# 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*

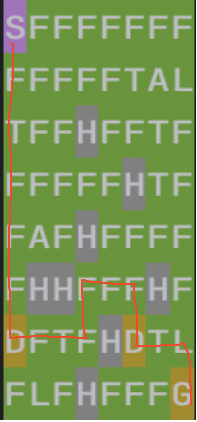
*מקדם הסיעוף בבעיה הוא 4, מכיוון שלכל היותר ניתן להגיע ל4 שכנים מצומת כלשהו (למשל, צומת שהוא לא אחד מקצוות הלוח או בור).*

*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.

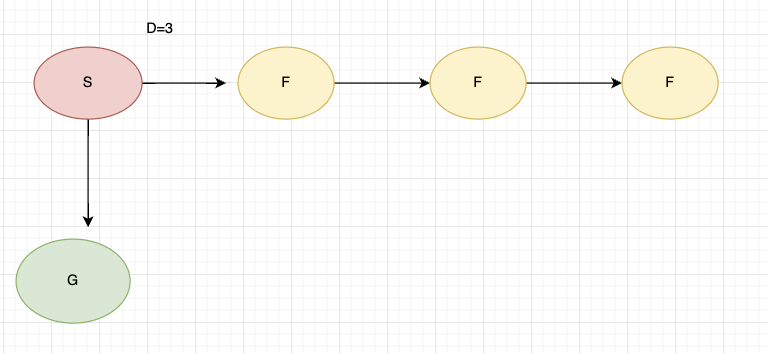
אולם, כן קיים מסלול מצומת ההתחלה לצומת מטרה בעומק וזו סתירה להגדרת מכיוון שהוא עובר על כל המסלולים בעומק .

c.

כנ״ל

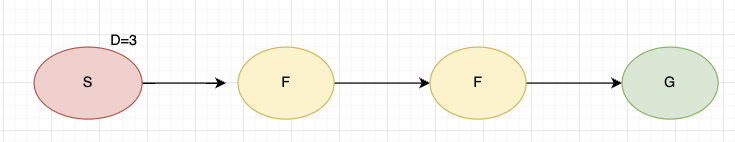
4.2

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.

## שאלה 6

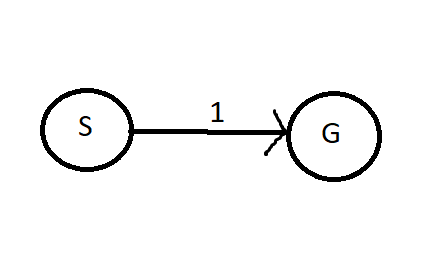
6.1

אלגוריתם UCS ואלגוריתם BFS יפעלו באותו האופן בבעיות חיפוש שבהן לכל הקשתות יש אותו משקל. זאת מכיוון ש UCS כל פעם מוציא מ OPEN את הצומת עם ה g המינימלי ומפתח אותו ולעומת זאת, BFS מפתח כל פעם את הצומת שהמרחק בקשתות ממנו למצב ההתחלתי הוא מינימלי (ניתן לראות את זה בכך שהוא מניח משקל אחיד על הקשתות ומוציא מ OPEN את הצומת עם המרחק המינימלי מהמצב ההתחלתי). בנוסף, נציין שב BFS אנו יודעים שהמסלול אופטימלי כבר כאשר צומת נכנס ל OPEN, ולכן מחזירים מסלול זה בעת יצירת הצומת, לעומת זאת, ב UCS אנו יודעים שהמסלול אופטימלי רק כאשר הצומת נכנס ל CLOSE אך כאשר מחיר הקשתות הוא אחיד, ב UCS ניתן היה להחזיר את המסלול האופטימלי כאשר הצומת נוצר, בדומה ל BFS.

6.2

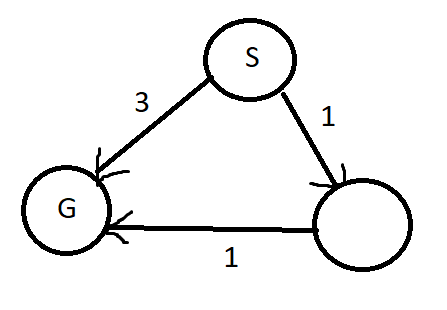
ראינו בתרגול שאלגוריתם UCS הוא שלם, אם פונקציית המחיר חסומה מלמטה ע"י ושבמקרה זה הוא גם קביל. זה מתקיים בבעיית כדורי הדרקון ולכן האלגוריתם שלם וקביל בבעיית כדורי הדרקון.

6.3



בגרף זה, האלגוריתם של שאדי יחזיר את המסלול היחיד בין S ל G שהוא המסלול הקצר ביותר. האלגוריתם ראשית יבדוק ש S לא צומת מטרה, יכניס אותו ל OPEN ואז יפתח אותו ויצור את G ויגלה שהוא צומת מטרה ונחזיר את המסלול.

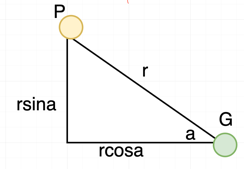
מצד שני, עבור:



בגרף זה, האלגוריתם של שאדי יחזיר מסלול לא אופטימלי. ראשית, הוא יכניס את S ל OPEN לאחר שיגלה שהוא לא צומת מטרה. לאחר מכן, הוא יוציא אותו מ OPEN ויצור את הילדים שלו. בעת יצירת הילד הראשון שהוא G, נגלה שהוא צומת מטרה ונחזיר את המסלול S->G. החזרנו מסלול במשקל 3 למרות שהמסלול הקצר ביותר במשקל 2 (המסלול הקצר ביותר הוא המסלול מ S לצומת הימני ואז G).

## שאלה 7

7.1

היוריסטיקה היא כן -קבילה כאשר ה ההדוק ביותר מתקבל עבור: .   
הסבר: ראשית, נבחין כי היוריסטיקה האופטימלית חסומה מלמטה ע״י המרחק האוקלידי בין 2 צמתים שיש ביניהם כביש. (בהנחה שהעולם שטוח) כלומר .  
בנוסף, נתבונן בציור הבא:

בציור מסומנים שתי נקודות כאשר המרחק האוקלידי ביניהם מסומן ב וסכום ההטלות על הצירים הוא מרחק מנהטן. מתקיים:

* הסבר למעבר: . נשים לב כי מתקיים ולכן המעבר תקין.

בשילוב 2 המסקנות נקבל בסה״כ לכל צומת :

כלומר היוריסטיקה היא – קבילה.

החסם הנ״ל הוא ההדוק ביותר, מכיוון שעבור , נקבל כי ולכן במקרה שבו ניקח קטן יותר, קיימת דוגמא נגדית שבה מרחק מנהטן יהיה גדול יותר. (ההיטלים יצרו משולש 90,45,45).

7.2

היוריסטיקה קבילה עבור

הסבר: במשולש ישר זווית היתר גדול מכל אחד מהניצבים, לכן בשילוב המסקנה מסעיף קודם, (היורסיטקה האופטימלית חסומה מלמטה ע״י המרחק האוקלידי) נקבל אי השוויון:

זהו ה ההדוק ביותר מכיוון שהוא מקבל את הערך הכי קטן שניתן לפי ההגדרה.

7.3

לפי גוגל זה קביל, אבל צריך לבדוק..

7.4

יהי היוריסטיקות קבילות ויהיו האפסילונים ההדוקים ביותר בהתאמה.

נגדיר . נראה כי היא קבילה עבור האפסילון ההדוק

הוכחה:

זהו האפסילון ההדוק ביותר, מכיוון שאם נניח בשלילה שקיים

*אז מתקיים: ,*

*אבל בנוסף מתקיים:   
להמשיך*

*7.5*

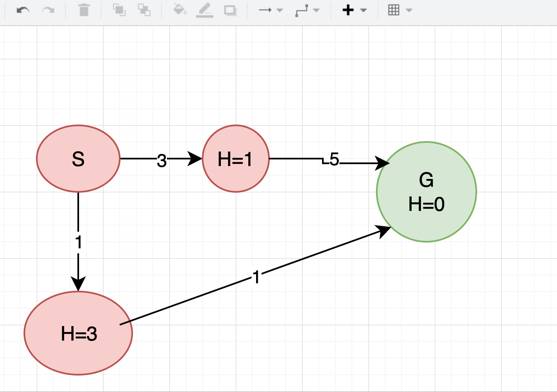
## שאלה 8

8.1

האלגוריתם שלך אך אינו קביל.  
 מכיוון שהמרחב סופי (ניתן לבקר ב 256 מצבים לכל היותר) וקשיר, האלגוריתם שלם (היוריסטיקה קובעת רק את סדר הפיתוח של הצמתים). האלגוריתם אינו קביל, כי ייתכן שהיוריסטיקה תטעה אותנו כך שנגיע למטרה דרך מסלול לא אופטימלי ונחזיר אותו.

למשל בגרף הבא:

הgreedy יבחר לפתח את הצומת עם H=1 ולכן יחזיר מסלול בעלות 8 למרות שהמסלול האופטימלי בעלות 2.



8.2

יתרון: חוסך זיכרון, אנחנו שומרים פחות צמתים, וגם חוסך זמן ריצה כי אנחנו מפתחים גם פחות צמתים.

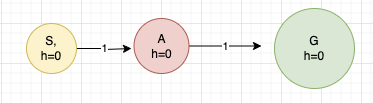
חיסרון: עלולים לא להגיע לפתרון או לא למצוא פתרון אופטימלי, כלומר האלגוריתם לא שלם בניגוד לGreedy.

## שאלה 9

9.1 רטוב

9.2

a. הטענה לא נכונה, נתבונן בדוגמא נגדית פשוטה, עבור היוריסטיקה הטריוויאלית, שהיא כמובן קבילה.



עבור נקבל שהמסלולים המוחזרים יהיו: , כלומר בסתירה לטענה.

b.

אותה דוגמא נגדית עבור הסעיף הקודם תפריך גם את הסעיף הנ״ל. (להשוות, שאלה מוזרה)

## שאלה 10

10.1

יתרון של IDA\* הוא שהוא חוסך בזיכרון כיוון שהוא משתמש ב DFS-f ובכך הוא שומר בכל איטרציה רק את הצמתים לאורך המסלול הנוכחי ואת הבנים שלהם לכן סיבוכיות הזיכרון שלו היא O(bd), לעומת זאת, סיבוכיות הזיכרון של A\* היא O(b^d).

יתרון של A\* הוא שהוא חוסך בזמן ריצה בכך שהוא לא מפתח את העץ מחדש בכל איטרציה, לעומת זאת ב IDA\* אנו מפתחים מחדש את העץ בכל פעם שמגדילים את f-limit.

היינו מעדיפים להשתמש ב A\* כאשר אין מגבלת זיכרון, וב IDA\* כאשר אנו מוגבלים בזיכרון.

10.2

לא

## שאלה 11

11.1 רטוב

11.2

יתרון: ניתן להפעיל היוריסטיקה נוספת ושונה על הקבוצה שלנו, זה יכול להיות שימושי אם ידוע לנו מידע נוסף על העולם, ניתן להגיע לדרך יותר מהירה לפתרון וכך האלגוריתם ירוץ יותר מהר. (האלגוריתם יותר ״מיודע״).

חיסרון: בהינתן שהיוריסטיקה קבילה, חיובית וחסומה מלמטה אז באלגוריתם מובטח לנו שנקבל פתרון אופטימלי, ולעומת זאת באלגוריתם מובטח שנקבל פתרון אופטימלי עד כדי פקטור , כלומר הבטחה פחות חזקה מ .

11.3

לדעתנו, יוריסטיקה שמייצגת בצורה טובה את המרחק אל היעד יכולה להתאים, למשל נבחר את שהוגדרה בתרגיל.

הסיבה לכך, היא שהצומת שיבחר מבין קבוצת הFocal, יהיה הצומת שמקרב אותנו הכי ליעד. (בדר״כ, לא תמיד אבל זה משקף את העולם בצורה טובה) וכתוצאה מכך, האלגוריתם יכול לפעול יותר מהר מ. כמובן שזה עלול לעלות במחיר של לא למצוא את הפתרון האופטימלי ולכן יש פה trade-off.

ברמה הפרקטית, בדקנו את היוריסטיקה אל מול שימוש ב . קיבלנו כי מספר הפיתוחים קטן משמעותית עבור עבודה עם והמסלול שנבחר הוא לא המסלול האופטימלי, עלותו גדולה יותר מהמסלול האופטימלי. המסלול האופטימלי נבחר כאשר השתמשנו ב – כלומר הבחירה לעבוד עם תניב פתרון אופטימלי, אבל בעלות של יותר פיתוח צמתים לעומת היוריסטיקה שהצענו.

11.4

אם למעשה אין חשיבות אלא רק ליורסטיקה של ה. כלומר, האלגוריתם הופך להיות אלגוריתם חיפוש גרידי עבור פונקציית היוריסטיקה .

הסבר: בכל איטרציה, קבוצת ה תהיה למעשה כל הצמתים בopen.

## שאלה 12

12.1 רטוב

12.2 התוצאות שקיבלנו אינן תואמות את הציפיות שלנו.

ראשית נתייחס לריצות השונות של :

אנחנו ציפינו כי ככל ש יעלה אז האלגוריתם יתכנס בצורה יותר מהירה אל היעד, כלומר פחות צמתים יפותחו, אבל איכות הפתרון תרד. לעומת זאת, בתוצאות שקיבלנו נראה שאיכות הפתרון נשארה קבועה עבור כל שהורץ, ודווקא כמות הצמתים שפותחו גדלה ככל שהגדלנו את .

נתייחס כעת לביצועי BFS:

לכאורה, BFS לא לוקח בחשבון את עלות המסלולים, ולכן ציפינו שיחזיר תוצאות רעות במיוחד. אולם, הופתענו לגלות שהBFS החזיר תוצאות סבירות, כמובן שלא אופטימליות אך לא רחוק במיוחד מהעלות האופטימלית.

## שאלה 13

13.1

נשים לב ש ולכן לפיכך נקבל

13.2

תשובה: עבור המסלול הארוך ביותר הוא באורך 3, ועבור המסלול האורך ביותר באורך 2. הסבר:

אם נניח כי , מספר הצעדים המקסימלי שהאלגוריתם יכול לבצע הוא 3, במקרה זה הוא ילך במסלול A->B->F->G כי זה המסלול הארוך ביותר האפשרי וכל מצב משפר את המצב הקודם.

אם נניח כי אז לא משפר במסלול זה ולכן האלגוריתם יעצור ב F. (ייתכנו מסלולים אחרים עם שלושה צמתים, אבל לא יהיה מסלול ארוך יותר) ולכן במקרה זה מספר הצעדים המקסימלי שהאלגוריתם יכול לבצע הוא 2.

13.3

אם בצעד הראשון האלגוריתם עבר למצב c, הוא לא יתכנס למקסימום הגלובאלי כיוון שמהצומת c נלך לצומת h כי הוא הצומת היחיד שאפשר להגיע אליו מ c והוא צומת משפר, לאחר מכן אפשר רק לחזור ל c אבל זה לא ישפר ולכן נעצור ב h ונקבל ערך 3. ניתן לראות שזה לא המקסימום הגלובאלי כי יש צומת עם ערך 4.

13.4

13.4

אם אז

אם אז

הסברים:

במקרה הראשון, המסלולים הלא אופטימליים יהיו A->B->G, A->C->H, A->D

עבור המסלול A->B->G נקבל

עבור המסלול A->C->H נקבל

עבור המסלול A->D נקבל

לכן בסה״כ נקבל

במקרה השני, המסלולים הלא אופטימליים יהיו A->C->H,A->D (בכל מקרה שאנו עוברים מ A ל B אנו נתכנס לפתרון אופטימלי, כי אם ואנו עוברים מ A ל B, נלך במסלול A->B->F או A->B->G ובמקרה שבו ואנו עוברים מ A ל B, נלך במסלול A->B->G או A->B->F->G)

עבור המסלול A->C->H נקבל

עבור המסלול A->D נקבל

לכן בסך הכל נקבל

13.5

לאף ערך .

על מנת להגיע למקסימום הגלובאלי תוך בדיוק 3 צעדים, אנו חייבים ללכת על המסלול A->B->F->G (מסלולים אחרים שהאלגוריתם הולך אליהם בהכרח קצרים יותר, זאת מכיוון שמהקודקוד H לא ניתן להמשיך כי אין שיפור)

כדי ללכת על המסלול A->B->F->G חייב להתקיים כי כדי להגיע מ F ל G הכרחי ש G ישפר את F. נקבל

נדרוש שזה יהיה גדול מ ונקבל

כלומר, קיבלנו וגם ולפיכך לא קיים כזה.