Games (KI) – Aufgaben 1–5: Minimax, Alpha–Beta, Heuristiken & Mehrspieler

Aufgabe 1: Vergleich *verlässlicher Quellen* vs. LLM-Antworten

Beispiel I ("historisches Ereignis"): Bevölkerungsentwicklung Kiribati

Aufgabe an das LLM(DeepAI): "Nenne die Bevölkerungszahlen von Kiribati für 1931, 1947, 1963, 1968, 1973, 1978, 1985, 1990, 1995, 2000, 2005, 2010, 2015, 2020."

LLM-Antwort (verkürzt): 1931 $\approx$ 16,000; 1947 $\approx$ 17,500; ... 2010 $\approx$ 36,000; 2015 $\approx$ 103,000; 2020 $\approx$ 119,000.

Vergleich mit Quellen (Zensus/amtliche/UN-Reihen):

1931: 29 751, 1947: 31 513,
1963: 43 336, 1968: 55 529,[1] 1973: 51 926,
1978: 56 213, 1985: 63 883,
1990: 72 335, 1995: 77 658,
2000: 84 494, 2005: 92 533,
2010: 103 058, 2015: 110 136,
2020: 119 940.[2]

Befund: Die LLM-Zahlen sind in den frühen Jahren um den Faktor $\sim$2–3 zu niedrig (Halluzination/Fehlquelle). 2010 ist grob falsch (36k statt 103k). Ab 2015/2020 stimmen die Größenordnungen, die exakten Zensuswerte weichen jedoch.
Ursache: Fehlende oder unzuverlässige Quelle; außerdem fehlte der Verweis auf amtliche Tabellen (Kiribati NSO 1968–2010; Census Atlas 2022 mit 1947–2020).[3]

[3] Kiribati NSO: Census-Tables 1968–2010. `https://nso.gov.ki/census/census-tables-1968-2010/`. Kiribati Census Atlas 2022: historische Tabellen 1947–2020. `https://spccfpstore1.blob.core.windows.net/digitallibrary-docs/files/ef/ef3b780972ce41ee779c6aed507dbbff.pdf`

Beispiel II (Standardproblem Informatik): AVL-Baum

LLM-Lösung (Java, auszugsweise):

- `isAVL(root)` prüft nur Höhenbalance, *nicht* die BST-Eigenschaft.
- Umwandlung: *inorder* in Liste → `sortedListToAVL`; dabei wird *nicht* sortiert.

Beurteilung: *Nicht korrekt.*

(a) Ein AVL ist ein *balancierter Binärsuchbaum*. Ohne BST-Prüfung akzeptiert der Code z. B. `10` mit linkem Kind `20` fälschlich als AVL.

(b) Die Umwandlung setzt eine sortierte Folge voraus; *inorder* eines Nicht-BST ist unsortiert ⇒ erzeugter Baum ist u. U. weder BST noch AVL.

(c) Keine Regel für Duplikate; keine Rotationen (bei Einfüge-Variante).

Korrekturmöglichkeiten (kurz): (i) AVL-Test = BST-Invariante *und* Höhenbalance in $O(n)$; (ii) Für die Umwandlung alle Schlüssel sammeln und *sortieren* ⇒ `sortedListToAVL`; oder alle Schlüssel in einen leeren AVL mit Rotationen einfügen.

Aufgabe 2 (Games.02): Minimax & Alpha–Beta an Tic-Tac-Toe

```
def minimax(state, player):
    term, u = terminal(state)
    if term: return u, None
    best, best_move = (-inf, None) if player=='X' else (inf, None)
    for m in legal_moves(state):
        s2 = play(state, m, player)
        v, _ = minimax(s2, 'O' if player=='X' else 'X')
        if player=='X' and v>best or player=='O' and v<best:
            best, best_move = v, m
    return best, best_move

def alphabeta(state, player, alpha=-inf, beta=inf):
    term, u = terminal(state)
    if term: return u, None
    if player=='X':
        v, mv = -inf, None
        for m in order(legal_moves(state)):
            v2, _ = alphabeta(play(state,m,'X'),'O',alpha,beta)
            if v2>v: v, mv = v2, m
            alpha = max(alpha, v)
            if alpha>=beta: break
        return v, mv
    else:
        v, mv = inf, None
        for m in order(legal_moves(state)):
```

```
            v2, _ = alphabeta(play(state,m,'O'),'X',alpha,beta)
            if v2<v: v, mv = v2, m
            beta = min(beta, v)
            if alpha>=beta: break
        return v, mv
```

Vergleich Knotenzahlen (Beispiel-Szenarien).

- Startposition: Minimax $\approx 5.5 \cdot 10^5$ Knoten; $\alpha\beta$ ohne Ordnung $\approx 1.8 \cdot 10^4$; mit einfacher Ordnung (Zentrum $\rightarrow$ Ecken $\rightarrow$ Kanten) $\approx 7.3 \cdot 10^3$.
- Mittelspiel: 53 vs. 26 (ohne Ordnung) vs. 29 (mit Ordnung).
- Endspiel (wenige Züge): alle ≈ 5.

Aufgabe 3 (Games.03): Minimax vereinfachen (Nullsummenspiel)

Negamax. Durch Nullsummenannahme lassen sich `max` und `min` in *eine* Funktion integrieren:

```
def negamax(s, color): # color = +1 (MAX) oder -1 (MIN)
    if terminal(s): return color * utility(s)
    best = -inf
    for a in actions(s):
        best = max(best, -negamax(result(s,a), -color))
    return best
```

Alpha–Beta passt analog durch Negation/Tausch von (α, β).

Beispielbaum (Tiefe 2). Blätter (aus MAX-Sicht): $(+3, +5)$ links, $(+2, -1)$ rechts unter MIN.
Minimax: $\max(\min(3,5), \min(2,-1)) = \max(3,-1) = 3$.
Negamax liefert denselben Wert (Vorzeichenwechsel im Rekursionsschritt).

Aufgabe 4 (Games.04): Suchtiefenbegrenzung & Heuristik für Tic-Tac-Toe

Evaluationsfunktion (aus Vorlesung):

$$\text{Eval}(s) = 3X_2(s) + X_1(s) - (3O_2(s) + O_1(s)),$$

wobei X_n (bzw. O_n) die Zahl der *offenen* Reihen/Spalten/Diagonalen mit genau n X (bzw. O) ist. *Sinnvoll*, weil "fast fertige" Dreierreihen (n=2) stark gewichtet, Nullsummen-symmetrisch und billig berechenbar.

Beispielwerte (6 Zustände).

1. Terminal X gewinnt: Nutzen $+1$ (Eval $= +1$).
2. Terminal O gewinnt: -1 (Eval $= -1$).
3. Remis: 0 (Eval $= 0$).
4. Zwischenstand: Zentrum X, Ecke O $\Rightarrow (X_2, X_1, O_2, O_1) = (0, 3, 0, 2) \Rightarrow$ Eval $= 1$.
5. X mit zwei auf Hauptdiagonale: $(1, 3, 0, 1) \Rightarrow 5$.
6. X mit zwei oben in einer Reihe: $(1, 2, 0, 1) \Rightarrow 4$.

Aufgabe 5 (Games.05): Generalisierter Minimax (3 Spieler)

Regel: Am Knoten von Spieler i wähle das Kind, das die i-te Komponente des Nutzenvektors maximiert (keine Allianzen).
Ergebnis der angegebenen Blätter (links → rechts):
Spieler-1-Knoten über den Blättern: $A = (4, 2, 1)$, $B = (7, 4, -1)$, $C = (5, -1, -1)$, $D = (7, 7, -1)$, $E = (5, 4, 5)$.
Spieler-2-Knoten: links $(7, 4, -1)$, mitte $(5, -1, -1)$, rechts $(7, 7, -1)$.
Wurzel (Spieler 1): $(7, 4, -1)$ *oder* $(7, 7, -1)$ (beide optimal bzgl. x_1).
