ממן 11

מבוא לבינה מלאכותית 20551

סמסטר 2020א

22/11/2019

גיא כרמי 301726154

חלק מעשי

שאלות 1 עד 4

בחרתי לכתוב פונקציית חיפוש משותפת המקבלת כקלט את מבנה הנתונים בו היא תשתמש. בהתחלה כתבתי אותה בצורה פשוטה עבור שאלות 1 ו-2, ואז עבור שאלות 3-4 הוספתי את השימוש במשקל. על מנת לשמור על גנריות כתבתי מחלקת Wrapper המממשת את הממשק של Queue ו- Stack, אך מקבלת priority עבור הפעולה push (ומתעלמת ממנו).

שאלה 5

תחילה השתמשתי במערך לשמירת הפינות אשר ביקרנו בהן, אך נתקלתי במגבלה כיוון שמערך משתנה על פי סדר ההכנסה, ולכן שתי מצבים זהים (מבחינתי) בהם פקמן באותו מיקום ועבר באותן פינות ייחשבו כמצבים שונים. פתרתי זאת על ידי שימוש במבנה הנתונים set אשר מתעלם מסדר ההכנסה (כפי ששתי קבוצות שוות אך ורק על פי האיברים שבהן).
לבדיקת מצב המטרה, אם ישנה פינה שאינה בפינות שנבדקו אזי מוחזר שקר, אם כל הפינות כבר נבדקו מוחזר אמת כמובן.
להחזרת המצבים הממשיכים ממצב מסוים, נבדקים כמובן כל כיווני התנועה האפשריים ואם אין בהם קיר, נבדק גם אם המצב הוא פינה. אם מצב ממשיך הוא פינה אזי מוחזר המצב לאחר הוספת הפינה.

שאלה 6

מימשתי יוריסטיקה בסיסית וחמדנית. באופן כללי הערכות חמדניות אינן קבילות, אך במקרה פרטי זה, מכיוון שמיקום הפינות קבוע וידוע מראש, במידה ולא היו קירות במפה (הנחה מקלה אותה אני מבצע- שימוש במרחקי מנהטן) אז הדרך המהירה ביותר לאסוף את כל הפינות הייתה ללכת בצמוד לגבולות המפה. בכך, מובטח כי היוריסטיקה נותנת הערכה אופטימית ומכך קבילה.

שאלה 7

ברור כי יורסטיקה חמדנית בדומה לשאלה 6 אינה קבילה.

תחילה השתמשתי במרחק מנהטן של האוכל הכי רחוק, ברור כי זוהי יוריסטיקה קבילה כי את האוכל הכי רחוק בכל מקרה צריך יהיה לאסוף, ואיסופו במרחק מנהטן הוא כמובן הזול ביותר. קיבלתי תוצאה אופטימלית של 60, אך רק לאחר פיתוח של 9262 מצבים. לא מספיק טוב ביחס לטבלת הניקוד של השאלה.

השיפור הראשון שחשבתי עליו היה התייחסות גם לאוכל הקרוב ביותר, הרי ברור כי במצב אופטימי ביותר, כל נקודות האוכל נמצאות בין הקרובה ביותר לרחוקה ביותר, אם יש נקודה שאינה בדרך, זה רק יאריך את המסלול. כמו כן אם כולן בדרך בין הקרובה לרחוקה אך נתחיל מנקודה אחרת (ולא הקרובה ביותר) זה רק יאריך את המסלול. לכן יוריסטיקה המחזירה את מרחק מנהטן של מסלול שמתחיל מנקודת האוכל הקרובה ביותר, וממנה עובר לנקודת האוכל הרחוקה ביותר (וכך במצב האופטימי המתואר, אוכל בדרך את כל שאר המאכלים) היא קבילה. הפעם פיתחתי רק 8527 מצבים. עדיין לא מספיק טוב ביחס לטבלת הניקוד.

לאחר כמה ניסיונות נוספים המבוססים על בחירת המאכלים וחישוב מרחקי מנהטן ביניהם (למשל חלוקת המסך ל- 4 כאשר פקמן באמצע החלוקה, כלומר צפון-מערב לפקמן, צפון-מזרח לפקמן וכו', וחיפוש האוכל הרחוק ביותר בכל רבע, ואז ביצוע חישובים על המאכלים שנמצאו) אשר רובם לא היו קבילים וכולם החזירו תוצאות של מעל 8000, הבנתי שהשיפור יבוא משימוש בהערכת מרחק טובה יותר ולא בבחירת מאכלים טובה יותר, ואז נתקלתי בפונקציית העזר mazeDistance אשר משתמשת באלגוריתמי החיפוש שיצרנו ומציינת כי "This might be a useful helper function".

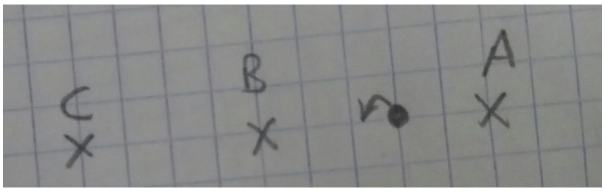
בשלב זה נתקלתי בבעיה נוספת, הריצה לא הסתיימה. לאחר קצת debugging וחפירה מה הבעיה, הבנתי שאני מבצע את אותן פעולות שוב ושוב, מקרה קלאסי לשימוש בתכנון דינמי, וכאן הרמז מתיאור היוריסטיקה בא לעזרתי: "If you want to *store* information to be reused in other calls to the heuristic, there is a dictionary called problem.heuristicInfo that you can use."

הפתרון הזה משלב את השימוש בפונקציית המרחק מבוססת BFS והשילוב של המרחקים לאוכל הקרוב ביותר וממנו לאוכל הרחוק ביותר, ובשלב זה קיבלתי פתרון אופטימלי לאחר 1889 צמתים בלבד!!

זוהי תוצאה מפתיעה מאוד ביחס לטבלת הניקוד, חשבתי שאצליח לרדת מ- 7000 רק במעט, לכן השקעתי רבות בניסיון להוכיח שהיוריסטיקה קבילה ועקבית, והבנתי שלמרות שהיא קבילה (כפי שהראתי קודם), היא אינה עקבית! (ישנו מצב בו צעד אחד יקטין את היוריסטיקה ביותר

מאחד כמתואר בשרטוט המצורף*) ולכן חזרתי ליוריסטיקה הראשונה (המרחק לאוכל הרחוק ביותר) אשר ברור כי היא עקבית (המרחק לאוכל הרחוק ביותר יכול לקטון לכל היותר באחד על כל צעד) אך עם בפונקציית המרחקים המדויקת יותר (ותכנון דינמי) וכך הגעתי לפתרון אותו אני מגיש, אשר מחזיר תוצאה אופטימלית לאחר פיתוח 4049 צמתים, עדיין תוצאה מצוינת והפעם גם עקבית.

*שרטוט המפריך עקביות של הפתרון שהצעתי:



ביצוע צעד אחד כמתואר על ידי החץ, מקטין את היוריסטיקה מ- 11 (המרחק ל- A + המרחק בין A ל- C) ל- 6 (המרחק ל- B שעכשיו קרוב ביותר + המרחק בין B ל- C שעדיין רחוק ביותר).

שאלה 8

כפי שנרמז בשאלה השלמתי את מבחן המטרה והחזרתי חיפוש BFS על הבעיה הנתונה.
פתרון פשוט זה פותר את המבוך bigSearch בפחות משנייה ובמחיר מסלול של 350, כפי שמצוין בשאלה.

חלק עיוני

שאלה 1

כידוע DFS אינו מחזיר פתרון אופטימלי, אלה את הפתרון הראשון שהוא נתקל בו בדרך שבה בחר ללכת. DFS כשמו כן הוא מחפש קודם לעומק, לכן ייתכן שלמרות שיש מסלול מקביל קצר מאוד, האלגוריתם ימצא מסלול ארוך מאוד אותו בחר לבדוק קודם.

מכאן, שזהו לא בהכרח (אך ייתכן שכן) פתרון זול ביותר.

שאלה 2

אלגוריתם חיפוש לרוחב אכן מוצא פתרון זול ביותר (כאשר פונקציית המחיר לא יורדת, כמו במקרה הזה כמובן), כיוון שקודם בודק את כל המסלולים הזולים ביותר, ואז הזולים קצת פחות.

כאשר הוא יתקל בפתרון, בהכרח לא היה פתרון זול יותר.

שאלה 4

הרצות על openMaze

אלגוריתם DFS מפתח מספר נמוך יחסית של מצבים, 315, אך מוצא מסלול לא אופטימלי כלל באורך 158.

אלגוריתמי BFS ו- UCS, אשר מתנהגים באופן זהה במקרה זה (כיוון שפונקציית המשקל זהה לכן אין יתרון לשימוש ב- UCS) מפתחים כמעט את כל המצבים האפשריים, 680 במספר, אך לבסוף מוצאים פתרון אופטימלי באורך 54.

אלגוריתם ASTAR עם שימוש ביוריסטיקת מנהטן הנתונה, מפתח 145 מצבים בלבד למציאת המסלול האופטימלי באורך 54. זהו מסלול ללא מכשולים רבים ולכן מרחקי המנהטן קרובים יחסית ל"מציאות" ונותנים הערכה מצוינת.