שם: עמית סידס. תעודת זהות: 206768780. קורס: בינה מלכותית 20551

השתמשתי ב- python 2.7.16.

שאלה 1

א. האם סדר הסריקה תואם לסדר שהייתם מצפים ?

כן. ניתן לראות שבכל פיצול במבוך, השחקן בוחר לנוע כל פעם בכיוון מסוים ע"פ ההעדפה קבועה (שמאלה > ימינה > למטה > למעלה). כלומר, לדוגמא אם השחקן מגיע לפיצול לימין ולשמאל (והוא הגיע מלמעלה או למטה) אז הוא יבחר קודם ללכת שמאלה, ורק אחרי שהוא בדק את כל אפשרויות המסלולים בכיוון הזה, הוא "ינסה" ללכת ימינה. בדיוק כפי שהיינו מצפים מ-DFS.

ב. האם זה פתרון זול ביותר? נמקו.

לרוב לא. הפתרון הראשון שנמצא הוא זה שיבחר, למרות שיכול להיות שבהסתעפות כלשהיא בהתחלה, במידה והייתה נבחרת פנייה אחרת היינו יכולים לקבל פתרון יותר טוב. דוגמא טובה לכך היא המסלול ב-mediumSearch שבו השחקן בוחר במסלול שהולך לכיוון שמאל (כי הוא מתעדף שמאל על פני כל כיוון אחר) והפתרון שהוא מוצא הוא ממש לא אופטימאלי.

שאלה 2

ג. האם אלגוריתם חיפוש לרוחב מוצא פתרון זול ביותר? נמקו .

כן. BFS מבטיח לנו את מציאת המסלול הכי אופטימלי, משום שהוא סורק באופן מקבילי את כל המסלולים האפשריים (סריקה לרוחב העץ) ולכל המסלולים באותה רמה יש אותו cost. לכן, המסלול הראשון שימצא יהיה המסלול ברמה הנמוכה ביותר שמגיע ליעד. מכיוון שה-cost רק עולה ככל שהרמה עולה, המסלול הזה יניב את הפתרון האופטימלי.

שאלה 4

ד. מה קורה ב-openMaze לאסטרטגיות החיפוש השונות?

1. DFS

כיוון ש-DFS מתעדף פניות לצדדים על פני פניות מטה, הוא מוצא מסלול ממש גרוע עבור openMaze שבו כמעט ואין קירות. המסלול שלו גורם לו ללכת מקצה אחד לקצה השני של המבוך ורק אז להחליט לרדת שורה לכיוון היעד.

2. BFS

כמובן שהמסלול שנמצא הוא המסלול האופטימלי, אך מכיוון שהיעד נמצא בנק' הכי רחוקה מהמיקום ההתחלתי של השחקן, אז כל מיקום במבוך נסרק במהלך החיפוש (ניתן לראות שכל משבצת נצבעת במפה). מה שיוצר הרבה מאוד צמתים.

3. UCS

באופן לא מפתיע, התוצאות המוחזרות הן זהות לאלו של BFS, וזאת משום שה-cost לכל צומת בגרף (צעד במסלול) הוא 1. ועל כן גם המסלול זהה לזה של BFS, וכך גם החיפוש שסורק את כל המבוך.

4. A*

כאשר משתמשים ב-a* עם פונקציית היוריסטיקה nullHeuristic, החיפוש זהה לזה של BFS, כיוון שאין העדפה לשום צומת. אבל, כאשר משתמש בפונקצייה manhattanHeuristic, ניתן לראות שהמסלול הנמצא הוא אופטימלי (זהה לזה של BFS) אך בכשליש מהצמתים שנדרשו לחיפוש ב-BFS. ברגע ש-a* הצליח למצוא את המסלול שמתגבר על אחד המכשולים בדרך ליעד, הוא ישר חיפש האם הוא יכול להגיע ליעד ממנו, מבלי לסרוק עוד מסלולים מיותרים.

שאלה 5

בחרתי לייצג מצב בתור נקודה שמייצגת את המיקום הנוכחי ורשימה של הפינות שעדיין לא ביקרנו בהן. בכל פעם שנבקר בפינה מסוימת, נסיר אותה מהרשימה. אם הרשימה ריקה, הרי שלא נותרו עוד פינות והגענו ליעד.

שאלה 6

ה. **האם היוריסטיקה קבילה?** כן.
הפונקציה *smallest_distance_of_4_points* מקבלת את המיקום הנוכחי ו-iterable המכיל 4 (או פחות) נקודות. היא מחזירה את אורך מסלול מנהטן הטוב ביותר (המסלול הקצר ביותר ללא מכשולים) אשר עובר בכל ארבעת הנקודות ומתחיל במיקום הנוכחי. הפונקציה הזו היא הדבר הכי אופטימי שקיים, היא מניחה שהיא תצליח ללכת לכל מקום אשר תבחר מבלי להתייחס לקירות. לפיכך, האורך שהיא תחזיר יהיה בהכרח קטן/שווה לאורך המסלול האופטימלי (שעלול להתארך במידה ויש קירות) ולכן הפונקציה הזו היא קבילה.

שאלה 7

ו. **האם היוריסטיקה קבילה?** כן.
הפונקציה *hardest_points_to_reach* מקבלת את המיקום הנוכחי, מערך נקודות ומספר 'amount' שמייצגת כמות מספרית של נקודות בין 0 ל-4. הפונקציה מחפשת את קומבינציית הנקודות (כלומר, אם amount הוא 4, היא מחפשת קומבינצייה של 4 נקודות) שהכי קשה לעבור בכולן ומחזירה את מסלול מנהטן האופטימלי שעובר בכולן מהמיקום הנוכחי (ללא הפרעה של קירות). בהנתן שיש יותר מ-amount נקודות במערך הנקודות, בהכרח האורך הזה הוא קטן מהמסלול האופטימלי, כיוון שהמסלול האופטימלי יצטרך גם הוא לעבור באותן נקודות ואפילו בעוד נקודות, ולכן ככל הנראה המסלול האופטימלי יהיה ארוך יותר (ישנם מקרים שבהם הוא יהיה זהה, זה קורה כאשר כל הנקודות הן על אותו הקו לדוגמא. אבל תמיד הפונקציה תחזיר מספר שהוא קטן/שווה לאורך המסלול האופטימלי).

ז. **האם היוריסטיקה עקבית?** כן.
נתחיל בכך שה-cost של כל פעולה יחידה (כלומר, צעד במבוך) הוא 1 תמיד. לכן, צריך בעצם להראות כי פונקציית היוריסטיקה אינה קטנה ביותר מ-1 במעבר בין משבצות סמוכות.
בהנחה שרשימת נקודות האוכל לא משתנית, כיוון שהיוריסטיקה מתבסס על חישוב מרחק מנהטן, אז בהנתן שזזנו כיוון אחד לכיוון המסלול שחושב, אז האורך שלו נהפך לקטן ב-1, ולכן גם ערך החזרה של פונקציית היוריסטיקה. אם זזנו לכיוון אחר למסלול, האורך יכול או לגדול ב-1 (אם זזנו לכיוון מנוגד) או להשאר זהה (אם יש מסלול סימטרי מהנקודה החדשה).
אם רשימת הנקודות כן משתנית, אז זה יכול לקרות מ-2 סיבות:

1. במקרה עברנו באחת מנקודות האוכל שאינה קשורה לקומבינציה שאליה מחשבים את המסלול. במקרה הזה היוריסטיקה לא תתחשב בכך שאכלנו אותה והערך שלה לא יושפע.
2. הגענו לאחת מהנקודות שאליה מחשבים את המסלול, ולכן היוריסטיקה חיפשה נקודה חדשה שתתווסף לקומבינציית הנקודות. כעת היוריסטיקה תחפש קומבינצייה חדשה של amount נקודות שהכי קשה להגיע לכולן. הערך שיוחזר עבור הקומבינצייה החדשה בהכרח יהיה גדול/שווה לקומבינצייה הקודמת כיוון שבקומבינצייה הקודמת השחקן היה במרחק של 1 מאחת הנקודות, ולכן הנקודה החדשה שתתווסף לקומבינצייה כנראה רק תגדיל את המרחק להגעה לכולן, ובוודאות תשאיר אותו גדול/שווה למרחק הקודם (אם המרחק הזה היה קטן יותר אז לא היינו בוחרים את הקומבינצייה הקודמת מלכתחילה).

כדאי לציין שהמעבר מ-amount=3 ל-amount=4 גם כן לא פוגע בזה, כיוון שהוא גורם לפונקצייה רק לגדול. הרי שלהגיע לשלוש הנקודות שהכי קשה להגיע אליהן זה יותר קל מאשר להגיע לארבעת הנקודות שהכי קשה להגיע אליהן.