מבוא לבינה מלאכותית

סמסטר חורף תשפ"ד

מטלה 1

תאריך הגשה: 23:55 30.01.24

הנחיות

- שאלות בנושא מטלה זו יש לשאול דרך המודל, בפורום "מטלה 1".
- הוראות להגשת המטלה מופיעים בסוף מסמך זה.
- הקבצים הנדרשים להרצת הקוד הינם:
  - heuristics.py
  - main.py
  - pancake_state.py
  - search.py
  - search_node.py
- העבודה להגשה בזוגות בלבד אלא אם כן המגישים קיבלו אישור להגשה שאינה בזוגות.
- 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.
- מסמך זה בנוי באופן הבא: תיאור המטלה, בעיית החיפוש, מרחב הבעיה, שאלות המטלה, פונקציות שיש להשלים והסבר על פונקציות קיימות.

תיאור המטלה

שון החליט לפתוח דוכן למכירת פנקייקים. לשם כך, שון קנה זרוע רובוטית שתמיין לו את ערימות הפנקייקים מהפנקייק הארוך ביותר לפנקייק הקצר ביותר. לזרוע יש מרית בעזרתה הופכת ערימות פנקייק ומחזירה אותן לערימה. שון גילה שאלגוריתם המיון של הזרוע מאוד לא יעיל וצורך הרבה חשמל. עזרו לשון לכתוב אלגוריתם **A*** המוצא את הפתרון האופטימלי תוך כדי התחשבות למשקל הפנקייקים. **משקל של כל פנקייק שווה לאורך הפנקייק**.

בעיית חיפוש

בהינתן ערימת פנקייקים. עלינו למצוא את רצף המצבים שבעזרתן הזרוע תוכל למיין את הפנקייקים.

מרחב הבעיה

רשימה בגודל N של אורכי הפנקייקים כך שאורך כל פנקייק p שונה, וגם אורכם תחום:

$1 \leq len(p) \leq N$.

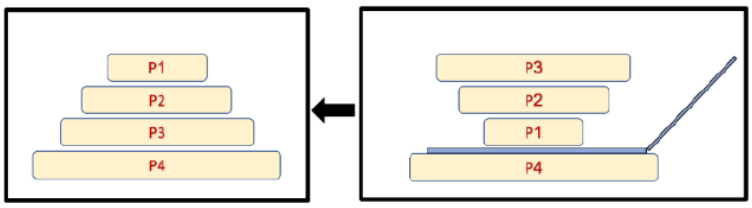
<u>מצב התחלתי</u>: רשימת פנקייקים ההתחלתית.

<u>מצב סופי</u>: רשימת הפנקייקים ממויינת כך שפנקייק באורך 1 נמצא בצד הימני ביותר ופנקייק באורך N נמצא בצד השמאלי ביותר ביותר . לדוגמה, עבור $N = 5$ המצב הסופי יהיה 5,4,3,2,1.

אופרטור:

הפיכת תת ערימה של פנקייקים (ניתן להפוך גם את כל הערימה). עלות הפעולה היא כעלות הפנקייקים שהזרוע מרימה. בדוגמה, עלות הפעולה היא $1 + 2 + 3 = 6$

דוגמא לפעולת Flip:



שאלות המטלה

א. עזרו לשון להשלים את הפונקציות החסרות (מצויינות בהמשך).

ב. שנו את היוריסטיקה base heuristic כך שתחשב את היוריסטיקה הבאה- יוריסטיקה של מצב תהיה סכום הפנקייקים מראש הערימה ועד לפנקייק האחרון שלא נמצא במקום (לפי המצב הסופי).
לדוגמה, עבור הערימה: 7,6,4,5,1,2,3 היוריסטיקה תהיה $4 + 5 + 1 + 2 + 3 = 15$.
פנקייקים 6 ו- 7 נמצאים במקומם לפי המצב הסופי ולכן לא נוספו לחישוב היוריסטיקה.
פנקיק 3 הוא הפנקייק הראשון מלמטה שלא נמצא במקומו. בדוגמה של פעולת הFlip (התמונה בתחילת העמוד) היוריסטיקה תהיה 3+2+1=6 בציור הימני ו-0 בשמאלי.

ג. שנו את היוריסטיקה advanced_heuristic כך שתחשב היוריסטיקה שלמצב מסויים בפנקייק. על היוריסטיקה להיות לשפר את היוריסיטיקה מסעיף ב' (ימדד בזמן ריצה) ולשמור על אופטימליות הפתרון.
הזוג שהריץ את הטסטים של סעיף ג' בזמן הממוצע קצר ביותר (ימדד בהרצת הטסט במודל), יקבל **2 נקודות בונוס לציון הסופי**.
שימו לב שכל הטסטים צריכים לעבור כדי שהזמן יחשב.
זוגות שרוצים להשתתף בבונוס צריכים לרשום את תעודות הזהות שלהם והזמן שקיבלו בהרצה בקובץ הזה.

הפונקציות שיש להשלים למימוש כל אחד מאלגוריתמי החיפוש

heuristics.py

base_heuristic(_pancake_state)

הפונקציה מקבלת אובייקט מסוג pancake_state ומחזירה את ערך היוריסטיקה עבור ערימת הפנקייקים בקלט. (יש למחוק את השורה שמחזירה 0)

advanced_heuristic (_pancake_state)

הפונקציה מקבלת אובייקט מסוג pancake_state ומחזירה את ערך היוריסטיקה עבור ערימת הפנקייקים בקלט. (יש למחוק את השורה שמחזירה 0)

pancake_state.py

get_neighbors(self)

הפונקציה לא מקבלת קלט ומחזירה רשימה של סדורות (tuples) כך שבכל סדורה יש את אובייקט מסוג pancake_state המייצג את אחד השכנים ועלות המעבר אל אותו שכן.

[(pancake_state1, cost1), (pancake _state2, cost2)...]

search.py

create_open_set()

הפונקציה אינה מקבלת ערכים, ומחזירה מבני נתונים מתאים לאלגוריתם החיפוש.

create_closed_set()

הפונקציה אינה מקבלת ערכים, ומחזירה מבני נתונים מתאים לאלגוריתם החיפוש.

add_to_open(vn, open_set)

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

open_not_empty(open_set)

הפונקציה מקבלת כקלת את מבני הנתונים open ומחזירה אמת אם הוא לא ריק, אחרת, מחזירה שקר.

get_best(open_set)

הפונקציה מקבלת כקלט את מבני הנתונים ומחזירה את הקודקוד הטוב ביותר, על פי אלגוריתם החיפוש.

add_to_closed(vn, closed_set)

הפונקציה מקבלת כקלט את מבני הנתונים closed ומצב vn ומכניסה את vn ל-closed .

duplicate_in_open(vn, open_set)

הפונקציה מקבלת כקלט את מבני הנתונים open ומצב vn ומחזירה אמת אם קיים קודקוד ב open עם מצב זהה ל vn.state וערך ה g שלו קטן או שווה ל vn.g. אם vn.state לא קיים ב open או ערך ה g של המצב ב open גדול מ vn.g , הפונקציה תחזיר שקר.

duplicate_in_close(vn, closed_set)

הפונקציה מקבלת כקלט את מבני הנתונים closed ומצב vn ומחזירה אמת אם קיים קודקוד ב closed עם מצב זהה ל vn.state וערך ה g שלו קטן או שווה ל vn.g. אם vn.state לא קיים ב closed או ערך ה g של המצב ב closed גדול מ vn.g , הפונקציה תחזיר שקר.

מידע על פונקציות ממומשות

pancake_state.py

get_state_str(self)

הפונקציה לא מקבלת קלט ומחזירה את יצוג המחרוזת של ערימת הפנקייקים (אין צורך לשנות את הפונקציה)

search_node.py

get_neighbors(self)

הפונקציה לא מקבלת קלט ומחזירה את כל המצבים השכנים של המצב הנוכחי.

search.py

print_path(path)

הפונקציה מקבלת מערך של קודקודים מסוג search_node ומדפיסה את המחרוזת המייצגת את המצב של ערימת הפנקייקים.

search(start_state, heuristic, goal_state)

הפונקציה מקבלת את המצב ההתחלתי, פונקצית היוריסטיקה ומצב סופי ומחזירה מערך של search_node המייצגים את המסלול האופטימלי מהמצב ההתחלתי למצב הסופי. בתא הראשון במערך יהיה המצב ההתחלתי בתא הבא יהיה השכן שלו במסלול שהוחזר וכך הלאה עד המצב הסופי בתא האחרון.

הגשת המטלה

- יש להגיש דרך מערכת המודל את ארבעת הקבצים הבאים:
  - heuristics.py
  - pancake_state.py
  - search.py
  - search_node.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. ישנם מספר טסטים הבודקים את הפתרון המוגש למטלה. אם התקבלה שגיאת קומפליציה יש עליכם להעלות קובץ חדש בכדי לקבל ציון לאחר התיקונים. בקבצים המקוריים שקיבלתם עם המטלה עלולות להיות שגיאות ריצה עבור פונקציות לא ממומשות.

בהצלחה!