מבוא לבינה מלאכותית
236501

<u>דו"ח הגשה – תרגיל בית רטוב 1</u>

מתן צחור 208936989

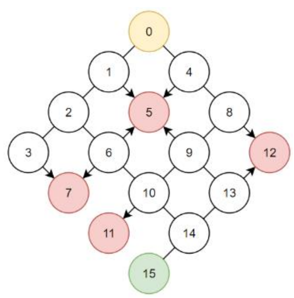
אלון פנפיל 318598166

שאלה 1

2. נגדיר את מרחב החיפוש שלנו:

$$S = \{(indx) \mid indx \in [0{,}63]\}$$
$$O = \{Down, Right, Up, Left\}$$
$$I = \{(0)\}$$
$$G = \{(63)\}$$

3. גודל מרחב המצבים S הוא 64, כמספר התאים במפת $8x8$ הנתונה.
4. $Domain(DOWN)$ תחזיר את כל המצבים פרט למצבים שמסמנים את השורה האחרונה במפה (מקיימים $s > 56$) ופרט למצבים סופיים (מוגדרים במפה כH, G). זאת מכיוון שעל פי הגדרה, פונקציית $Domain$ על אופרטור מסוים מחזירה את תת קבוצת המצבים שאפשר להפעיל את האופרטור עליהם.
5. הפונקצייה $Succ$ על המצב ההתחלתי 0 תחזיר את המצבים 1 ו- 8, כיוון והם המצבים שאליהם ניתן להגיע מהמצב ההתחלתי.
6. כן, קיימים מעגלים במרחב החיפוש, למשל המעגל: $1 \to 9 \to 10 \to 2 \to 1$, או המעגל $1 \to 2 \to 1$.
7. מקדם הסיעוף הוא $b = 4$, מכיוון וזה מספר המצבים המירבי אליו ניתן להגיע ממצב נתון.
8. גרף המצבים עבור המפת $4x4$ הנתונה הוא:



9. במקרה הגרוע סוכן $Random\ Agent$ לא יגיע לפתרון לעולם, מכיוון והוא עלול לעולם לא להגריל את המצב הסופי כתנועה הבאה.
במקרה האידיאלי שבו הסוכן האקראי מגיע במסלול הקצר ביותר ייקח לו 14 מהלכים להגיע למצב הסופי.
10. עבור מפה כללית כלשהי בסביבת הקמפוס בעלת מספר מצבי מטרה המסלול הזול ביותר לא בהכרח מביא למצב מטרה הקרוב ביותר (במונחים של מרחק מנהטן).
למשל עבור המפה הבאה:

S	L	L	L
F	H	H	L
F	H	H	L
G_1	F	F	G_2

קל לראות שבמפה הנ"ל המטרה G_1 קרובה יותר מהמטרה G_2 במרחק מנהטן (3 ל-G_1 לעומת 6 ל-G_2) אך המסלול הזול ביותר למטרה G_2 הוא בעל מחיר של 6 לעומת המסלול הזול ביותר למטרה G_1 שהוא בעל מחיר של 21.
לא ניתן לעבור ב-4 התאים המרכזיים כיוון והם בורות, לכן ניתן רק "לעקוף" אותם מלמעלה או משמאל.

שאלה 2

1. עבור בעיית הניווט בקמפוס עם מפה סופית בגודל NxN האלגוריתם BFS הוא שלם אך לא קביל. האלגוריתם שלם מכיוון ו-BFS הוא אלגוריתם שלם, אך הוא לא קביל מכיוון והמחיר על הקשתות לא אחיד.
2. על מנת שאלגוריתמים BFS ו-$BFS - G$ ייצרו ויפתחו צמתים זהים באותו סדר הוא שלא יהיו מעגלים בגרף. מכיוון ואם כן יהיו מעגלים בגרף האלגוריתם $BFS - G$ לא ייפתח את אותו צומת יותר מפעם אחת, לעומת BFS שכן ייפתח את אותו צומת יותר מפעם אחת, כך שסדר הפיתוח של צמתים ישתנה בין האלגוריתמים על אותו גרף חיפוש.
3. נגדיר את הפונקציה $T: G \rightarrow G'$ שמקבלת גרף G ומחזיריה גרף G' כך שב-G' אותם צמתים בדיוק כמו ב-G פרט לצמתי הבור אותם נסיר לגמרי מהגרף (ואת הקשתות המחברות אליהם), קשתות בעלות משקל 1 נשארות בדיוק כמו ב-G וקשתות עם משקל גבוהה יותר מ-1 מוחלפות במסלול בעל מספר קשתות וצמתים כמשקל הקשת המקורית.
נפעיל את הפונקציה T על G ואז נפעיל את $BFS - G$ על G'.
4. עבור מפה בגודל NxN ללא פורטלים וחורים כך שהמצב ההתחלתי הוא בפינה השמאלית העליונה והמצב הסופי הוא בפינה הימנית התחתונה, ייווצרו כל הצמתים בגרף ויפותחו כל הצמתים פרט לצמתים $N^2 - N - 1$ ו-$N^2 - 1$, כלומר ייווצרו N^2 צמתים, ויפותחו $N^2 - 2$. זאת מכיוון ואלגוריתם $BFS - G$ מפתח את הצמתים בצורה רוחבית (ולפי סדר הפעולות המוגדר לפיתוח $(Down, Right, Up, Left$ ולכן הצמתים יפותחו בסדר של אלכסונים מהפינה השמאלית העליונה (מהתחתון באלכסון ועד לעליון) עד לפינה ימנית תחתונה. כל צמתי הגרף יספיקו להיווצר, אך כשנגיע לפתח את צומת $N^2 - 2$ ניצור את צומת $N^2 - 1$ ונגלה שהוא צומת היעד, ולכן לא נפתח את צומת $N^2 - N - 1$ (שהוא הבא באלכסון), ולא נפתח את צומת המטרה.

שאלה 3

2. האלגוריתם $DFS-G$ עבור בעיית החיפוש בקמפוס עם מפה סופית בגודל NxN הוא שלם אך לא קביל. עבור בעיית החיפוש הנתונה וסדר פיתוח הצמתים, האלגוריתם יחפש באופן ספירלי על התאים החל מהתא השמאלי העליון מטה ← ימינה ← מעלה ← שמאלה, ומכיוון והאלגוריתם לא מפתח צומת יותר מפעם אחת, החיפוש יתכנס כלפי מרכז המפה. לכן ימצא פתרון כלשהו בסופו של דבר.
האלגוריתם לא קביל מכיוון שייתכן מסלול זול יותר, כמו למשל אם צומת המטרה היה הצומת שמיד מימין לצומת ההתחלה.
3. האלגוריתם DFS לא בהכרח היה מוצא פתרון כלשהו. אם צומת המטרה לא נמצא על הצלע השמאלית או התחתונה של המפה, האלגוריתם יגיע לתא אחד מעל התא התחתון הימני (באותו אופן כמו $DFS-G$), אך כעת מכיוון והאלגוריתם מפתח צמתים יותר מפעם אחת, יפתח שוב פעם את התא התחתון הימני (התא שממנו הגיע לתא הנוכחי), ומשם יכנס למעגל ולא יוכל למצוא פתרון.
4. 1. במהלך חיפוש $DFS-G$ יפותחו $2N-2$ צמתים וייווצרו $4N-3$ צמתים.
2. במהלך חיפוש $DFS-G\ Backtracking$ יפותחו $2N-2$ צמתים וייווצרו $2N-2$ צמתים (כיוון שצמתים נוצרים רק מיד לפני שהם מפותחים, וצומת מטרה מעולם לא מפותח)
היתרון באלגוריתם זה הוא חיסכון בזיכרון, צמתים שאינם מפותחים לא נוצרים.
5. 1. נרצה לשנות את מרחב החיפוש באופן הבא:
S, I, G יישארו זהים.
את O נשנה באופן הבא: נרצה להוסיף מכל מצב אפשרות להגיע גם למצבים הישיגים מהמצבים הישיגים שלו. למשל, עבור מצב 0, נרצה להוסיף אופרטורים שיעבירו אותנו למצבים 16, 9, 2 בנוסף למצבים 1, 8.

בצורה זו, איימי תוכל לעבור שני מצבים בבעיה המקורית במחיר של מצב אחד בבעיה החדשה, ובכך להגיע לצמתים שבבעיה המקורית היו בעומק d כשבבעיה החדשה הם בעומק $\frac{d}{2}$.

2. $b' = b + 8$.
3.

	DFS-L	DFS-L custom
סיבוכיות זמן ריצה	$O(b^d)$	$O\left((b+8)^{\frac{d}{2}}\right)$
סיבוכיות מקום	$O(b \cdot d)$	$O\left(\frac{d}{2} \cdot (b+8)\right) = O(b \cdot d)$

4. לשם הדוגמאות, נניח את סדר הפעלת האופרטורים בבעיית החיפוש החדשה הבא משמאל לימין: $(DD, RD, RR, RU, UU, LU, LL, LD, D, R, U, L)$ כאשר $D = Down, R = Right, U = Up, L = Left$.
ניקח את הדוגמאות מסביבת הקמפוס.
דוגמה בה $DFS-L$ טוב יותר במרחב החיפוש החדש לעומת במרחב החיפוש הקודם:

S	F	G
F	F	F
F	F	F

$DFS-L$ במרחב החיפוש החדש, יפתח בהתחלה את המצב ההתחלתי וייצור 5 מצבים. אחד מהם יהיה מצב המטרה ולכן הריצה תסתיים בשלב הראשון.

סה"כ יפותח מצב אחד, וייווצרו 6 מצבים (כולל המצב ההתחלתי).

$DFS-L$ במרחב החיפוש הקודם, יפתח את המצב ההתחלתי, ואז יפתח מצבים לפי סדר כפי שהוגדר בשאלה. ע"פ סדר הפיתוח, הריצה תמצא את מצב המטרה לאחר שהיא תיצור את כל המצבים, ותפתח 6 מצבים.

סה"כ יפותחו 6 מצבים וייווצרו 9 מצבים.

דוגמה בה $DFS-L$ טוב יותר במרחב החיפוש הקודם לעומת במרחב החיפוש החדש:

S	F
G	F

$DFS-L$ במרחב החיפוש החדש, יפתח את המצב ההתחלתי וייצור 3 מצבים. אחד מהם מצב המטרה, לכן הריצה תסתיים בשלב הראשון.

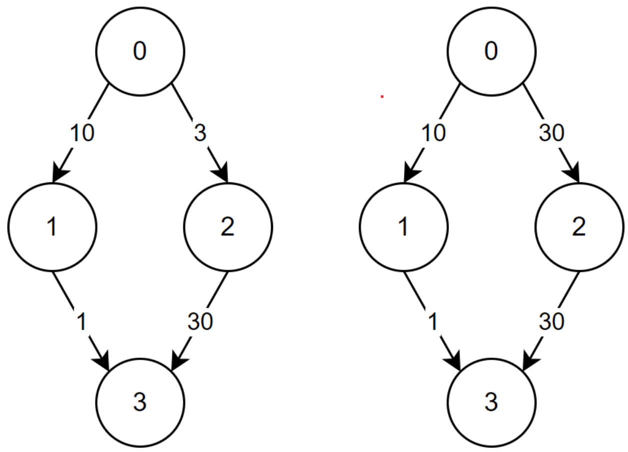
סה"כ יפותח מצב אחד וייווצרו 4 מצבים.

$DFS-L$ במרחב החיפוש הקודם, יפתח את המצב ההתחלתי וייצור 2 מצבים, אחד מהם מצב המטרה, לכן הריצה תסתיים גם היא בשלב הראשון.

סה"כ יפותח מצב אחד וייווצרו 3 מצבים.

שאלה 4

2. עבור בעיית החיפוש שלנו, עם מפה בגודל NxN, האלגוריתם UCS שלם וקביל. זאת כיוון והמחיר על הקשתות חסום מלמטה ע"י קבוע $\delta = 1$, ומקרה זה הוא גם קביל.
3. כאשר פונקציית המחיר על הקשתות אחידה לכל הקשתות בגרף האלגוריתמים $BFS - G$ ו-UCS יפעלו באותו אופן, כיוון והצמתים שנוצרים יכנסו באותו סדר לתור העדיפויות. ב-BFS הצמתים בעלי אותו **מרחק** מהשורש מוכנסים לפי הסדר הפעולות, וב-UCS אם אין הבדל בין המחיר על הקשתות אז ה-g המצטבר יהיה זהה לכל הצמתים בעלי אותו מרחק מהשורש ולכן יוכנסו באותו סדר לתור.
4. נתבונן בשני הגרפים הבאים (נקרא לשמאלי גרף 1 ולימני גרף 2) בהם צומת 0 הוא צומת התחלה וצומת 3 הוא צומת מטרה:



בגרף 1 המימוש של איימי לא יחזיר את המסלול הקל ביותר. במימוש של איימי צומת 0 יפותח ראשון, אז ייווצרו צמתים 1 ו-2. לקשת אל צומת 2 משקל קל יותר ולכן זה הצומת הבא שיפותח. צומת 2 יפותח, אז ייווצר צומת 3 וכיוון והוא הצומת האחרון, יוחזר המסלול אליו: $0 \to 2 \to 3$ עם משקל 33, בעוד שקל לראות שהמסלול הקל ביותר הוא $0 \to 1 \to 3$ עם משקל 11.

בגרף 2 המימוש לש איימי יחזיר את המסלול הקל ביותר. במימוש של איימי צומת 0 יפותח ראשון, אז ייווצרו צמתים 1 ו-2. לקשת אל צומת 1 משקל קל יותר ולכן זה הצומת הבא שיפותח. צומת 1 יפותח, אז ייווצר צומת 3 וכיוון והוא הצומת האחרון, יוחזר המסלול אליו: $0 \to 1 \to 3$ עם משקל 11. קל לראות שזה המסלול הקל ביותר בגרף.

שאלה 5

1. 1) נפריך: נניח גרף בעל 2 צמתים, צומת התחלה a וצומת סיום b כך שהמרחק מ-a ל-b הוא 10.
נניח ש-h_1, h_2 מקיימות $h_1(a) = h_2(a) = 7$ וגם $h_1(b) = h_2(b) = 0$. אזי מתקיים:
$0 \le h_1(a), h_2(a) \le h^*(a)$ וגם $h_1(b) = h_2(b) = h^*(b) = 0$, לכן קבילות. אך $h = h_1 + h_2$
אינה קבילה כי מתקיים $14 = h(a) > h^*(a) = 7$.

2) נוכיח: תהא נא h_1, h_2 יוריסטיקות קבילות, לכן לכל צומת g בגרף מתקיים עבורן:
$$0 \le h_1(g) \le h^*(g), \qquad 0 \le h_2(g) \le h^*(g)$$
(כאשר h^* הינה הערכה אידיאלית)
לכל g נסכום את המשוואות הנ"ל ונקבל:
$$0 \le h_1(g) + h_2(g) \le 2 \cdot h^*(g)$$
$$0 \le \frac{h_1(g) + h_2(g)}{2} \le h^*(g)$$
מתקיים לכל g בגרף ולכן h קבילה.

2. 1) נפריך: נשתמש באותה דוגמה נגדית כמו בשאלה 1.1. מתקיים ש $h_1(a) = h_2(a) = 7$ וש $h_1(b) = h_2(b) = 0$ ושמחיר הקשת מa לb הוא 10.
על כן, $h_i(a) - h_i(b) \le cost(a, b)$ עבור $i \in \{1, 2\}$, ומכיוון ואלה כל הצמתים והקשתות בגרף, נקבל ש h_1, h_2 יוריסטיקות קבילות.
לעומת זאת, מתקיים כי $14 = h(a) - h(b) > cost(a, b) = 10$ לכן h אינה קבילה.
2) נוכיח: תהא נא h_1, h_2 יוריסטיקות עקביות על מרחב חיפוש נתון כלשהו ונגדיר $h(s) = \frac{h_1(s)+h_2(s)}{2}$.
יהא $s \in S$ ו $s' \in succ(s)$. מתקיים:
$$h(s) - h(s') = \frac{h_1(s) + h_2(s)}{2} - \frac{h_1(s') + h_2(s')}{2} = \frac{h_1(s) - h_1(s')}{2} + \frac{h_2(s) - h_2(s')}{2} \le$$
$$\le \frac{cost(s, s')}{2} + \frac{cost(s, s')}{2} = cost(s, s')$$
כאשר באי שוויון השתמשנו בהנחה על היותן של h_1, h_2 עקביות.
לכן על פי הגדרה, היוריסטיקה h גם עקבית.

4. היוריסטיקה h_{CAMPUS} קבילה לכל מפה בבעיית הניווט בקמפוס, כיוון והערכת המרחק במרחק מנהטן תמיד קטנה מהמחיר האמיתי כדי להגיע לתא כלשהו במפה (המחיר הקטן ביותר הוא 1, וניתן לנוע רק באותו אופן כמו שמחושב מרחק מנהטן – תנועה ימינה, שמאלה, למטה, למעלה) וחסומה מלמעלה ע"י מחיר של פורטל (אם מרחק מנהטן גדול יותר ממחיר פורטל לצומת מטרה כלשהו) וגם תמיד חיוביות, לכן מקיימת את הגדרת הקבילות ליוריסטיקה.
5. היוריסטיקה h_{CAMPUS} עקבית.
נשים לב ש$|h_{CAMPUS}(s) - h_{CAMPUS}(s')| \le 1$ עבור $s \in S, s' \in Succ(s)$
זאת מכיוון שבעת התקדמות למצב עוקב קורים אחד מהדברים הבאים:
מתרחקים או מתקרבים בנורמת מנהטן יחידה אחת ממצב מטרה כלשהו ואז ההפרש הוא 1.
אם המרחק בנורמת מנהטן ממצב מקבל גדול מהמחיר של שימוש בפורטל, אז על פי הגדרת נורמת קמפוס ערך היוריסטיקה לא ישתנה וההפרש בין היוריסטיקה של המצבים יהיה 0.

כעת, מכיוון שע"פ הגדרת הבעיה, מחיר מינימלי להתקדמות ממצב אחד למצב אחר חסום מלמטה על ידי 1, מתקיים התנאי בהגדרת יוריסטיקה עקבית.

לכן h_{CAPMUS} יוריסטיקה עקבית.

שאלה 6

1. האלגוריתם על מפה כללית עבור בעיית הניווט בקמפוס שלם. ראינו בהרצאה שכל מרחב החיפוש סופי וקשיר האלגוריתם ימצא פתרון במידה וקיים ולכן שלם.
   באופן כללי האלגוריתם לא קביל, זאת מכיוון שהגישה החמדנית שgreedy best first search נוקט בה נקבעת אך ורק על פי היוריסטיקה, ולא למשל על פי המרחק מצומת המקור.
   לכן, יכול להיווצר מצב בו הבחירות של האלגוריתם יובילו לבחירת קשתות כבדות יותר מאשר בחירת מצבים עבורם היוריסטיקה גבוהה יותר וכך עלול לחזור פתרון לא אופטימלי.
   לכן האלגוריתם לא יהיה קביל.
2. יתרון של $Greedy\ Best\ First\ Search$ לעומת $Beam\ Search$:
   כפי שצוין קודם, תחת תנאים של מרחב חיפוש סופי וקשיר, $Greedy\ Best\ First\ Search$ הוא אלגוריתם שלם. לעומת זאת, תחת אותן הנחות, חיפוש אלומה לא בהכרח יהיה שלם. זאת מכיוון שייתכן מקרה בו המסלול היחיד שמגיע לפתרון באחד הפיצולים יכול להיות עם ערך יוריסטי גבוה בפיצול, ולכן חיפוש אלומה עלול להשמיט אותו מהחיפוש, וכך לפספס את הפתרון, בעוד ששימוש ב$Greedy\ Best\ First\ Search$ מבטיח למצוא את הפתרון.
   חסרון של $Greedy\ Best\ First\ Search$ לעומת $Beam\ Search$:
   חיפוש אלומה מאפשר הגבלת מספר המסלולים שיפותחו מכל מצב, דבר שמגביל את סיבוכיות הזיכרון של האלגוריתם, בעוד שב$Greedy\ Best\ First\ Search$ אין שום הגבלה על כך ולכן סיבוכיות הזיכרון שלו גבוהה יותר.

שאלה 7

2. איימי צודקת. שינוי פונקציית ההערכה לא תשנה את תוצאת החיפוש מכיוון שגם לאחר השינוי סדר ההכנסה וההוצאה של מצבים מהתור לא תשתנה. זאת מכיוון שהשינוי לא משפיע על ערכי g ו-h אלא רק על ערכי f, השינוי הוא לינארי ולכן המונוטוניות בסדר הכנסת האיברים לתור לא תשתנה. כתוצאה מכך סדר פיתוח הצמתים יישאר זהה, וכן סדר היציאה מהתור, המחיר לא מושפע ולכן נקבל את אותו המסלול עם אותו המחיר כמו שימוש ב-f.
4. יתרון של IDA^* לעומת A^*:

באלגוריתם IDA^* מספר הצמתים שעוקבים אחריהם פרופורציוני למספר הצמתים במסלול שאותו נחזיר כפתרון. בA^* מספר הצמתים שעוקבים אחריהם הוא פרופורציוני למספר הצמתים שנוצרו. לכן, סיבוכיות המקום של IDA^* טובה לפחות כמו סיבוכיות המקום שלA^* (ויכול להיות שטובה יותר).
חסרון של IDA^* לעומת A^*:
בעוד שA^* מממש חיפוש בגרף ונמנע מפיתוח צמתים יותר מפעם אחת, IDA^* לא נמנע מכך. יתר על כן, בכל איטרציה שלו מתחילים את הפיתוח לגמרי מההתחלה עם גבול חדש. לכן, מבחינת סיבוכיות זמן, קיימים מקרים בהם A^* יהיה עדיף על IDA^*.
נעדיף להשתמש באלגוריתם A^* במצבים בהם נתעדף זמן על משאבי זיכרון (למשל במרחב חיפוש מצומצם), ונעדיף להשתמש באלגוריתם IDA^* במצבים בהם נתעדף חתימת זיכרון נמוכה יותר על חשבון זמן מציאת הפתרון (למשל אם עובדים על מרחב חיפוש גדול, או אם עובדים עם מערכת עם משאבי זיכרון מצומצמים).

5. יתרון של A_ε^* לעומת A^*:

בתיאוריה, A_ε^* מאפשר לקבוע גמישות בבחירת הצמתים הבאים לפיתוח, בכך שהוא מגדיר רשימה של מועמדים לפיתוח שחורגים מהמצב האופטימלי עד כדי אפסילון. בפועל, שיטה זו מאפשרת לנו לתעדף מהירות על חשבון אופטימליות הפתרון, כאשר שולטים בפקטור החריגה המקסימלית מהפתרון האופטימלי. על כן, באופן תיאורטי, A_ε^* יכול למצוא פתרון מהר יותר מאשר A^*.
חסרון של A_ε^* לעומת A^*:
כמו שהסברנו, A^* מבטיח פתרון אופטימלי תחת תנאים מסוימים, בעוד A_ε^* מאפשר לחרוג מהאופטימליות הזאת. לכן הוא עשוי להחזיר פתרון לא אופטימלי בניגוד לA^*.
נעדיף להשתמש בA_ε^* כאשר נרצה למצוא פתרון מהר, במחיר שהוא לא יהיה אופטימלי.
נעדיף להשתמש בA^* כאשר נרצה לקבל פתרון אופטימלי, גם אם זה יהיה יקר בזמן.

שאלה 8

2. ניתוח עבור חיפושים לא מיודעים:
עבור $DFS-G$ וUCS ניתן לצפות ש:
באופן כללי, ככל שהמפה תגדל נצפה לראות שמפותחים יותר מצבים ושמחיר המסלול יעלה. זאת מכיוון שככל שהמפה גדולה יותר, המרחק בין מצב ההתחלה למצב מטרה עשוי לגדול.
כדי להימנע מאלמנט הסתברותי (של מפה אקראית כלשהי) נתייחס למפות הנתונות, שבכולן מצב ההתחלה בפינה השמאלית העליונה ומצב המטרה יחיד ונמצא בפינה הימנית התחתונה, כך שהמרחק בין מצב התחלתי למצב מטרה אכן גדל בפועל ככל שהמפות גדלות.
אכן אנחנו רואים שמחירי המסלולים שמוחזרים על ידי האלגוריתמים עולים ככל שהמפה גדולה.
כמו כן, נצפה לראות ש$DFS-G$ יפתח פחות מצבים במחיר של החזרת פתרון לא אופטימלי, בעוד ש UCS יחזיר פתרון אופטימלי במחיר של פיתוח יותר מצבים.
במקרה שלנו אנחנו רואים הבדל משמעותי (בין פי 3 לפי 6, תלוי במפה) בין מספר המצבים המפותחים על ידי UCS לעומת $DFS-G$.
מבחינת מחירים, אנחנו עדים לכך שUCS אכן מחזיר מסלולים משמעותית זולים יותר ממחירי המסלולים המוחזרים על ידי $DFS-G$.
ניתוח עבור חיפושים מיודעים:
ראשית, נשים לב שA^* שקול ל$WA^*(0.5)$. לכן, כל אלגוריתמים החיפוש המיודעים שאנחנו מפעילים הם וריאציות של WA^* שמסתמכים יותר או פחות על היוריסטיקה שהוגדרה לבעיה.
באופן כללי, ככל שערך המשקולת גדול יותר, כך האלגוריתם נותן משקל גדול יותר ליוריסטיקה בהחלטה איזה מצב לפתח.
לכן, נצפה שככל שהמשקולת קטנה יותר, נחזיר פתרון יותר קרוב לאופטימלי, במחיר של פיתוח יותר מצבים.
ככל שהמשקולת גדולה יותר, נצפה לראות שמפתחים פחות מצבים, אבל שהפתרון מתרחק מהפתרון האופטימלי.
מכיוון שהראינו שהיוריסטיקה בה אנו עושים שימוש קבילה ועקבית, נצפה לראות עבור משקולת שקטנה מחצי WA^* וA^* יחזירו את אותו פתרון (האופטימלי) ושWA^* יפתח יותר מצבים מאשר A^*.
בפועל, עבור הקלט הנתון, אנו רואים שמחיר הפתרונות אכן זהה לכל המפות כמצופה, אבל מספר המצבים שפותחו גדול יותר בקצת (4 מצבים) רק עבור המפה הראשונה ועבור שאר המפות מספר המצבים זהה.
כעת, ככל שהמשקולת גדולה יותר, נצפה לראות פתרונות רחוקים יותר מהפתרון האופטימלי, אך מספר מצבים שפותחו קטן יותר.
בפועל, האלגוריתם $WA^*(0.7)$ מצליח למצוא מסלול במשקל אופטימלי (בהשוואה לאלגוריתמים שרצים עם משקולת נמוכה או שווה לחצי) אך הוא כן מפתח פחות מצבים (בין 4 ל20 מצבים פחות, תלוי במפה).
האלגוריתם $WA^*(0.9)$ לא מצליח למצוא את הפתרון האופטימלי עבור אף מפה, פרט למפה הקטנה ביותר.
המחירים של הפתרון שהוא מחזיר חורגים מהפתרון האופטימלי ב23 יחידות עבור המפה הבינונית, וב27 יחידות עבור המפה הגדולה.
לעומת זאת, עבור כל המפות הוא מפתח משמעותית פחות מצבים משאר האלגוריתמים. בהשוואה ל$WA^*(0.7)$ הוא מפתח בין 63 ל148 מצבים פחות כתלות בגודל המפה (כצפוי, ככל שהמפה גדלה, כך ההפרש גדל).

אם היינו משתמשים ביוריסטיקה מיודעת יותר, תוצאות האלגוריתמים הלא מיודעים לא היו מושפעות, אך ניתן להניח שהאלגוריתמים ממשפחת WA^* עם משקולת גדולה מחצי היו נהנים מביצועים יותר טובים (בהקשר של מחיר הפתרון המוחזר) ביחס לביצועים עם היוריסטיקה הנוכחית.

שאלה 9

1. נגדיר את מרחב המצבים להיות כל הסידורים האפשריים של n המילים המופיעות במסמך.
2. $n!$, מספר המצבים זהה למספר התמורות על n במילים השונות במסמך, יש $n!$ תמורות שונות כאלו.
3. האלגוריתם המוצע ימצא בהכרח פתרון. בהסתמך על האופרטור הנתון, כל מעבר ממצב אחד למצב אחר יגרור את אחד השינויים הבאים בפונקציית הערך:

   להוריד את ערכה ב2: למשל אם החלפנו בין שתי מילים שנמצאו במקום שלהן במצב הקודם.

   להוריד את ערכה ב1: למשל אם החלפנו בין מילה שהייתה במקום שלה למילה שלא הייתה במקום שלה.

   לא לשנות את ערכה: למשל אם החלפנו בין שתי מילים שלא היו במקום שלהן במצב הקודם ולא במקום שלהן במצב החדש.

   להעלות את ערכה ב1: למשל אם החלפנו בין שתי מילים שלא היו במקום שלהן במצב הקודם ולאחר ההחלפה מילה אחת הגיעה למקום שלה והמילה האחרת עדיין לא במקום שלה.

   להעלות את ערכה ב2: למשל אם החלפנו בין זוג מילים שהיו במקומות אחת של השנייה.

   כעת, מהגדרת האלגוריתם המוצע, כל עוד אפשר לבצע פעולות שישפרו את פונקציית הערך ב-2 הן תתבצע. לאחר מכן, כל עוד ניתן לבצע פעולות שישפרו את פונקציית הערך ב-1 הן תתבצע.

   אם אין פעולות שמעלות את פונקציית הערך, כלומר אין אף זוג מילים שהחלפת המקומות ביניהן יעלה את פונקציית הערך. נניח בשלילה שקיימת מילה שלא נמצאת במקום שלה, נניח שזאת מילה שאמורה להיות במקום הj במסמך. במקום הj במסמך יש מילה שלא אמורה להיות שם, נסמן את המקום שלה בi. אזי החלפת המילים יעלה את פונקציית הערך בלפחות 1 בסתירה לכך שלא נשארו מעברים שמשפרים את פונקציית הערך. כלומר, כשמגיעים למצב בו אין אף מעבר שמשפר את פונקציית הערך, כל המילים נמצאות במקום הנכון. לכן האלגוריתם המוצע ימצא את הסדר הנכון.
4. 1. האלגוריתם ימצא פתרון. האלגוריתם המוצע מבצע צעדים הצידה (שלא משפרים את פונקציית הערך) רק אם אין צעדים זמינים שישפרו אותה. כפי שהסברנו בסעיף הקודם, במצב כזה אנו יודעים בוודאות שהגענו לפתרון. לכן אלגוריתם זה גם ימצא פתרון.

   2.אלון טועה. האלגוריתם יבצע צעדים הצידה רק במידה ואין צעדים עוקבים שישפרו את פונקציית הערך. במקרה זה, כבר הגענו לפתרון, ולכן שונות האלגוריתם לא תבוא בכלל לידי ביטוי בבעיה הזו.
5. האלגוריתם ימצא פתרון.

באלגוריתם זה, המעבר נבחר מתוך מעברים שמשפרים את פונקציית הערך, בהסתברות שמתאימה לכמות השיפור שהם מציעים. לכן, האלגוריתם ירוץ עד שלא יישארו עוד מעברים משפרים, ואז, כפי שכבר ציינו, נגיע לפתרון לבעיה.