מבוא לבינה מלאכותית

תרגיל 1

דנה שבתאי 314822214
רואי גרייף 315111401

שאלה 1

1) בוצע

2) $S = \{0, 1, ..., 63\}$

$$O = \{LEFT, RIGHT, UP, DOWN\}$$

$$I = \{(S, 0, 0)\}$$

$$G = \{(G, 7, 7)\}$$

3) 64 = |S| כי יש 64 מצבים להיות עליהם בלוח - כל משבצת היא מצב.

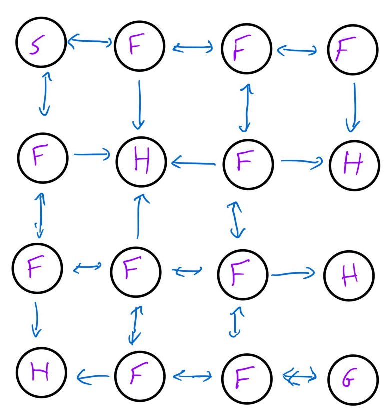
4) הפונקציה domain על אופרטור down תחזיר לנו את כל המצבים **מלבד למצבים בשורה האחרונה ולמצבים שהם חור**. מכיוון שאין אפשרות לרדת למטה בלוח בשורה האחרונה, ואין אפשרות להמשיך לכל כיוון אחרי שמגיעים לחור.

5) מכיוון שהוגדר לנו שהמצב התחלתי הוא למעלה מצד שמאל אז הפונקציה SUCC תחזיר את המצבים שנמצאים מימין ומלמטה למצב ההתחלתי, ואת המצב ההתחלתי עצמו. כלומר יוחזרו המצבים 0 (עבור הפעולות שמאלה ולמעלה) , 1 (עבור פעולת ימינה) ו-8 (עבור פעולת למטה).

6) כן. לדוגמה מהצמה ההתחלתי לעשות פעולת DOWN ואז פעולת UP.

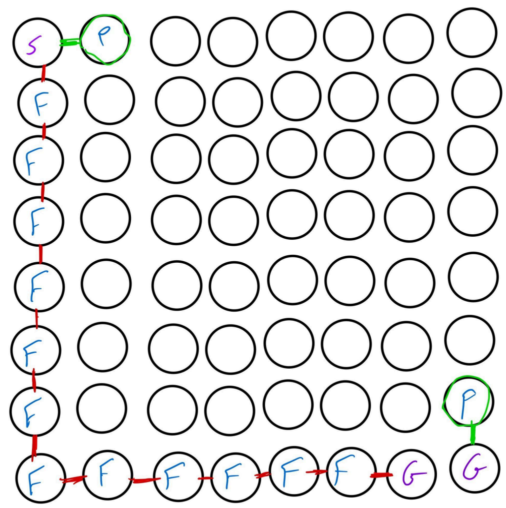
7) מכיוון שיש 4 פעולות כל מצב יכול לתת לנו לכל היותר 4 מצבים חדשים ולכן מקדם הסיעוף הינו 4.

8)



<u>הערה:</u> כל מצב חוזר לעצמו אם לא ניתן לבצע עליו את אחת הפעולות (לא כתוב בגרף כי לא רציתי שיהיה מבולגן).

9) במקרה הגרוע ביותר הוא לא יגיע ליעד לדוגמא אם ילך ימינה ואז שמאלה ויחזור על כך (בהנחה שאין portals במשבצות האלו). במקרה הטוב ביותר הוא יגיע תוך 2 צעדים, נניח עבור דוגמא בו יש portal מימין למצב ההתחלה והportal השני נמצא ליד הgoal.

10) 

ניתן לראות בדוגמא הנגדית כי המסלול האדום הוא הקצר ביותר למטרה במונחים של מרחק מנהטן והמחיר שלו זה 122 ולעומת זאת המחיר של המסלול הירוק הוא 102 למרות שהוא יותר רחוק לפי מרחק מנהטן.

שאלה 2

1) בהנחה שקיים פתרון (כלומר אנחנו לא חוסמים את המעבר על ידי חורים) האלגוריתם BFS יהיה שלם כי בסופו של דבר נעבור על כל האופציות עד נגיע לפתרון כלשהו. הפתרון לא קביל לדוגמא אם הצומת מטרה שלנו נמצאת 5 משבצות מתחת להתחלה בBFS נלך לכל השכנים הקרובים במקום ללכת ישר למטרה ולכן זה לא פתרון אופטימלי ולכן לא קביל.

2) ההבדל הוא שבBFS על גרף אנחנו שומרים close וopen ולכן לא נעבור באותו צומת פעמיים לכן על ידי הוספת close וopen לעץ שהוא יהיה זהה לBFS על גרף.

   לפי מה שלמדנו בשיעור, התנאי בגרף להכנסה ל -OPEN הוא:

   child.state is not in CLOSE and child is not in OPEN

   במידה ויהיו שני מסלולים לאותו צומת, תנאי זה לא בהכרח יתקיים ונרצה שיתקיים תמיד. לכן התנאי שנבחר הוא שלכל צומת בגרף קיים לכל היותר מסלול אחד יחיד אליו מצומת ההתחלה. בנוסף, נדרוש שלצומת ההתחלה אין אב כדי שתהיה התאמה בין הגרף לעץ.

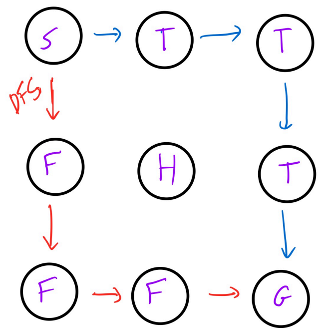
3) על מנת לקבל פתרון אופטימלי עם אלגוריתם BFS נגדיר T פונקציה שלוקחת את הגרף G ובין כל שתי צמתים מכניסה דרכם $w(e)$ צמתים כאשר e זאת צלע כלשהי וw זאת הפונקציה המחזירה את המחיר של הצלע. בין כל הצמתים החדשים שהכנסנו משקל של כל 2 צמתים עוקבים יהיה 1 בדיוק ולכן סה"כ כשנסכום את הצלעות שהוספנו נקבל בדיוק $w(e)$. וכעת מכיוון שBFS עושה חיפוש לרוחב אנחנו בוודאות נגיע לG עם המסלול הקצר ביותר.

4) מכיוון שאנחנו עובדים עם BFS וזהו חיפוש לרוחב והצומת מטרה נמצאת הכי רחוק מהצומת התחלה נקבל כי יווצרו כל הצמתים בגרף ולכן סה"כ יווצרו n^2 צמתים. לעומת זאת, יפותחו רק $n^2 - 2$ כי לא נפתח את הצומת מטרה עצמו וגם את הצומת שמעל המטרה (זה נובע מכך שהצעד הראשון שלנו על המפה הוא למטה במקום ימינה.

שאלה 3

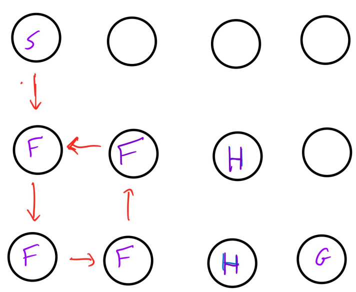
1) בוצע

2) האלגוריתם שלם, מכיוון שאנחנו בסופו של דבר נעבור על כל המצבים האפשריים ונגיע לgoal אם זאת האלגוריתם אינו קביל. דוגמא נגדית:



ניתן לראות כי קיים מסלול שהוא יותר זול (המסלול הכחול).

3) לא כי בDFS (על עץ) אנחנו לא בודקים בclose אם היינו כבר בצומת כלשהי ולכן עלולים להיכנס למעגל, לדוגמא:



4) 1) נלך למטה וכאשר לא יהיה אפשר לרדת יותר נלך ימינה עד שנגיע לצומת מטרה כלומר סה"כ 2N-2 יפותחו כי לא נספור שוב את צומת ההתחלה והמטרה. ובכל צומת בו נעבור יווצרו כל הצמתים מסביב לאותו צומת (אם ניתן) ולכן סה"כ יווצרו 4N-5 צמתים.

2) עם backtracking האלגוריתם יצור את הצמתים באופן עצל ולכן סה"כ יווצרו 2N-1 צמתים ויפותחו 2N-2 כי לא נפתח את צומת המטרה.

5) 1) נשנה את מרחב החיפוש על ידי בחירת פעולות חדשות במקום $O = \{\leftarrow, \rightarrow, \uparrow, \downarrow\}$ נגדיר את $O' = \{\leftarrow, \rightarrow, \uparrow, \downarrow, \leftarrow\uparrow, \leftarrow\downarrow, \leftarrow\leftarrow, \rightarrow\uparrow, \rightarrow\downarrow, \rightarrow\rightarrow, \uparrow\leftarrow, \uparrow\rightarrow, \uparrow\uparrow, \downarrow\leftarrow, \downarrow\rightarrow, \downarrow\downarrow\}$ כלומר כעת יהיו לנו 16 פעולות אפשריות. דבר זה יאפשר לנו ללכת שתי משבצות בפעולה אחת ובכך נצליח להגיע ב$\frac{d}{2}$ צעדים.

2) מקדם הסיעוף החדש הינו 4b. כי $b = 4$ ומתקיים כי $|O'| = 16$.

3) סיבוכיות של DFS-L היא $time = O(b^d),\ space = O(bd)$ ומכיוון שאצלנו העומק הוא $d' = \frac{d}{2}$

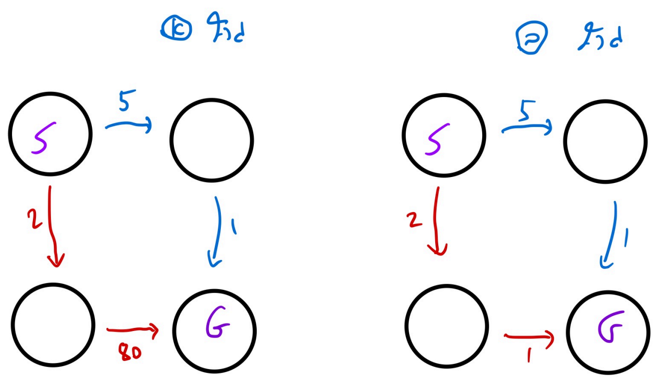
וגם $b' = 4b$ אזי $time = O(4b^{\frac{d}{2}}),\ space = O(2bd)$ כלומר סה"כ נקבל:

$time = O(b^{\frac{d}{2}}),\ space = O(bd)$

4) דוגמא בה מרחב החיפוש החדש יותר טוב היא אם אנחנו בלוח N על N בלי חורים עם אותם ערכים בכל המשבצות וgoal אחד בפינה הימנית למטה. מכיוון שנלך למטה עד תחתית הלוח ואז ימינה עד המטרה. במרחב החיפוש החדש אנחנו נבצע בכל פעולה 2 צעדים קדימה ואילו במרחב הקודם היינו מבצעים צעד אחד בכל פעולה. ולכן יפותחו חצי מהצמתים במרחב החיפוש החדש. דוגמא בה המרחב המקורי עדיף היא אם נפתח יותר מדי צמתים וזה יכול לקרות כי כעת נפתח פי 4 צמתים (מקדם הסעיוף שלנו הוא 16 במקום 4).

שאלה 4

1) בוצע
2) האלגוריתם שלם כי אנחנו נעבור על כל האופציות ולכן בהכרח נקבל פתרון. האלגוריתם קביל כי אנחנו כל פעם נמשיך במסלול שלקח לנו הכי פחות להגיע אליו ולכן בהכרח נקבל את המסלול הקצר ביותר.
3) האלגוריתמים יפעלו בדרך זהה במידה והמשקלים של הקשתות בכל עומק יהיו יותר גדולים מהמשקלים של העומק שקדם לו. כך נגרום לזה שUCS יעשה חיפוש לרוחב כי בכל עומק שהוא יגיע הוא לא יוכל להתקדם עד לאחר שעבר על כל הקשתות בעומק n, כי בעומק n+1 הקשתות יהיו בהכרח יותר גדולות.
4) עבור גרף א האלגוריתם של איימי לא יחזיר את המסלול הקצר ביותר. במקום להחזיר 6 הוא יחזיר 82. ואילו בגרף ב האלגוריתם של איימי יחזיר 3 שזהו המסלול המינימלי.



שאלה 5

תהיינה שתי יוריסטיקות קבילות h1,h2:

1) 1) לא נכון, הדוגמא הנגדית היא היוריסטיקה האוקלידית עבור מסלול ריבועי כאשר כל צלע בעלות 1 ואנחנו רוצים להגיע מקודקוד 1 לקודקוד ההופכי לו לכן $h^* = 2$ ו$h_1 = h_2 = h_{ecl} = \sqrt{2}$ ולכן h_1, h_2 הינם קבילות אבל הסכום שלהם לא קביל כי: $h = h_1 + h_2 = 2 * \sqrt{2} > 2 = h^*$ ולכן זה סתירה להגדרת הקבילות.

2) נכון, כי $h_1 \le h^*,\ h_2 \le h^*$ ולכן הסכום שלהם יהיה $h_1 + h_2 \le 2h^*$ ואז נחלק ב2 ונקבל $0 \le h = \frac{h_1+h_2}{2} \le h^*$ ולכן h קבילה.

2) 1) לא נכון, כי אנחנו יודעים שעקביות גורר קבילות ולכן לא קבילה גורר לא עקבית והראנו בסעיף הקודם כי $h_1 + h_2$ לא קבילה ולכן לא עקבית.

2) נכון, כי אם $h_1(s) - h_1(s') \le cost(s, s')$ וגם $h_2(s) - h_2(s') \le cost(s, s')$ אזי אם נחבר נקבל:

$$h_1(s) - h_1(s') + h_2(s) - h_2(s') \le 2cost(s, s')$$

$$h_1(s) + h_2(s) - (h_1(s') + h_2(s')) \le 2cost(s, s')$$

$$\frac{h_1(s) + h_2(s)}{2} - \frac{h_1(s') + h_2(s')}{2} \le cost(s, s')$$

$$h(s) - h(s') \le cost(s, s')$$

ולכן h עקבית לפי הגדרה.

4) כן, נסתכל על המקרה הראשון בו לא עוברים בportals לכן העלות הכי נמוכה שיש בלוח היא 1 וכדי להגיע מS לG צריך לפחות $|G_{row} - S_{row}| + |G_{col} - S_{col}|$ ולכן $|G_{row} - S_{row}| + |G_{col} - S_{col}| \le h^*$ נשים לב שזו יוריסטיקת מנהטן כלומר מתקיים $hmanhatan(s, g) \le h^*$ (עבור g הכי קרוב לs) ולכן:

$$min\{hmanhatan(s, g) | gG\} = |G_{row} - S_{row}| + |G_{col} - S_{col}| \le h^*$$

אם כן עוברים בportal במהלך המסלול אז במידה ונבחר את המינימום בין מרחק מנהטן למחיר הportal ולכן נבחר את המחיר של הportal במידה והportal תגרום לנו לעבור מעל 100ל צעדים ונקבל כי $hmanhatan(s,g) \le h^*$ כי הראנו שבבעית החיפוש שלנו מתקיים: $100 \le hmanhatan(s,g) \le h^*$.

5) כן, מכיוון ש: $h_{campus} = min\{min\{h_{manhattan}(s,g)|g \in G\},\ 100\}$ ולכן לפי הגדרה נחלק ל2 מקרים:

מקרה 1: צעד רגיל:

$1 = h_{campus}(s) - h_{campus}(s') \le cost(s,\ s')$

ולפי הגדרת המפה אנחנו יודעים כי $1 \le cost(s,s')$ ולכן עקבית.

מקרה 2: כאשר הצעד הוא בתוך portal נקבל:

$h_{campus}(s) - h_{campus}(s') \le cost(s,\ s') = 100$

כאשר $h_{campus}(s) - h_{campus}(s') \le 100$ ולכן אי השוויון מתקיים וזה גורר כי סה"כ נקבל ש h_{campus} עקבית.

שאלה 6

1) האלגוריתם שלם כי אנחנו נפתח בסוף את כל המצבים עד שנגיע לפתרון (פתרון מובטח לנו עבור בעיית הניווט בקמפוס). האלגוריתם אינו קביל כיוון שראינו בשיעור כי יוריסטיקה יכולה להטעות אותנו.
2) **יתרון** לbeam search הוא שאנחנו חוסכים בזיכרון וזמן ריצה בכך שאנחנו מוגבלים לK צמתים שמותר לשמור בopen, כלומר יש אפשרות שנפספס מסלולים מסוימים כי נוציא אותם מהopen ולכן לא נצטרך לפתח אותם ואת המסלולים שלהם ולכן נחסוך בזיכרון וזמן ריצה.

חיסרון לbeam search הוא שאנחנו עלולים לפגוע בטיב הפתרון כי אולי הפתרון הטוב יותר היה דווקא מתגלה בצמתים שמחקנו מהopen. בGBFS אנחנו לא מוגבלים לK צמתים בopen ולכן לא נפספס מסלולים.

שאלה 7

2) מה שעשינו הוא להכפיל את f המקורית בקבוע שהוא חצי כלומר לכל צומת הערך f שלה הוכפל בחצי ולכן מכיוון שאנחנו משווין ביניהם כדי לקבל מינימום כל פעם אנחנו לא נפגע בהשוואה. במילים אחרת, $f_1 < f_2$ אזי מתקיים גם $0.5f_1 < 0.5f_2$ ולכן $f_1' < f_2'$ ולכן נקבל אותו מסלול וזה גורר כי גם העלות תהיה שווה.

4) <u>יתרון וחסרון של האלג' A-ID* ביחס ל-A*:</u>

יתרון: הקטנת צריכת זיכרון, כלומר במידה ואנחנו יודעים שיש הרבה מצבים בהם נצטרך לעבור.

חסרון: אנחנו עלולים לפתח מצבים שכבר היינו בהם בעבר. לכן נרצה להשתמש בA* כאשר אין לנו הגבלה על הזיכרון ואילו בA*-ID כאשר כן יש לנו הגבלה על הזיכרון.

5) <u>יתרון וחיסרון של האלג' A*-epsilon ביחס ל-A*:</u>

יתרון: נותן לנו יותר גמישות בלפתוח את הצומת הבאה כי יש אפשרות שתהיה לנו יוריסטיקה לא קבילה שקרובה ליוריסטיקה האופטימלית כלומר h*, כלומר יהיה לנו מבחר יותר גדול.

חסרון: אנחנו מוותרים על האופטימליות של A* לדוגמא: אם המחיר האופטימלי הוא 200 אז עבור אפסילון כלשהו אנחנו יכולים לקבל מחיר שהוא בטווח של 150-250, ולכן אנחנו עלולים לקבל פתרון שהוא 250 וזה פחות טוב.

שאלה 8

map	DFS-G cos	DFS-G num	UCS cost	UCS num	A* cost	A* num of	W-A* (0.3)	W-A* (0.3)	W-A* (0.7)	W-A* (0.7)	W-A* (0.9)	W-A* (0.9)	num of expanded nodes
map12x12	121	33	87	97	87	92	87	94	87	89	89	26	
map15x15	181	47	106	167	106	167	106	167	106	150	129	44	
map20x20	282	57	175	309	175	308	175	308	175	298	202	56	

התוצאות אכן תואמות לציפיות שלנו לפי מה שלמדנו בכיתה. לדוגמא האלגוריתם של DFS הוא הכי מהיר כי הוא מפתח הכי קצת צמתים, הוא עושה חיפוש לעומק בניגוד לשאר האלגוריתמים שעוברים בין מלא מסלולים כדי לנסות לקבל עלות נמוכה ככל שאפשר דבר אשר גורם לכך שהם מפתחים יותר צמתים וזה גורם לכך שהחיפוש יהיה ארוך יותר.

ניתן גם לראות שככל שh_weight מתקרב ל0 אנחנו מקבלים תוצאות שדומות יותר לUCS וזה גם תואם למה שראינו בכיתה.

וכאשר הh_weight קרוב יותר ל1 אנחנו מקבלים אלגוריתם שרוצה להגיע למטרה כמה שיותר מהר ובאמת ניתן לראות שהוא מפתח הכי פחות צמתים ולכן הוא מגיע מהר למטרה אבל במחיר גבוה יחסית.

ניתן גם לראות שA* נותן תוצאות שהם יחסית באמצע בין UCS לGBFS כפי שראינו בכיתה.

שאלה 9

1) S={כל הקבצים עם n מילים כך שכל מילה מופיעה בדיוק פעם אחת} זה שקול לכך שיש פונקציה חח"ע ועל שממפה את המילה הi למקום הji. כאשר i היא אחת המילים מתוך n וji הוא המיקום בו נמצאת המילה i.

2) $|S| = |\{\pi|\pi: n \to n \ \ is\ a\ bijection\ function\}| = n!$ כי S שקולה לקבוצת הפונקציות שממפות כל מילה שונה מקום אחד. וראינו בקורס אחר כי יש בדיוק n! פונקציות כאלה.

3) כן, תהא w1 מילה שאינה במקום הנכון, אזי בהכרח גם המילה שנמצאת במקומה של המילה w1 נמצאת במקום הלא נכון נסמנה w2. ולכן אם נחליף בין w1 וw2 אזי w1 תהי במקום הנכון וw2 אולי כן ואולי לא תהיה במקום הנכון אבל בכל מקרה שיפרנו את המצב ולכן תמיד יהיה מצב שאליו יהיה עדיף לעבור אליו.

4) 1) כן, לפי סעיף 3 הוכחנו כי לא נגיע למצב שבו אין לנו אפשרויות להתקדם. ולכן מכיוון שזהו אותו אלגוריתם מלבד לsideways steps (שגם ככה לא נשתמש בהם) ניתן להסיק כי נקבל אותו תשובה ובהכרח נמצא פתרון.

2) זה יהיה אותו אלגוריתם כי אנחנו לא "נתקע". בנוסף, כל פעם ניתן לתקן מילה אחת או 2 מילים לכל היותר ולכן בשתי האלגוריתמים הם יבחרו תמיד לתקן 2 מילים אם ניתן ואם לא, אז מילה אחת (תמיד יש) עד שנגיע למטרה. ההבדל היחיד שיכול להיות זה שאולי האלגוריתם with sideways steps יעשה פעולה שונה מהאלגוריתם השני מה שיגרום לכך שהקובץ ישתנה "לטובתו" כך שיהיה ניתן בשלב הבא לתקן 2 מילים ולאלגוריתם השני יהיה ניתן לשפר רק במילה אחת.

5) כן, מכיוון שאנחנו תמיד נשפר בכל צעד את הפתרון אז לכל היותר תוך n צעדים נגיע למטרה (הוכחנו בסעיף 3 שתמיד יש מצב שמשפר).