

מבוא לבינה מלאכותית

סמסטר חורף תשפ"ו

מטלה 1

תאריך הגשה: 23:55 25.11.25

הנחיות

- שאלות בנושא מטלה זו יש לשאול דרך המודל, בפורום "מטלה 1".
- הוראות להגשת המטלה מופיעים בסוף מסמך זה.
- הקבצים הנדרשים להערכת הקוד הינם:
 - heuristics.py
 - main.py
 - color_blocks_state.py
 - search.py
 - search_node.py
- העבודה להגשה בזוגות בלבד אלא אם כן המগישים קיבלו אישור להגשה שאינה בזוגות.
- לפני שניגשים למשם את המטלה מומלץ לעין רבות בהסבירים וכן בקוד הק"ם.
- פתרון המטלה שתגלו יבדק מול שאר ההצעות על ידי תוכנת העתקות.
- מי שימצא כי העתיק יכשל בקורס וכן יועבר לוועדת משמעת אוניברסיטאית.
- הפרויקט נכתב ויבדק בשפת התכנות python3.10.
- מסמך זה בניין באופן הבא: תיאור המטלה, בעיית החיפוש, מרחב הבעיה, שאלות המטלה, פונקציות שיש להשלים והסבר על פונקציות קיימות.

תיאור המטלה

כדי להרשים את חברי, דני בנה רובוט שטמיין מגדל של קוביות צבעוניות. המגדל מורכב מקוביות כר-ש-2 מהפאות הצדדיות צבעונות בצד אחד ו- 2 הפאות האחוריות צבעונות בצד אחר (אין צבע לפחות העליונה והתחתונה ולא נשתמש בהן). בכל נקודת זמן יש צבע אחד שפונה אל החברים (החלק הקדמי) וצבע שפונה הצדיה מהחברים (החלק הצדדי). חברי של דני יכול לראות רק את החלק הקדמי של מגדל הקוביות. הרובוט יכול לשובב קובייה אחת בכל מיקום במגדל כך שצבע אחר יוצג כלפי החברים (סיבוב של 90 מעלות). כמו כן, הרובוט יכול גם להפוך בזריזות את החלק התחתית של מגדל הקוביות כך שסדר וצבע הקוביות נשמר אך בסדר הפוך. כדי להפוך את תחתית המגדל הרובוט חייב לתפוס את הקובייה התחתית ביותר ועוד קובייה במגדל. הרובוט יהפוך רק את הקוביות שנמצאות בין הקוביות שתتفس. אם הרובוט תפס את הקובייה התחתית ביותר ואת הקובייה העליונה ביותר, הוא יהפוך את כל המגדל כך שהקוביה העליונה תהיה תחתית והקוביה התחתית תהיה בראש המגדל. כמו כן, כל הסדר בין קוביות יתפרק. כדי להפחית את הסיכוי שהרובוט יFAIL את המגדל, עזרו לדני למשם אלגוריתם * A כדי למזער את כמות הפעולות המופעלות על המגדל.

בעיית חיפוש

בהתנחת מגדל קוביות צבעוניות וסידור הצבעים הגלויים הממוין (גם קלט). علينا למצוא את רצף הממצבים המינימלי שבצערתן הרובוט יוכל למיין את המגדל המקורי שיציג את הסידור הצבעים הגלויים במצב הסופי.

מרחב הבעיות

רשימה בגודל N של קוביות כך שלכל קובייה יש 2 צבעים:

$$\forall i_{1 \leq i \leq N}: color_1(cube_i) \in \mathbb{Z}, color_2(cube_i) \in \mathbb{Z}$$

לכל קובייה יש צבע אחד גלי (החלק הקדמי) וצבע אחד נסתר (החלק הצדדי)

לכל קובייה יש שני צבעים שונים:

$$\forall i_{1 \leq i \leq N}: color_1(cube_i) \neq color_2(cube_i)$$

אר יכולות להיות 2 קוביות שונות עם צבע משותף.

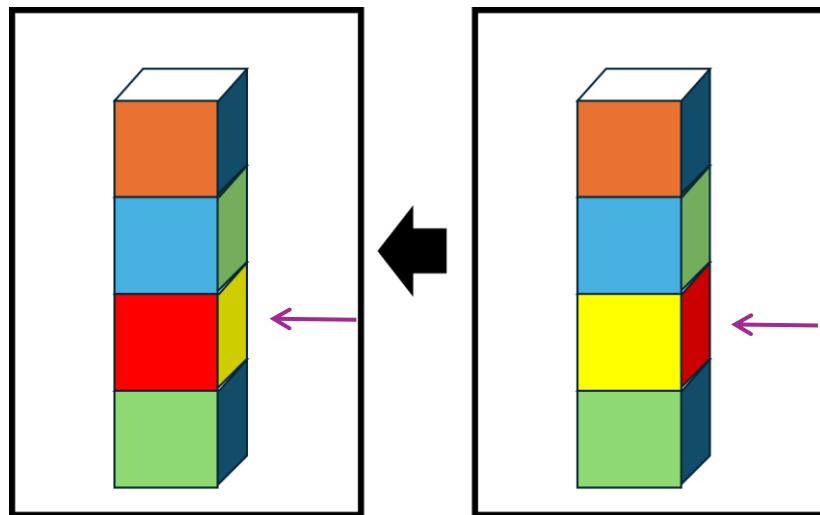
מצב התחלתי: רשימת קוביות צבעוניות לכל קובייה יש 2 צבעים ואחד הצבעים הוא הצבע הגלוי. הרשימה בנויה מזוג מספרים המסמנים את הצבעים של הקובייה כך שהמספר השמאלי הינו הצבע שמוצג והימני מוסתר. ראש המגדל הינו בצדיו השמאלי של הרשימה והקובייה בתחום המגדל בצד הימני של הרשימה.

מצב סופי: סדר הצבעים הגלויים של הקוביות.

אופרטורים:

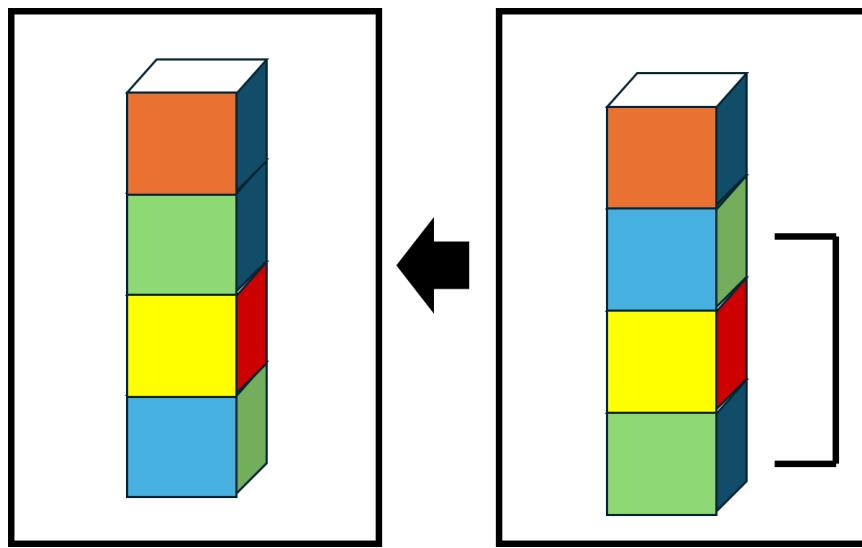
1. chin-Skip-סיבוב של קובייה יחידה (90 מעלות) מבלי לשנות את סדר הקוביות במגדל.

דוגמא לפעולות סיבוב של קובייה (הקוביה השנייה מלמטה):



.2. flip- הפיכה של תות מגדל מתחתית המגדל.

דוגמא לפעולות הפיכה:



שימוש לב: הסידור של כל הקוביות מתחת ערימה מתהפרק, בגלל שסכום הקוביות שהתחכפו אי זוגית, הקוביה הצהובה נשארה במקום. **עלות כל פעולה הינה 1.**

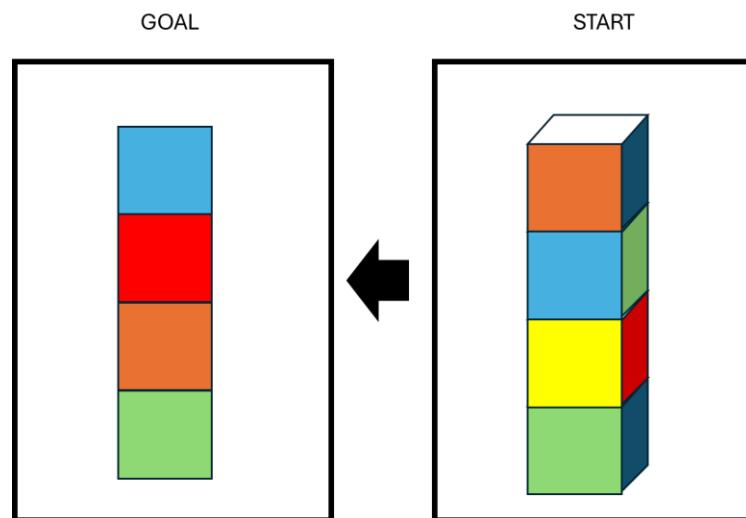
שאלות המטלה

א. עזרו לדני להשלים את הפונקציות החסומות (מציענות בהמשך).

ב. שנו את היוריסטיקה base heuristic כך שתחשב את היוריסטיקה הבאה- עבור כל זוג קוביות צמודות – אם קיימת קומבינציה צבעים של 2 הקוביות במצב הסופי – נסיף 0 ליוריסטיקה. אחרת, נסיף 1. קומבינציה צבעים הינה בחירת צבע מכל קובייה(סדר הקוביות לא חשוב).

שימוש לב: הסדר של שני הצלבים אינו חשוב העיקר שקיים צימוד צזה במצב הסופי.

לדוגמה:



עבור המצב הinitial מימין והמצב הסופי משמאלי, היריסטיקה תחזיר $1 = 0 + 0 + 0$.
 זוג הקוביות העליון מוסף 0 כי קיימת קומבינציה צבעים של כתום וירוק במצב הסופי. זוג הקוביות הבא גם מוסף 0 לאחר ותכלת ואדום מופיע יחד במצב הסופי. עבור 2 הקוביות התחתיות: לא קיים מצב סופי שבו יש זוג של {ירוק, צהוב} או {ירוק, אדום} או {כחול, צהוב}
 או {כחול, אדום} או {סדר הפוך של הצבעים הצמודים} {צהוב, ירוק} או {אדום, ירוק} או {צהוב, כחול} או {אדום, כחול}. لكن נוסף 1 ליריסטיקה.

ג. שנו את היריסטיקה advanced_heuristic כך שתחשב היריסטיקה של מצב מסוים בבעיה. על היריסטיקה לשפר את היריסטיקה מסעיף ב' (ימדד בזמן ריצה) ולשמור על אופטимальות הפתרון.
 הזוג שהריץ את הטפסים של סעיף ג' בזמן הממוצע הקצר ביותר (ימדד בהרצת הטפסים במודל ואולי בטפסים נוספים רק כדי להזכיר את הבונוס), קיבל **2 נקודות בונוס לציון הסופי**.
 שימושו לב שלל הטפסים צריכים לעבור כדי שהזמן ייחסב. בנוסף, כדי למדוד את פונקציית היריסטיקה בצורה מדויקת יותר, הוספנו בטפסים מימוש של קובץ ה search ושם נקרא לפונקציות היריסטיקה (והגדרת המצביעים) שלכם כדי למדוד את זמן הריצה עם אותו מימוש של מבנה נתונים ולוגיקת חיפוש. גם כאן **2 נקודות בונוס לציון הסופי**.
 זוגות שרוצים להשתתף בבונוס צריכים לרשום את תעוזות הבעיות שלהם והזמן שקיבלו בהרצאה **בקבוץ זהה**.
 (זוגות שלא רוצים להשתתף בבונוס צריכים לרשום רק את תעוזת הבעיות בלבד הזמן שקיבלו)

עבור הבונוסים - במידה ויהי קשה להבדיל בין זמני הריצה (למשל כמה צוותים יקבלו זמן ריצה של 0.0) נריץ עבור הוצאות האלו טפסים קשים יותר (מספר שונה של קוביות וצבעים).
 הטפסים הנוספים לא משפיע על הציון והציוון שאתה רואה במודל הינו הציון לעובדה ולא לבדוק אתכם יותר מכך.

הfonקציות שיש להשלים למימוש כל אחד מאלגוריתמי החיפוש

[heuristics.py](#)

init goal for heuristics(goal blocks)

הfonקציה מקבלת "זוג של המצב הסופי כמחרוזת ולא מחזירה כלום".

שימוש לב: הfonקציה ההזו תקרה לפני כל ריצה חדשה – תווודה לפחות את מבני הנתונים במידה והשתמשם

base_heuristic_color_blocks_state)

הfonקציה מקבלת אובייקט מסוג color_blocks_state ומחזירה את ערך היריסטיקה עבור מגדל הקוביות בקלט. (יש למחוק את השורה שמחזירה 0)

`advanced_heuristic (_color_blocks_state)`

הfonקציה מקבלת אובייקט מסווג `color_blocks_state` ומחזירה את ערך היריסטיקה עבור מגדל הקוביות בקלט. (יש למחוק את השורה שמחזירה 0)

`color_blocks_state.py`

`init_goal_for_search(goal_blocks)`

הfonקציה מקבלת ייצוג של המצב הסופי כמחרוזת ולא מחזירה כלום.

שימוש לב: הfonקציה זו תקרה לפני כל ריצה חדשה – תודאו לאפס את מבני הנתונים במידה והשתמשם

`is_goal_state(_color_blocks_state)`

פונקציה סטטית המקבלת מצב של מגדל קוביות ומחזירה אמת אם המצב הינו מצב סופי, אחרת תחזיר שקר.

`get_neighbors(self)`

הfonקציה לא מקבלת קלט ומחזירה רשימה של סדרות (tuples) כך שבכל סדרה יש את אובייקט מסווג `color_blocks_state` המציג את אחד השכנים ועלות המעבר אליו שקר.

`[(color_blocks_state1, cost1), (color_blocks_state2, cost2)...]`

`get_state_str(self)`

הfonקציה לא מקבלת קלט ומחזירה את ייצוג המחרוזת של מגדל הקוביות (אין צורך לשנות את הfonקציה) - פונקציה זו לא חובה למימוש. תוכלו להיעזר בה כדי לדבג את הקוד.

`search.py`

`create_open_set()`

הfonקציה אינה מקבלת ערכים, ומחזירה מבני נתונים מתאימים לאלגוריתם החיפוש.

`create_closed_set()`

הfonקציה אינה מקבלת ערכים, ומחזירה מבני נתונים מתאימים לאלגוריתם החיפוש.

`add_to_open(vn, open_set)`

הfonקציה מקבלת כקלט את מבני הנתונים `open` ומצב `vn` ומכניסה את `vn` ל-`open`.

`open_not_empty(open_set)`

הfonקציה מקבלת כקלט את מבני הנתונים `open` ומחזירה אמת אם הוא לא ריק, אחרת, מחזירה שקר.

`get_best(open_set)`

הfonקציה מקבלת קבלת את מבני הנתונים ומחזירה את הקודקוד הטוב ביותר, על פי אלגוריתם החיפוש.

`add_to_closed(vn, closed_set)`

הfonקציה מקבלת קבלת את מבני הנתונים `closed` ומצב `vn` ומכניסה את `vn` ל-`closed`.

`duplicate_in_open(vn, open_set)`

הfonקציה מקבלת קבלת את מבני הנתונים `open` ומצב `vn` ומחזירה אמת אם קיימ קודקוד ב `open` עם מצב זהה ל `vn.state` וערך `g` שלו קטן או שווה ל `vn.g`. אם `vn` לא קיים ב `open` או ערך `g` שלו המצביע ב `open` גדול מ `vn.g`, הfonקציה תחזיר שקר.

`duplicate_in_close(vn, closed_set)`

הfonקציה מקבלת קבלת את מבני הנתונים `closed` ומצב `vn` ומחזירה אמת אם קיימ קודקוד ב `closed` עם מצב זהה ל `vn.state` וערך `g` שלו קטן או שווה ל `vn.g`. אם `vn` לא קיים ב `closed` או ערך `g` שלו המצביע ב `closed` גדול מ `vn.g`, הfonקציה תחזיר שקר.

מידע על פונקציות ממומשות

`search_node.py`

`get_neighboors(self)`

הfonקציה לא מקבלת קבלת ומחזירה את כל המצביעים השכנים של המצביע הנוכחי.

`search.py`

`print_path(path)`

הfonקציה מקבלת מערך של קודקודים מסווג `search_node` ומדפיסה את המחרוזת המייצגת את המצביע של מגדל הקוביות.

`search(start_state, heuristic, goal_state)`

הfonקציה מקבלת את המצביע הראשוני, פונקציית היוריסטייקה ומצב סופי ומחזירה מערך של `search_node` המייצגים את המסלול האופטימלי מהמצביע הראשוני למצב הסופי. בתא הראשון במערך יהיה המצביע הראשוני בתא הבא יהיה השכן שלו במסלול שהוחזר וכך להלאה עד המצביע הסופי בתא האחרון.

הגשת המטלה

- יש להגיש דרך מערכת המודל את ארבעת הקבצים הבאים:
 - heuristics.py ○
 - color_blocks_state.py ○
 - search.py ○
 - search_node.py ○

הגשת המטלות תתבצע ישרות מול מערכת המודל בצורה אלקטרוני.

שים לב: ישנה אפשרות להגיש את המטלה התיכונית מספר פעמים ובכל הגשה לקבל חיוי, ככלומר תקבלו באופן מיידי את הציון לביצוע התרגיל.

ניתן לראות את הפידבק להרצה (קומפילציה, מספר טפסטים בעברו, שגיאות זמן ריצה וכו'...). לאחר סיום הרצאה תקבלו התוצאות. ישם מספר טפסטים הבזקדים את הפתרון המוגש למטלה. אם התקבלה שגיאת קומפלייזיה יש עליכם להעלות קובץ חדש בצד לקביל ציון לאחר התיקונים. בקבצים המקוריים שקיבלתם עם המטלה עלולות להיות שגיאות ריצה עברו פונקציות לא ממושכות.

טיפים לעבודה

במידה שהטפסטים בVPL לא עוברים, אין סיבה להסיק שיש בעיה בטפסטים (זה יכול לקרות אך הסבירות נמוכה). יש לבדוק את הקוד שלכם קודם:

- האם הריצתם דוגמאות פשוטות בmain?
- האם וידאתם שהיוריסטייקה שלכם מחזירה את הערך שציפיתם לו?
- האם בדקתם את תקינות היוריסטייקה המתקדמת?
- אם יש לכם בעיות זמן ריצה- האם השתמשתם במبني נתונים ייעילים? למשל- מה הסיבוכיות של בדיקת כפליות בOPEN? האם אפשר לייעל?

בהצלחה!