Ostbayerische Technische Hochschule
Amberg-Weiden

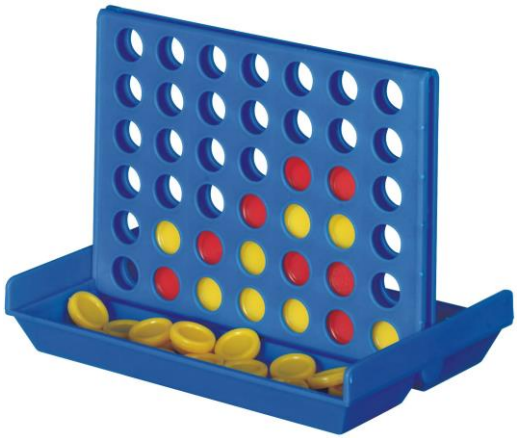
Benotete Aufgabe 2

Benotete Aufgabe 2

Ostbayerische Technische Hochschule
Amberg-Weiden

Erstellen Sie eine KI (Planer) für das normale Vier gewinnt

- Schreiben Sie eine KI, welche für eine gegebene Spielsituation selbstständig den nächsten Zug plant und ausführt (ein entsprechendes Framework wird bereitgestellt)
- Die KI muss sowohl als Spieler 1 oder Spieler 2 offline ausführbar sein
- Durch händisches Setzen eines Flags (Variable in Ihrem Programm) wird nach jedem Zug der aktuelle Fortschritt der Partie (Bildschirmausgabe wie in Aufgabe 1) in einer Datei "log.txt" angehängt
- Erstellen Sie eine kurze Präsentation (~5 Folien), in der Sie ihre Ideen, Ansätze und Ergebnisse präsentieren, insbesondere gegen welche der bereitgestellten KIs (`ai2_stageX`) Ihre KI welches Ergebnis basierend auf jeweils 100 durchgeführten Partien erzielte



Benotete Aufgabe 2

Ostbayerische Technische Hochschule Amberg-Weiden

Mögliche Ansätze (in aufsteigender Stärke)

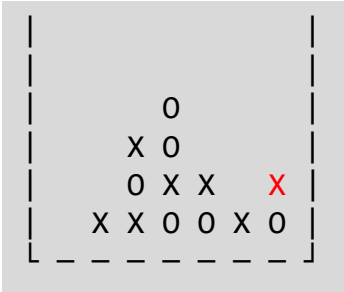
- zufällige Züge
- Heuristik (in `ai2_stage1` implementiert)
- Baumsuche + schlechte Heuristik
- Baumsuche + gute Heuristik + Alpha-Beta-Pruning (in `ai2_stage7` implementiert)
- … (irgendwas komplett anderes)

Benotete Aufgabe 2

Ostbayerische Technische Hochschule Amberg-Weiden

Möglicher Ansatz: Zufällige Züge

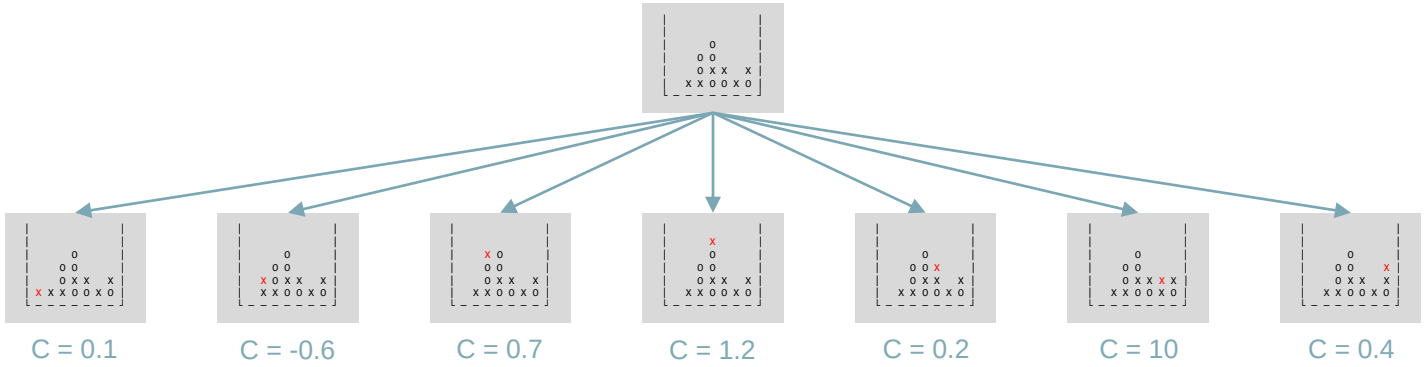
- Idee: Wähle eine zufällige (idealerweise nicht volle) Reihe in die der eigene Spielstein geworfen wird



Benotete Aufgabe 2

Möglicher Ansatz: Heuristik

- Idee: Berechne eine Zahl (Heuristik) C für jede Reihe, die in Abhängigkeit verschiedener Kriterien C_1, C_2, … angibt, wie gut oder schlecht es ist einen eigenen Stein in die jeweilige Reihe zu werfen
- Wähle die Reihe für den nächsten Zug aus, die den höchsten Wert C hat



Benotete Aufgabe 2

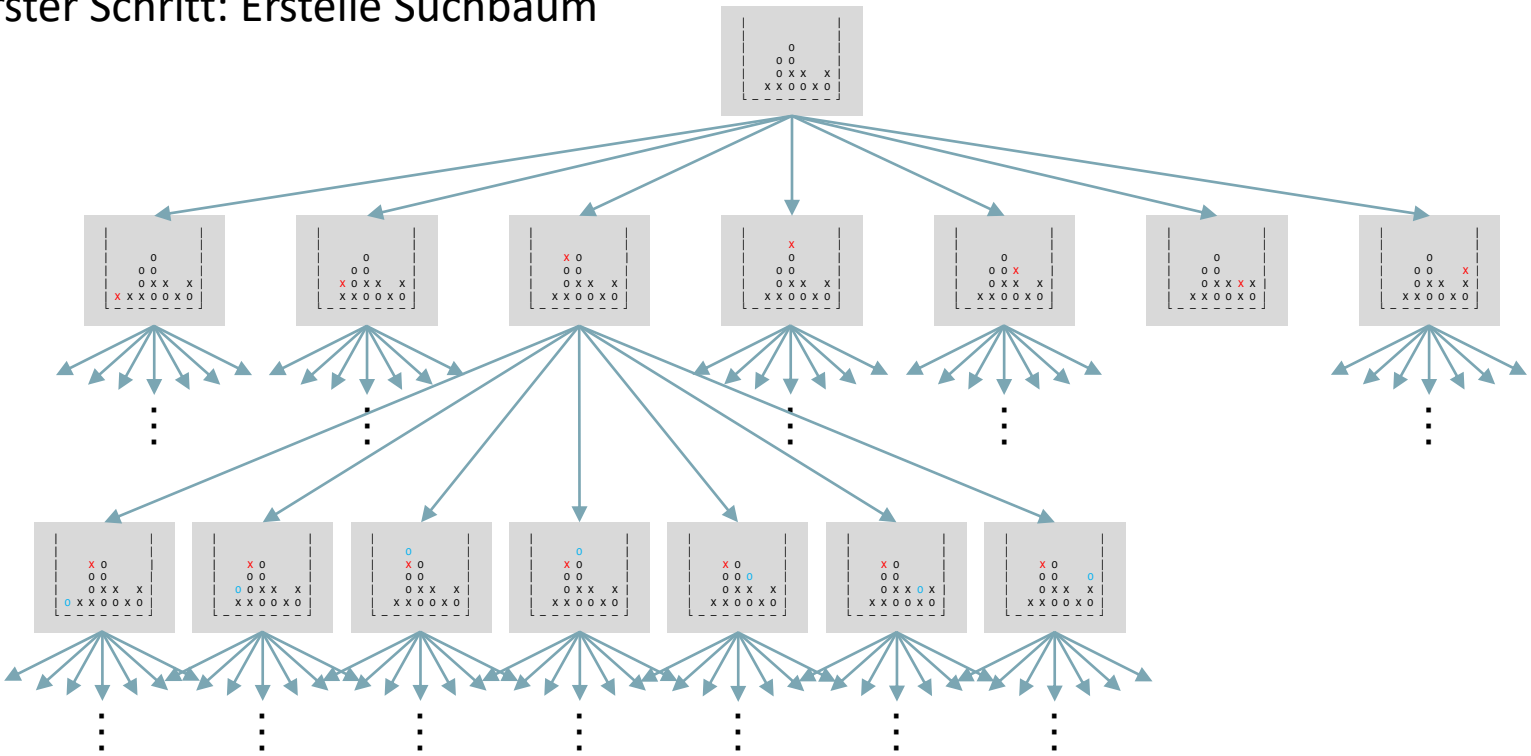
Ostbayerische Technische Hochschule
Amberg-Weiden

- Eine Idee für eine mögliche Kostenfunktion ist
  $C = \alpha \cdot C_1 + \beta \cdot C_2 + \cdots$
  mit händisch zu bestimmenden Gewichtungsfaktoren α, β, …
- Ideen für mögliche Kriterien C_i sind
  - Gewinne ich oder der Gegner?
  - Ist es die mittlere Reihe bzw. ein Feld in der Spielfeldmitte?
  - Wieviele eigene/gegnerische Steine sind in den angrenzenden acht Feldern?
  - Wieviele waagerechte/senkrechte/diagonale Reihen mit 2/3/4 eigenen Steinen gibt es, wenn ich den Stein in die jeweilige Reihe werfe?
  - …

Benotete Aufgabe 2

Möglicher Ansatz: Baumsuche + Heuristik

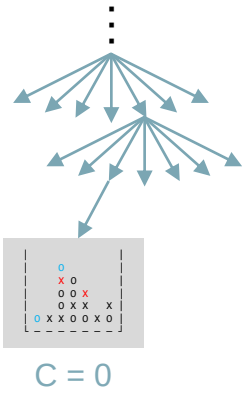
- Idee: Berechne n eigene + gegnerische Züge (Knoten des Baums) im Voraus und wähle unter allen Zügen die beste Sequenz aus
- Erster Schritt: Erstelle Suchbaum



Benotete Aufgabe 2

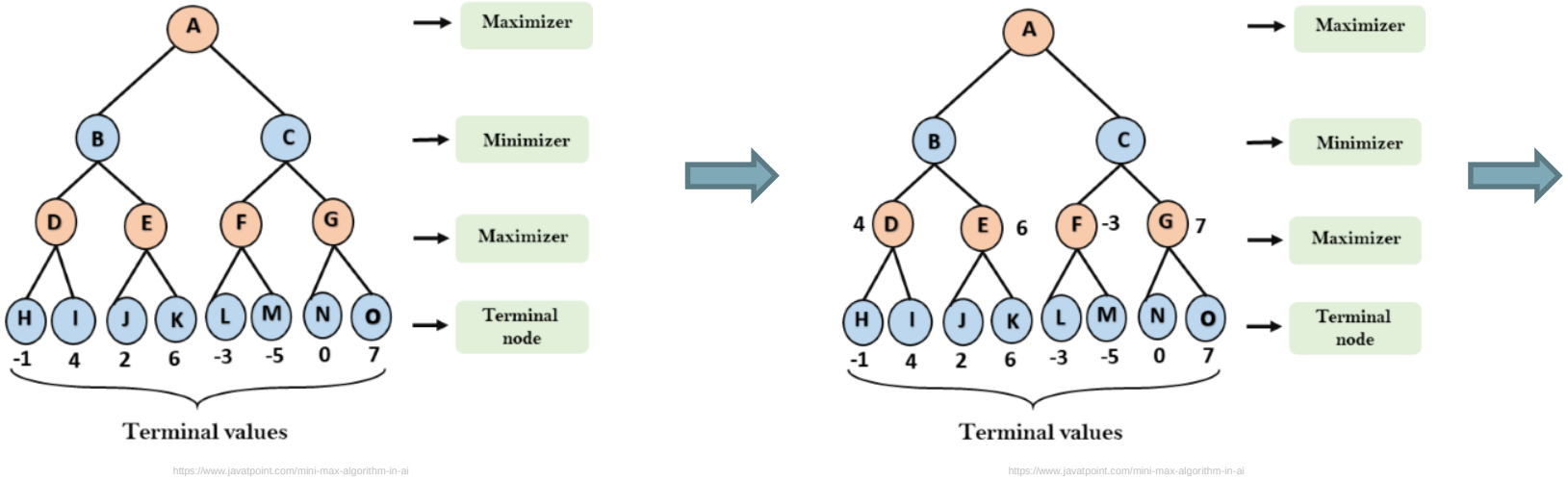
- Sobald der Suchbaum komplett erstellt wurde, wird für jedes Blatt (Spielsituation am Ende der Züge) eine Heuristik C angegeben, die evaluiert wie gut oder schlecht die jeweilige Situation für den Spieler ist
- Idee für simple Heuristik:
  $C = 1$ falls Spieler gewinnt
  $C = -1$ falls Gegner gewinnt
  $C = 0$ sonst
- Es dürfen auch andere, komplexere Heuristiken (siehe vorherige Folien) verwendet werden

→ hohe Werte von C sind gut für den Spieler, niedrige Werte von C sind gut für den Gegner



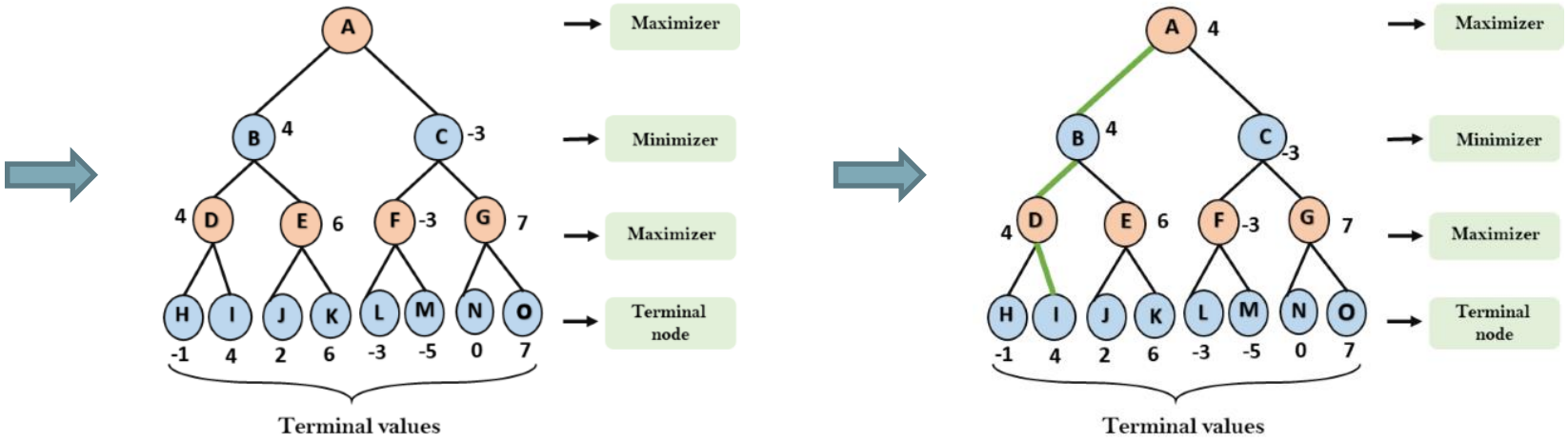
Benotete Aufgabe 2

- Minimax-Algorithmus: Gehe ausgehend von den Blättern rückwärts durch den Baum und wähle Wert für alle Knoten wie folgt aus:
  - wenn Spieler am Zug ist, wähle den Zug aus, der den höchsten C-Wert hat (Maximierer)
  - wenn Gegner am Zug ist, wähle den Zug aus, der den niedrigsten C-Wert hat (Minimierer)



Benotete Aufgabe 2

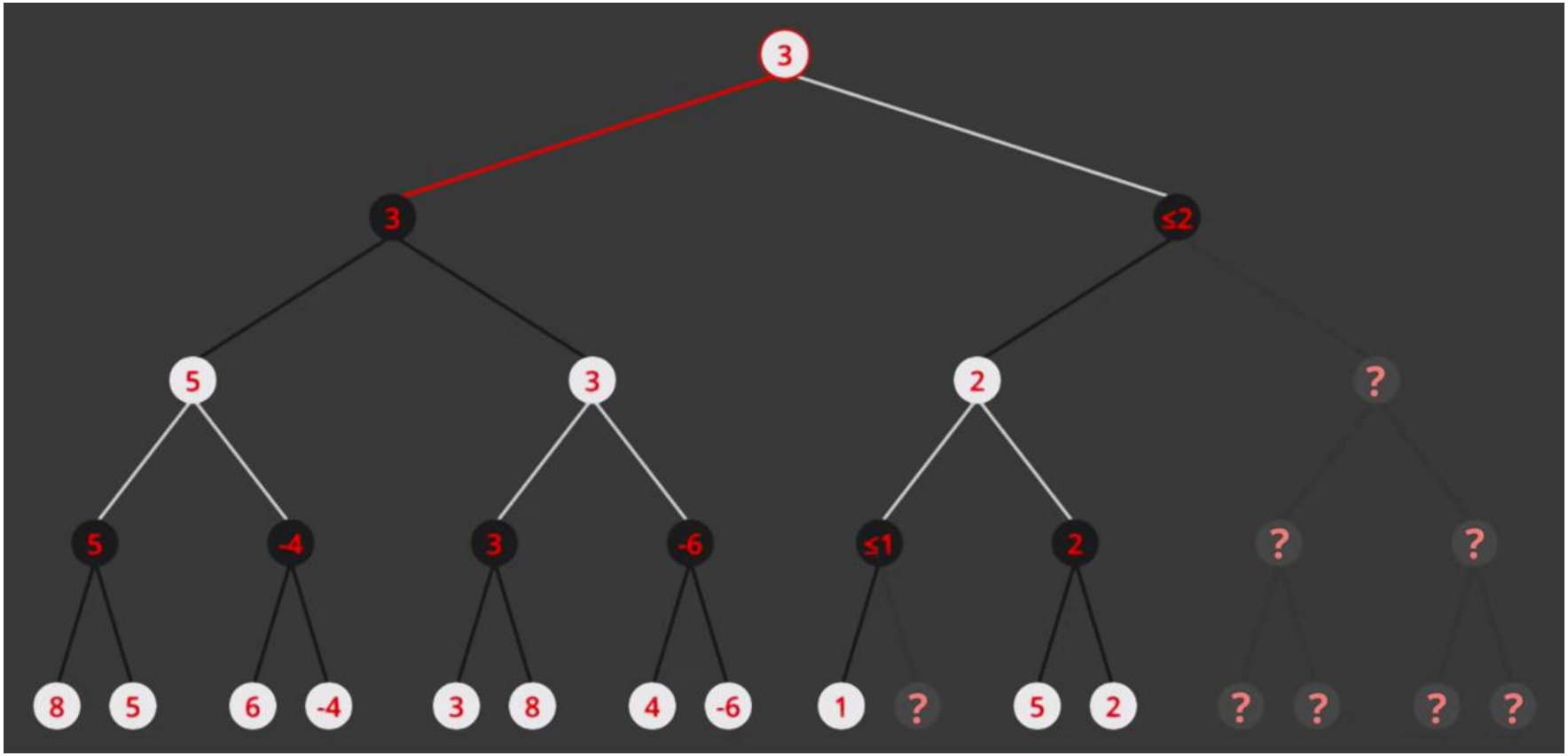
- Am Ende: Wähle den Zug als nächsten eigenen Zug aus, der den höchsten C-Wert hat (Sequenz von Zügen, die zum besten Ergebnis führt)



Benotete Aufgabe 2

Ostbayerische Technische Hochschule
Amberg-Weiden

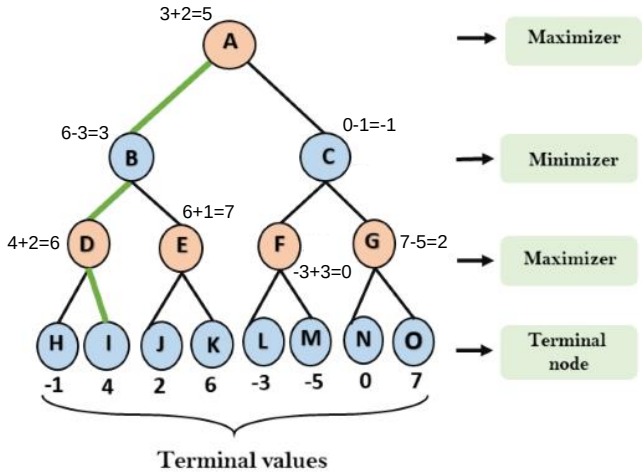
Mögliche Erweiterung: Alpha-Beta-Pruning beschleunigt das Durchsuchen des Suchbaums, tlw. um einen Faktor >100 (gutes Erklärungsvideo)



https://www.youtube.com/watch?v=I-hh51ncgDI

Benotete Aufgabe 2

Mögliche Erweiterung: Update Werte der Parent-Knoten nicht nur mit Heuristik der Child-Knoten, sondern mit Heuristik der Parent-Knoten + Heuristik der Child-Knoten (evaluiere Qualität des aktuellen Zugs + Qualität der zukünftigen Züge)



Benotete Aufgabe 2

Ostbayerische Technische Hochschule Amberg-Weiden

Hilfe zur Aufgabenstellung:

- Der Suchbaum / Minimax-Algorithmus lässt sich elegant mittels Rekursion realisieren
- Die Ausführungszeit lässt sich mit dem `time` Modul erfassen
- In der finalen Submission soll nur ihre stärkste KI lauffähig sein, Sie dürfen aber gerne (und sollen) ihre anderen Version unten in der `ai1.py` Datei als auskommentierten Code anfügen und auch in Ihrer Präsentation vorstellen
- Das Programm erhält mehrere KIs verschiedener Stärker zur Evaluation ihrer KI. Hierzu benötigen sie das Modul `pyarmor`, welches sie z.B. in einer Anaconda Eingabeaufforderung installieren können:
  In den Unterordnern `ai2_stage1` (leichteste KI) bis `ai2_stage7` (schwierigste KI) finden Sie unterschiedliche, von mir implementierte KIs. Kopieren Sie den Inhalt des Ordners in den jeweiligen Überordner (`ai2.py` wird hierbei ggf. überschrieben) und starten Sie ihr Programm

Benotete Aufgabe 2

Ostbayerische Technische Hochschule Amberg-Weiden

Hinweise zur Abgabe:

- **Falls Fragen/Unklarheiten/Probleme/Fehler existieren → mich fragen**
- Die Bearbeitung der Aufgabe erfolgt einzeln
- Die Bearbeitung der Aufgabe erfolgt in der Programmiersprachen Python, Version 3.7 oder höher
- Schreiben Sie ihr komplettes Programm in der Datei `ai1.py`
- Ihr Programm muss durch Kopieren des Codes von `ai1.py` nach `ai2.py` auch als Spieler 2 ausführbar sein
- Verändern Sie in ihrer finalen Submission nicht die Dateien `config.py`, `vier_gewinnt.py` und `helper.py`
- Verwenden Sie nur Bibliotheken, die in der Python Standard Library enthalten sind sowie pyarmor
- Das Programm muss ohne weitere manuelle Konfiguration durch Aufruf der `vier_gewinnt.py` Datei lauffähig sein
- Das Programm braucht zur Planung und Ausführung des nächsten Spielzugs auf einem durchschnittlichen Notebook nicht mehr als 300ms
- Fügen Sie ihrer Submission eine `readme.txt` Datei hinzu, in der folgende Information stehen: Verwendete Python-Version sowie die Benutzung des hochgeladenen Programms (als Backup zum Debuggen, falls es nicht sofort läuft)
- Die Präsentation muss im Powerpoint- oder PDF-Format sein
- Am 18. Januar besteht Anwesenheitspflicht für alle, dort werden Sie einerseits Ihre programmierte KI (zweiter Teil der Modularbeit) vergleichen, andererseits werde ich Ihnen ein paar Fragen zum Code stellen um zu überprüfen inwiefern Sie Ihn verstehen
- **Die Submission erfolgt via moodle, die Deadline ist der 17. Januar 2023, 23:59:00**

Benotete Aufgabe 2

Ostbayerische Technische Hochschule Amberg-Weiden

Bewertungskriterien:

- **Lauffähigkeit**: Der Code ist fehlerfrei und lauffähig
- **Korrektheit**: Die Submission erfüllt die gestellten fachlichen Anforderungen
- **Performance**: Der Code implementiert eine möglichst starke KI

} am wichtigsten

- **Struktur und Lesbarkeit**: Der Code ist klar strukturiert, modularisiert, gut lesbar und nachvollziehbar
- **Eleganz und Effizienz**: Der Code ist elegant, effizient und verwendet – sofern verfügbar – bereits vorhandene Python-Funktionen zur Bearbeitung der Aufgabe
- **Verwendete Ansätze**: Implementierte Techniken und Kreativität bei der Entwicklung