestimmt die mögliche Zugrichtung. Ein Zug bewegt eine Figur von ihrem Ausgangsfeld auf ein Zielfeld, welches von der Figur gemäß den Zugmöglichkeiten erreichbar sein muß. Gelangt man mit einer eigenen Figur auf ein vom Gegner besetztes Feld, wird dessen Figur aus dem Spiel geschlagen.

Turm (t), Läufer (l) und Dame (d) können beliebig weit ziehen, aber nur über freie Felder. Weder fremde noch eigene Figuren dürfen dabei übersprungen werden.

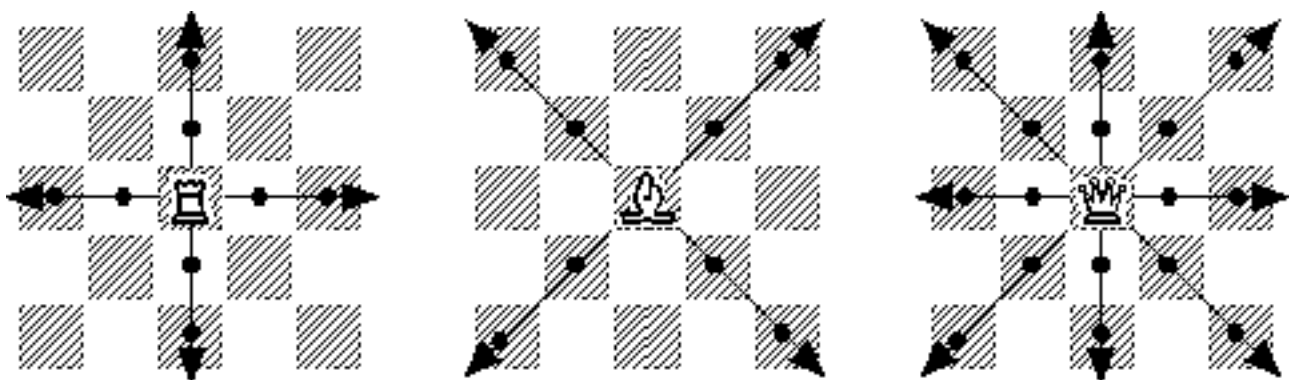


Abb. 3: Zugmöglichkeiten für Turm, Läufer und Dame

Der König (k) bewegt sich wie die Dame, zieht aber höchstens ein Feld weit. Der Springer (s) kann als einzige Figur fremde Figuren überspringen.

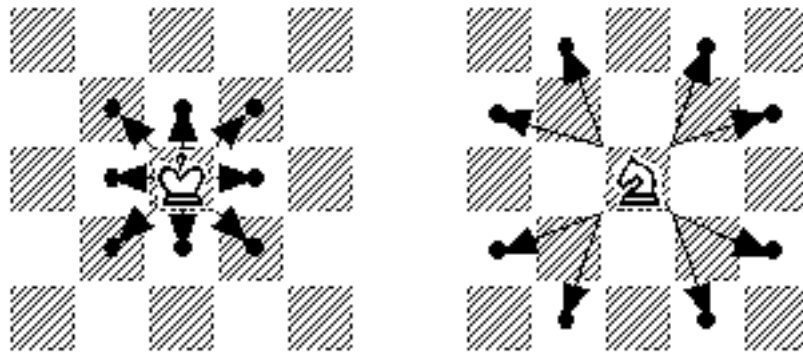


Abb. 4: Zugmöglichkeiten für König und Springer

Der Bauer (p) kann ein Feld gerade nach vorne auf ein freies Feld ziehen (Abb. 5a). Er ist die einzige Figur, die nur in eine Richtung ziehen darf. Gegnerische Figuren können vom Bauern nur direkt diagonal geschlagen werden (Abb. 5b). Erreicht der Bauer den gegenüberliegenden Brettrand, so verwandelt er sich automatisch in eine Dame, egal ob der Spieler noch seine Dame besitzt oder nicht (Abb. 5c).

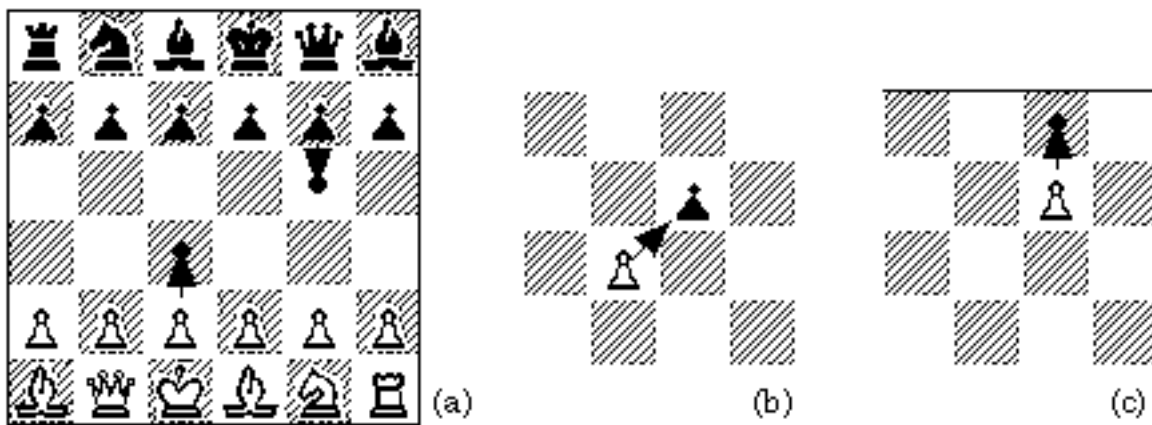


Abb. 5: Zugmöglichkeiten für den Bauern

Im Gegensatz zum normalen Schach gibt es keine Rochade und kein „Schlagen im vorübergehen“. Auch darf ein Bauer niemals zwei Felder nach vorne ziehen.

Gewonnen hat der Spieler, der als erster *keinen* Zug mehr machen kann. Dies kann geschehen, wenn alle seine Figuren geschlagen wurden oder weil er keinen gültigen Zug mehr durchführen kann (z.B. er besitzt nur noch blockierte Bauern).

Beim Räuberschach herrscht *Schlagzwang*. Kann man im nächsten Zug eine gegnerische Figur schlagen, so muß man dies tun. Kann man mehrere Figuren schlagen, darf man zwischen diesen Möglichkeiten wählen. Der König nimmt keine Sonderstellung ein, er kann genauso geschlagen werden wie jede andere Figur, das Spiel geht trotzdem weiter.

Eine gute Strategie ist, die eigenen Figuren dem Gegner so anzubieten, daß man selber nicht zurückschlagen muß. Im Gegensatz zu einer Schachpartie ist eine Räuberschachpartie im allgemeinen recht kurz.

## Hinweise zur Programmierung:

Die ersten Teilaufgaben sind Repräsentation, Initialisierung und Darstellung des Brettes:

- Deklarieren Sie Konstanten für die Brettgröße.
- Deklarieren Sie die Datentypen SpielerFarbe, Figur, Groesse, Zeile und Spalte.
- Deklarieren Sie einen Datentyp Feld als Verbund.
- Deklarieren Sie einen Datentyp Brett als zweidimensionales Feld.
- Schreiben Sie eine Prozedur, welche ein Brett mit der Grundaufstellung initialisiert. Das Brett soll Parameter dieser Prozedur sein.
- Schreiben Sie eine Prozedur, welche ein Brett auf dem Bildschirm ausgibt. Benutzen Sie einfache Zeichen oder Zeichenkombinationen für die Figuren (z.B. wd = weiße Dame), denn die Ausgabe dient ja nur zu Testzwecken.

Nachdem das Brett als der statische Teil des Spiels definiert wurde, sollen jetzt Züge auf diesem Brett ausgeführt werden. Noch ist es dabei völlig egal, ob ein Zug legal ist (d.h. den Regeln entsprechend) oder nicht. Lediglich seine Repräsentation und die Ausführung ist wichtig.

- a) Deklarieren Sie einen Datentyp Zug, welcher einen Spielzug repräsentiert (zwei Felder: Woher wird gezogen und wohin).
- b) Schreiben Sie eine Prozedur, die einen Zug  $z_1s_1z_2s_2$  von der Tastatur einliest. Dabei sind  $z_i$  und  $s_i$  jeweils Ziffern, die Zeile und Spalte eines Feldes darstellen,  $z_1s_1$  das Startfeld und  $z_2s_2$  das Zielfeld eines Zuges.
- c) Schreiben Sie eine Prozedur, die einen Zug auf dem Bildschirm ausgibt, ebenfalls in der Form  $z_1s_1z_2s_2$ .
- d) Schreiben Sie eine Prozedur, die einen Zug ausführt. Eingabeparameter dieser Prozedur sind ein Brett und ein Zug, Ausgabeparameter ist das gleiche Brett, auf welchem der Zug ausgeführt wurde. Beachten Sie den Spezialfall der Bauernumwandlung auf der Grundlinie des Gegners.

Testen Sie die Prozeduren: Initialisieren des Brettes und wiederholtes Ein- und Ausgeben eines Zuges und des geänderten Brettes.

Schreiben Sie ein Modul ZugStapel (siehe Aufgabe 31), welches einen Stapel über dem Datentyp Zug realisiert. Der Stapel soll 30 Züge aufnehmen können. Dieses Modul werden wir auf dem nächsten Übungsblatt benötigen.

Nun tun wir endlich etwas für unseren Computergegner. Unser Programm bekommt beigebracht, wie sich „gute“ und „schlechte“ Stellungen unterscheiden.

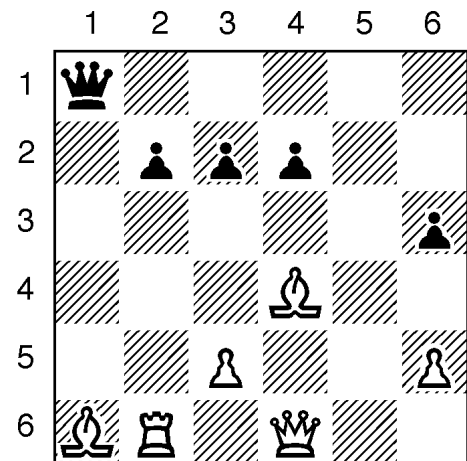
Definieren Sie einen Datentyp Stellungswert als Unterbereichstyp von INTEGER. Schreiben Sie nun eine Prozedur (in einem neuen Modul Stellungsbewertung), welche ein Brett als Eingabe bekommt. Ausgabe dieser Prozedur soll ein Stellungswert sein, welcher die Stellung durch bloßes „Hinsehen“ bewertet. Der Wert soll positiv sein, wenn Weiß besser steht, negativ, wenn Schwarz besser steht und 0 für eine ausgeglichene Stellung. Hier ist ihre Phantasie gefordert! Im Test gegen andere Programme wird sich dann zeigen, wie gut die Bewertung wirklich ist.

Beispiele:

- Zählen Sie die Figuren, die jeder Spieler noch besitzt.
- Gewichten Sie die Figuren. Noch einen Läufer zu besitzen ist z.B. schlechter als eine Dame. Beispiel: Turm = 10, Läufer = 10, Dame = 6, König = 4, etc. Führen Sie einige Probespiele durch um festzustellen, wie „unangenehm“ der Besitz verschiedener Figuren sein kann.
- Eine eigene Figur in gegnerischen Reihen ist ungünstig, da sie wahrscheinlich Figuren des Gegners schlagen muß.

Nun lernt unser Programm die Räuberschach-Regeln. Schreiben Sie für jeden Figurtyp eine Funktion, die ein Brett, Zeile, Spalte, Farbe des ziehenden Spielers und zwei ZugStapel als Parameter bekommt. Diese Funktionen legen alle legalen Züge der auf dieser Position stehenden Figur auf den Stapeln ab. Auf dem ersten Stapel alle Züge, bei denen nicht geschlagen wird, auf dem zweiten alle Züge, bei denen eine gegnerische Figur geschlagen wird.

Nebenstehendes Brett sei der Funktion Läufer mit Zeile 4, Spalte 4 und Weiß am Zug übergeben worden. Beim Läufer sind nacheinander die vier Diagonalen auf legale Züge zu überprüfen. Ziehen wir zuerst nach oben rechts. Das Feld 35 ist frei, also ist 4435 ein legaler Zug, welcher auf den ersten ZugStapel gelegt wird. Die Figur kann nun weiter in diese Richtung ziehen. Das Feld 26 ist ebenfalls frei, also kommt 4426 ebenfalls auf den ersten Stapel. Noch weiter kann die Figur nicht ziehen, da sie an den Rand stößt, weshalb die Suche in diese Richtung beendet wird. Weiter geht es nach oben links. 4433 wird ebenfalls auf den ersten Stapel gelegt. Feld 22 ist vom Gegner besetzt, also wird 4422 auf den zweiten Stapel gelegt und nicht weiter in diese Richtung gesucht, da der Bauer auf 22 ein weiteres Ziehen nach 11 verhindert. Auf 53 steht eine eigene Figur, die wir nicht schlagen dürfen und uns ebenfalls blockiert. Dieser Zug wird also auf keinem Stapel abgelegt. Genauso finden wir noch 4455 und 4466 als legale Züge.



Wie man sieht, ist es sinnvoll, eine Prozedur zu verwenden, die alle möglichen Positionen einer Figur „in einer Richtung“ findet und auf „leeres Feld“, „Rand“, „Gegner“ und „eigene Figur“ entsprechend reagiert. Machen Sie sich klar, daß für den Turm das gleiche Vorgehen verwendet werden kann, nur eben die Zugrichtung anders ist. Gleiches gilt für die Dame, welche wie Läufer und Turm kombiniert zieht.

Für König und Springer gilt, daß sie nur „einmalige“ Züge machen, weil sie nicht beliebig weit ziehen. Auch hier reicht bei geschickter Programmierung eine Funktion! Der Bauer bedarf einer eigenen Funktion, da er anders schlägt als zieht.

Schreiben Sie nun eine Prozedur ZugFinder, die zu einem Brett, Farbe des ziehenden Spielers und einem ZugStapel alle legalen Züge eines Spielers findet und auf den Stapel legt. Hierfür suchen Sie alle Figuren des Spielers auf dem Brett und sammeln jeweils alle Züge der Figuren auf zwei neuen Stapeln. Ist der zweite Stapel am Schluß nicht leer, so konnte der Spieler schlagen, also sind alle legalen Züge auf dem zweiten Stapel (wegen Schlagzwang) und der erste Stapel wird ignoriert. Ist der zweite Stapel leer, sind alle legalen Züge auf dem ersten Stapel. Man mache sich klar, daß der Stapel automatisch leer bleibt, wenn der Spieler nicht mehr ziehen kann. Testen Sie den ZugFinder mit den bisher erstellten Modulen.

Im Idealfall würde ein Spielprogramm folgendermaßen vorgehen: Zu einer gegebenen Stellung sucht es alle legalen Züge („Denktiefe 1“). Dann sucht es für alle diese Züge jeweils alle möglichen Antwortzüge („Denktiefe 2“), dann darauf wieder alle möglichen Antwortzüge („Denktiefe 3“), usw. Jener (erste) Zug, bei dem alle möglichen Antworten zum Gewinn führen, wird ausgeführt. Angenommen, in einer Stellung sind durchschnittlich 15 Züge möglich. Dann ist die Zahl der Antwortzüge auf diese Züge  $15^2 = 225$  Züge. Die Zahl der Züge wäre dann  $15^{\text{Denktiefe}}$ . Soll unser Programm eine Denktiefe von nur 6 Zügen haben, d.h. jeder Spieler zieht 3 mal, so müßte es schon  $15^6 \approx 11$  Millionen Züge ausprobieren und diese Bedenkzeit wächst mit jeder zusätzlichen Denktiefe um das fünfzehnfache! Dies ist nicht mehr vollständig in akzeptabler Zeit zu berechnen.

Die meisten Spielprogramme arbeiten in etwa folgendermaßen: Für eine vorgegebene Denktiefe (z.B. 4) werden alle möglichen Zugfolgen berechnet. Die danach entstandenen Stellungen werden bewertet und dieser Wert als Wert der jeweiligen Zugfolge zurückgegeben. Der Zug, der die „beste“ Bewertung der Zugfolge bekommt, wird als nächster Zug ausgewählt.

Schreiben Sie eine rekursive Prozedur `ZugFinder`, die zu einem gegebenen Brett, einer Spielerfarbe, die am Zug ist und einer Denktiefe den „besten“ Zug (aus der Sicht des Programms) findet und ihn zusammen mit seiner Bewertung zurückliefert. Die Prozedur ist *wesentlich* kürzer, als Sie auf den ersten Blick denken mögen, bei geschickter Programmierung weniger als 50 Zeilen! Sie arbeitet folgendermaßen:

Zuerst werden für den ziehenden Spieler alle legalen Züge ermittelt. Ist keiner möglich, hat der Spieler gewonnen und `ZugFinder` gibt einen beliebigen Zug zurück, er wird ja nicht mehr benutzt. Ein gewonnenes Spiel bedeutet ein sehr gutes Resultat für den ziehenden Spieler. Gemäß Stellungswert liefert der `Zugfinder` in diesem Fall also einen sehr hohen Wert zurück, falls Weiß am Zug ist oder eine sehr niedrige Zahl, falls Schwarz am Zug ist.

Ansonsten wird für alle durchführbaren legalen Züge folgende Schleife durchgeführt:

Das Brett wird kopiert, der Zug auf der Kopie ausgeführt und das entstandene Brett bewertet. Letzteres geschieht, indem entweder der `Zugfinder` sich selber rekursiv aufruft (!), dann natürlich mit einer anderen Spielerfarbe am Zug und einer um eins verringerten Denktiefe, oder aber, falls Denktiefe 0 erreicht wurde, wird der Stellungsbewerter aufgerufen. Falls dieser Zug besser war als der bisher beste (oder es der erste war, der probiert wurde), wird der Zug und die zugehörige Bewertung gemerkt.

Wenn alle Züge so durchprobiert wurden, ist der beste Zug gefunden. Dieser und die dazugehörige Bewertung werden vom `Zugfinder` zurückgegeben.

Die Bewertung, wann ein Zug besser ist als ein anderer, folgt dem sogenannte Minimax-Prinzip, das an einem Beispiel verdeutlicht werden soll: In der Ausgangsstellung (siehe Abb. 1) sei Weiß am Zug. Es soll gezeigt werden, wie die fettgedruckten Stellungswerte von unten nach oben ermittelt werden. Das Prinzip ist, daß sich der Wert eines Zuges aus den Werten aller möglichen Antwortzüge ergibt, oder, falls die vorgegebene Denktiefe erreicht wurde, aus dem Wert des Stellungsbewerter.

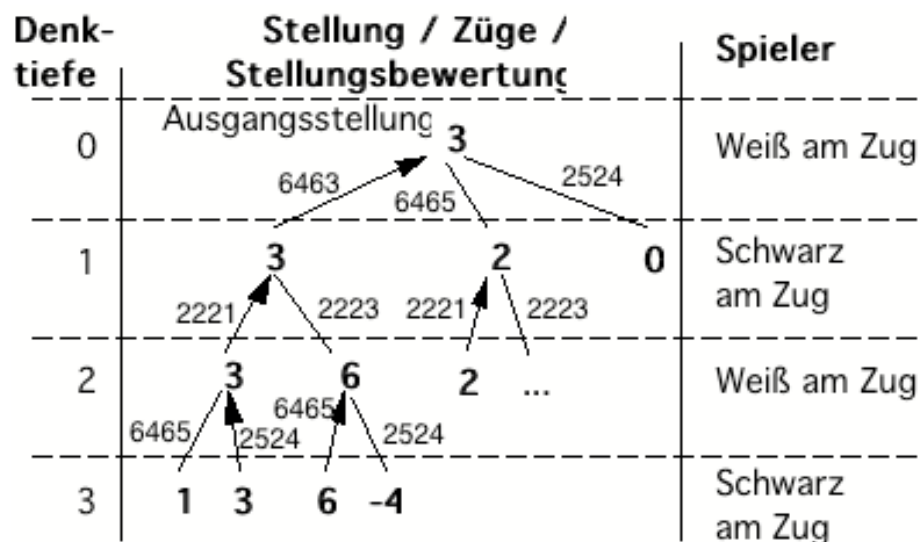


Abb.1: Spielbaum

Weiß habe in der fiktiven Ausgangsstellung drei legale Züge: 6463, 6465 und 2524. Wenn er 6463 ziehen würde, hätte Schwarz darauf zwei legale Antwortzüge: 2221 oder 2223. Auf den ersten Zug davon hat Weiß wieder zwei Zugmöglichkeiten: 6465 oder 2524. Diese werden nun mit dem Stellungsbewerter bewertet (die fest vorgegebene Denktiefe ist hier 3). Der Stellungsbewerter ermittle für die nach der Zugfolge 6463–2221–6465 entstandene Stellung den Wert 1 (fettgedruckte Zahl), die nach der Zugfolge 6463–2221–2524 den Wert 3. Da Weiß immer den besten Zug machen möchte und eine *größere* Zahl eine bessere Stellung für Weiß verspricht, würde Weiß, falls vorher 6463–2221 gezogen würde, Zug 2524 wählen, also den Zug, der die meisten Punkte ergibt. Aus der nach 6463–2221 entstandenen Stellung heraus kann Weiß nun im besten Fall einen Zug machen, der eine Stellung mit Wert 3 erlaubt. Dies ist damit auch der Wert des Zuges 2221 für Schwarz in der darüberliegenden Ebene! Dort versucht Schwarz natürlich eine Stellung mit möglichst *kleiner* Zahl zu erreichen, d.h. er wählt den Zug, gegen den Weiß den schlechtesten Antwortzug hat! Nun wird also zunächst der Wert für den Zug 2223 ermittelt. Der beste Zug, den Weiß nach Ausführung von 6463–2223 machen kann, ergibt eine Stellung mit dem Wert 6 und ist deshalb schlechter für Schwarz. Also wählt Schwarz den Zug, dessen Antwort nur maximal 3 Punkte zuläßt: 2221. Dies ist dann wiederum auch der Wert des allerersten Zuges für Weiß: 6463. Weiß versucht nun wiederum, den Stellungswert zu maximieren. Die Werte für die anderen Züge mögen mit dem gleichen Verfahren 2 und 0 ergeben. Diese sind alle kleiner, also schlechter für Weiß. Deshalb wird Weiß letztendlich den ersten Zug, nämlich 6463 als den besten Zug wählen. Dies ist das Ergebnis des Zugfinders und wird zusammen mit dem Wert 3 zurückgeliefert. Jetzt zeigt sich, wie gut Ihr Stellungsbewerter arbeitet! Ergeben zwei Züge den gleichen Wert, so ist es egal, für welchen man sich entscheidet. Hier könnte allerdings auch ein Zufallszahlengenerator entscheiden, so daß Ihr Zugfinder auch mal andere, gleichwertige Züge findet.

Machen Sie sich klar, daß jeder Teilbaum einen rekursiven Aufruf des Zugfinders mit verändertem Brett und veränderter Denktiefe darstellt und nur die Blätter einen Aufruf des Stellungsbewerter darstellen. Lassen Sie sich durch die Rekursion nicht verwirren. Wählen Sie zum Testen kleine Denktiefen (<5)! Falls die Rechenleistung es zuläßt, können Sie die Denktiefe erhöhen.

Diese Suchmethode, die ja alle Zugmöglichkeiten beachtet, wird in der Praxis „brute-force“, Methode der brutalen Gewalt, genannt. Durch die Technik „alpha-beta-pruning“, die gewisse Teile des Spielbaumes ausläßt, läßt sich die Geschwindigkeit drastisch erhöhen. Aus Zeitmangel wird dies hier jedoch nicht mehr durchgeführt.