שאלה 1:

סעיף 2) תחילה נרצה להגדיר את מרחב החיפוש כפי שנלמד בתרגול. הגדר את $(S, 0, I, G)$ עבור סביבת האגם הקפוא. כאשר S זה מרחב המצבים, O, זה מרחב האופרטורים, I, זה המצב ההתחלתי וG הוא קבוצת מצבי המטרה. מה גודל מרחב המצבים S? הסבירו.

עבור לוח בגודל NxN נתן לכל מקום בלוח מספר בהתאם לנוסחה הבאה: $(i, j) \equiv N * i + j$

S - נגדיר את מרחב הצמתים להיות אוסף כל המספרים מ-0 עד (N-1)(N+1) המסמנים כל התאים מ-(0,0) עד (N-1,N-1) בלוח, כלומר עבור לוח 8x8 הנתון: $S = \{ s \mid s \in \mathbb{N} : 0 \leq s \leq 63\}$

O – קבוצת האופרטורים $O = \{DOWN, RIGHT, UP, LEFT\}$

I – קבוצת המצבים ההתחלתיים בלוח הנתון הינה $I = \{0\}$

G – קבוצת מצבי המטרה בלוח הנתון הינה $G = \{63\}$

סעיף 3) מה תחזיר לנו הפונקציה Domain על אופרטור 2 (UP)?

$$Domain(UP) = \{ s \in S \mid s \; is \; not \; a \; hole\}$$

מכל מצב s בלוח שהוא בשורה הראשונה ($s \in [0,7]$) נקבל כי UP(s) = s

מכל מצב אחר s בלוח שהוא לא חור קיים מצב אחר 's עבורו : UP(s)=s'

אם מגיעים לחור בבעיית האגם נתקעים והמשחק מסתיים ולען אי אפשר להמשיך ולהפעיל אף אופרטור על חורים.

סעיף 4) מה תחזיר לנו הפונקציה Succ על המצב ההתחלתי 0?

$$Succ(0) = \{8,1,0\}$$

מתקיים כי: $DOWN(0) = 8, \quad RIGHT(0) = 1, \quad UP(0) = LEFT(0) = 0$

סעיף 5) האם קיימים מעגלים במרחב החיפוש שלנו? כן, עבור מצב 1 למשל, אם נבצע UP(DOWN(1)) נחזור למצב 1 כלומר זהו מעגל.

סעיף 6) מה הוא מקדם הסיעוף בבעיה? b=4, מכל מצב ניתן להגיע לכל היותר ל-4 מצבים שונים אחרי הפעלת 4 האופרטורים המוגדרים בסעיף 1.

סעיף 7) במקרה הגרוע ביותר, כמה פעולות ידרשו לסוכן כללי להגיע למצב הסופי? אינסוף, במקרה הגרוע ביותר נתקע במעגלים שאינם מכילים מצב מטרה או חור ולכן נבצע אינסוף פעולות מבלי שהמשחק יסתיים.

סעיף 8) במקרה הטוב ביותר, כמה **פעולות** ידרשו לסוכן כללי להגיע למצב הסופי? (מדובר בלוח 8x8 הנתון)

במקרה הטוב ביותר הסוכן יבצע את 9 הפעולות הבאות: $0 \to 0 \to 0 \to 1 \to 0 \to 0 \to 0 \to 1 \to 1$

נשים לב כי בלוח הנתון הפורטלים מצמצמות את מספר הפעולות הנדרשות להגיע למצב המטרה, כלומר נצטרך 4 פעולות להגיע ממצב ההתחלתי לפורטל הכי קרוב, ואז 5 פעולות להגיע מהפורטל השני למטרה, סה"כ 9 פעולות.

S	F	F	F	F	F	F	F
F	F	F	F	F	T	A	L
T	F	F	H	F	F	T	F
F	P	F	F	F	H	T	F
F	A	F	H	F	P	F	F
F	H	H	F	F	F	H	F
F	H	T	F	H	F	T	L
F	L	F	H	F	F	F	G

** הדרך מסומנת בירוק בגרף הנ"ל

סעיף 9) עבור לוח כללי בסביבת ה frozen lake המסלול הקל ביותר הוא המסלול שמגיע למצב מטרה שהכי קרוב למצב ההתחלתי (במונחים של manhattan distance)? (מדובר בלוח כללי ולא בהכרח את הלוח הנתון)

הטענה לא נכונה, דוגמה נגדית:

S	F	F	F	F	F	F	F
L	H	H	H	H	H	H	F
L	H	H	H	H	H	H	F
L	H	H	H	H	H	H	F
L	H	H	H	H	H	H	G1
L	H	H	H	H	H	H	H
L	H	H	H	H	H	H	H
L	L	L	L	L	L	L	G2

יש בלוח הנ"ל שני מצבי מטרה G1 ו-G2 ושני מסלולים אופטימליים להגיע מהמצב ההתחלתי לכל אחד משני מצבי המטרה, לפי המחירים הנתונים מתקיים כי:

המסלול המסומן בירוק הינו המסלול המוביל למצב מטרה 1 שהוא בעל מרחק מנהאתן של 11 ממצב ההתחלתי אבל מחיר המסלול הינו $10 * 10 + 1 = 101$

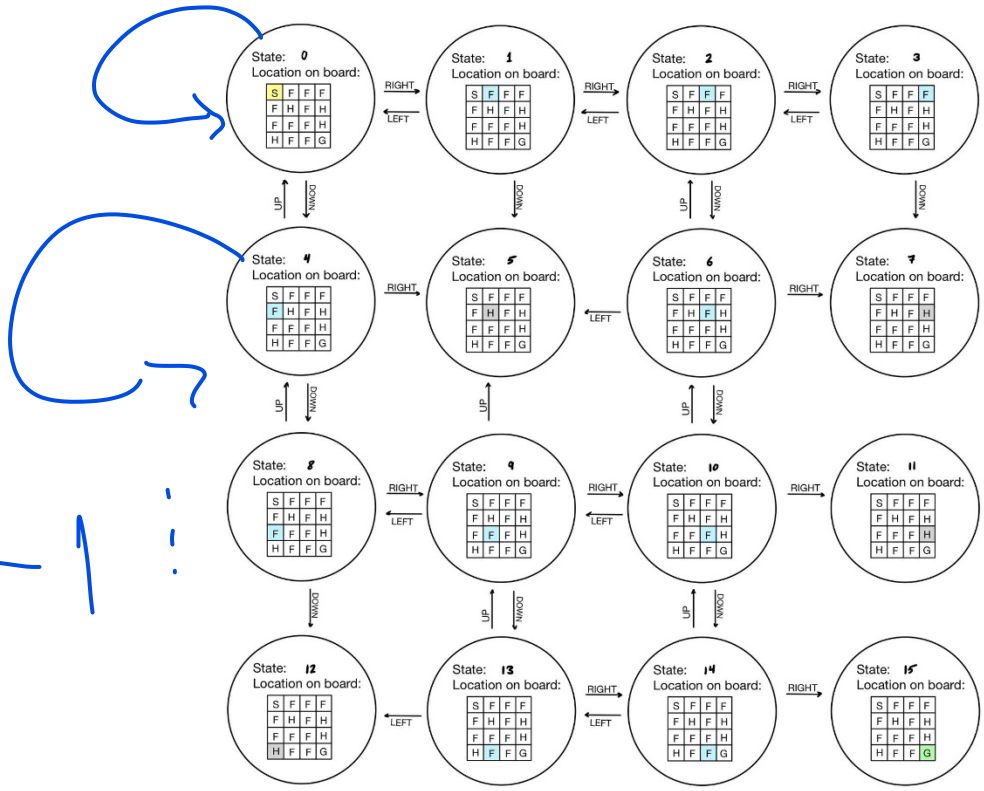
מצד שני המסלול המסומן בצהוב מוביל למצב מטרה 2 שהוא בעל מרחק מנהאתן של 14 ממצב ההתחלתי אבל מחיר המסלול הינו $14 * 1 = 14$

שאלה 2:

סעיף 2) מה צריך להיות התנאי על גרף החיפוש (לא בהכרח בבעיית האגם הקפוא) כך שBFS על גרף ו-BFS על עץ ייצרו ויפתחו צמתים זהים באותו הסדר?

כדי שאלגוריתם BFS יעל באותה צורה גל גרף ועל עץ מרחב החיפוש שלנו צריך להיות חסר מעגלים, ההבדל בין BFS-G ו-BFS רגיל הוא שב-BFS-G האלגוריתם שומר רשימה של הצמתים שהוא כבר פיתח ויצר ואז לא יפתח או יצור אותה צומת פעמיים, ב-BFS על עץ הוא יפתח אותה צומת פעמיים במקרה של מעגלים במרחב החיפוש, ולכן כדי ששניהם יפעלו באותה צורה הגרף צריך להיות חסר מעגלים.

סעיף 3) עבור הלוח "4x4" שמופיע במחברת, ציירו את גרף המצבים.



סעיף 4) נתון לוח בגודל NxN שלא מכיל portals. הציעו דרך להשתמש באלגוריתם BFS-G כך שיחזיר פתרון אופטימלי (עלות מינימלית) והסבירו.

נעשה רדוקציה לגרף בצורה הבאה :

לכל צומת בעלת ערך קבוע A אנחנו ניצור על אותו מסלול בגרפף החדש A צמתים דמה בגרף שמתארות את העלות הנדרשת ף כלומר אם למשל אנחנו בצומת T כך שעלותה היא 3 אז יש לנו מסלול T1→T2→T3→next_state ואם נתקענו בבור אז יש לנו מסלול חסום(מסלול שגוי – כך שאין מסלול לצומת המטרה) כך של S ו G ו L יש צומת אחד , A שני צמתי דמה , T שלושה וכו

וכאשר משתמשים ב BFS-G מובטח לנו שייוצר את המסלול הקצר ביותר למטרה ועם הרדורציה שעשינו מחיר הצמתים הוא המחיר המינימלי

סעיף 5) נתון לוח בגודל NxN, ללא חורים, ללא Portals, המכיל N^2-2 משבצות רגילות (F,T,A,L) מצב התחלתי בפינה השמאלית עליונה ומצב מטרה בפינה הימנית תחתונה. כמה צמתים יפותחו וייווצרו במהלך חיפוש BFS-G? הסבירו?

ייווצרו: N^2

ייפתחו: N^2-2

האלגוריתם יוצר לכל צומת 4 בנים (המתקבלים מהפעלת 4 האופרטורים על הצומת), אבל כיוון שהוא פועל על גרף ולא על עץ לכן הוא לא מייצר אותה צומת יותר מפעם אחת (לפי פוסט @69 בפיאצה), ולכן, כיוון שצומת המטרה נמצאת ברמה הכי תחתונה בגרף החיפוש (הכי רחוקה מהמצב ההתחלתי), לכן הוא מייצר כל צומת בגרף, סה"כ N^2

הוא מפתח כל צומת פעם אחת, חוץ מצומת המטרה, כיוון שניתן להגיע אליה משני צמתים שונים שהם באותה רמה ברף המצבים, לכן הוא יפתח את הצומת הראשון ומגלה שהגיע למטרה ואז לא ממשיך לפתח את הצומת השני, סה"כ N^2-2

שאלה 3:

סעיף 2) עבור בעיית האגם הקפוא עם לוח NxN, האם האלגוריתם שלם? האם הוא קביל?
שלם: כן, כיוון שהאלגוריתם רץ על גרף (כלומר לא מפתח אותו צומת פעמיים) בעל קבוצת מצבים סופית לכן כפי שלמדנו בקורס הוא אכן שלם ומובטח שיחזיר פתרון.
קביל: לא, האלגוריתם לא מוצא את הפתרון בעל המרחק (או המחיר) הקטן ביותר, ניתן לראות את זה בקלות עבור הדוגמה הנתונה, בשאלה 1 סעיף 8 מצאנו את הפתרון בעל מספר הצעדים הקטן ביותר, אבל אם נריץ DFS-G נקבל את הפתרון הבא (המסומן בירוק):

S	F	F	F	F	F	F	F
F	F	F	F	F	T	A	L
T	F	F	H	F	F	T	F
F	P	F	F	F	H	T	F
F	A	F	H	F	P	F	F
F	H	H	F	F	F	H	F
F	H	T	F	H	F	T	L
F	L	F	H	F	F	F	G

כלומר האלגוריתם לא קביל (למדנו בקורס).

סעיף 3) האם אלגוריתם DFS (על עץ), עבור בעיית האגם הקפוא על לוח NxN, היה מוצא פתרון כלשהו? אם כן, מה המסלול שיתקבל? אם לא, כיצד האלגוריתם היה פועל?

לא, ניזכר בהתנהגות סוכן כלשהו על בעיית האגם:

1. סוכנים יוצרים את הבנים של צומת $s \in S$ לפי הסדר $DOWN(s) \to RIGHT(S) \to UP(s) \to LEFT(s)$
2. עבור צמתים $\{s \in S \mid (N-1)*N \le s \le (N-1)(N+1)$ כלומר מצבים בשורה התחתונה במפה מתקיים: $DOWN(s) = s$

אלגוריתם DFS על עץ לא שומר את קבוצת המצבים שהוא כבר פיתח ולכן הוא תמיד יעשה DOWN (יבצע פעולות אחרות אם הוא מגיע לחור) עד שמגיע לשורה הכי תחתונה, (אם הוא בצע N RIGHT+ פעמים אז הוא יגיע למטרה אם היא נמצאת בפינה הימנית אבל מדובר בדרך כלל עבור לוח כללי ללא הנחות עליו), בסוף הוא יגיע לשורה התחתונה אם קיימת דרך ואז יבצע DOWN ללא שינוי במקום ואז הוא יתקע בלולאה אינסופית ולא ימצא פתרון. דבר דומה יכול לקרות עבור כל האופרטורים ולכן הסיכוי שסוכן DFS יתקע במקום (בלולאה אינסופית) ולא ימצא פתרון עבור לוח כללי כלשהו הוא גדול.

סעיף 4) נתון לוח בגודל NxN, ללא חורים, ללא Portals, המכיל N^2-2 משבצות רגילות (F,T,A,L) מצב התחלתי בפינה השמאלית עליונה ומצב מטרה בפינה הימנית תחתונה. כמה צמתים יפותחו וייווצרו במהלך חיפוש DFS-G? הסבירו?

ייווצרו: $4N-5$

ייפתחו: $2N-2$

כפי שהוסבר בסעיף הקודם עבור DFS על עץ רק במקום שהוא יתקע ולא ימצא פתרון, הוא שומר קבוצת המצבים שהוא פיתח ולא יחזור על אותו מצב פעמיים, ולכן הוא ירד עד לשורה התחתונה ואז הולך ימינה עד מצב המטרה.

הוא יפתח כל צומת במסלול הזה:

כלומר יפתח את המצב ההתחלתי, כל צומת אחר בעמודה הראשונה (N-1), כל צומת בשורה האחרונה חוץ ממצב המטרה שהוא עוצר אחרי שהוא יגלה אותו ולכן לא מפתח אותו (נשים לב כי הצומת בפינה השמאלית התחתונה משותף עבור השורה הראשונה והעמודה האחרונה והאלגוריתם יפתח אותו פעם אחת ולא פעמיים), סה"כ מקבלים:

$$I + (first\ column - I) + (last\ row - G) - (bottom\ left\ corner) = 1 + (N-1) + (N-1) - 1 = 2N - 2$$

עבור כל צומת במסלול שהוא עובר הוא יוצר את כל השכנים שלו (חוץ מאלה שהוא כבר יצר בעבר לפי פוסט @69 בפיאצה והוא לא יוצר אותו צומת פעמיים), עבור כל צומת נשים לב כי הם בקצה הלוח וכיוון שהם שכנים אז עבור כל צומת הוא רק יוצר שני שכנים (חוץ מהפינה השמאלית התחתונה והשכן שלה שהוא רק יוצר בן אחד עבורם):

$$\begin{aligned} 2 * I + 2 * (first\ column - I - bottom\ left\ corner) + 1 * (bottom\ left\ corner) + 1 \\ * (bottom\ left\ corner's\ neghbor) + 2 \\ * (last\ row - bottom\ left\ corner\ and\ its\ neighbor - G) + G \\ = 2 + 2 * (N-1-1) + 1 + 1 + 2(N-1-1-1) + 1 = 4N - 5 \end{aligned}$$

סעיף 5) נתון לוח בגודל NxN, ללא חורים, ללא Portals, המכיל $N^2 - 2$ משבצות רגילות (F,T,A,L) מצב התחלתי בפינה השמאלית עליונה ומצב מטרה בפינה הימנית תחתונה. כמה צמתים <u>יפותחו</u> <u>וייווצרו</u> במהלך חיפוש backtracking DFS-G? הסבירו?

ייווצרו: $2N-1$

ייפתחו: $2N-2$

אלגוריתם backtracking DFS-G הינו אלגוריתם DFS-G "עצלן", כלומר הוא מפתח צומת ואז יוצר הבן שלה ומיד אחריה הוא עובר לבן (במקרה שהוא לא פותח כבר), הוא לא שומר רשימה של הצמתים שהוא יצר ורוצה לפתח בסדר כלשהו, אלא הוא מפתח כל צומת שהוא יוצר מיד אחרי היצירה (אם לא פיתח אותה בעבר) ואם נתקע הוא חוזר אחורה ע"י רקורסיה, ולכן הוא מפתח אותו מספר של צמתים כמו DFS-G בנוסף למצב המטרה שהוא רק יוצר ולא מפתח, סה"כ נקבל:

$$(first\ column - bottom\ left\ corner) + (last\ row) = N - 1 + N = 2N - 1$$

שאלה 4:

ג'רי רוצה למצוא מסלול בסביבת האגם הקפוא עם DFS-L. ידוע כי אורך המסלול הקצר ביותר לצומת מטרה הוא d אך ריק מגביל את החיפוש של ג'רי לעומק $\frac{d}{2}$.

סעיף a) הציעו שינוי **לבעיית החיפוש** (S, O, I, G) כך שג'רי יוכל למצוא פתרון מבלי להפר את הגבלת העומק שריק הטיל עליו. הסבירו למה כעת ניתן למצוא פתרון.

נשנה את מרחב החיפוש ל (S', O', I', G') באופן הבא:

S - מרחב הצמתים $S' = S = \{ s \mid s \in \mathbb{N} : 0 \le s \le (N-1) * (N+1) \}$

O – קבוצת האופרטורים

$$O' = \{OP1xOP1 \mid OP1, OP2 \in O\} \cup O =$$

$$\left\{ \begin{matrix} DOWNxDOWN, & DOWNxRIGHT, & DOWNxUP, & DOWNxLEFT, \\ RIGHTxDOWN, & RIGHTx\ RIGHT, & RIGHTxUP, & RIGHTx\ LEFT, \\ UPxDOWN, & UPx\ RIGHT, & UPxUP, & UPxLEFT, \\ LEFTxDOWN, & LEFTxRIGHT, & LEFTx\ UP, & LEFTxLEFT, \\ DOWN, & RIGHT, & UP, & LEFT \end{matrix} \right\}$$

נגדיר את הפעלת האופרטורים על צומת $s \in S$ הנ"ל באופן הבא:

$$\forall\ OP \in O \ : \ OP(s) \equiv OP(s) \quad as\ defined\ in\ the\ previos\ search\ space$$

$$\forall\ OP1, OP2 \in O \ : \ OP1xOP2(s) \equiv OP2(\ OP1(s)\ )$$

I – קבוצת המצבים ההתחלתיים בלוח הנתון הינה $I' = I = \{0\}$

G – קבוצת מצבי המטרה בלוח הנתון הינה $G' = G = \{63\}$

כלומר נשנה את קבוצת האופרטורים כך שנוסיף לה גם שני צעדים קדימה עבור כל זוג אופרטורים, סדר יצירת הבנים של כל צומת הוא כסדר הופעתם ב-O', כלומר $DOWNxDOWN \rightarrow DOWNxRIGHT \rightarrow DOWNxUP \rightarrow \cdots \rightarrow LEFT$

כיוון שעכשיו קבוצת האופרטורים מכילה גם שני צעדים קדימה לכן ניתן למצוא פתרון שנמצא במרחק של d בבעיה המקורית בעומק של d/2 בגרף החיפוש החדש.

סעיף b) האם השתנה מקדם הסיעוף? מה מקדם הסיעוף החדש b'? אם כן רשמו את התשובה כתלות בb (מקדם הסיעוף בבעיה המקורית).

כן, $b' = b * b + b = b^2 + b$, כי מתקיים: $|\{OP1xOP1 \mid OP1, OP2 \in O\}| = b * b$

סעיף c) מה סיבוכיות הזמן והמקום החדשים? ענו במונחים של b, d והשוו את התשובה לDFS-L רגיל עם עומק d. כיצד תשובתכם הייתה משתנה עם היינו משתמשים בDFS-L עם בקטרקינג?

לפי מה שלמדנו בקורס נקבל:

אלגוריתם	סיבוכיות זמן	סיבוכיות מקום
DFS-L עם מרחק d	$O(b^d)$	$O(bd)$
DFS-L עם מרחק d/2	$O(b'^{d'}) = O((b^2+b)^{d/2})$	$O(b'd') = O((b^2+b) * d)$ $= O(b^2 d)$
backtracking DFS-L	$O(b^d)$	$O(d)$

סעיף d) ספקו דוגמא לבעיה שבה DFS-L במרחב החיפוש החדש (לאחר השינויים שביצעתם בa) יותר טובה מאשר DFS-L במרחב החיפוש הקודם ודוגמה לבעיה שבה DFS-L במרחב המקורי עדיף. בתשובתכם התייחסו למספר צמתים שפותחו. דוגמאות יכולות להיות כלליות ולא בהכרח מסביבת ה frozen lake.

נצמצם את מרחב החיפוש הנתון (כדי להקל את הדוגמה):

$$S = \{ s \mid s \in \mathbb{N} : 0 \leq s \leq 24\}$$

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

$$I = \{0\}$$

$$G = \{23\}$$

דוגמה 1 (עבורה DFS-L עם מרחק d/2 יותר טוב):

S	F	F	F	F
F	F	F	F	F
F	F	F	F	F
F	G	F	F	F
F	F	F	F	F

אלגוריתם DFS-L עם מרחק d/2 עובר על הצמתים במסלול המסומן בירוק, כלומר מפתח את הצמתים:

$$s1 = S$$

$$s2 = DOWNxDOWN(S)$$

$$s3 = DOWNxDOWN(s2)$$

$$s4 = DOWNxRIGHT(s3)$$

$$s5 = DOWNxRIGHT(s4)$$

$$s6 = DOWNxRIGHT(s5)$$

$$s7 = DOWNxRIGHT(s6)$$

$$s8 = DOWNxRIGHT(s1)$$

$$s9 = DOWNxDOWN(s8) = GOAL$$

ואז הוא יגלה שצומת המטרה היא s9 ולכן עוצר.

S	F	F	F	F
F	F	F	F	F
F	F	F	F	F
F	G	F	F	F
F	F	F	F	F

אלגוריתם DFS-L עם מרחק d עובר באותו אופן על הצמתים במסלול המסומן בצהוב, כלומר מפתח את הצמתים:

$$s1 = S$$

$$s2 = DOWN(s2)$$

$$s3 = DOWN(s2)$$

$$s4 = DOWN(s3)$$

$$s5 = DOWN(s4)$$

$$s6 = RIGHT(s5)$$

$$s7 = RIGHT(s6)$$

$$s8 = RIGHT(s7)$$

$$s9 = RIGHT(s8)$$

$$s10 = RIGHT(s1)$$

$$s11 = DOWN(s10)$$

$$s12 = DOWN(s11)$$

$$s13 = DOWN(s12) = GOAL$$

ואז הוא יגלה שצומת המטרה היא s13 ולכן עוצר.

דוגמה 1 (עבורה DFS-L עם מרחק d יותר טוב):

S	F	F
G	F	F
F	F	F

אלגוריתם DFS-L עם מרחק d עובר באותו אופן על הצמתים במסלול המסומן בצהוב, כלומר מפתח את הצמתים:

$$s1 = S$$

$$s2 = DOWN(s2) = GOAL$$

S	F	F
G	F	F
F	F	F

אלגוריתם DFS-L עם מרחק d/2 עובר על הצמתים במסלול המסומן בירוק, כלומר מפתח את הצמתים:

$$s1 = s$$

$$s2 = DOWNxDOWN(s1)$$

$$s3 = RIGHTxDOWN(s2)$$

$$s4 = RIGHTxDOWN(s3)$$

$$s5 = DOWNxRIGHT(s1)$$

$$s6 = RIGHTxRIGHT(s5)$$

$$s7 = RIGHTxRIGHT(s1)$$

$$s8 = DOWN(s1) = GOAL$$

שאלה 5:

הניחו כי יש לנו ידע מקדים על חסם עליון למרחק למצב מטרה, נסמנן D. בת (Beth) הציעה את האלגוריתם החיפוש ReverseDFS:

סעיף a) האם האלגוריתם שלם? אם כן, הוכיחו. אם לא, ספקו דוגמה נגדית
כן, האלגוריתם שלם.
נתאר התנהגות האלגוריתם:
בהינתן חסם על מרחק מצב מטרה, האלגוריתם יפעיל DFS-L עם L שהולך וקטן עד שהוא לא ימצא פתרון עבור L קטן כלשהו ואז מחזיר את הפתרון שהוא מצא באיטרציה הקודמת.
נתון כי האלגוריתם משתמש ב-DFS-L ונתון כי מרחק מצב המטרה חסום ונתון כי D הינו חסם עליון על מצב המטרה הזה, לכן מובטח שבאיטרציה הראשונה האלגוריתם (שמפעיל DFS-L עם מרחק D שהוא שלם במקרה זה לפי מה שלמדנו בקורס) ימצא פתרון, ואז בכל איטרציה אחריה אם הוא לא ימצא פתרון אז הוא יחזיר את הפתרון שמצא באיטרציה הקודמת, ולכן מובטח שיחזיר פתרון אם קיים אחד.

סעיף b) האם האלגוריתם אופטימלי? אם כן, הוכיחו. אם לא, ספקו דוגמה נגדית
כן האלגוריתם קביל.
תהי בעיית חיפוש בעלת פתרון אופטימלי הנמצא בעומק $d \leq D$ בעץ החיפוש, למדנו כי DFS-L הינו שלם עבור מרחק גדול מ-d, נביט בהתנהגות האלגוריתם החדש:

איטרציה	עומק הפתרון שנמצא ע"י *DFS-L* באלגוריתם *ReverseDFS*
1	קטן שווה *D*
2	קטן שווה *D-1*
3	קטן שווה *D-2*
4	קטן שווה *D-3*
...	...
i	קטן שווה *D-(i-1)*
...	...
D-d+1	קטן שווה *d*

כיוון שהפתרון האופטימלי הינו בעומק *d* לכן באיטרציה ה- *D-d+2* האלגוריתם לא ימצא פתרון והוא יחזיר את הפתרון שהוא מצא באיטרציה הקודמת, כלומר יחזיר פתרון בעומק *d* אופטימלי כלומר האלגוריתם קביל לפי הגדרה.

סעיף *c*) ספקו דוגמה בה ReverseDFS עדיף על ID-DFS ודוגמה בה ID-DFS עדיף על ReverseDFS. הדוגמאות יכולות להיות כלליות ולא בהכרח מסביבת התרגיל.

דוגמה 1 (עבורה ID-DFS-L יותר טוב):

S	F	G	F
F	F	F	F
F	F	F	F
F	F	F	F

המטרה נמצאת במרחק d=2 מהמצב ההתחלתי.

לפי הגדרת ID-DFS-L הוא ימצא פתרון באיטרציה השנייה ומחזיר אותו, לעומת ReverseDFS עם D=6 הוא יצטרך 5 איטרציות עד שימצא איטרציה שבה לא קיים פתרון ואז הוא יחזיר את הפתרון שהוא מצא באיטרציה הרביעית.
כלומר ID-DFS-L יבצע פחות איטרציות מאשר ReverseDFS.

דוגמה 2 (עבורה ReverseDfs יותר טוב):

S	F	F	F
F	F	F	F
F	F	F	F
F	F	F	G

המטרה נמצאת במרחק d=6 מהמצב ההתחלתי.

לפי הגדרת ID-DFS-L הוא ימצא פתרון באיטרציה הששית ומחזיר אותו, לעומת ReverseDFS עם D=6 הוא יצטרך שתי איטרציות בהן הוא מוצא פתרון באיטרציה הראשונה ואז עובר לאיטרציה הבאה שבה לא קיים פתרון ואז הוא יחזיר את הפתרון שהוא מצא באיטרציה הקודמת.
כלומר ID-DFS-L יבצע יותר איטרציות מאשר ReverseDFS.

סעיף d) הציעו כיצד ניצן לייעל את האלגוריתם. רמז: האם אתם יכולים לחשוב על צעד עדכון עדיף לL?
נתחיל עם $L \leftarrow D$ כמו שתואר מקודם, בכל איטרציה אם מצאנו פתרון נמשיך עם $L \leftarrow L/2$, וכך ממשיכים עד האטרציה הראשונה שבה לא נמצא פתרון, נניח שבה מתקיים $L = l$, אז מבצעים $L \leftarrow l * 2$ וממשיכים עם ReverseDFS כרגיל כמו שתואר בשאלה.
הסבר:
לפי הגדרת שני האלגוריתמים:
אלגוריתם ID-DFS-L עדיף אם החסם D גדול מאוד מהמרחק האמתי של מצב המטרה האופטימלי.
אלגוריתם ReverseDFS עדיף אם החסם D קרוב מאוד למרחק האמיתי של מצב המטרה האופטימלי.
ולכן מה שהעדכון הנ"ל עושה הוא שמאפשר לנו לדלג על הרבה אטירציות מיותרות שהן רחוקות מדי מהמרחק האמיתי של מצב המטרה האופטימלי, למשל עבור D=20 ו-optimal_d=5, נתאר התנהגות אלגוריתם ReverseDFS החדש:

מספר איטרציה	L	האם מצא פתרון
נתחיל עם L ואז בכל אטרציה מחלקים ב-2 עד שלט נמצא פתרון		
1	D=20	כן
2	L=20/2=10	כן
3	L=10/2=5	כן
4	L=5/2=3	לא
לא מצאנו פתרון ולכן נבצע L=2L ואז קוראים ל-ReverseDFS שהוגדר מקודם		
5	L=3*2=6	כן
6	L=6-1=5	כן
7	L=5-1=4	לא

ואז באיטרציה השביעית לא ימצא פתרון ומחזיר את הפתרון שמצאונו באטרציה 6 שהוא הפתרון האופטימלי בעל מרחק 5.

לעומת *ReverseDFS* שהוגדר מקודם, שהיה מצטרך $20 - 6 + 1 = 15$ אטרציה עד שיגיע לאותו פתרון.

שאלה 6:
סעיף 2) עבור אילו בעיות חיפוש אלגוריתם UCS ואלגוריתם BFS יפעלו באותו האופן? הסבירו.

עבור בעיות שבהן מחיר כל הקשתות באותה רמה שווים (וחיוביים).

אלגוריתם BFS מפתח את הצומת הכי קרוב למצב ההתחלתי, אלגוריתם UCS מפתח הצומת בעל המחיר הכי קטן, אלגוריתם BFS הינו מקרה פרטי של UCS שבו מחיר של הקשתות הוא a קבוע חיובי :

במקרה שבו המסלולים שמגיעים לצמתים באותה רמה הן שווי מחיר אז אלגוריתם UCS מפתח את הצמתים לפי הרמות (כיוון שהמחיר יגדל ככל שיורדים ברמות לכן הוא מפתח את הרמה הראשונה שהיא בעלת מחיר שווה, ואז הרמה השנייה שהיא בעלת מחיר שווה הגדול מהרמה הקודמת, וכך הלאה), כלומר הוא מפתח את הצמתים באותו אופן דומה ל-BFS שהוא חיפוש לרוחב.

סעיף 3) האם בבעיית החיפוש שלנו, עבור לוח NxN, האלגוריתם הוא שלם? האם הוא קביל? כן וכן

שלם: כיוון שפונקציית המחיר חסומה מלמטה ע"י 1 שהוא גדול מ-0 וכיוון שמחיר מצב חור הוא ∞ לכן לפי מה שלמדנו בקורס, האלגוריתם אכן שלם. כיוון שכל עוד יש צומת נגיש שהוא לא חור שעדיין לא פיתחנו אז האלגוריתם יפתח אותו, וזה כולל צומת מטרה, כל עוד קיים צומת מטרה שהוא נגיש מהמצב ההתחלתי אז האלגוריתם יפתח אותו לפני פיתוח חורים שהם מסיימים את המשחק.

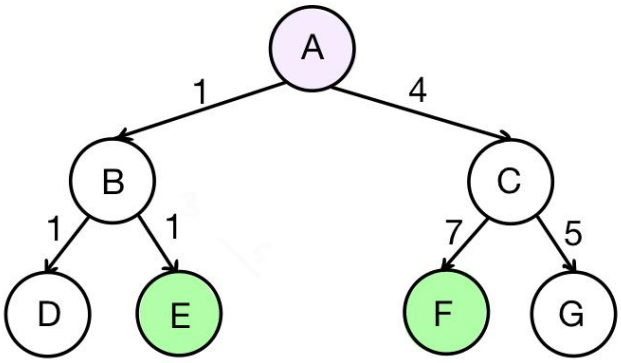
קביל: האלגוריתם הוא כן קביל מבחינת חיפוש הפתרון בעל המחיר הקטן ביותר (כמו שלמדנו והוכחנו בקורס), אבל מבחינת מרחק הוא לא מובטח להחזיר את הפתרון בעל המרחק הקטן ביותר.

סעיף 4) דן טעה במימוש של אלגוריתם UCS ובטעות בדק בעת יצירת הצומת האם היא צומת מטרה במקום בפיתוח שלה. הביאו דוגמה לגרף חיפוש שעבורו דן יחזיר בכל זאת את המסלול הקל ביותר ודוגמה לגרף חיפוש שעבורו דן לא יחזיר את

המסלול הקל ביותר. עבור כל דוגמה הסבירו מה המסלול והעלות ש-UCS השגוי החזיר, ומה המסלול והעלות שהאלגוריתם הנכון היה מחזיר. נדגיש שגרף החיפוש לא בהכרח צריך לייצג את בעיית האגם הקפוא. אתם יכולים לתת דוגמה לגרף שמייצג בעיית חיפוש אחרת. הגרף צריך להכיל קשתות מכוונות ואת העלות של כל קשת.

נביט בדוגמה הבאה ונתאר את התנהגות האלגוריתם שדן יצר:

דוגמה 1 (עבורה דן מחזיר את הפתרון הנכון):

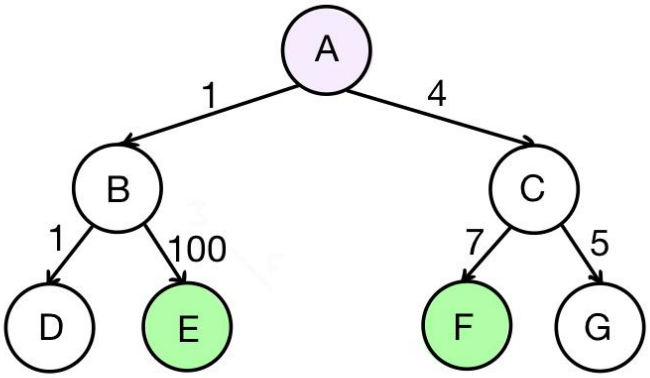


צומת התחלתי: A
צמתי מטרה:E ו-F

המסלול המוחזר מאלגוריתם *UCS* שלמדנו בקורס הוא: $A \to B \to E$ כיוון שזהו המסלול בעל המחיר (2) הקטן ביותר המוביל למטרה.
עבור השינוי שדן בצע נקבל את הבא:

1. ניצור את *A* ונבדוק אם שה מצב מטרה
2. נפתח את *A*: ניצור *B* ו-*C* ונבדוק אם הם צמתי מטרה, נגלה שלט ונוסיף אותם ל-*opened* בהתאם לסדר
3. נפתח את *B*: ניצור את *D* ו-*E* ונבדוק האם הם צמתי מטרה, נגלה ש-*E* הוא אכן מטרה ואז מחזירים את המסלול: $A \to B \to E$ שהוא בעל ממחיר 2 כלומר אופטימלי מבחינת מחיר.

דוגמה 2 (עבורה דן לא מחזיר את הפתרון הנכון):



צומת התחלתי: A
צמתי מטרה:E ו-F

המסלול המוחזר מאלגוריתם *UCS* שלמדנו בקורס הוא: $A \to C \to F$ כיוון שזהו המסלול בעל המחיר (11) הקטן ביותר המוביל למטרה.
עבור השינוי שדן בצע נקבל את הבא:

4. ניצור את *A* ונבדוק אם שה מצב מטרה
5. נפתח את *A*: ניצור *B* ו-*C* ונבדוק אם הם צמתי מטרה, נגלה שלט ונוסיף אותם ל-*opened* בהתאם לסדר
6. נפתח את *B*: ניצור את *D* ו-*E* ונבדוק האם הם צמתי מטרה, נגלה ש-*E* הוא אכן מטרה ואז מחזירים את המסלול: $A \to B \to E$ שהוא בעל ממחיר 101 כלומר אינו אופטימלי מבחינת מחיר.

שאלה 7:

סעיף 1) בהיתן שתי יוריסטיקות קבילות h_1, h_2. האם $h = \min\{h_1, h_2\}$ קבילה? אם כן, הוכיחו. אם לא, הפריכו.

נתון:

$$\forall s \in S : h1(s) \le h^*(s)$$

$$\forall s \in S : h2(s) \le h^*(s)$$

יהי $s \in S$:

$$h(s) = \min\{h1(s), h2(s)\} \le h1(s) \; and \; h2(s) \le \; h^*(s)$$

נקבל כי h קבילה לפי הגדרה.

סעיף 2) בהיתן שתי יוריסטיקות קבילות h_1, h_2. האם $h = \max\{h_1, h_2\}$ קבילה? אם כן, הוכיחו. אם לא, הפריכו.

נתון:

$$\forall s \in S : h1(s) \le h^*(s)$$

$$\forall s \in S : h2(s) \le h^*(s)$$

יהי $s \in S$:

נניח בה"כ כי מתקיים: $h1(s) \le hs2(s)$

$$h(s) = \max\{h1(s), h2(s)\} = hs(s) \le \; h^*(s)$$

נקבל כי h קבילה לפי הגדרה.

סעיף 3) בהיתן שתי יוריסטיקות עקביות h_1, h_2. האם $h = \min\{h_1, h_2\}$ עקבית? אם כן, הוכיחו. אם לא, הפריכו.

נתון:

$$\forall s \in S, \quad s' \in Succ(s): \quad h1(s) - h1(s') \le cost(s, s') \quad \rightarrow \quad h1(s) \le cost(s, s') + h1(s')$$
$$\forall s \in S, \quad s' \in Succ(s): \quad h2(s) - h2(s') \le cost(s, s') \quad \rightarrow \quad h2(s) \le cost(s, s') + h2(s')$$

יהי $s \in S$:

נניח בה"כ כי מתקיים: $h1(s) \le hs2(s)$:

$$h(s) = \min\{h1(s), h2(s)\} = h1(s)$$

נחלק למקרים:

1. $h1(s') \le h2(s') \rightarrow h(s') = h1(s') \; :$

$$h(s) - h(s') = h1(s) - h1(s') \le cost(s, s')$$

2. $h1(s') \ge h2(s') \rightarrow h(s') = h2(s') \; :$

$$h(s) - h(s') = h1(s) - h2(s') \le h2(s) - h2(s') \le cost(s, s')$$

כלומר נקבל כי h עקבית לפי הגדרה.

סעיף 4) בהיתן שתי יוריסטיקות עקביות h_1, h_2. האם $h = \max\{h_1, h_2\}$ עקבית? אם כן, הוכיחו. אם לא, הפריכו.

נתון:

$$\forall s \in S, \quad s' \in Succ(s): \quad h1(s) - h1(s') \le cost(s, s') \quad \rightarrow \quad h1(s) \le cost(s, s') + h1(s')$$
$$\forall s \in S, \quad s' \in Succ(s): \quad h2(s) - h2(s') \le cost(s, s') \quad \rightarrow \quad h2(s) \le cost(s, s') + h2(s')$$

יהי $s \in S$:

נניח בה"כ כי מתקיים: $h1(s) \ge hs2(s)$:

$$h(s) = \max\{h1(s), h2(s)\} = h1(s)$$

נחלק למקרים:

1. $h1(s') \ge h2(s') \rightarrow h(s') = h1(s') \; :$

$$h(s) - h(s') = h1(s) - h1(s') \le cost(s, s')$$

2. $h1(s') \le h2(s') \rightarrow h(s') = h2(s') \; :$

$$h(s) - h(s') = h1(s) - h2(s') \le h1(s) - h1(s') \le cost(s, s')$$

כלומר נקבל כי h עקבית לפי הגדרה.

סעיף 5) נגדיר יוריסטיקה חדשה עבור בעיות עם מצב מטרה יחיד $|G| = 1$:

$$h_{SAP}(s) = \min\{h_{Manhatan}(s,g), Cost(p)\}$$

כאשר הביטוי הראשון הוא מרחק מנהטן מהמצב הנוכחי למצב הסופי והביטוי השני הוא עלות קשת המביאה למשבצת שיגור (קבוע).
האם היוריסטיקה h_{SAP} קבילה על כל לוח? אם כן הסבר, אם לא הבא קודמה נגדית.

נחלק למקרים:

1. המסלול האופטימלי ממצב s למצב המטרה עובר דרך portal:

ולכן מתקיים:

$$h_{SAP}(s) = \min\{h_{manhattan}(s,g)\,, cost(p)\} \le cost(p) \le cost\ of\ route\ passing\ a\ portal \le h^*(s)$$

2. המסלול האופטימלי ממצב s למצב מטרה לא עובר דרך portal:

ולכן מתקיים:

$$h_{SAP}(s) = \min\{h_{manhattan}(s,g), cost(p)\} \le h_{manhattan}(s,g)$$
$$= shortest\ route\ from\ s\ to\ g\ in\ a\ board\ with\ no\ holes\ or\ portals\ with\ cost\ 1\ (minimal) for\ all\ edges$$
$$\le actual\ optimal\ route\ from\ s\ to\ g = h^*(s)$$

כלומר נקבל כי h קבילה לפי הגדרה.

סעיף 6) האם היוריסטיקה h_{SAP} עקבית על כל לוח? אם כן הסבר, אם לא הבא דוגמה נגדית.

תהיינה $s \in S$, ו- $s' \in Succc(s)$, נחלק למקרים לפי ערך h_{SAP} של כל אחד מהצמתים:

1. $h_{SAP}(s) = h_{manhattan}(s,g)\ \ and\ \ h_{SAP}(s') = h_{manhattan}(s',g)$:

כיוון ש-s ו-s' עוקבים לכן מתקיים (לפי הגדרת מרחק מנהאתן וצמתים עוקבים בלוח שלנו):

$$h_{SAP}(s) - h_{SAP}(s') = h_{manhattan}(s,g) - h_{manhattan}(s',g) \le 1$$
$$= minimal\ cost\ of\ an\ edge\ in\ the\ frozen\ lake\ problem \le cost(s,s')$$

2. $h_{SAP}(s) = h_{manhattan}(s,g)\ \ and\ \ h_{SAP}(s') = cost(p)$:

כיוון ש- $h_{SAP}(s) = h_{manhattan}(s,g) \to cost(p) \ge h_{manhattan}(s,g)$:

$$h_{SAP}(s) - h_{SAP}(s') = h_{manhattan}(s,g) - cost(p) \le cost(p) - cost(p) = 0$$
$$< minimal\ cost\ of\ an\ edge\ in\ the\ frozen\ lake\ problem \le cost(s,s')$$

3. $h_{SAP}(s) = cost(p)\ \ and\ \ h_{SAP}(s') = h_{manhattan}(s',g)$:

כיוון ש- $h_{SAP}(s) = cost(p) \to cost(p) \le h_{manhattan}(s,g)$:

$$h_{SAP}(s) - h_{SAP}(s') = cost(p) - h_{manhattan}(s',g) \le h_{manhattan}(s,g) - h_{manhattan}(s',g) \le 1$$
$$= minimal\ cost\ of\ an\ edge\ in\ the\ frozen\ lake\ problem \le cost(s,s')$$

4. $h_{SAP}(s) = cost(p)\ \ and\ \ h_{SAP}(s') = cost(p)$:

$$h_{SAP}(s) - h_{SAP}(s') = cost(p) - cost(p) = 0 < minimal\ cost\ of\ an\ edge\ in\ the\ frozen\ lake\ problem$$
$$\le cost(s,s')$$

ולכן לפי הגדרה, h עקבית.

סעיף 7) נכליל את היוריסטיקה לבעיות עם מספר מצבי מטרה על ידי:

$$h_{MSAP}(s) = \min\{h_{Manhatan}(s,g), Cost(p) | g \in G\}$$

שימו לב שבמקרה זה אנחנו לוקחים את המינימום על פני כל צמתי היעד.

האם היוריסטיקה h_{MSAP} קבילה על כל לוח? אם כן הסבר, אם לא הבא קודמה נגדית

נחלק למקרים:

1. המסלול האופטימלי ממצב s למצב המטרה האופטימלי עובר דרך portal:

ולכן מתקיים:

$$h_{SAP}(s) = \min\{h_{manhattan}(s, g\_optimal)\,, cost(p)\} \le cost(p)$$
$$\le cost\ of\ route\ to\ g\_optimal\ which\ passes\ through\ a\ portal \le h^*(s)$$

2. המסלול האופטימלי ממצב s למצב המטרה האופטימלי לא עובר דרך portal:

ולכן מתקיים:

$$h_{SAP}(s) = \min\{h_{manhattan}(s, g_{optimal}), cost(p)\} \le h_{manhattan}(s, g_{optimal})$$
$$= shortest\ route\ from\ s\ to\ g_{optimal} in\ a\ board\ with\ no\ holes\ or\ portals\ with\ cost\ 1\ (minimal) for\ all$$

$$edges \le actual\ optimal\ route\ from\ s\ to\ g\_optimal = h^*(s)$$

כלומר נקבל כי h קבילה לפי הגדרה.

סעיף 8) האם היוריסטיקה h_{MSAP} עקבית על כל לוח? אם כן הסבר, אם לא הבא דוגמה נגדית.

תהיינה $s \in S$, ו- $s' \in Succc(s)$, נחלק למקרים לפי ערך h_{SAP} של כל אחד מהצמתים:

1. $h_{SAP}(s) = h_{manhattan}(s, g\_optimal1)\ \ and\ \ \ h_{SAP}(s') = h_{manhattan}(s', g\_optimal2)$:

כיוון ש-s ו-s' עוקבים לכן מתקיים (לפי הגדרת מרחק מנהאתן וצמתים עוקבים בלוח שלנו):

$$h_{SAP}(s) - h_{SAP}(s') = h_{manhattan}(s, g\_optimal1) - h_{manhattan}(s', g\_optimal2)$$
$$\le h_{manhattan}(s, g\_optimal2) - h_{manhattan}(s, g\_optimal2) \le 1$$
$$= minimal\ cost\ of\ an\ edge\ in\ the\ frozen\ lake\ problem \le cost(s, s')$$

2. $h_{SAP}(s) = h_{manhattan}(s, g\_optimal1)\ \ and\ \ \ h_{SAP}(s') = cost(p)$:

כיוון ש- $h_{SAP}(s) = h_{manhattan}(s, g\_optimal1) \to cost(p) \ge h_{manhattan}(s, g)$:

$$h_{SAP}(s) - h_{SAP}(s') = h_{manhattan}(s, g\_optimal1) - cost(p) \le cost(p) - cost(p) = 0$$
$$< minimal\ cost\ of\ an\ edge\ in\ the\ frozen\ lake\ problem \le cost(s, s')$$

3. $h_{SAP}(s) = cost(p)\ \ and\ \ \ h_{SAP}(s') = h_{manhattan}(s', g\_optimal2)$:

כיוון ש- $h_{SAP}(s) = cost(p) \to cost(p) \le h_{manhattan}(s, g\_optimal2)$:

$$h_{SAP}(s) - h_{SAP}(s') = cost(p) - h_{manhattan}(s', g\_optimal2)$$
$$\le h_{manhattan}(s, g\_optimal2) - h_{manhattan}(s', g\_optimal2) \le 1$$
$$= minimal\ cost\ of\ an\ edge\ in\ the\ frozen\ lake\ problem \le cost(s, s')$$

4. $h_{SAP}(s) = cost(p)\ \ and\ \ \ h_{SAP}(s') = cost(p)$:

$$h_{SAP}(s) - h_{SAP}(s') = cost(p) - cost(p) = 0 < minimal\ cost\ of\ an\ edge\ in\ the\ frozen\ lake\ problem$$
$$\le cost(s, s')$$

ולכן לפי הגדרה, h עקבית.

שאלה 8:

סעיף 1) האם האלגוריתם שלם? האם הוא קביל? כן ולא בהתאם

שלם: כיוון שמרחב החיפוש הנתון הינו סופי ובמקרה הגרוע יצטרך האלגוריתם לעבור על על כל המסלולים האפשריים (שלא מכילים מעגלים) מהמצב ההתחלתי עד למצב הסופי לכן לפי מה שלמדנו בקורס האלגוריתם הוא אכן שלם ומובטח שיחזיר פתרון.

לא קביל: ראינו בקורס שאלגוריתם *Greedy Best First Search* אינו קביל גם אם המרחב סופי, וניתן לראות את זה ע"י הדוגמה הנתונה שבה אלגוריתם *GBFS* מחזיר את המסלול:

$$1 \to 1 \to 1 \to 1 \to 1 \to 1 \to 1 \to 0 \to 0 \to 0 \to 0 \to 0 \to 0 \to 0$$

בעל מחיר 113 , אבל המסלול האופטימלי מבחינת מחיר בלול הנתון הינו בעל מחיר 93.

סעיף 2) תנו יתרון וחיסרון של אלגוריתם Greedy Best first Search לעומת Beam Search.

יתרון: חוסך בזיכרון

חסרון: לא שלם

הסבר: אלגוריתם beam search ואלגוריתם GBFS פועלים בצורה דומה רק אלגוריתם beam search מגביל את רשימת opened (צמתים לפיתוח) ל-K מסוים. בעת חריגה מה-K הזה האלגוריתם "זורק" את הצומת בעל הערך היוריסטי הגדול ביותר מהרשימה.

כיוון שגודל הרשימה מוגבל לכן beam search חוסך בזיכרון ומספר הפיתוחים לעומת GBFS שמפתח את כל הצמתים האפשריים.

מצד שני ייתכן מקרה שבו צומת ש"נזרק" בגלל ערכו היוריסטי הגדול הוא בפועל הצומת היחיד שמוביל לצומת מטרה, ולכן במקרה זה אלגוריתם beam search לא ימצא פתרון לבעיה בכלל.

שאלה 9:

השאלות בחלק זה מתבססות על הלוח "8x8" שמופיע במחברת.

בהינתן $w_1 < w_2 \leq 1$, נסמן את המסלולים המחוזרים על ידי W-A* תחת הפורמולציה $f = g + w \cdot h$ בp_1, p_2 עבור w_1, w_2 בהתאמה. אזי $cost(p_1) < cost(p_2)$עבור:

סעיף a) יוריסטיקה קבילה h. אם כן הסבירו. אם לא, ספקו דוגמה נגדית.
לא נכון, נגדיר למשל: $w1 = 0.3 < w2 = 0.4 \leq 1$, עם $h(s) = h_{MANHATTAN}(s, g)$:

1. למדנו והוכחנו בקורס כי מרחק מנהאתן הינו יוריסטיקה קבילה.
2. אחרי הרצת האלגוריתם עם שני ערכי *w* הנ"ל, קיבלנו את הפלט:

$$p1 = p2 = 1, \to 1 \to 1 \to 1 \to 1 \to 0 \to 1 \to 0 \to 0 \to 1 \to 0 \to 0 \to 0 \to 0$$

$$cost(p1) = cost(p2) = 93$$

כלומר קיבלנו שהטענה לא נכונה.

סעיף ב) יוריסטיקה כללית (לא בהכרח קבילה) h. אם כן הסבירו. אם לא, ספקו דוגמה נגדית.
לא נכון, נגדיר למשל: $w1 = 0.3 < w2 = 0.4 \leq 1$, עם $h(s) = 5$:

1. היוריסטיקה *h* הינה ערך קבוע 5 ובפרט לא קבילה כיוון ש-*h(goal) = 5*
2. אחרי הרצת האלגוריתם עם שני ערכי *w* הנ"ל, קיבלנו את הפלט:

$$p1 = p2 = 1, \to 1 \to 1 \to 1 \to 1 \to 0 \to 1 \to 0 \to 0 \to 1 \to 0 \to 0 \to 0 \to 0$$

$$cost(p1) = cost(p2) = 93$$

כלומר קיבלנו שהטענה לא נכונה.

שאלה 10 :

IDA יש לו סיבוכיות מקום פולי בהשוואה ל A* (אקספו) וחסרון של IDA זה שהוא יותר מורכב לממימוש מ A*, כך שאם יש לנו B (BRANCHING FACTOR) גדול מדי כך שזה יצריך להשתמש בסיבוכיות מקום גדולה אז עדיף להשתמש ב IDA אחרת משתמשים ב A*

שאלה 11 :

1.יתרון של A*-אפסילון זה שאנחנו מקבלים פתרון בצורה מהירה יותר וחסרון זה שאנחנו מתפשרים על איכות הפתרון

2 . הצעה חיוריסטיקה היא פונקציית משקל בין g ל h כך ש $h'(e)=x^* h(e)+(1-x)g(e)$

כמן בWA* כך שעם ה w אנחנו מקבעים את מידת הדגש בין הפונקציות, כלומר אם אנחנו רוצים להישען יותר על חיפוש באיזור צומת ההתחלה (נותנים משקל גדול יותר לg)או שנסתמך יותר על היוריסטחקה

מספר פיתוחים: כפי שציינו קודם אם משתמשים רק בg אז אנחנו מפתחים יותר צמתים קרובים לצומת ההתחלה בניגוד ליוריסטחקה כך זהיא מפתחת בהתאם למשקלים בין g ל h כך שהיא יכולה לפתח צמתים מפוזרים יותר על גרף החיפוש ולפיכך יותר צמתים מ g

מסלול שנבחר: g מבטיח לנו שיחזיר את המסלול האופט שהוא בעל המחיר הכולל הנמוך ביותר (ucs)

יוריסטיקה: לא מבטיח להחזיר לנו את המסלול האופט ביותר, כך שזה תלוי ביוריסטיקות ובמשקל שניתן ביו שתי הפונקציות

עלות המסלול שנבחר : עם g אנחנו יודעים שהעלות היא הנכונה והאופט לעומת עם H זה תלוי W- כמה משקל אנחנו נותנים לשתי הפונקציות ו היוריסטיקה H המקורית

שאלה 12:

מבחינת מחיר UCS הוא המחזיר את המסלול האופטימלי והמחיר הזול ביותר אבל מבחינת מספר הצמתים שפותחו הוא היקר ביותר, אם נשווה עם WA(0.5)* המחיר הוא מינימלי והמסלול שיוחזר יהיה האופט, וכפי שנלמד בגלל ש WA(0.5)* הוא A* הוא חוסך בכמות הצמתים שייפותחו לעומת UCS, עם WA*(0.7) באופן מפתיע הוא חוסך במספר הצמתים שייפותחו בהשוואה עם UCS ו WA(0.5)* וגם איכות הפתרון שהוא המחיר המינימלי והמסלול האופטימלי לא נפגע בניגוד למה שמצופה, שאם אנחנו יותר נשענים על היוריסטיקה אנחנו נוכל לפגוע באיכות הפיתרון, אבל עם WA(0.9)* זה תואם את ציפיותינו, כך שלא מוחזר הפיתרון האופט אבל מספר הצמתים שפותחו קטן יותר משאר האלגוריתמים, אם יכולנו להשתמש ביורסטיקה יותר מיודעת עם האלג WA(0.9)* היינו יכולים לקבל תוצאות יותר מדיוקות.

<u>שאלה 13 :</u>

.1

P(b|a)=3/7

P(c|a)=2/7

P(d|a)=2/7

.2 6 wrong -1

.3 כן no explanation in any section -2

.4 3\7

.5 11/2>=b wrong -2