

# מבוא לבינה מלאכותית

## סמסטר חורף תשפ"ו

### מטלה 2

תאריך הגשה: 23:55 10.12.25

#### הנחיות

- שאלות בנושא מטלה זו יש לשאול דרך המודל, בפורום "מטלה 2".
- הוראות להגשת המטלה מופיעים בסוף מסמך זה.
- הקבצים הנדרשים להרצת הקוד הינם:
  - minimax\_isoKnight.py
  - player\_agent.py
  - heuristics.py
  - alpha\_beta\_isoKnight.py
  - heuristic\_alpha\_beta\_isoKnight.py
  - game\_state.py
  - game\_engine.py
- העבודה להגשה בזוגות בלבד אלא אם ניתן אישור מיוחד להגיש אחרת.
- לפני שניגשים לממש את המטלה מומלץ לעיין רבות בהסברים וכן בקוד הקיים.
- פתרון המטלה שתגישו ייבדק מול שאר ההגשות על ידי תוכנת העתקות.
- **מי שימצא כי העתיק יכשל בקורס וכן יועבר לוועדת משמעת אוניברסיטאית.**
- הפרויקט נכתב וייבדק בשפת התכנות python.
- יש להתקין את החבילה numpy.
- מסמך זה בנוי באופן הבא: תיאור המטלה, בעיית משחק שני שחקנים, מרחב הבעיה, שאלות המטלה, הסבר על המימוש הקיים והסבר על הגשת המטלה.

#### תיאור המטלה

נתון משחק שני שחקנים בשם IsoKnight. בהינתן לוח משחק  $m * n$ , לכל שחקן יש דמות המתחילה במיקום מסוים על הלוח. כל שחקן בתורו יכול להזיז את הדמות שלו בדיוק כמו פרש בשחמט (הסבר בהמשך). הדמות יכולה לזוז לכל משבצת בכיוון מסוים אם שלושת התנאים הבאים מתקיימים:

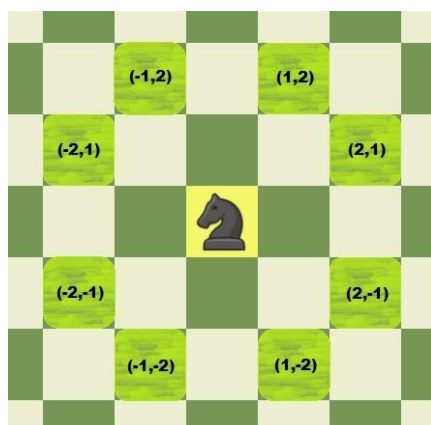
1. לא ניתן להגיע למשבצת שכבר ביקרו בה
2. ניתן לדלג מעל משבצות שביקרו בהן
3. ניתן לדלג מעל לדמות השחקן השני

כיצד פרש זז? – לפרש יש שני צעדים אפשריים:

שני צעדים באחד מארבעת הכיוונים (למעלה, למטה, ימינה או שמאלה) ולאחר מכן צעד אחד הצידה. צעד אחד באחד מארבעת הכיוונים (למעלה, למטה, ימינה או שמאלה) ולאחר מכן שני צעדים הצידה.

תמונה הממחישה את כל הצעדים האפשריים בעמוד הבא.

הפרש יכול לדלג מעל כל המשבצות פרט לצעד האחרון (התא אליו הוא עובר).



## בעיית משחק שני שחקנים

בהינתן מיקום התחלתי של שני השחקנים, עליכם לממש אלגוריתמים שיחשבו את ערך ה-maximin של המשחק.

## מרחב הבעיה

בכל שלב במשחק, הדמות נמצאת באחד התאים במטריצה, כאשר המטרה שלה היא שלדמות השניה יגמרו המהלכים לפניה.

דוגמה ללוח משחק:



## שאלות המטלה

- עליכם לממש את אלגוריתם `alphabeta`. לשם כך האלגוריתמים `minimax` ו-`maximin` ממומשים בקובץ `minimax_isoKnight.py`, ואילו את אלגוריתם `alphabeta` יש לשנות ולהתאים בקובץ `alpha_beta_isoKnight.py`. האלגוריתם הממומש כעת בקובץ `alpha_beta_isoknight.py` הינו גם כן `minimax` ו-`maximin`, ויש לשנות את הפונקציות בקובץ כך שהאלגוריתם שיממוש יהיה `(alpha beta pruning)`. שימו לב, כאשר מבצעים גיזום יש להחזיר את `v` וגם `None`, מכיוון שתוכנית זו שומרת את הצעד שיש לבצע עבור השחקן הפורס את עץ המשחק (כלומר, "return v, None"). ניתן לשנות את חתימות הפונקציה בשביל נכונות האלגוריתם `alphabeta`.

- עליכם לממש את אלגוריתם alphabeta גם בקובץ heuristic\_alpha\_beta\_isoKnight.py (אותם שינויים כמו בסעיף א'). בנוסף, כדי להתמודד עם עצי משחק גדולים יותר- עליכם לממש יוריסטיקה עבור קודקודים שאינם מציינים סיום משחק. השלימו את הפונקציה base\_heuristic בקובץ heuristics.py כך שתקבל מצב של משחק ותחזיר את ההפרש בין כמות הצעדים החוקיים של השחקן הראשון לבין הצעדים החוקיים של השחקן השני. כלומר, עליכם תמיד להחזיר p2\_moves - p1\_moves (לא משנה תור מי לשחק בצעד הנתון).

- הינכם יכולים לממש יוריסטיקה מתקדמת בפונקציה advanced\_heuristic. היוריסטיקה הטובה ביותר תזכה את המגשים ב 2 נקודות בונס לציין הסופי. כדי להחליט על היוריסטיקה הטובה ביותר, נערוך טורניר בין קבוצות המגשים כך שהאסטרטגיה של השחקן תשתמש ב advanced\_heuristic על מנת להחליט על כל אחד מצעדי המשחק. הקבוצה שתנצח בטורניר תקבל את נקודות הבונס.

**שימו לב:** כאשר אתם מריצים את אלגוריתם minimax אתם תוכלו לחשב משחקים לוח בגודל 5X5 (ולפעמים 6). כדי לנסות לשחק על לוח בגודל 6X6 עליכם לממש את alphabeta pruning. בשביל לוחות גדולים יותר תצטרכו להשתמש ביוריסטיקה ולהגביל את עומק החיפוש.

## הגשת המטלה

- יש להגיש **שלושה קבצים בלבד** לאתר המודל בדומה להגשת המטלה הקודמת. הקבצים הנדרשים:

heuristics.py   ○  
 alpha\_beta\_isoKnight.py   ○  
 heuristic\_alpha\_beta\_isoKnight.py   ○

הגשת המטלות תתבצע ישירות מול מערכת המודל בצורה אלקטרונית.

- **שימו לב:** ישנה אפשרות להגיש את המטלה התכנותית מספר פעמים ובכל הגשה לקבל חיווי, כלומר תקבלו באופן מיידי את הציון לביצוע התרגיל.
- ניתן לראות את הפידבק להרצה (קומפילציה, מספר טסטים שעברו, שגיאות זמן ריצה וכו'...).

לאחר סיום ההרצה יתקבלו התוצאות.