**מבוא לבינה מלאכותית - 236501**

**תרגיל בית 1**

**מרחבי חיפוש**

מטרות התרגיל

· נתמודד עם בעיות פרקטיות ותיאורטיות של חיפוש במרחבי מצבים.

· נתרגל את הנלמד בהרצאות ובתרגולים.

· נתנסה בתכנות ב-python לפתרון בעיות פרקטיות.

הנחיות כלליות

· **תאריך הגשה** 2/3 **יום שבת**, בשעה 23:59.)לשנות מועד ולפתוח פיאצה)

· את המטלה יש להגיש **בזוגות בלבד**.

· יש להגיש מטלות מוקלדות בלבד בעברית או באנגלית. פתרונות בכתב יד לא ייבדקו.

· ניתן לשלוח שאלות בנוגע לתרגיל [**בפיאצה**](https://piazza.com/technion.ac.il/winter2024/236501/home) **בלבד**.

· המתרגל האחראית על תרגיל: **שאדי דיאב** .

· בקשות דחיה **מוצדקות** (מילואים, אשפוז וכו') יש לשלוח למתרגל האחראי (**ספיר טובול**) בלבד.

· במהלך התרגיל ייתכן שנעלה עדכונים, למסמך הנ"ל – תפורסם הודעה בהתאם.

· העדכונים הינם **מחייבים**, ועליכם להתעדכן עד מועד הגשת התרגיל.

· שימו לב, התרגיל מהווה כ- 15% מהציון הסופי במקצוע **ולכן העתקות תטופלנה בחומרה!**

· ציון המטלה יורכב מהגורמים הבאים:

o **65% - המסמך היבש.**

o **35% - הקוד המוגש.**

· אנו יודעים שעבור חלקכם זו התנסות ראשונה בכתיבת קוד בפיתון ותרגיל זה מתוכנן בהתאם לכך.

· שימו לב שלא יענו שאלות בסגנון: ״איך מוצאים את עלות הפתרון שהוחזר?״ / ״איך ניגשים למפות הכבישים מתוך המימוש של הפונק׳ ההיא?״ / ״באיזה שדה שמור ה...?״ וכדומה.

· אנחנו רוצים לעודד אתכם לעיין בקוד ולמצוא פרטים אלו בכוחות עצמכם. הכרת סביבת העבודה שסיפקנו לכם והתמצאות בה הן למעשה חלק מהתרגיל.

· בתרגילי הבית בקורס הרצת הניסויים עשויה לקחת זמן רב. לכן מומלץ מאוד להימנע מדחיית העבודה על התרגיל ו/או כתיבת הדו״ח לרגע האחרון. לא תינתנּה דחיות על רקע זה.

· מסמך זה כתוב בלשון זכר מטעמי נוחות בלבד, אך מתייחס לנשים וגברים כאחד.

אנחנו קשובים לפניות שלכם במהלך התרגיל ומעדכנים את המסמך הזה בהתאם. גרסאות עדכניות של המסמך יועלו לאתר. הבהרות ועדכונים שנוספים אחרי הפרסום הראשוני יסומנו כאן בצהוב. בנוסף, לכל עדכון יהיה מספר גרסה כדי שתוכלו לעקוב. ייתכן שתפורסמנה גרסאות רבות – אל תיבהלו מכך. השינויים בכל גרסה יכולים להיות קטנים.

**הנחיות לחלק היבש**

1. ככלל אצבע, בהינתן שאלה ראשית ספקו את התשובה המיידית ולאחר מכן תרחיבו ותסבירו. למשל, אם שואלים מה סיבוכיות הזמן של אלגוריתם BFS תשובה תהיה ״, מכיוון שבקרה הכי גרוע נאחסן את כל עץ החיפוש של הבעיה בCLOSE״.

**הנחיות לחלק הרטוב**

1. אנו מעודדים אתכם לעבור על הקבצים המצורפים ולהבין כיצד הסביבה בנויה ובאילו פונקציות תוכלו להשתמש במימוש שלכם.
2. הקוד שלכם ייבדק בקפדנות על ידי טסטים. הטסטים יבדקו את הפתרונות המוחזרים על ידי האלגוריתמים שלכם אל מול המימוש שלנו על פני בעיות שונות. אנו מצפים ממכם (אלא אם צוין אחרת) להחזיר את אותם ערכים בדיוק. אנחנו נבדוק את המסלול המוחזר, מספר הצמתים שפתחו ואת עלות הפתרון המוחזר. הטסטים יהיו מוגבלים בזמן אך תקבלו זמן גדול מאוד לכל טסט.
3. ספקו קוד ברור ונקי הניתן לבדיקה ידנית.

## **מבוא ורקע**

התרגיל מתפרש על פני מסמך זה והמחברת המצורפת. מומלץ לענות על השאלות לפי הסדר במסמך זה.

במטלה זו נעסוק בהפעלת אלגוריתמי חיפוש על מרחבי מצבים לבעיות ניווט. מומלץ לחזור על שקפי ההרצאות והתרגולים הרלוונטיים לפני תחילת העבודה על התרגיל.

### סיפור מסגרת

לקאקרוטו וגוהאן יש 5 כדורי דרקון וחסר להם שני כדורים, והם ממש צריכים אותם כדי להזמין הדרקון שן-ראן ולבקש ממנו להחזיר את החברים שלהם לחיים, לכן הם הלכו לכוכב לכת נאמיק כדי לחפש כדורי הדרקון, קאקרוטו הציע שיחפשו על הכדור דרך ה ג׳י.פי.אס שלהם אבל גוהאן מסביר לקאקרוטו שיש לו חברים שלוקחים הסמסטר את קורס ״מבוא לבינה מלאכותית״. גוהאן מבקש ממכם לעזור לו לתכנן את המסלול הטוב ביותר כדי לאסוף כדורי הדרקון ולהגיע לקאקרוטו שמחקה לו.

A cartoon of a person and a child

Description automatically generated

### שאלה 1 – מבוא (8 נק׳):

השאלות בחלק זה מתבססות על הלוח “8x8” שמופיע במחברת אלא אם נכתב אחרת:



* 1. **רטוב**: עברו על המחברת עד שאתם מגיעים לחלק של BFS-G ועצרו שם.
  2. יבש (1 נק׳): תחילה נרצה להגדיר את מרחב החיפוש כפי שנלמד בתרגול. הגדר את עבור סביבת כדורי הדרקון. כאשר זה מרחב המצבים, , זה מרחב האופרטורים, , זה המצב ההתחלתי ו הוא קבוצת מצבי המטרה. מה גודל מרחב המצבים S? הסבירו.

S: every state in the shape of (0-63,0-1,0-1), for there are 64 tiles possible with 2 dragonballs for every tile.

|S| = ncol\*nrow\*binaryball1\*binaryball2=8x8x2x2=256

There are 64 indices within each index there is a state which has no DragonBalls, one of the Dragon Balls respectively or two of them.

O: from a tile t a neighboring tile the operations are {DOWN, RIGHT, UP, LEFT}.

I: the initial state (0,0,0) as in top left corner.

G: Goal state (63,1,1) as in bottom right corner with both of the dragonballs collected.

* 1. יבש (1 נק׳): מה תחזיר לנו הפונקציה Domain על אופרטור 2 (UP)?

Returns S. (all the states) given that an up operation on a tile in the first row for example doesn’t move the agent and is still a legal move.

* 1. יבש (1 נק׳): מה תחזיר לנו הפונקציה Succ על המצב ההתחלתי 0?

It returns the set of states which can be accessed with every operator, for example:

Down->(8,false,false)

Right->(1,false,false)

Up->(0,false,false)

Left->(0,false,false)

* 1. יבש (1 נק׳): האם קיימים מעגלים במרחב החיפוש שלנו?

Yes, it is possible to reach a field of F values in which no direction has a preference which may cause circles. For example, indices (1,2,3,9,10,11,17,18).

* 1. יבש (1 נק׳): מה הוא מקדם הסיעוף בבעיה?

**BF =** 4**,** as there are a maximum of four available moves for our agent or available nodes to open.

* 1. יבש (1 נק׳): במקרה הגרוע ביותר, כמה פעולות ידרשו לסוכן כללי להגיע למצב הסופי?

Infinite number of moves, as stated above there are circles in our search space.

* 1. יבש (1 נק׳): במקרה הטוב ביותר, כמה פעולות ידרשו לסוכן כללי להגיע למצב הסופי?

14 moves according to the table above using the Manhattan distance without collecting the balls.

16 is the optimal including collecting the balls. First the ball at index 48 (6 steps) then second ball at 53 going around the hole (7 steps) and finally to the goal at 63 (3 steps).

* 1. יבש (1 נק׳): עבור לוח כללי, המסלול הקל ביותר הוא המסלול שמגיע למצב מטרה שהכי קרוב למצב ההתחלתי (במונחים של manhattan distance)? אם כן, הוכיחו. אם לא, ספקו דוגמא נגדית.

No for this example:

S F D D F G

A A A A A A

Manhattan = 2\*10+2\*1+1 = 23

Optimal = 6\*1 + 1 + 1 = 9

### שאלה 2 – Breadth First Search-G (7 נק׳):

השאלות בחלק זה מתבססות על הלוח “8x8” שמופיע במחברת אלא אם נכתב אחרת.

* 1. **רטוב**: ממשו את אלג׳ BFS-G (על גרף) במחברת ע״פ ההנחיות המופיעות שם.
  2. יבש (1 נק׳): מה צריך להיות התנאי על גרף החיפוש (לא בהכרח בבעיית כדורי הדרקון) כך שBFS על גרף ו-BFS על עץ ייצרו ויפתחו צמתים זהים באותו הסדר?

When the graph is acyclic, and is limited to only one pass on any specific node, then we get the same open/close sequences with both the tree and the graph.

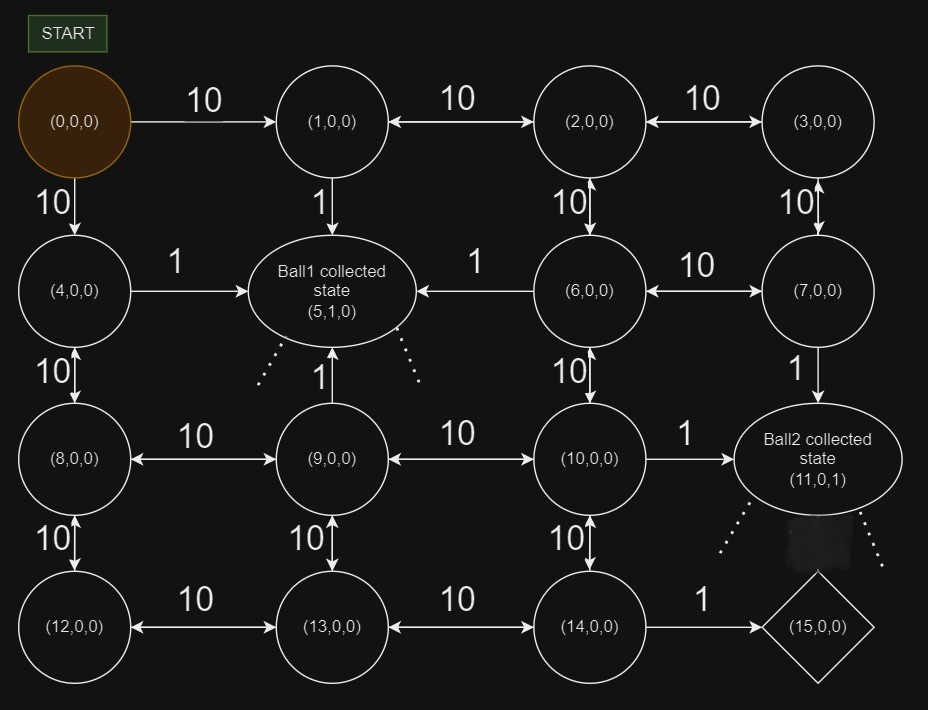
* 1. יבש (2 נק׳): עבור הלוח“4x4” שמופיע במחברת, ציירו את גרף המצבים.

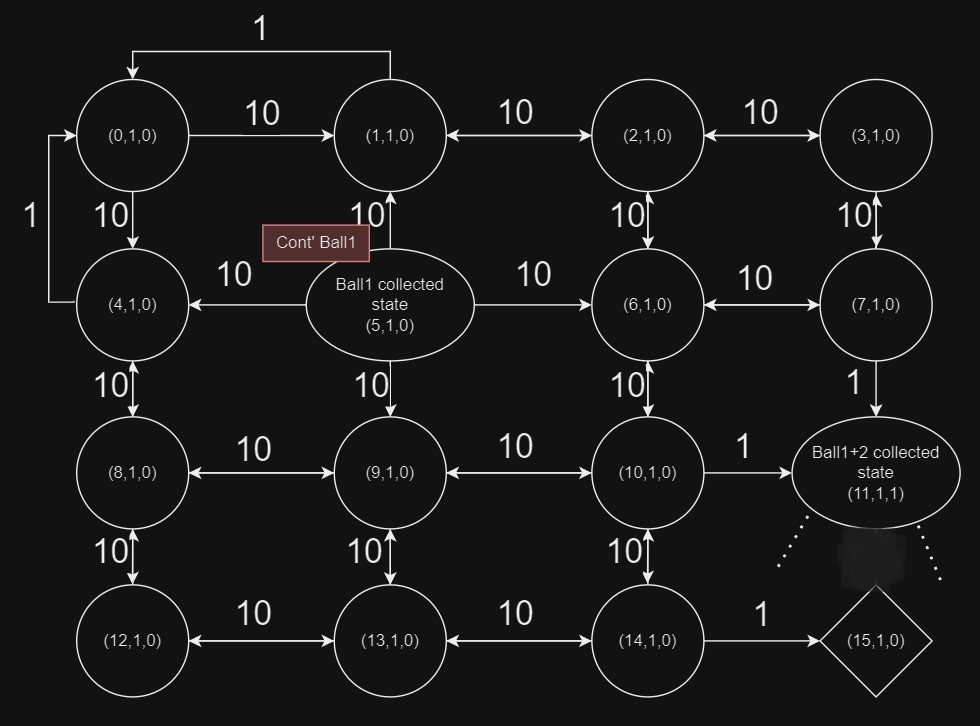
SFFF

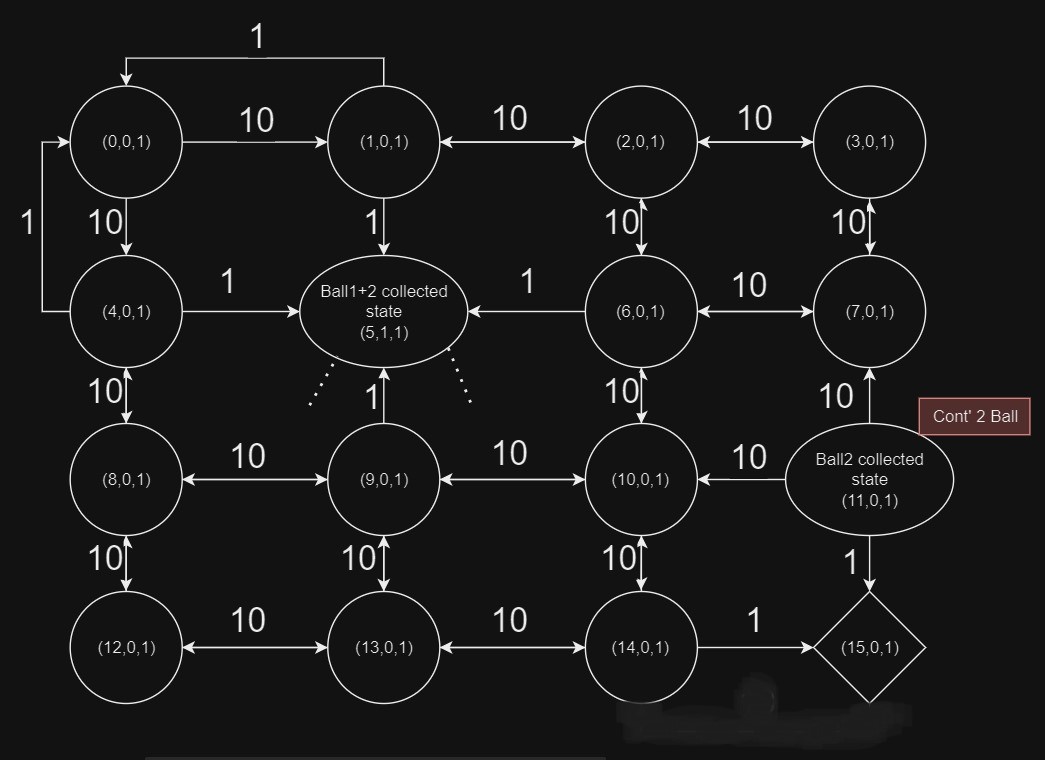
FDFF

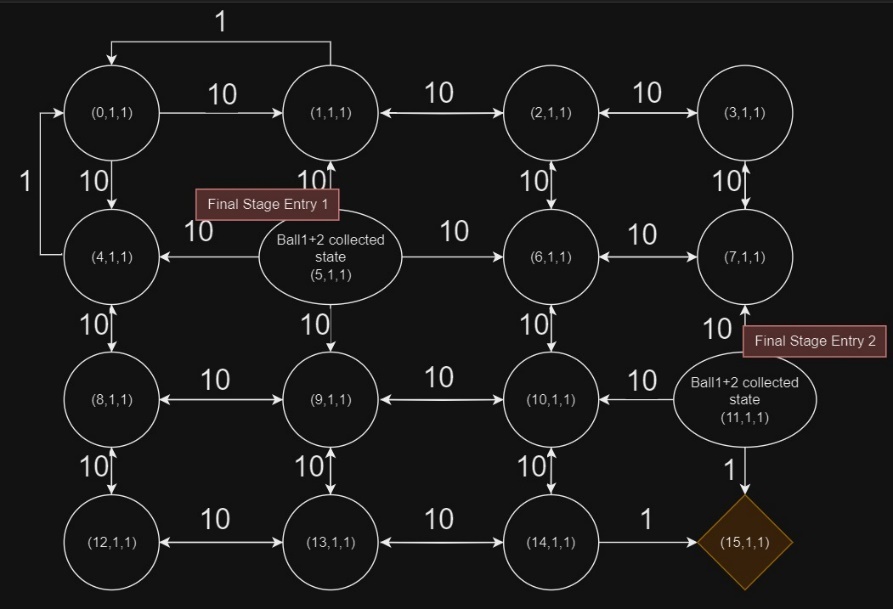
FFFD

FFFG







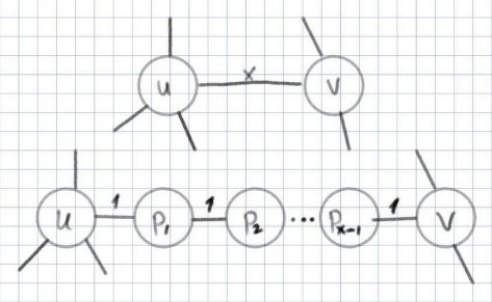


* 1. יבש (2 נק׳): נתון לוח בגודל NxN. הציעו דרך להשתמש באלגוריתם BFS-G כך שיחזיר פתרון אופטימלי (עלות מינימלית) והסבירו.
* רמז: עליכם לספק פונקציה המקבלת את גרף המצבים ויוצרת גרף חדש ובעזרתה למצוא את המסלול האופטימלי בגרף .

ננצל את העובדה שאלגוריתם BFS הוא קביל כאשר משקל הקשתות הוא אחיד.

לכן נגדיר שתהפוך את הגרף המקורי שמשקל כל קשת תלוי בצומת אליה עוברים (למשל, אם צומתv הוא בעל מצב F אזי משקל הקשת (u,v) יהיה 10) לגרף שמשקל כל הקשתות בו שווים באופן הבא:

לכל שני צמתים u,v שמחוברים בקשת (u,v) בעל משקל X בגרף , נחליף בסדרת צמתים מהצורה הבאה:



כך שלכל צומת pi אין קשתות נוספות חוץ מהקשתות שציינו, ולכן מאובטח ששמרנו על נכונות המעברים בגרף, ואז אלגוריתם BFS יעבור בין u ל- v ב-X שלבים במקום שלב אחד.

כעת אם נריץ BFS על הגרף החדש, האלגוריתם יחזיר את המסלול האופטימלי שמגיע לצומת המטרה.

* 1. יבש (2 נק׳): נתון לוח בגודל NxN, ללא חורים, המכיל 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?

יפותחו: N2-1 ייווצרו: N2

We can take a look at the first steps of the search:

* At first, we discover the start node
* Then we expand it and discover two nodes
* Then we expand the two neighbors and discover three nodes

We can see that each time we discover we discover a diagonal of these nodes, starting from the start to the main diagonal to the goal, thus resulting in discovering and expanding of all the nodes of the graph, apart from the goal node that gets discovered by one of his neighbors but doesn’t get expanded.

### שאלה 3 – Depth First Search-G (6 נק׳):

1. יבש (1 נק׳): עבור בעיית כדורי הדרקון עם לוח NxN, האם האלגוריתם שלם? האם הוא קביל?

**כן שלם לא קביל**

האלגוריתם שלם כי מספר הצמתים בגרף הוא סופי ואין מעגלים ולכן מאובטח שיגיע בחיפוש לצומת מטרה.

האלגוריתם לא קביל כי הפתרון שנמצא יהיה תלוי בסדר בחירת הצומת הבא לפיתוח במהלך החיפוש, ולא באיכות הפתרון.

1. יבש (1 נק׳): האם אלגוריתם DFS (על עץ), עבור בעיית כדורי הדרקון על לוח NxN, היה מוצא פתרון כלשהו? אם כן, מה המסלול שיתקבל? אם לא, כיצד האלגוריתם היה פועל?

ייתכן תחת תנאים מסוימים אבל לא מאובטח שימצא פתרון.

הדבר החשוב פה הוא זה שעכשיו מותר לו לחזור לאותו מצב, בניגוד לחיפוש DFS בגרף.

כיוון שבכל שלב בחיפוש, אלגוריתם DFS בוחר בצומת הבא לפיתוח באופן זהה, ( למשל, תמיד הבן השמאלי קודם) , אזי בעת התחלתו מצומת התחלה יהיה רק ארבע אפשרויות להמשך המשחק:

* או שינסה לחפש למעלה או שמאלה, ואז יישאר במקום, וייתקע בלולאה אינסופית, ולא ימצא פתרון.
* או שיחפש ימינה או למטה, ואז ימשיך בקו ישר N פעמים עד שיגיע לצד שני של הלוח או למטה בהתאם, ואם לא יגיע לפתרון בדרכו, אזי ייתקע בלולאה אינסופית ולא ימצא פתרון.

אזי רק אם קיים פתרון שהוא קו ישר ימינה/למטה שאורכו עד N, שנמצא עליו כל כדורי הדרקון וגם מצב הסיום, ושהוא יעדיף ללכת בכיוון זה, ורק אז הוא ימצא פתרון

1. יבש (2 נק׳): נתון לוח בגודל NxN, ללא חורים, המכיל משבצות רגילות (F,T,A,L) מצב התחלתי בפינה השמאלית עליונה ומצב מטרה בפינה הימנית תחתונה (תניחו כי שני כדורי הדרקון הם בפינה ימינית תחתונה) . כמה צמתים יפותחו וייווצרו במהלך חיפוש  DFS-G? הסבירו?

מקרה אופטימלי: יפותחו: 2N-2 ייווצרו: 4N-5

מקרה גרוע ביותר Nזוגי : יפותחו: N2-N ייווצרו: N2

מקרה גרוע ביותר Nאי-זוגי : יפותחו:N2-2N+1 ייווצרו: N2-N+1

כיוון ששני כדורי הדרקון נמצאים על מצב המתרה, אזי מספיק פשוט להגיע למצב המתרה כדי לנצח.

התשובה תלויה לגמרי בסדר העדפות האלגוריתם בבחירת הכיוון אליו יתקדם לפתח.

מקרה אופטימלי יתקבל כאשר האלגוריתם יעדיף את ימין ומטה על שמאל ומעלה, וכך הוא ייקח 1-N צעדים ימינה ואז 1-N צעדים מטה (או בסדר הפוך) ויגיע למטרה.

מקרה גרוע ביותר יתקבל כאשר הסדר הוא למשל:

למעלה->ימינה->שמאלה->למטה או שמאלה-> למטה->למעלה->ימינה

ובשני המקרים נקבל סריקה מסודרת של שורות או עבודות הגרף.

1. יבש (2 נק׳): נתון לוח בגודל NxN, ללא חורים, המכיל משבצות רגילות (F,T,A,L) מצב התחלתי בפינה השמאלית עליונה ומצב מטרה בפינה הימנית תחתונה (תניחו כי שני כדורי הדרקון הם בפינה ימינית תחתונה). כמה צמתים יפותחו וייווצרו במהלך חיפוש backtracking DFS-G? הסבירו?

מקרה אופטימלי: יפותחו: 2N-2 ייווצרו: 2N-1

מקרה גרוע ביותר Nזוגי : יפותחו: N2-N ייווצרו: N2-N+1

מקרה גרוע ביותר Nאי-זוגי : יפותחו:N2-1 ייווצרו: N2

כמו בסעיף קודם, התשובה תלויה לגמרי בסדר העדפות האלגוריתם בבחירת הכיוון אליו יתקדם לפתח.

וסדרי הבחירה שמובילים למקרים אלו הם גם כמו סעיף קודם.

התשובה קצת שונה בגלל lazy-expand של אלגוריתם זה, שגורם להתנהגות שונה, שמייעלת את המקרה האופטימלי, כי כעת יחסוך את יצירת הצמתים המיותרים במסלול, אבל פוגעת במקרה הגרוע ביותר, כי אז במקום לחסוך את השורה האחרונה הוא יצטרך לעבור על כל הלוח לפני שיגיע למטרה.

### שאלה 4 – ID-DFS (6 נק׳):

* 1. (1 נק׳) האם האלגוריתם שלם? אם כן, הוכיחו. אם לא, ספקו דוגמה נגדית.

כיוון שיש מספר סופי של צמתים, אזי אם יש פתרון בגודל X כלשהוא היינו יכולים למצוא באמצעות DFS כי הוא שלם עבור גרף סופי בלי מעגלים. ולכן נניח ש- DFS-ID לא מצא פתרון לכל X>L, אזי בהכרח ימצא את הפתרון עבור X=L כי הוכחנו שיש הרצת DFS באורך X שתגיע לפתרון.

* 1. (1 נק׳) נניח כי עלות כל פעולה היא 1, האם האלגוריתם קביל? אם כן, הוכיחו. אם לא, הסבירו**.**

האלגוריתם קביל.

האלגוריתם שלם, נניח בשלילה שהוא **לא** קביל, כלומר ייתכן שעבור שני פתרונות באורך X,Y כאשר Y>X, האלגוריתם יחזיר את הפתרון Y, כלומר עבור הרצת L-DFS עם X=L הוא לא מצא את הפתרון שבאורך X וזו סתירה כי L-DFS שלם כאשר קיים פתרון בעומק קטן או שווה ל-L.

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

**function ReverseDFS (*problem, D*):**

L D

result failure

**While** Not Interrupted:

*new\_result DFS-L (problem, L)*

*if new\_result = failure:*

*break*

L L - 1

result new\_result

**return** result

1. בשאלות הבאות הניחו כי יש מספיק זמן לסיום האיטרציה הראשונה.
   1. (1 נק׳) ספקו דוגמה בה ReverseDFS עדיף על ID-DFS ודוגמה בה ID-DFS עדיף על ReverseDFS. הדוגמאות יכולות להיות כלליות ולא בהכרח מסביבת התרגיל.

דוגמה בה ReverseDFS עדיף על ID-DFS:

SFFF

FFFF

FFFF

FFDG

בדוגמה הזו ייתכן ש- ReverseDFS ימצא את מצב המטרה ישירות, ו-DFS-ID ייקח לו משמעותית יותר זמן להגיע

ל- 6=L ולמצוא את מצב המטרה.

ודוגמה בה ID-DFS עדיף על ReverseDFS:

SGFF

DFFF

FFFF

FFFF

בדוגמה הזו מוחזר חסם לדוגמה 6=D כי מובטח לנו שהוא חסם עליון אבל לא הכי טוב שיש ולכן ייתכן ש- ReverseDFS ייקח לא זמן יותר עד שימצא הפתרון למרות שהוא קרוב יותר לנקודת ההתחלה כי עובדים עם חיפוש לעומק, אבל DFS-ID ימצא את הפתרון האופטימלי מהיר יותר.

* 1. (2 נק׳) הציעו כיצד ניצן לייעל את האלגוריתם. רמז: האם אתם יכולים לחשוב על צעד עדכון עדיף לL?

נשנה את האלגוריתם בצורה הבאה :

ניתן לחסוך הרצות L-DFS על ידי הורדת L בקפיצות גדולות יותר למשל 2 במקום 1, ורק בהרצה אחרונה שלא נמצא פתרון, האלגוריתם בודק פעם נוספת את הרצת L-DFS שפספס:

function ReverseDFS( problem , D)

L <- D

Result <- failure

While not interrupted

new\_result <- DFS\_L( problem , L)

if new\_result = failure

L<-L+1

new\_result <- DFS\_L( problem , L)

if new\_result != failure

result <- new\_result

break

L<-L-2

result <- new\_result

return result

### שאלה 6 - UCS (4 נק׳):

השאלות בחלק זה מתבססות על הלוח “8x8” שמופיע במחברת אלא אם נכתב אחרת.

1. יבש (1 נק׳): עבור אילו בעיות חיפוש אלגוריתם UCS ואלגוריתם BFS יפעלו באותו האופן? הסבירו.

שני האלגוריתמים יפעלו באותו אופן כאשר ערך כל הקשתות הוא זהה,

כי אז BFS יפעל כרגיל ויוצר צמתים בצורה שכבתית, וכיוון שערך הקשתות זהה, אזי אלגוריתם UCS גם יוצר צמתים בצורה שכבתית, כי תמיד המחיר של הצמתים באותה שכבה יהיה זהה, ובפרט קטן ממחיר כל צומת בשכבה הבאה, ולכן יעדיף לייצר כל הצמתים בשכבה מסוימת לפני צמתי השכבה הבאה.

1. יבש (1 נק׳): האם בבעיית החיפוש שלנו, עבור לוח NxN, האלגוריתם הוא שלם? האם הוא קביל?

**כן שלם כן קביל**

האלגוריתם שלם כי מספר המצבים סופי ואין מעגלים ומחיר הקשתות הוא חסום מלמטה על ידי 1, ולכן יעבור על כל המצבים עד שימצא פתרון, ולכן מאובטח שימצא פתרון אם קיים.

האלגוריתם קביל אם בנוסף לתנאים שציינו של שלמות, משקל הקשתות הוא בהתאם למחיר הצעד כפי שמוגדר בבעיה, ורק אז אפשר להוכיח שאם יש שני פתרונות באורך X,Y כאשר Y>X , אזי בהכרח מסלול החיפוש לא בוחר את הפתרון שבאורך Y כי זה סותר את אופן פעילות UCS שהוא בוחר בצעד שמניב את המחיר הכי נמוך ( במקרה זה X) , ולכן לא ידלג על X ויבחר אותו כפתרון אופטימלי.

1. יבש (2 נק׳): שאדי טעה במימוש של אלגוריתם UCS ובטעות בדק בעת יצירת הצומת האם היא צומת מטרה במקום בפיתוח שלה. הביאו דוגמה לגרף חיפוש שעבורו שאדי יחזיר בכל זאת את המסלול הקל ביותר, ודוגמה לגרף חיפוש שעבורו שאדי 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 כדור הדרקון. אתם יכולים לתת דוגמה לגרף שמייצג בעיית חיפוש אחרת. הגרף צריך להכיל קשתות מכוונות ואת העלות של כל קשת.

A diagram of a diagram

Description automatically generated

* הגרף העליון הוא גרף חיפוש שעבורו שאדי יחזיר בכל זאת את המסלול הקל ביותר :

המסלול הוא כזה, יתחיל מ- START ואז יגלה 1N ו- 2N , ואז יבחר אחד מהם לפיתוח, ואז יגלה GOAL, וכך יקבל את המסלול הקל ביותר.

* הגרף התחתון הוא גרף חיפוש שעבורו שאדי לא יחזיר את המסלול הקל ביותר :

המסלול הוא כזה, יתחיל מ- START ואז יגלה 1N ו- 2N, ואז יבחר לפתח את 1N כי המחיר שלו הוא מינימלי ברשימה OPEN, ואז יוצר את צומת GOAL, ולפני שהוא יפתח אותו הוא ישירות מחזיר את המסלול (START->N1->GOAL) שזה לא המסלול הכי קצר כי ברור שהמסלול השני יותר קצר.

### שאלה 7 - יוריסטיקות (8 נק׳):

יהי מרחב חיפוש (G,I,O,S) , נסתכל על בעיית הניווט לכדור דרקון יחיד. המטרה היא למצוא מסלול זול ביותר מהמוצא ליעד יחיד G . פונק׳ העלות מוגדרת כאורך הכביש המחבר בין שתי נקודות, יהיה חסם תחתון  על אורך הכבישים .ניתן להניח כי העולם שטוח . מלבד זאת, לא ניתן להניח דבר נוסף על מרחב החיפוש.

הגדרה : יוריסטיקה h היא ε-קבילה אם קיים כך שלכל מצב מתקיים .

נזכיר כי הינה פונקציית המחיר המסלול האופטימאלי מ-s לצומת היעד .

עבור כל אחת מהיוריסטיקות הבאות קבעו האם קיים כך שהיוריסטיקה תהיה ε-קבילה . אם כן מצאו את ה-ε ההדוק ביותר המקיים את זאת. נמקו היטב .

1. יבש (1 נק׳): מרחק מנהטן :

בהנחה שמותר צעדים באלכסון ולכל כיוון, ובהנחה שאורך פתרון אופטימלי חסום מלמטה על ידי המרחק האווירי בין נקודת התחלה וסיום.

לפי תכונות מתמטיות אנו יודעים כי מרחק מנהטן חסום מלמעלה על ידי המרחק האוקלידי (מרחק אווירי) כפול .

נסמן מרחק אוקלידי ב- heuclidian(p) ונקבל:

heuclidian(p)

ובנוסף אנו יודעים ש- חסום מלמטה על ידי המרחק האוקלידי, ולכן נקבל :

heuclidian(p)

heuclidian(p)

מחיבור שתי המשוואות נקבל:

hMD(p) heuclidian(p)

hMD(p)

ולכן מקיים את הדרישה שעבורו היוריסטיקה -קבילה.

1. יבש (1 נק׳):

ידוע לנו כי הפתרון האופטימלי חסום מלמטה על ידי המרחק האוקלידי (אווירי),

וידוע לנו מגאומטריה אוקלידית כי מרחק אודלידי בין שתי נקודות גדול או שווה להפרש בערך קוארדינטה X\Y, כלומר:

heuclidian(p) >= max{Gx-Px , Gy-Py}

ולכן עבור מתקיים:

h(p) =< heuclidian(p) =< h\*(p) = \* h\*(p)

h(p) =< \* h\*(p)

1. יבש (1 נק׳): :

ידוע לנו לפי תכונות אלגבריות כי נורמה L3 קטנה או שווה לנורמה L2 (מרחק אוקלידי) **במרחב דו ממדי,**

כלומר :

h(p) =< heuclidian(p)

ובנוסף ידוע כי כמו בסעיפים קודמים, מרחק אוקלידי הוא חסם תחתון לאורך פטרון אופטימלי, ולכן נקבל:

h(p) =< heuclidian(p) =< h\*(p) = \* h\*(p)

כלומר קיבלנו את הנדרש עבור , וזה: h(p) =< \* h\*(p)

1. יבש (1 נק׳): נתונות יוריסטיקות שהן קבילות בהתאמה וכי הם האפסילונים ההדוקים ביותר.

הראו כי היא – קבילה , מצאו את ההדוק ביותר והוכיחו .

Given and we get:

Meaning that h­3 is the best epsilon will be because otherwise it contradicts that the epsilon given is the best for the first two h’s

נגדיר יוריסטיקה חדשה :

* D היא קבוצת כדורי הדרקון, .

הערה: בנוסחת המרחק מתייחסים למיקום של צומת.

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

1. יבש (1 נק׳): האם היוריסטיקה קבילה על כל לוח? אם כן הסבר, אם לא הבא קודמה נגדית.

**YES**

given every Board NxN and under the rules of the game (only moves are DOWN RIGHT LEFT UP) hMSAP(s) is the minimum of the Manhattan distances from s to every goal, meaning that if the closest goal which can be a dragonball or final goal is smaller than the actual optimal solution because the value of each step on the board is bigger or equal to 1, we can visualize Manhattan distance as steps with value 1.

Finally, we get hMSAP(s) is in fact smaller or equal than the optimal value of our solution.

1. יבש (1 נק׳): האם היוריסטיקה עקבית על כל לוח? אם כן הסבר, אם לא הבא דוגמה נגדית.(לחשוב אם היא עקבית ולתקן בהתאם)

**YES**

Because the largest [h(s)-h(s’)] value that we can get from any taken step is 1, because its measured by Manhattan distance and the farthest it can get by one step is exactly one step, and in addition, the value of cost(s,s’) is always greater or equal to 1,thus leaving us with the following inequation :

h(s)-h(s’) =< 1 =< cost(s,s’)

which is what’s needed to show consistency.

נגדיר יוריסטיקה חדשה :

* D היא קבוצת כדורי הדרקון, .

1. יבש (1 נק׳): האם היוריסטיקה קבילה על כל לוח? אם כן הסבר, אם לא הבא קודמה נגדית.

**YES**

Yes, given every Board NxN and under the rules of the game (only moves are DOWN RIGHT LEFT UP),

We know that the Manhattan distance is the lower boundary for the length of the optimal solution, and since hnew(s) is less or equal to the Manhattan distance, we can say that hnew(s) =< h\*(s) , thus its admissible.

1. יבש (1 נק׳): האם היוריסטיקה עקבית על כל לוח? אם כן הסבר, אם לא הבא דוגמה נגדית.

**YES**

Because the largest [h(s)-h(s’)] value that we can get from any taken step is 1, because its measured by Manhattan distance and the farthest it can get by one step is exactly one step, and in addition, the value of cost(s,s’) is always greater or equal to 1,thus leaving us with the following inequation :

h(s)-h(s’) =< 1 =< cost(s,s’)

which is what’s needed to show consistency.

### שאלה 8 – Greedy Best First Search (3 נק׳):

השאלות בחלק זה מתבססות על הלוח “8x8” שמופיע במחברת אלא אם נכתב אחרת.

1. יבש (1 נק׳): האם האלגוריתם שלם? האם הוא קביל?

**כן שלם לא קביל**

Finite number of states and a connected graph means we will have an answer but not necessarily an optimal one, in the tutorial we learned that here Greedy Best First Search using a heuristic function only sets the order of nodes to expand and starts with the best one first.

1. יבש (2 נק׳): תנו יתרון וחיסרון של אלגוריתם Greedy Best first Search לעומת Beam Search.

Upside: Greedy Best First Search finds a better solution generally because it is not limited by the number of nodes to expand and check which is a downside with Beam Search over some values of K.

Downside: Greedy Best First Search uses more memory and slower due to its ability to check out a bigger number of nodes while searching.

### שאלה 9 – W-A\* (2 נק׳):

השאלות בחלק זה מתבססות על הלוח “8x8” שמופיע במחברת.

1. **רטוב**: ממשו את החלקים החסרים באלג׳ W-A\* בקובץ ע״פ ההנחיות המופיעות שם. עליכם להשתמש ביוריסטיקה .
2. (יבש 2 נק׳) בהינתן , נסמן את המסלולים המחוזרים על ידי W-A\* תחת הפורמולציה ב עבור בהתאמה. אזי עבור:
   1. יוריסטיקה קבילה . אם כן הסבירו. אם לא, ספקו דוגמה נגדית.

h=0

h=10

11

לא נכון,

עבור גרף זה עם יוריסטיקה אכן קבילה אבל לכל שני משקלים שמקיימים את התנאי המחיר של המסלול המוחזר ע"י \*A-W שווה.

* 1. יוריסטיקה כללית (לא בהכרח קבילה) . אם כן הסבירו. אם לא, ספקו דוגמה נגדית.

לא נכון, אותה דוגמה כנ"ל רק נשנה h(s)=12 כלומר יוריסטיקה לא קבילה.

### שאלה 10 – IDA\* (2 נק׳):

1. יבש (1 נק׳): ספקו יתרון וחסרון של IDA\* ביחס לA\*. באילו מקרים הייתם מעדיפים להשתמש בכל אחד מהם?

Upside: A\* Doesn’t expand nodes which are already closed without the possibility of finding a better solution, while IDA\* expands repetitively nodes which he met in the past.

Downside: A\* Memory usage proportional to the number of nodes expanded (in Open and Close), while IDA\* Memory usage is linearly proportional to the solutions length.

We prefer using A\* when there are no memory limits and we want a faster solution.

We prefer using IDA\* when we have a large state space and we are limited with memory.

1. יבש (1 נק׳): ספק המחשה שלב אחר שלב של אלגוריתם IDA\* על הלוח (4x4) שמופיע במחברת, המראה כיצד החיפוש מתקדם באמצעות העמקה איטרטיבית ?

### שאלה 11 – A\* epsilon (6 נק׳):

1. **רטוב**: ממשו את החלקים החסרים באלג׳ W-A\* בקובץ ע״פ ההנחיות המופיעות שם. עליכם להשתמש ביוריסטיקה .

### יבש (2 נק׳): תנו יתרון וחיסרון של A\*-epsilon לעומת A\*.

A\* gives an optimal solution while A\*-epsilon gives a good solution up to a factor of (1+epsilon) from the optimal solution or a not so optimal solution.

A\* epsilon is faster than A\* because it eases up on optimality in hope of finding a solution faster.

1. יבש (3 נק׳): תנו הצעה ליוריסטיקה כדי לבחור את הצומת הבאה לפיתוח מתוך FOCAL. תארו את היוריסטיקה והציגו השוואה בין השימוש ביוריסטיקה זו לעומת השימוש ב-, מבחינת מספר פיתוחים, מסלול שנבחר ועלות המסלול שנבחר.

We can use f-value as the heuristic for choosing the next node to expand from the FOCAL list, rather than the g-value.

This lets choosing the next node to expand be dependent on not only distance from the start, but also predicted distance to the goal (h-value), which means considering the overall path length.

This change would lead to a more direct search, and as a result, likely **less expansions**.

The returned path after the change highly depends on the quality of the heuristics for each node, and since we are using Manhattan distance, which in this case seems to be highly informative as a heuristic, we can conclude that after the change, the algorithm would consider the overall path length and would likely result in a **shorter (better) path** than the unchanged version.

1. יבש (1נק׳): אם נגדיר שאפסילון שווה לאינסוף איך תהיה ההתנהגות של האלגוריתם עם סביבת כדורי הדרקון.

If we set epsilon to infinity, then we find that for each node we expand, all its unvisited neighbors should be in the FOCAL list, and then we choose the node in FOCAL list with the lowest g-value to expand.

We ended up with UCS algorithm where the FOCAL list in our case behaves exactly like the OPEN list in the UCS.

### שאלה 12 – Benchmarking (2 נק׳):

בשאלה זאת נשווה בין אלגוריתמי חיפוש שונים על בעיות שונות. הריצו את החלק הרלוונטי במחברת ותיראו שנוצר קובץ csv. (ניתן לפתוח עם Excel).

1. **רטוב**: הריצו את החלק הרלוונטי במחברת ותיראו שנוצר קובץ csv. (ניתן לפתוח עם Excel).
2. יבש (2 נק׳): הסבירו את התוצאות. האם הן תואמות לציפיות שלכם? האם התוצאות היו משתנות עם יוריסטיקה יותר מיודעת? נתחו והסבירו את התוצאות במונחים של מספר פיתוחים, מסלול מוחזר ומחיר הפתרון. שימו לב שבסעיף זה אין תשובה נכונה או לא נכונה אבל נדרש ממכם לספק הסבר מפורט ומבוסס.

### שאלה 13 – Local Search (5 נק׳):

בהינתן מרחב המצבים הבא, כאשר הינו המצב ההתחלתי, הינה פונקציית ערך והערך עבור כל מצב מצוין בצומת. המטרה שלנו היא למצוא מצב שממקסם את ערך .

נשתמש באלגוריתם Stochastic Hill Climbing.

כמו כן ידוע כי .

1. יבש (1 נק׳): מה ההסתברויות למעבר מהצב ההתחלתי לכל אחד מהמצבים . רשמו את .

First, we calculate the utility value for each neighbor:

∆UB = 2 , ∆UC = 2 , ∆UD = 1

Then the probabilities are:

p(b|a) = 2/5 , p(c|a) = 2/5 , p(d|a) = 1/5

1. יבש (1 נק׳): מה הוא מספר הצעדים המקסימלי שהאלגוריתם יכול לבצע? צעד מוגדר כמעבר בין מצבים.

2 or 3.

Depends on the value of :

If >4 then the longest path is 3 steps long.

If <=4 then the longest path is 2 steps long.

1. יבש (1 נק׳): בהיתן שבצעד הראשון האלגוריתם עבר למצב . האם האלגוריתם יתכנס למקסימום הגלובלי?

NO.

The global maximum is either F or G nodes, depending on value, and since the algorithm only takes steps towards nodes with higher values (neither equal nor less), and in this case the algorithm will certainly step into H (local maximum) since it’s the only possible step to take, which means that the algorithm will NOT reach a global maximum.

1. יבש (1 נק׳): מה ההסתברות שהאלגוריתם יתכנס לפתרון לא אופטימלי (שאינו מקסימום גלובלי)?

יש שני מקרים :

We can calculate the probability of Not getting the optimal solution by calculating the opposite event rule, which means calculating 1-p(getting optimal solution).

We consider two cases:

< 4:

the optimal solution is F node:

p(F) = p(b|a)\*p(f|b)

p(b|a) = 2/5 from question 13.1.

p(f|b) = ∆UF/(∆UF+∆UB) = 2/(2+-2) = 2/

so the probability of NOT getting an optimal solution would be:

P(not optimal) = 1- p(optimal) = 1-(2/5)\*(2/) = **(5 – 4) / (5 *)***

>= 4:

In this case, if we reach node B then we will definitely get the global maximum solution.

And if the first step is not towards B, then we will definitely NOT get an optimal solution.

Thus:

P(not optimal) = P(c|a) + P(d|a) = **3/5**

1. יבש (1 נק׳): עבור אילו ערכים של ההסתברות להגיע מהמצב ההתחלתי למקסימום הגלובלי תוך בדיוק 3 צעדים גדול מ ?

NOT POSSIBLE,

No value satisfies the demand.

In order to reach an optimal solution in exactly 3 steps, the optimal solution should be node G, and we have to go through the path: A->B->F->G

This can happen only when > 4,

P(|3 steps) = p(G|F)\*p(F|B)\*p(B|A) = 1\* (2/) \* (2/5) = 4/(5)

We demand that P(|3 steps) > (1/5) and calculate:

4/(5) > (1/5)

20 > 5

4 >

We got a contradiction.

So we concluded that no value yields a probability greater than 1/5.

הוראות הגשה:

עליכם להגיש קובץ יחד בשם AI1\_<id1>\_<id2>.zip (בלי הסוגריים המשולשים) המכיל:

1. קובץ בשם AI1\_<id1>\_<id2>.pdf שמכיל את התשובות לחלק היבש.
2. קובץ בשם Algorithms.py המכיל את המימוש לאלגוריתמי החיפוש.