# מבוא לבינה מלכותית – תרגיל בית 1

## שאלה 1

1.2

נגדיר את בעיית החיפוש באופן הבא:

* *S יהיה tuple בעל 3 איברים (n,d1,d2) כאשר:*
  + *n מייצג את המיקום של המצב בלוח הוא מחושב ע״י הנוסחה כאשר row, col הם השורה והעמודה של המצב בלוח, ובמקרה של הלוח הנתון row\_length=8. (ומתחילים לספור מ-0)*
  + *מייצג האם אספנו את כדור הדרגון בול הראשון שחסר, יקבל את הערך 1 במידה ואספנו, ו0 במידה ולא.*
  + *כנ״ל כמו עבור כדור הדרגון השני שחסר.*
* *O יהיה מוגדר \*(צריך להסביר פה משהו*? )
* *I בלוח הנתון הוא*
* *G בלוח הנתון הוא*

גודל מרחב המצבים S הוא , משום שיש 64 אפשרויות לקורדינטה ועוד 2 אפשרויות לכל כדור דרגון בול, סה״כ .

1.3

הפונקציה Domain תחזיר את הקבוצה הבאה:

מכיוון שלא ניתן להפעיל את הפעולה על כל השורה העליונה בלוח, ובנוסף לא ניתן לבצע שום פעולה מתוך בור, לכן גם התאים שהם בור בלוח הנתון לא נכללים בקבוצה.

1.4

הפונקציה Succ תחזיר את הקבוצה הבאה:

*מכיוון ש Succ מחזירה את כל המצבים שניתן מ S (קיים אופרטור שניתן להפעיל על S ולקבל את המצב החדש),שהם השכן מימין והשכן מלמטה במקרה זה.*

*1.5*

*כן, למשל מכל מצב (שהוא לא הקצה הימני של הלוח או בור) ניתן לבצע פעולה ימינה ולאחר מכן פעולה שמאלה, ובכך לחזור לאותו מצב.*

*1.6*

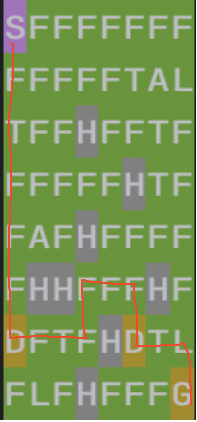
*מקדם הסיעוף בבעיה הוא 4, מכיוון שלכל היותר ניתן להגיע ל4 שכנים מצומת כלשהו (למשל, צומת שהוא לא אחד מקצוות הלוח או בור).*

*1.7*

*במקרה הגרוע ביותר, הסוכן ייתקע במעגל, ולא יסיים את ריצתו לעולם ולכן לא יגיע למצב סופי.*

*1.8  
במקרה הכי טוב הסוכן שלנו יבצע 16 פעולות בלבד. (לבדוק תשובה\*)*

*זאת מכיוון שהמסלול האופטימלי (מבחינת כמות פעולות, ולא עלות) למצב הסופי הוא:*

**

*זאת מפני שכל מסלול אחר ידרוש מאיתנו לעשות עיקופים נוספים.*

*כמות הפעולות במסלול הנ״ל היא: 16. לכן לכל הפחות הסוכן שלנו יצטרך לבצע 16 פעולות.*

*1.9*

*לא, נתבונן בדוגמא נגדית:*

## שאלה 2

2.1 מימשנו

2.2 הדרישה על גרף החיפוש היא שלא יהיו 2 מסלולים שונים מצומת ההתחלה, לכל צומת אחר בגרף. להשוות.

2.3 אחר כך

2.4

בהינתן לוח NxN נבנה את גרף המצבים G ונבצע את הטרנספורמציה הבאה:   
ראשית, נגדיר גרף שהוא:

בהינתן צומת כלשהו ושכן המחשוב ע״י הפונקצייה נגדיר צמתים נוספים בגרף G’, שנסמנם שיש ביניהם קשתות, ולבסוף קשת בין לצומת הראשון שיצרנו וקשת ביןהצומת האחרון שיצרנו לבין בכך למעשה הגדרנו מסלול באורך בין ל-. בנוסף, עבור משבצות מסוג ״H״, לא נגדיר קשתות בכלל בגרף ולמעשה הן יהיו ״בור״ (במונח של גרף).  
נעיר כי אנחנו מניחים כי פעולות על גרף המצבים, בהינתן שעוברים ממשבצת “D” מוגדרות כך שה-State החדש יכיל True עבור כדור הדרגון המתאים לאותה המשבצת. (ללא הנחה זו, צריך לשכפל את הגרף ולהוסיף קשתות בין כל צומת שמכיל “D” לשכפול אחר של הגרף).  
כעת מנכונות BFS נקבל את המסלול הקצר ביותר.

2.5

כמות הצמתים שייוצרו: מכיוון שנצטרך לסרוק את כל הלוח.

כמות הצמתים שיפותחו: מכיוון שבאיטרציה האחרונה, אנחנו נפתח צומת סמוך לצומת המטרה, ונזהה את המטרה ונסיים, לכן בסה״כ פיתחנו את כל המשבצות בלוח חוץ מצומת המטרה, ואחת הצמתים הסמוכים אליה (העליון או השמאלי).

## שאלה 3

3.1 עבור לוח האלגורית שלם אך איננו קביל, מכיוון שהלוח איננו אינסופי, וגם עבור מקרה הקצה של הבור, בגרף החיפוש לא יהיו לו בנים ולכן DFS פשוט יחזור ממנו ללא הצלחה בחיפוש, בסה״כ הDFS יסיים את ריצתו וימצא פתרון, ולכן שלם.

לעומת זאת, האלגוריתם איננו קביל, מכיוון שDFS מחפש לעומק ולכן עלול להחזיר מסלול לצומת המטרה למרות שקיים מסלול יותר קצר.

3.2

כן, האלגוריתם היה מוצא פיתרון והמסלול שיתקבל הוא המסלול שבו האלגוריתם הולך בכל צעד אל הכיוון המועדף ביותר שהוא יכול ללכת אליו (למשל אם הסדר הוא DRUL הוא קודם ינסה ללכת למטה אלא אם כן הוא נתקע בקיר ואז הוא ינסה ללכת ימינה).

3.3

יפותחו: , הסיבה: כפי שהוסבר מקודם, האלגוריתם יתעדף כיוון כלשהו, לכן נוכל להניח בה״כ שהוא ילך ימינה ואז למטה, כלומר יפתח את כל השורה הראשונה והעמודה האחרונה, פרט כמובן לצומת המטרה, בסה״כ (את הפינה הימנית העליונה סופרים רק פעם אחת).

ייווצרו: ל (עבור התשובה 1 כמובן) הסבר: הפעם חושפים גם את השורה הראשונה והשנייה והעמודה האחרונה + אחת לפני האחרונה. פרט לאיבר האחרון בעמודה האחת לפני האחרונה – זאת מכיוון שצומת המטרה לא פותח. סה״כ .

3.4

יפותחו + ייווצרו: הסבר: יפותחו בדיוק כמו מקודם, אך הפעם בBacktracking DFS הצמתים נוצרים ביחד עם הפיתוח ולכן נקבל שכמות הצמתים שנוצרו היא גם .

## שאלה 4

4.1

a.   
כן, נוכיח: (בהנחה שמקדם הסיעוף הוא סופי)  
נניח שקיים פתרון כלשהו ונסמן את אורכו ב –. נתבונן בריצת DFS-L בעומק d.

בריצה זו, DFS יחשוף את כל המסלולים בגרף ממצב התחלה בעומק d לצומת כלשהו.   
מהנחה, קיים מסלול בעומק d מצומת ההתחלה לצומת הפתרון ולכן נמצא את מסלול זה ונחזיר את התשובה בריצה זו.

b.

לא ברור, צריך להבין מה הם רוצים

כן, נוכיח:  
נניח כי עלות כל פעולה היא 1, ונניח שקיים פתרון ובנוסף נסמן את עומקו של הפתרון האופטימלי ב-.  
נניח בשלילה שהאלגוריתם לא קביל,  
הוכחנו בסעיף הקודם שהאלגוריתם שלם, לפיכך האלגוריתם יחזיר פתרון בעומק .  
נתבונן בריצה של עבור ,

מהנחת השלילה, האלגוריתם לא ימצא אף מסלול מצומת התחלה לצומת יעד מאחר ו.

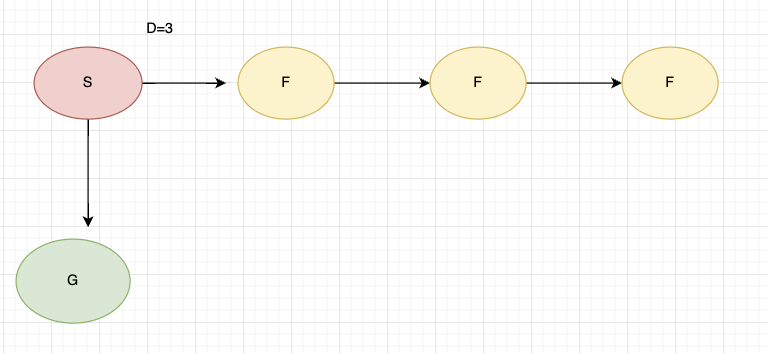
אולם, כן קיים מסלול מצומת ההתחלה לצומת מטרה בעומק וזו סתירה להגדרת מכיוון שהוא עובר על כל המסלולים בעומק .

c.

כנ״ל

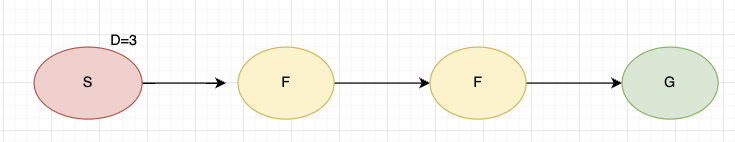
4.2

נתבונן בדוגמא הבאה: a.



נניח בדוגמא הנ״ל כי DFS יתעדף פנייה ימינה על פני פנייה למטה.  
במקרה זה, ID-DFS יחזור אחרי פיתוח של צומת ההתחלה בלבד (כלומר ייצור את צומת המטרה וצומת F אחד), ולעומת זאת עבור , האלגוריתם ReverseDFS יחזור לאחר שפיתח את כל הענף הימני (מלבד פיתוח הצומת האחרון בענף), ולכן במקרה זה ID-DFS עדיף.

נתבונן בדוגמא נוספת:



בדוגמא זו, תבצע שלוש פעמים עבור לעומת עם שיחזור לאחר ריצת עם בלבד.

b.

ניתן לבצע לעדכן את L באופן דומה לחיפוש בינארי:

נאתחל את האלגוריתם עם גבולות ובכל פעם נחפש פיתרון בעומק:   
 . אם מצאנו פתרון נעדכן , אחרת נעדכן .  
נעצור כאשר ונחזיר את הפתרון בעומק הנ״ל.