תרגיל בית 1 מבוא לבינה מלאכותית

מגישים: ליאור דרור (212775530) יהונתן לחמן (212783088)

שאלה 1:

1. ☑
2. נגדיר את הרביעייה $(\mathbf{S}, O, I, G)$ (נסמן בהדגשה לייצוג מרחב החיפוש **S** ואת המצב ההתחלתי נסמן ללא הדגשה S). בנוסף נסמן Board להיות המפה של הלוח.
   מרחב המצבים - $\mathbf{S} = \{0, 1, ..., 63\} \times \{false, true\} \times \{false, true\}$
   $|S| = 64 \cdot 2 \cdot 2 = 256$ (64 אפשרויות לריבוע, 2 אפשרויות עבור אם אספנו dragonball או לא לכל אחד מה-2).
   מרחב האופרטורים: $O = \{D = 0, R = 1, U = 2, L = 3\}$
   מצב התחלתי: $I = (0, false, false)$
   קבוצת מצבי סיום:
   $\{g \in S | g[1] = true \wedge g[2] = true \wedge Board[g[0]] = "G"\} = \{(63, true, true)\}$
3. נתאר את קבוצת המצבים לאחר הפעלת UP:
   $Domain(UP = 2) = \{s \in \mathbf{S} | Board[s[0]] \neq "H"\}$
4. נחשב את המצבים העוקבים למצב ההתחלתי 0:
   $Succ(S) = \{(0, false, false), (1, false, false), (8, false, fasle)\}$
5. יש מעגלים במרחב החיפוש. לדוגמה, אם "נעים" שמאלה מהמצב ההתחלתי מקבלים שוב פעם את המצב ההתחלתי, ועל כן יש מעגל בגודל 1.
6. מכל מצב שיש לו התקדמות, אפשר להתקדם ב-4 כיוונים: $\{D = 0, R = 1, U = 2, L = 3\}$. לכן $b = 4$.
7. כיוון שראינו כי קיים מעגל בגרף, במקרה הגרוע הסוכן אף פעם לא יגיע למטרה ולכן כמות הצעדים אינסופית.
8. במקרה הטוב ביותר, הסוכן יצטרך לעשות 16 צעדים - הסוכן ירד ישירות לdragonball הראשון, ומשם ימשיך לdragonball במסלול הקצר ביותר, ומשם ילך למטרה במסלול הקצר ביותר. עבור מסלול זה, משקל המסלול לפי איך שהוגדר בתרגיל הוא 104.
9. הדבר אינו בהכרח נכון. ניתן דוגמה נגדית:

G_1	L	F	S

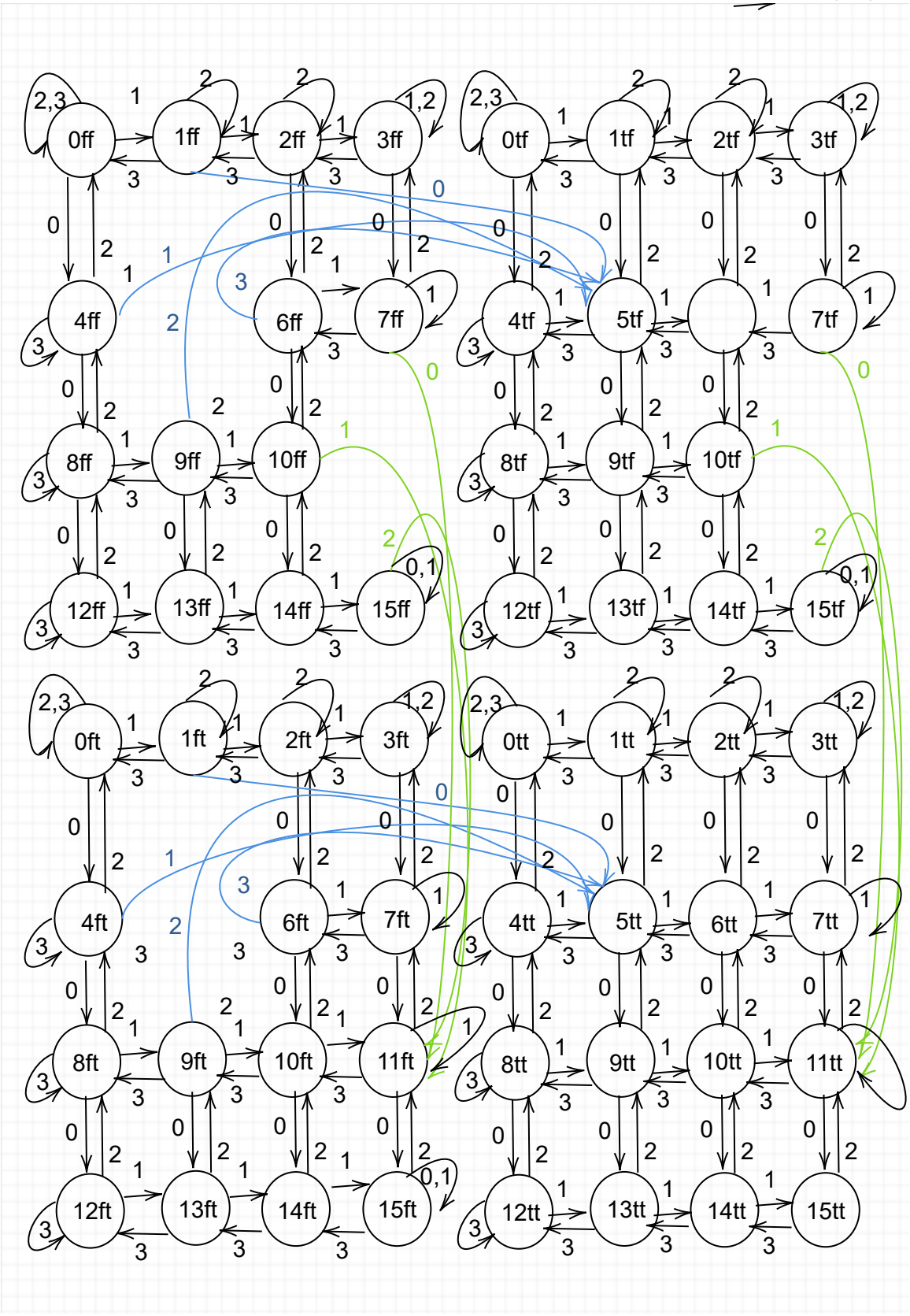
L	F	L	F
L	F	L	F
L	D	D	G_2

אפשר לראות כי G_1 קרוב יותר למצב ההתחלתי מאשר G_2 לפי מטריקת manhattan distance (מרחקו של G_1 3 בעוד שמרחקו של G_2 6). אבל, נשים לב שהמסלול הקל ביותר מ- S ל- G_2 הוא 6 (מתחילים ב S, יורדים דרך ה L שכל אחד עם משקל 1, אוספים את 2 ה- D שכל אחד עם משקל 1, ומסיימים ב G_2 שמשקלו 1). לעומת זאת, בכדי להגיע ל G_1 כמצב סופי, צריך גם כן לאסוף את 2 ה- D באותו אופן (זהו המסלול הקל ביותר) אך משם צריך לעלות שלושה L ולהתקדם ימינה, מה שייצור מסלול במשקל 9. לכן המסלול הקל ביותר הוא למצב הסופי הרחוק יותר במטריקת manhattan distance בניגוד לטענה.

שאלה 2:

1. ☑
2. ההבדל המרכזי בין חיפוש בעץ לחיפוש בגרף הוא שחיפוש בגרף לא מפתח צמתים שכבר פיתח בעבר. לכן, אם נרצה שחיפוש בעץ יחזיר את אותו הסדר כמו חיפוש בגרף, נדרוש שהחיפוש בעץ לא יפתח את אותו הצמת פעמיים. חיפוש בעץ יפתח צמת פעמיים אם הצמת מופיע במעגל כלשהו.
כלומר, גרף החיפוש שלנו יהיה חסר מעגלים. כל גרף חיפוש אשר מקיים תנאי זה יגרום לכך ש BFS על גרף ו- BFS על עץ ייצרו ויפתחו צמים זהים באותו הסדר.
3.

להלן גרף המצבים:



4. יהיה גרף $N \times N$. ניצור פונקציה $T: G \to G'$ אשר מקבלת גרף מצבים G ומחזירה G' כך שהרצת BFS על G' תחזיר את הפתרון האופטימלי. הרעיון שנבצע הוא שנרצה להביע את המשקל של כל צמת במרחק, כך שBFS, אשר מוצא מסלול קצר ביותר, יצטרך "להתחשב" במשקל. להלן הפונקציה:
   1. עבור הצמתים S, D, G, L, H, הפונקציה תשאיר את המצבים כמו שהם בגרף החדש.
   2. $T(A \in G) = A', A''$ כך ש-כל הקשתות שנכנסות ל A ב- G ייכנסו רק ל A', כל הקשתות היוצאות מ- A ב- G ייצאו מ- A'', ועוברת קשת יחידה $A' \to A''$.
   3. $T(T \in G) = T_1, T_2, T_3$ כך ש-כל הקשתות שנכנסות ל T ב- G ייכנסו רק ל T_1, כל הקשתות היוצאות מ- T ב- G ייצאו מ- T_3, ועוברת קשת יחידה $T_1 \to T_2, T_2 \to T_3$.
   4. $T(F \in G) = F_1, F_2, ..., F_{10}$ כך ש-כל הקשתות שנכנסות ל T ב- G ייכנסו רק ל F_1, כל הקשתות היוצאות מ- F ב- G ייצאו מ- F_{10}, ועוברת קשת יחידה
   $F_1 \to F_2, F_2 \to F_3, ..., F_9 \to F_{10}$

נשים לב שעבור הגרף החדש, מספר הצמתים במסלול חדש מצמת המקור לצמת היעד הוא כמשקל המסלול ב- G. משקל כל קשת במסלול הקודם התארך לכמספר הצמתים ב G'. אם נריץ BFS על הגרף החדש (בהנחה ויש פתרון), האלגוריתם יחזיר מסלול קצר ביותר. נבצע את הטרנספורמציה ההפוכה מ G' ל- G על מנת לקבל את המסלול הקל ביותר: כל רצף מהצורה
$(A', A''), (T_1, T_2, T_3), (F_1, F_2, .., F_{10})$ יכווץ לצמת בודד, וכך יתקבל מסלול חוקי בגרף G. לא יכול להיות שרצף צמתים מהגרף החדש "נחתך באמצע" כלומר שיפיעו חלק מהצמתים ברצף שתואר לעיל אבל לא כולם, כיוון שלכל צמתי הביניים יש קשת אחת נכנסת (מהצמת הקודם ברצף) לצמת הבא ברצף, ורק בסוף הרצף יש אולי קשתות לצמת שאינו חלק מהרצף. זה מסלול אופטימלי כיוון ש-BFS מחזיר מסלול קצר ביותר (אופטימלי מבחינת אורך) וכיוון שנעשתה פה רדוקציה (לא פורמלית) אז גם הגרף המקורי יחזיר מסלול אופטימלי (אופטימלי מבחינת משקל).
באופן כללי, אם היינו רוצים להכליל את הרדוקציה לגרף כללי שלא תואם למשקלים הספציפיים של המשחק, היינו משכפלים כל צמת v שנכנסת אליו קשת עם משקל w ל w צמתים בגרף החדש, כך שיש קשת יחידה מצמת ברצף לצמת הבא ברצף, כל הקשתות הנכסנות ייכנסו לצמת v_1 וכל הקשתות היוצאות מ- v ייצאו מ v_w. כלומר, ליצור מעין שרוך בגודל w מכל צמת במשקל w.

5. האלגוריתם יפתח $N^2 - 2$ צמתים וייצור N^2 צמתים לאורך הריצה. נשים לב לתכונה מעניינת על BFS (כאשר אין D ולכן ישנה ריצה רגילה של BFS קלאסי):
לכל צמת v במרחק קצר ביותר i מצמת מקור s בריצת BFS, בשלב פיתוח v, לא יהיה צמת $v' \in OPEN$ כך ש המרחק המינימלי מ- s ל- v' קטן ממש מ i (הוכחה באלגוריתמים 1).
נשים לב שמציאת צמת המטרה קורה רק כאשר מפתחים צמת שכן. ידוע שצמת המטרה נמצא במרחק $2N - 2$ מצמת המקור (יורדים $N - 1$ צעדים למטה ואז $N - 1$ צעדים ימינה). לכן המרחק של השכן של צמת היעד הוא $2N - 3$. לכן כל צמת במרחק קטן ממרחק זה מצמת המקור כבר יפותח (ייצא מ- OPEN). כיוון שעוצרים את האלגוריתם ברגע שמגיעים לשכן כלשהו של צמת היעד, לא מפתחים את השכנים האחרים של צמת היעד. כיוון ששכן זה נמצא ב OPEN, זה מועיד על כך שגילינו אותו קודם לכן, ולכן ייספר כחלק מהצמתים שנוצרו. בלוח שלנו יש שכן אחד נוסף לצמת היעד, לכן הוא לא יפותח. בנוסף, צמת היעד עצמו לא יפותח מהגדרת BFS, לכן 2 צמתים לא יפותחו, אך הוא כן ייווצר (מחזירים את הפתרון אחרי שמגלים צמת יעד). לכן $N^2 - 2$ צמתים יפותחו ב- BFS, אבל כל

N^2 הצמתים יווצרו.

שאלה 3:

- **Complete**: Guaranteed to find a solution if one exists?
- **Optimal**: Guaranteed to find the least cost path?

1. עבור בעיית dragonballs עם לוח $N \times N$, האלגוריתם DFS-G הוא שלם מכיוון שיש הבטחה שכל מצב שפיתחנו לא נפתח שוב (מכניסים אותו לרשימת המצבים הסגורים CLOSED), ולכן הוא סופי. מובטח כי האלגוריתם ימצא פתרון כלשהו מכיוון ש-DFS-G עובר על כל המסלולים האפשריים מהמצב ההתחלתי למצב הסופי, ולא יוצר לולאות אינסופיות, ולכן אם קיים מסלול שעובר בשני ה-dragonballs ומגיע למצב הסופי, הוא יעבור גם בו.

האלגוריתם אינו קביל. למשל, עבור הדוגמה הבאה:

G_1	D	D	S
F	L	F	F
F	L	F	F
G_2	F	F	F

המסלול האופטימלי יהיה לזוז ימינה שלושה צעדים ואז במחיר של 3 (כולל המחיר 1 של צמת היעד) האלגוריתם אסף את ה dragonballs וגם הגיע לצמת יעד. לעומת זאת, ב DFS האלגוריתם יילך קודם כל למטה (כי זו הפעולה הראשונה שממנה הוא מרחיב את שאר המסלול בצורה רקורסיבית). משם הוא יזוז אחד ימינה, יעלה עד הסוף למעלה, יזוז אחד ימינה, יירד עד למטה ורק **אז** יסיים את ריצתו. אפשר לראות שהמחיר שDFS יחזיר יהיה גדול בהרבה מהמחיר של המסלול האופטימלי.

2. אלגוריתם DFS על עץ לאו דווקא היה מחזיר פתרון במקרה זה, מכיוון שהלוח מכיל מעגלים- למשל עבור לוח שמכיל Fים בתאים שבגבולות הלוח, ו-D,G בתאים שלא נמצאים בגבולות, האלגוריתם יעבור על גבולות הלוח, ויצליח לחזור למצב ההתחלתי ולעשות זאת שוב ושוב מכיוון שאין רשימת CLOSED שמכילה את הצמתים שכבר ביקרנו בהם ואין צורך לבקר שוב.

3. מספר הצמתים שיפותחו באלגוריתם הוא: $\mathbf{2n-2}$ (כולל הצומת ההתחלתית). מספר הצמתים שיווצרו במהלך האלגוריתם הוא: $2n+2(n-2)-1 = \mathbf{4n-5}$
זאת מכיוון שהכיוון ההתחלתי שבו צמתים יפותחו הוא או ימינה או למטה, ואז כל צומת שתפותח תיצור 2 בנים, צומת לידה ולמלטה אליה, ותבחר לפתח את האחת באותו כיוון שממנו הגיעה, עד שנגיע לסוף הגרף. עד כאן, פותחו N צמתים ונוצרו 2N צמתים. לאחר מכן ישנו כיוון- למטה או ימינה בהתאם, וימשיכו ביצירת ובפיתוח עד שיגיעו לפינה הימנית למטה (אותה הוא לא יפתח). כאן מפתחים N-2 צמתים ונוצרים $2(n-2)-1$ צמתים.

4. backtracking הוא אלגוריתם דומה ל-DFS, פרט לכך שהוא יוצר צמתים בצורה עצלה - יוצר צמתים עוקבים רק מיד לפני הפיתוח שלהם, ולא את כל הצמתים העוקבים לצומת ספציפית כאשר מפתחים

אותה. בעקבות כך, בעת המעבר על המסלול של השאלה, יווצרו $2n-1$ צמתים ויפותחו $2n-2$ צמתים (כולל הצומת ההתחלתית). זאת כי כמות הצמתים שיפותחו תישאר זהה מאופן הגדרתו של ה backtracking, אבל אנחנו ניצור גם את צמת היעד לכן יווצר צמת אחד יותר (ולא יוצרים את השכנים מהסעיף הקודם).

שאלה 4:

1)

a) עבור בעיית dragonballs 8x8 האלגוריתם $ID-DFS$ שלם. יש הבטחה שכל מצב שפיתחנו עבור L מסוים לא יפותח שוב מהגדרת DFS (שמים ב CLOSED של אותה איטרציה). הסברנו בשאלה 3 כי DFS יתכנס לפתרון עבור מסלול בגודל כלשהו, לכן בהכרח קיים L אשר DFS-L מתכנס לפתרון, ולכן $ID-DFS$ גם כן.

b) האלגוריתם אכן קביל. נניח בשלילה שאינו קביל. נזכור שהראנו בסעיף א שהאלגוריתם שלם, לכן האלגוריתם מחזיר מסלול תקין אך לא אופטימלי. בפרט, האלגוריתם $ID-DFS$ החזיר מסלול באורך L עבור לוח עם מסלול אופטימלי $L' < L$. אזי, באיטרציה L' של $ID-DFS$ האלגוריתם עבר על כל המסלולים באורך L', בפרט המסלול האופטימלי, אך לא החזיר אותו. סתירה להגדרת $DFS-L$. לכן האלגוריתם קביל.

2)

a) יתרון של $ID-DFS$ על $Reverse-DFS$ הוא במקרה ואין חסם עליון טוב על אורך המסלול המינימלי שמקיים פתרון. אם $D \gg L$ כאשר L האורך למסלול DFS המינימלי, אז $Reverse-DFS$ יצטרך לבצע הרבה יותר איטרציות, שכל איטרציה מכילה פיתוח צמתים רב יותר, מאשר $ID-DFS$.
מצד שני, אם קיים חסם עליון הדוק על מרחק המסלול המינימלי לDFS, עדיף להשתמש ב $Reverse-DFS$. למשל, אם $D = L$ עובר L גדול, אז לאחר איטרציה בודדת האלגוריתם יסתיים בעוד שב- $ID-DFS$ יעשה L איטרציות.

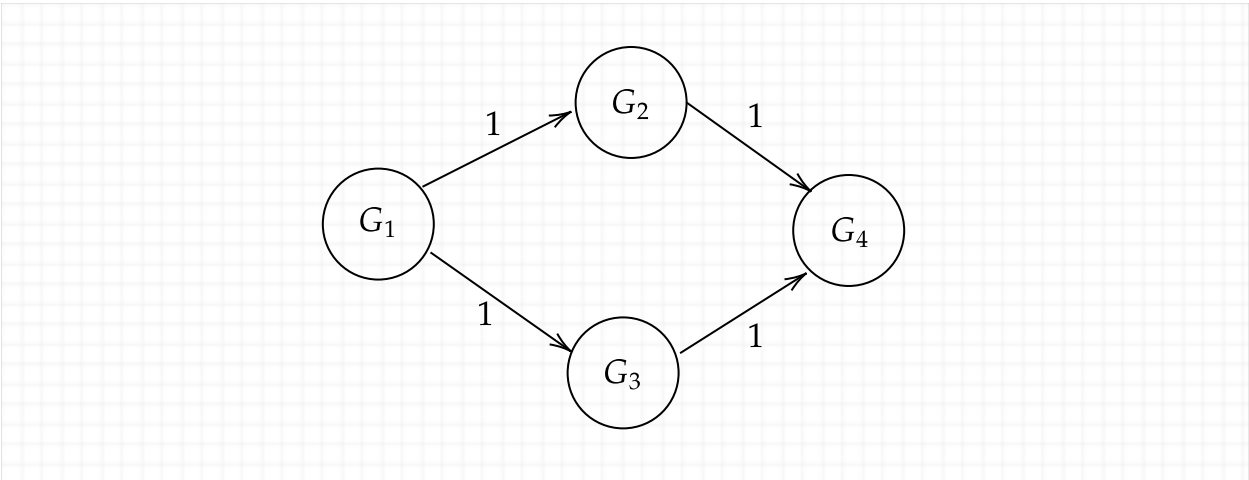
b) נשתמש במושג בסיסי של מדעי המחשב לייעול האלגוריתם: חיפוש בינארי. נשמור עותק של L בשם L_{old} (המרחק למסלול ב $DFS-L$), וכל איטרציה שיש פתרון נחלק את L ב-2 במקום להחסיר ב1. אם באחת האיטרציות קיבלנו שאין פתרון, נעשה ממוצע בין המספר ששמרנו באיטרציה הקודמת, לבין ה L העדכני כלומר $L = \frac{L+L_{old}}{2}$. נעצור כאשר יש שיוויון וכאשר $L == \frac{L+L_{old}}{2}$ (הכוונה פה ש- L_{old} או זהה או אחד מעל L הנוכחי ואז אפשר להחזיר אותו).

שאלה 6:

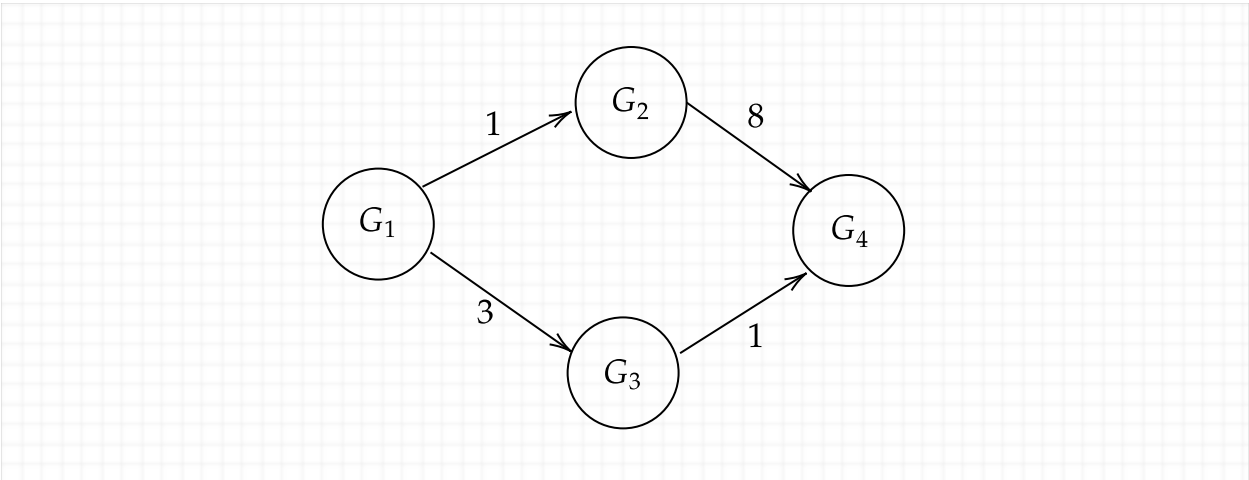
1. בעיות החיפוש עבורן UCS ו-BFS יפעלו באותה דרך עבור בעיות שבהן המשקל על הקשתות שווה. BFS אינו מתייחס למשקל על הקשתות, רק על המרחק מהשורש, בעוד ש-UFS כן. אם משקל כל הקשתות יהיה שווה, אז UCS יסתכל על המרחק מהשורש של כל צומת, ולכן שני האלגוריתמים יפעלו באותו אופן.

2. האלגוריתם אכן קביל, ולכן הוא גם שלם. ראינו בהרצאה שעבור משקל קשת חסום d, האלגוריתם קביל. בבעיית החיפוש שלנו, משקלי הקשתות (של איברי הלוח) גדולים או שווים ל1, לכן אם נבחר $d = 0.5$ נקבל חסם תחתון לכל הקשתות, ולפי המשפט האלגוריתם קביל ולכן גם שלם.

3. דוגמה לגרף שעבורו שאדי עדיין יחזיר את המסלול הקל ביותר:



מכיוון שכל המסלולים בעלי אותו משקל.
דוגמה לגרף שעבורו שאדי לא יחזיר את המסלול הקל ביותר:



בגרף זה המסלול $G_1 \to G_3 \to G_4$ שוקל 4, לעומת המסלול השני ששוקל 9, ולכן הוא הקצר ביותר. למרות זאת, האלגוריתם של שאדי תחילה ייצור את G_2, G_3, ולאחר מכן יפתח את G_2 מכיוון שהמשקל שלו קטן יותר, מה שיגרום ליצירת G_4 עם משקל 8 ולבחירה במסלול הזה שעולה 9 לפי

האלגוריתם של שאדי, למרות שהמסלול השני הוא קל יותר ממנו.

שאלה 7:

היוריסטיקה תיקרא ϵ קבילה אם קיים $\epsilon \geqslant 1$ כך שלכל מצב $s \in S$ מתקיים $h(s) \leqslant \epsilon \cdot h^*(s)$.

1. מרחק מנהטן $h_{MD}(p) = ||P - G||_1 = |G_x - P_x| + |G_y - P_y|$: ϵ קבילה, בפרט $\sqrt{2}$ קבילה.

תחילה נסביר למה היא $\sqrt{2}$ קבילה: יוריסטיקת מנהטן היא הפעלת נורמת L_1 על הנקודה p ביחס ליעד g. ממפשט קושי-שוורץ, מתקיים ש:

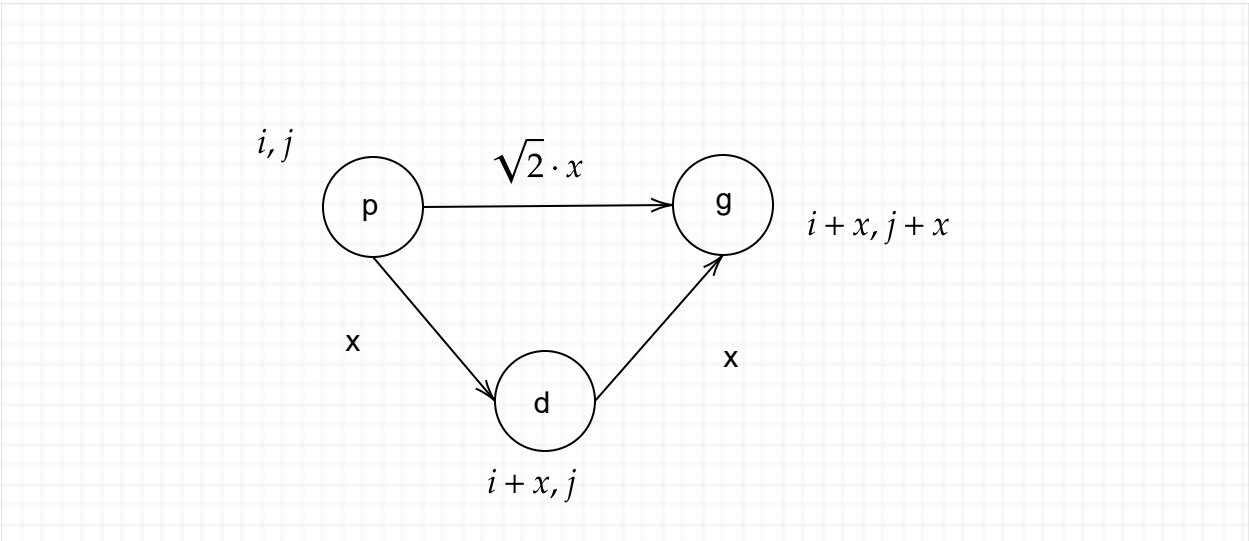
$$L_1(v) \leqslant \sqrt{d} L_2(v)$$

כאשר v הוא הנקודה ו- d הוא כמות המימדים של הנקודה. במקרה שלנו, מתקיים $L_1(v) \leqslant \sqrt{2} L_2(v)$. בנוסף, נשים לב ש L_2 (המרחק האוקלידי) חסם תחתון ליוריסטיקה האופטימלית, כיוון שמרחק בין שתי קואורדינטות יכול להיות לכל הפחות המרחק האוקלידי שלהם. כיוון שלא לכל זוג קואורדינטות יש כביש בינהן, המרחק האוקלידי קטן או שווה למרחק הקצר ביותר לפי הכבישים בגרף, שמתואר ע"י היוריסטיקה האופטימלית. לכן נקבל סה"כ:

$$h_{MD}(p) \leqslant \sqrt{2}||p||_2 \leqslant \sqrt{2} h^*(p) = h^*(p)$$

אם נבחר $\epsilon = \sqrt{2}$ נקבל ϵ-קבילות.

כעת נסביר למה זה החסם ההדוק ביותר, בכך שנראה דוגמה בה השיוון מתקיים ממש.



נשים לב שמרחק מנהטן עבור הגרף הנוכחי (כאשר p צמת המקור, g צמת היעד) יהיה $2x$, שהרי (בה"כ) $d_x = p_x + x,\ g_y = p_y + x$. מצד שני, המרחק האוקלידי יהיה

$$||g - p||_2 = \sqrt{(g_x - p_x)^2 + (g_y - p_y)^2} = \sqrt{x^2 + x^2} = \sqrt{2}x$$

אם נשתמש בהגדרה של ϵ קבילות, נקבל כי $h_{MD}(p) = 2x = \sqrt{2} \cdot \sqrt{2}x = L_2(p) = h^*(p)$.

לכן מצאנו חסם הדוק ביותר.

2. $h(p) = min\{G_x - P_x,\ G_y - P_y\}$ ϵ קבילה, בפרט 1 קבילה. נוכיח:

$$min\{G_x - P_x,\ G_y - P_y\} \leqslant min\{|G_x - P_x|,\ |G_y - P_y|\} \overset{*}{\leqslant} L_2(p) \leqslant h^*(p)$$

* - נשים לב ש$min\{|G_x - P_x|,\ |G_y - P_y|\}$ מייצג את המינימום מבין שני צירים (ציר x או ציר y).

לכן נקבל שהמרחק האוקלידי גדול מהמרחק מאחד הצירים, שכן כל ציר הוא הטלה של המרחק האוקלידי. לכן נקבל שהיוריסטיקה קבילה. כיוון ש- $\epsilon \geqslant 1$, מצאנו את החסם ההדוק ביותר שיכול להיות.

3. L_3 : $h(p) = ||P - G||_1 = \sqrt[3]{|G_x - P_x|^3 + |G_y - P_y|^3}$ ϵ קבילה, בפרט 1 קבילה. נשים לב שממשפט קושי-שוורץ מתקיים $||p||_3 \leqslant ||p||_2$. בסעיף א הסברנו כי L_2 חסם תחתון ליוריסטיקה האופטימלית, לכן מתקיים כי עבור $\epsilon = 1$, $||p||_3 \leqslant ||p||_2 \leqslant h^*(p) \cdot 1$. נשים לב שזהו החסם הנמוך ביותר, שכן $\epsilon \geqslant 1$.

4. יהיו שתי יוריסטיקות h_1, h_2 ϵ_1, ϵ_2 קבילות (בהתאמה) עם חסמים הדוקים ϵ_1, ϵ_2. נראה כי $h_3 = h_1 + h_2$ היא ϵ_3 קבילה, עבור $\epsilon_3 = \epsilon_1 + \epsilon_2$. יהי $s \in S$. מתקיים:

$$h_3(s) = h_1(s) + h_2(s) \overset{h_1,h_2 \text{ are } \epsilon_1,\epsilon_2 \text{ admissable}}{\leqslant} h^*(s) \cdot \epsilon_1 + h^*(s) \cdot \epsilon_2 = (\epsilon_1 + \epsilon_e)h^*(s) = \epsilon_3 h^*(s)$$

בנוסף, אפשר לראות כי חסם זה הוא ההדוק ביותר כי ϵ_1, ϵ_2 הדוקים ביותר. לכן הוכחנו את הנדרש.
■

5. נראה בסעיף 6 שהיוריסטיקה עקבית, וראינו בתרגול שיוריסטיקה עקבית היא גם כן קבילה ולכן היוריסטיקה קבילה.

6. נוכיח עקביות של היוריסטיקה. יהיו s, s' כך שקיימת קשת מ s ל s'. נרצה להראות כי $h(s) - h(s') \leqslant cost(s, s')$. נשים לב כי $h(s) - h(s') \leqslant 1$, כי כאשר אנחנו מתקדמים צעד אחד למשבצת אחרת, אנחנו הולכים צעד אחד בכיוון אנכי או כיוון אופקי. אם אנחנו נשארים באותה המשבצת, מתקיים כי היוריסטיקה זהה ולכן מתקיים התנאי. כעת נראה מה קורה למחיר הקשת:
   (a) אם נשארנו באותו המקום ($s = s'$) אזי המחיר שעבר הינו 0, כלומר $h(s) - h(s') = h(s) - h(s) = 0 = cost(s, s')$.
   (b) אם זזנו למשבצת אחרת ($s \neq s'$) אזי המשקל $1 \leqslant$, כי כל סוגי המשבצות שמוגדרות בלוח עם משקל גדול או שווה ל-1. לכן, מתקיים $h(s) - h(s') \leqslant 1 \leqslant cost(s, s')$.

   בכל המקרים קיבלנו $h(s) - h(s') \leqslant cost(s, s')$ לכל שני צמתים בלוח, ולכן לפי הגדרת עקביות, היוריסטיקה עקבית.

7. היוריסטיקה h_{new} אינה קבילה. דוגמה נגדית:

$$\begin{pmatrix} S & D_1 & G_2 \\ D_2 & F & F \\ F & F & G_1 \end{pmatrix}$$

אם היוריסטיקה קבילה אז מתקיים $h_{new}(s) \leqslant h^*(s)$ לכל $s \in S$. ביוריסטיקה האופטימלית, נרצה שיתקיים $h^*(G_1) = 0$ עבור G_1 מצב יעד כלשהו. נניח והגענו ל G_1 לאחר איסוף שני כדורי הדרקון (נניח במצב $(8, t, t$ אולם, ביוריסטיקה הזו מתקיים כי $h_{new}(g') = max\{h_{Manhattan}(G_1, g) | g \in D \cup G\} = h_{Manhattan}(G_1, G_2) = 2 > 0$ ולכן

היוריסטיקה לא קבילה.

8. ראינו בתרגול כי יוריסטיקה עקבית היא גם קבילה, ולכן יוריסטיקה לא קבילה היא גם לא עקבית. כיוון שהראנו בסעיף הקודם שהיורסטיקה h_{new} אינה קבילה, היא גם לא עקבית, ואותה דוגמה נגדית חלה גם על סעיף זה.

שאלה 8:

1. האלגוריתם שלם אם מובטח שהוא יחזיר לנו פתרון אם קיים פתרון. האלגוריתם Greedy Best Search שלם, מכיוון שכל המצבים יפותחו בסוף, מכיוון שמספר המצבים בלוח 8X8 הנתון הוא סופי. בסופו של דבר כל המצבים בבעייה יפותחו, והסדר ביניהם יקבע לפי ההיוריסטיקה. יש איסור לבקר בצמתים שכבר ביקרנו בהם, ולכן לא ניקלע למסלול אינסופי.
האלגוריתם לא קביל, מכיוון שיכול להיות שהוא ימצא מסלול שהוא נכון אבל אינו אופטימלי.זאת בהתבסס בהיוריסטיקה של המצבים.

2. חיסרון של greedy first search לעומת beam search הוא שהוא לוקח הרבה יותר זיכרון וזמן- כי אין לו הגבלה על התור הפתוח של המצבים כמו של beam search שמגביל את התור לגודל K כלשהו.
יתרון שלו על פני beam search הוא שהוא מדוייק יותר. ב-beam search מכיוון שיש הגבלה על גודל התור, יכול להיות שנפספס מסלול כלשהו שבהתחלה ההיוריסטיקה שלו הייתה גבוהה מהשאר, אבל בטוטאל הוא הכי עדיף מביניהם.

שאלה 9:

1. ☑
2. בהינתן $w_1 < w_2 \leqslant 1$, נסמן את המסלולים המוחזרים ע"י $W - A^*$ בפורמולציה $f = g + w*h$ ב- p_1, p_2 עבור w_1, w_2 בהתאמה.
   1. עבור יוריסטיקה קבילה, לא תמיד מתקיים $cost(p_1) < cost(p_2)$. למשל, עבור היוריסטיקה $h = 0$ נקבל שהפורמולציה המתקבלת היא UCS (לפי התרגול). לכן w אינו משפיע בשני המסלולים, ושניהם יתבססו אך ורק לפי המשקלים של הגרפים. כלומר, $cost(p_1) = cost(p_2)$ לכן הטענה אינה נכונה.
   2. עבור יוריסטיקה לא בהכרח קבילה, לא מתקיים $cost(p_1) < cost(p_2)$. למשל, עבור אותה יוריסטיקה מסעיף קודם, נקבל שוב שיוויון בין שני מחירי המסלולים ולכן הטענה גם כאן לא נכונה.

שאלה 10:

1. יתרון של IDA^* על A^* הוא זכרון המקום. בעוד ש IDA^* שומר על כמות זכרון לינארית ככמות הצמתים שפותחו במסלול הנוכחי (כמו DFS), אלגוריתם A^* משתמש שומר זכרון לכל הצמתים שנוצרו. לעומת זאת, עדיף להשתמש ב A^* אם רוצים להתחשב בכמות הצמתים שפותחו (סיבוכיות הזמן). A^* זוכר את הצמתים שכבר פיתח ולא יפתח אותם שוב אם אין שיפור, ולעומתו IDA^* מפתח את המסלולים עבור כל איטרציה של f_limit מחדש.
2. האלגוריתם יפעל כמו שראינו בתרגול. לצערי אין זמן לצייר 64 מצבים לכל איטרציה.

שאלה 11:

1. ☑
2. יתרון של $A^* - epsilon$ על פני A^* הוא שאפשר לקבל יעילות טובה יותר של האלגוריתם. כיוון שבכל צעד באיטרציה לא בודקים את המסלול האופטימלי, ורק מסתפקים בקירוב טוב שלו (אנחנו קובעים כמה "להתפשר" בקירוב טוב המדובר), נקבל פתרון יותר מהיר. מצד שני, החסרון ב- $A^* - epsilon$ על פני A^* הוא הקבילות - כבר אי אפשר להבטיח שהפתרון שנחזיר דרך $A^* - epsilon$ יהיה האופטימלי.
3. נציע את היוריסטיקה שהוגדרה בשאלה 7 - יוריסטיקת h_{msap}. נתאר את הריצה של $A^* - epsilon$ עם עם עם g בתור הצומת המינימלי שבוחרים מ-FOCAL:

```
AStarEpsilon_agent = AStarEpsilonAgent()
actions, total_cost, expanded = AStarEpsilon_agent.search(env, epsilon=100)
print(f"Total_cost: {total_cost}")
print(f"Expanded: {expanded}")
print(f"Actions: {actions}")
✓ 0.0s

Total_cost: 103.0
Expanded: 79
Actions: [0, 0, 0, 0, 0, 0, 1, 1, 1, 2, 1, 1, 0, 1, 1, 0]
```

עם h_{map} בתור הצומת המינימלי שבוחרים מ-FOCAL:

```
AStarEpsilon_agent = AStarEpsilonAgent()
actions, total_cost, expanded = AStarEpsilon_agent.search(env, epsilon=100)
print(f"Total_cost: {total_cost}")
print(f"Expanded: {expanded}")
print(f"Actions: {actions}")
✓ 0.0s

Total_cost: 103.0
Expanded: 88
Actions: [0, 0, 0, 0, 0, 0, 1, 1, 1, 2, 1, 1, 0, 1, 1, 0]
```

אפשר לראות שהעלות עם g זהה לעלות עם היוריסטיקה (לכן הפתרון עם היוריסטיקה עדיין בטווח של קירוב ϵ עבור $\epsilon = 100$) אבל מצד שני כמות הצמתים שפותחה עם היורסטיקה גדולה יותר.

1. נשים לב שאם מגדירים $\epsilon = \infty$, אז אין סינון של צמתים הנכנסים לfocal. נזכור שהקבוצה נבנת בצורה הבאה:

$$FOCAL = \left\{ v \in OPEN \mid f(v) \leqslant (1+\epsilon) \min_{v' \in OPEN} f(v') \right\}$$

לכן, כל צמת שנמצא ב succ של מצב כלשהו s יימצא בתוך FOCAL. משם, אנחנו נבחר את הצמת בעל הערך היוריסטי הנמוך ביותר. בתרגיל הוגדר שהערך היוריסטי יהיה לפי g ולכן הצמת בעל המרחק המינמלי מהמקור ייבחר. נשים לב שזה אופן פעולת UCS, לכן אם $\epsilon = \infty$ נקבל הרצה של UCS.

שאלה 12:

	A	B	C	D	E	F	G	H	I
1	map	BFS-G cost	BFS-G expanded	WA* (0.5) cost	WA* (0.5) expanded	WA* (0.7) cost	WA* (0.7) expanded	WA* (0.9) cost	WA* (0.9) expanded
2									
3	map12x12	140	445	118	223	118	200	118	240
4									
5	map15x15	215	858	178	650	178	604	195	707
6									
7	map20x20	203	1045	188	683	188	587	188	1002
8									

בטבלה ניתן לראות השוואה בין אלגוריתם BFS שבנינו לבין אלגוריתם wA* שבנינו עם משקלים שונים.

נשים לב שעבור המפה 12x12, BFS מצא מסלול במשקל 140, בעוד שכל אלגוריתמי wA* מצאו מסלולים במשקל 118. זה הגיוני מכיוון ש-BFS הוא אלגוריתם שמוצא מסלול לפי מרחק מהשורש, הוא לא מתחשב במשקלים של כל מעבר במסלול, המטרה שלו זה למצוא את המסלול הכי קצר. לעומת זאת, weightedA* מתחשב בשכלול בין משקלים של המסלולים והיוריסטיקה h_{map} שמוצאת מרחק מנהטן מינימלי לצמתים שיש לעבור בהן (קרבה לצמתי מטרה). לכן יש בו התחשבות במסלול עצמו, האופי שלו והמחיר שלו, ולא רק בקרבה לצומת ההתחלתי, ולכן הוא מוצא מסלולים עם מחיר נמוך יותר. המשקלים השונים ב-wA* מסמלים את הדגש של האלגוריתם- האם אכפת לו יותר למנמז את משקל המסלול, או את משקל ההיוריסטיקה. אנחנו משערים שבמקרה זה אין הבדל בין מחירי המסלולים עבור משקלים שונים ב-wA* מכיוון שעבור לוח זה כנראה ששתי הפונקציות הללו מתנהגות בצורה יחסית דומה לאורך המסלולים שנבדקו.
עבור המפה השנייה- 15x15, ניתן לראות שעדיין מחיר המסלול שמצא BFS הוא היקר ביותר, אבל הפעם המחיר של wA* עבור w=0.9 גבוה יותר מהמחיר עבור w=0.5, w=0.7. אמנם לא בהרבה, אבל עדיין. נזכיר שככל ש-w גבוה יותר, ניתן יותר משקל להיוריסטיקה, אנחנו מקרבים את האלגוריתם שלנו ל-greedy first best, שזהו אלגוריתם מיודע לחלוטין ולא קביל. הוא מהיר, אבל לא קביל- לא נמצא פתרון אופטימלי, ועובדה שהפתרון שמצאנו עם משקל גבוה משל האופטימלי. לעומת זאת, ככל שנקטין את w, ניתן יותר משקל למשקל המסלול, כך אנחנו הופכים את האלגוריתם לפחות מיודע (לאן הוא צריך ללכת, לעולם שמסביבו), אבל הוא יהיה יותר שלם וקביל, וגם איטי. ניתן גם לראות שככל שהקטנו את w, מצאנו פתרון קטן יותר, שיכול להיות שהוא האופטימלי.
עבור המפה השלישית 20x20 מתקיים משהו דומה למה שפירטנו מקודם ב-12x12.

נשים לב לכמות הexpanded nodes- כמות המצבים שפותחו. עבור BFS, תמיד מגיעים לכמות צמתים שהיא גדולה יותר מכמות הצמתים שפותחו עבור wA* בכל משקל. אנחנו משערים שזה בגלל ש-BFS לא מבצע בחירה חכמה של הצמתים הבאים לפיתוח (כמו שמשתמשים בהיוריסטיקה ובמשקל המסלול כדי למצוא את הצומת המינימלי ב-open ב-wA*), ולכן בסופו של דבר הוא צריך לפתח יותר צמתים על מנת להגיע למסלול שעונה על הדרישות- הוא הרבה פחות ספציפי. נשים לב גם שעבור כל המפות שנבדקו:

$$WA*(0.7)\ expanded < WA*(0.5)\ expanded < WA*(0.9)\ expanded$$

כלומר, עבור w=0.7 גם כמות הצמתים שפותחו היא מינימלית וגם מחיר המסלול הוא המינימלי, ולכן נשער שלפחות עבור מפות אלה והבעיה שאנחנו מתמודדים איתה, זהו המשקל האופטימלי מבין השלושה שנבדקו. כלומר, יש לבצע אלגוריתם מיודע יחסית, אבל עדיין להשאיר מקום למשקל המסלול, שהופך את האלגוריתם

לקביל. אם ההיוריסטיקה יותר מיודעת יש כבר בעיה במחיר המסלול האופטימלית שהיא מוצאת, ואם האלגוריתם פחות מיודע יקח לו יותר זמן להגיע לפתרון, כי הוא יחקור יותר מצבים שלא רלוונטים.

שאלה 13:

1. ההסתברויות למעבר מהמצב ההתחלתי לכל אחד מהמצבים: b, c, d הן:

$$p(d|a) = \frac{1}{5}$$
$$p(b|a) = \frac{2}{5}$$
$$p(c|a) = \frac{2}{5}$$

2. מספר הצעדים המקסימלי שהאלגוריתם יכול לבצע הוא 3: $A \to B \to F \to G$, שזהו מסלול שעונה על האלגוריתם stochastic hill climbing

3. בהינתן שבצעד הראשון האלגוריתם עובר למצב c, האלגוריתם לא יתכנס מכיוון שאחרי c, המצב שהאלגוריתם יעבור אליו הוא H מכיוון שיש לו את ההסתברות בפרופרציונלית הגבוהה ביותר לשיפור, ולכן לא נגיע ל-F, או ל-G, שהם האופציות לצומת עם מקסימום U.

4. האופציות לכך שהאלגוריתם יתכנס לפתרון לא אופטימלי:

- במידה והאלגוריתם בחר במסלול: $A \to D$ הוא יתכנס על הערך 1 וזה לא הערך האופטימלי, זה קורה בהסתברות 0.2.
- במידה והאלגוריתם בחר במסלול $A \to C \to H$ הוא יתכנס על הערך 3 וגם זה לא האופטימלי, זה קורה בהסתברות 0.4.
- האלגוריתם לא יבחר במסלול $A \to B \to E$ מכיוון שאין שיפור.

עכשיו נחלק למקרים:

-$3 < \beta \leqslant 4$: אז אם האלגוריתם יבחר ללכת במסלול $A \to B \to F$ הוא ימצא את הערך האופטימלי.

אחרת, המסלול $A \to B \to G$ הוא לא האופטימלי וההסתברות שהוא ילך שם היא:

$$\frac{2}{5} \cdot \frac{\beta - 2}{\beta - 2 + 4 - 2} = \frac{2\beta - 4}{5\beta}.$$

לכן, עבור מקרה זה ההסתברות שהאלגוריתם יתכנס לפתרון לא אופטימלי היא:

$$0.2 + 0.4 + \frac{2\beta - 4}{5\beta} = \mathbf{0.6 + \frac{2\beta - 4}{5\beta}}$$

- $4 < \beta$: יגיע לפתרון האופטימלי בכל מקרה (כי נגיע ל-G מ-B). לכן, עבור מקרה זה ההסתברות שהאלגוריתם יתכנס לפתרון לא אופטימלי היא:

$$0.4 + 0.2 = 0.6$$

5. נבדוק אילו ערכים של β עבורם ההסתברות להגיע מהמצב ההתחלתי למצב מקסימום גלובלי תוך בדיוק 3 צעדים היא גדולה מ-0.2 :

יש לחשב את הערכים של β שיגרמו למסלול הפתרון להיות: $A \to B \to F \to G$. נזכור מסעיפים קודמים שחייב להתקיים $\beta > 4$ על מנת ש- G יהיה המקסימום הגלובלי. נחשב:

$$\frac{2}{5} \cdot \frac{2}{2+\beta-2} > \frac{1}{5}$$

$$\frac{4}{\beta} > 1$$
$$\beta < 4$$

עם זאת, על מנת שזה יהיה מצב אופטימלי, צריך להתקיים $\beta > 4$.
וזאת סתירה, ולכן אין ערכים כאלה.