תרגיל בית 1 – מבוא לבינה מלאכותית

שאלה 1:

2. נגדיר את מרחב החיפוש:

*3. גודל מרחב המצבים הוא 64, ישנן 64 משבצות על הלוח מתוכן השחקן יכול להימצא בכל אחת ואחת מהן בזמן נתון במשחק.*

*4. נגדיר את פונקציית ה-Domain על אופרטור Down:*

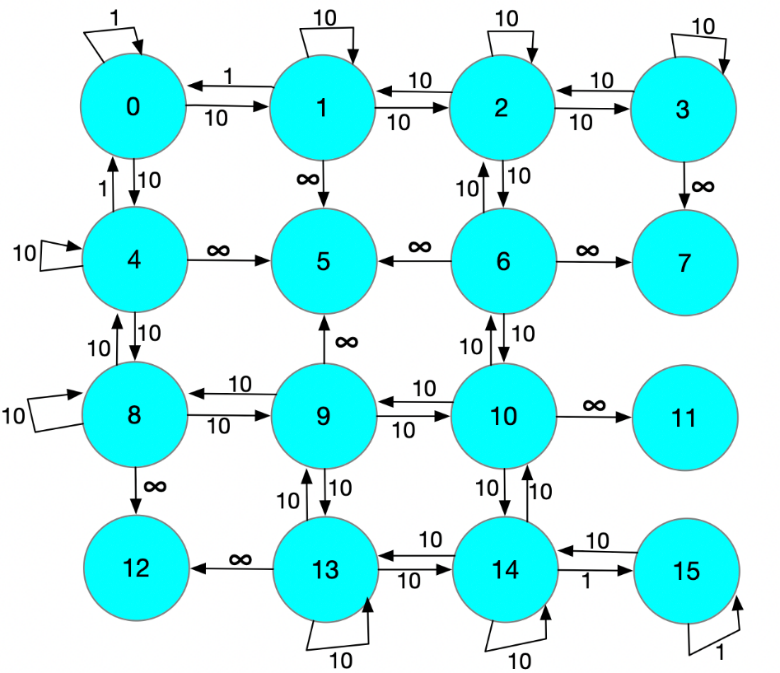
*5. נכתוב את קבוצת המצבים העוקבים של המצב ההתחלתי:*

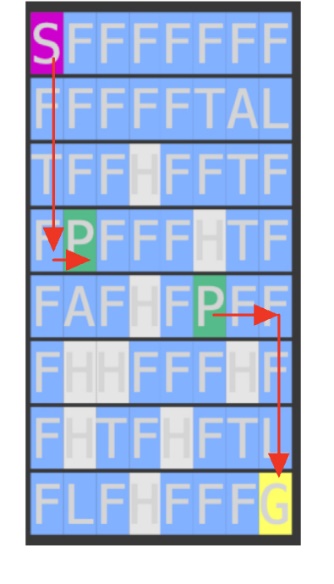
*נסביר את התשובה:*

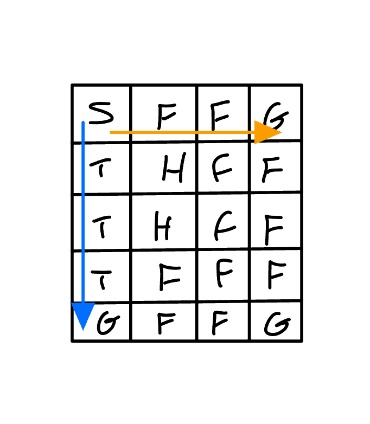
*6. קיימים מעגלים במרחב החיפוש, למשל שחקן יכול להחליט לרדת למטה ממצב כלשהו, לאחר מכן לעלות מצב אחד למעלה והתקבל מעגל.*

*7. מקדם הסיעוף בבעיית החיפוש בקמפוס הוא 4.*

*8. גרף המצבים עבור מפה* 4x4*:*

**

*9. במקרה הגרוע* random agent *לא יגיע למצב היעד, אם למשל יבחר לעלות למעלה בכל פעולה החל מהמצב ההתחלתי, כלומר פעולות.  
במקרה הטוב ביותר בהנחה שהשחקן מנחש נכון את הדרך המהירה ביותר יידרשו 9 פעולות:*

**

*10.* לא, נראה דוגמה נגדית עבור מפה כלשהי:

במסלול הכתום (הקרוב יותר לנקודת ההתחלה) עלות המסלול  
היא 1+10+10+1=22.

ואילו עבור המסלול הכחול (הרחוק יותר מנקודת ההתחלה) עלות המסלול  
היא 1+3+3+3+1=11.

שאלה 2:

1. BFS הינו אלגוריתם שלם מאחר ועובר על כל המצבים ומספר המצבים סופי. הוא אינו קביל מאחר והמחיר על הקשתות אינו אחיד.
2. בהרצאה נלמד כי בגרף החיפוש אנו לא מפתחים צומת שפותח בעבר ואילו בעץ החיפוש כן ניתן לפתח צומת שכבר פותח בעבר. לכן כדי לדרוש תוצאה זהה עבור ריצת BFS-G ו-BFS, נדרוש כי הגרף שממנו יוצרים את גרף החיפוש יהיה חסר מעגלים ובפרט כל צומת יהיה בעל הורה יחיד. בצורה זו כל צומת יפותח פעם אחת ובפרט נקבל את הסדר הזהה לריצת BFS.
3. על מנת שריצת BFS-G תחזיר פתרון אופטימלי נצטרך שעלויות הקשתות בגרף שעליו נריץ את האלגוריתם יהיו אחידות. ולכן נגדיר את G' בצורה הבאה:

נבנה את גרף החיפוש G מתוך המפה. לאחר מכן עבור כל צומת שעלותו גדולה מ-1 נשכפל למספר צמתים כעלות הצומת בצורה של שרוך. נוסיף לתחילת השרוך את הקשתות שנכנסות לצומת שאותו משכפלים מ-G. בסוף השרוך נחבר את הקשתות שיוצאות מהצומת שאותו משכפלים מ-G.

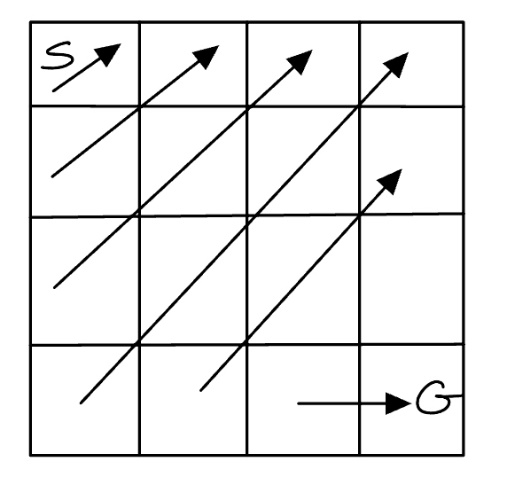
עם הצומת הוא בעל עלות 1 נכניס אותו לגרף G' כפי שהוא, עם הקשתות שיוצאות ממנו ונכנסות אליו.

עבור צומת H נכניס אותו כפי שהוא מאחר ואין ממנו קשתות יוצאות ובפרט הוא לא מביא לפתרון.

1. מספר הצמתים שיוצרו :

בהנחה שאם יצרנו צומת אז אנחנו לא יוצרים אותו שוב אם הגענו אליו משכן אחר , ובהנחה שלקחנו בחשבון את היצירה של S.

מספר הצמתים שיפותחו :

לצומת היעד G יש 2 שכנים שנוצרו אבל כדי להגיע ל G מספיק שאחד מהם יפותח. ברגע הפיתוח נגלה את צומת G ונסיים את ריצת האלגוריתם. לפי סדר הפעולות לפיתוח הצומת ההתקדמות של האלגוריתם נראית כך :

שאלה 3 :

1. שאלת רטוב
2. A grid with letters and arrows

   Description automatically generatedהאלגוריתם שלם כי N סופי (אין מסלולים אינסופיים) וכל צומת נפתח פעם אחת בלבד (אין מעגלים).

האלגוריתם אינו קביל , למשל עבור הלוח הבא:

מכאן ניתן לראות שהמסלול של DFS (באדום, מצויר חלק ממנו)

כבר יקר יותר מהריצה של BFS (טורקיז)

1. הרצת אלגוריתם DFS על עץ לא מוצא פתרון.

אחרי הרצת DFS והגעתו לצומת (הצומת בשורה האחרונה והעמודה הראשונה במפה) ואז כל פעם האלגוריתם ירד למטה ויחזור על אותו צומת שוב ושוב (כיוון שהרצנו על עץ יכולים לחזור על אותו צומת שוב ושוב) ובכך נתקענו בלולאה אינסופית.

1. 1. מספר הצמתים שפותחו :

DFS יתחיל מצומת S וכל פעם ירד למטה עד להגעתו לצומת ,כלומר יעבור על N צמתים. ואז DFS יתחיל ללכת ימינה עד הגעתו לצומת היעד G (אי אפשר לעבור על אותו צומת יותר מפעם אחת ולרדת למטה יחזיר אותנו לאותו צומת). צומת היעד רק נוצר ולא פותח ולכן לא נספור אותו בצמתים שפותחו , כלומר פותחו עוד צמתים נוספים. וסה"כ :

מספר הצמתים שנוצרו :

תחת ההנחה שצומת ההתחלה S נוצר באופן אוטומטי בתחילת הריצה עד לצומת התחתון ביותר בעמודה הראשונה (צומת ) אנו יוצרים את הצמתים במסלול מטה ואת שכניהם מצד ימין. לכן למסלול מטה אנו יוצרים צמתים, לאחר מכן אנו הולכים ימינה עד לצומת , ובכך נעבור על צמתים עד לצומת היעד כאשר את שני הצמתים הראשונים לא נספור ביצירה ובנוסף, לא נספור את השכן העליון של הצומת השני. מכאן נייצר עוד מאחר ולא סופרים את השכן העליון של צומת היעד (לא מפתחים את צומת היעד). בסה"כ יצרנו .

2. מספר הצמתים שפותחו:

תחת ההנחה שצומת היעד אינו מפותח, אנו נפתח את הצמתים החל מצומת ההתחלה S ועד לצומת התחתון ביותר בעמודה הראשונה (צומת ), עד כאן פיתחנו צמתים. מכאן נלך ימינה עד לצומת היעד G (לא נספור שוב את הצומת השמאלי התחתון ביותר). אנו עוברים על צמתים עד לצומת G שאותו לא מפתחים ולכן בסה"כ קיבלנו: .

מספר הצמתים שנוצרו:

תחת ההנחה שצומת ההתחלה S נוצר אוטומטית (לכן לא נכלל בספירה).

במסלול מטה מצומת ההתחלה S אנו יוצרים צמתים. מכאן אנו ממשיכים ימינה לצומת היעד G. נעבור בעוד צמתים ואותם נייצר. לכן בסה"כ יווצרו צמתים.

1. 1) נציע את השינוי הבא לבעיית החיפוש:

נאפשר לאיימי יותר אופרטורים שיתבטאו כצעדים הבאים:

הסבר – אנו מאפשרים כעת לאיימי לבצע את הצעדים הרגילים מבעיית החיפוש ללא השינוי – . בנוסף, נאפשר לאיימי לבצע קבוצת פעולות חדשה – ביצוע ולאחר מאחר כך שאיימי יכולה לחבר שתי פעולות יחדיו, למשל איימי יכולה לבחור שהיא מבצעת פעולת ולאחר מכן . לאיימי יתאפשרו לבצע כל קומבינציה אפשרית של חיבור שתי פעולות אחת אחרי השנייה.

כעת ניתן למצוא פתרון בעומק וזאת מכיוון שקבוצת האופרטורים החדשה מאפשרת לאיימי לבצע פעולה אחת שללא השינוי נאלצה לבצע בשתי פעולות. לכן אם אורך המסלול הקצר ביותר הוא בעומק , לאחר הוספת האופרטורים החדשים, עומק הפתרון יהיה .

2) מאחר ושינינו את קבוצת האופרטורים, מקדם הסיעוף אכן ישתנה מאחר ויתכנו יותר עבור צמתים מסוימים. מקדם הסיעוף מהבעיה ללא שינוי הוא ואליו יתווסף ערך נוסף המתבטא כתוצאה מהוספת האפשרות לשני צעדים בפעולה יחידה, יש בנים לאחר פעולה אחת ולכל אחד מהם יש בנים (פעולה שנייה). לכן בסה"כ נקבל:

3) סיבוכיות הזמן החדשה – .

סיבוכיות המקום החדשה –

עבור בעיית החיפוש ללא שינוי נקבל:

סיבוכיות הזמן –

סיבוכיות המקום –

4) מהסעיף הקודם ניתן לראות שסיבוכיות המקום עבור שינוי הבעיה גבוהה מאשר הבעיה ללא השינוי. השיקול יהיה מהירות ההגעה לפתרון (מספר הצמתים שפותחו) לעומת שימוש בזיכרון (מספר הצמתים שאנו שומרים בזיכרון).

דוגמה להעדפה של DFS-L עבור בעיית החיפוש החדשה שהגדרנו:

מפות שבהן המסלולים לצומת היעד ישרים, זיגזג, או מסלול לצומת היעד שהוא צירוף של שתי פעולות בכל צעד.

דוגמה להעדפה של DFS-L עבור בעיית החיפוש הישנה:

מפות שבהן המסלול לצומת היעד דורש צעד יחיד.

עבור הדוגמה הראשונה, נפתח מחצית מהצמתים בדרך לפתרון לעומת בעיית החיפוש הרגילה.

עבור הדוגמה השנייה נפתח אותו מספר צמתים אך בבעיית החיפוש החדשה נשמור יותר צמתים בזיכרון מאחר וכמות ה-succ גדלה.

*שאלה 4 :*

1. *ממומש.*
2. *האלגוריתם שלם , לפי מה שלמדנו בתרגול האלגוריתם שלם אם קיים חסם תחתון לפונקציית המחיר שגדול מ-0 והיא חסומה ע"י 1 (עלות הקשת הנמוך ביותר).*

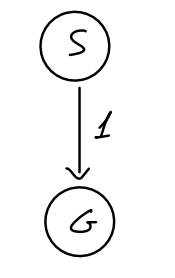
*האלגוריתם קביל מאחר ואופן הפעולה של* UCS *הוא תמיד לבחור את הקשת בעלת העלות הנמוכה ביותר של המסלול שמגיע עד אליה. אם נבחר תמיד במסלול הקל ביותר, נגיע לפתרון המחזיר את העלות הנמוכה ביותר עד לצומת היעד, מכאן שהאלגוריתם קביל.*

1. האלגוריתמים י*פעלו באותו אופן אם כל הקשתות בגרף בעלי עלות אחידה.*

*ההבדל בין האלגוריתמים הוא סדר פיתוח הצמתים ב-*OPEN*:*

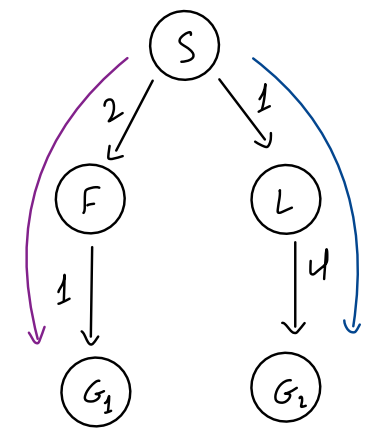
BFS *מפתח לפי סדר* FIFO *ו-*UCS *מפתח לפי ערךg מינימלי. אם מחיר הקשתות אחיד סדר ההוצאה של הצמתים יהיה לפי סדר הכנסתם ל-*OPEN*.*

1. *דוגמה עבורה האלגוריתם השגוי מחזיר את המסלול הקל ביותר :*

**

*יש מסלול אחד בלבד והוא הזול ביותר ולכן יחזיר אותו והמחיר שלו יהיה 1 .*

*דוגמה עבורה האלגוריתם השגוי לא מחזיר את המסלול הקל ביותר:*

**

*האלגוריתם השגוי יבחר את צומת* L *בגלל הקשת שערכה 1 ולאחר פיתוח צומת* L *הוא ייצור את צומת ויגלה שזו צומת מטרה. בגלל אופן מימוש האלגוריתם, הוא יבחר במסלול כחול ויחזיר אותו כאשר עלות המסלול היא 5.*

*באלגוריתם הנכון יפותחו שני צמתי המטרה ולכן המסלול הסגול יתגלה כזול יותר והוא יוחזר. עלות המסלול היא 3.*

*שאלה 5:*

1. *A diagram of a chemical reaction

   Description automatically generatedעבור שתי היוריסטיקות קבילות מתקיים: .*
   1. *נפריך את הטענה:*

*נציע את הדוגמה הנגדית הבאה:*

*כאשר מתקיים:*

*ההיוריסטיקות ו- קבילות מאחר והן מקיימות את תנאי אי השוויון.*

*ולפי הדוגמה מתקיים: מכאן שהחיבור שלהן אינו קביל.*

* 1. *נוכיח את הטענה:*

*היוריסטיקה קבילה.*

1. *עבור שתי היוריסטיקות עקביות מתקיים:*

* 1. *נפריך את הטענה:*

*נציע את הדוגמה הנגדית הבאה:*

*מתקיים:*

*בנוסף,*

*ולכן עקביות, אך:*

* 1. *נוכיח את הטענה:*

*עבור שתי ההיוריסטיקות מתקיים:*

*ולכן:*

*ההיוריסטיקה קבילה.*

1. *נגדיר את*
2. *נוכיח ש- קבילה:*

*עבור מצב מסויים, כדי להגיע ממנו לצומת יעד במחיר הזול ביותר ישנן 2 אפשרויות:*

1. *להשתמש ב-*portal *ואז:*
2. *לא להשתמש ב-*portal*:*

*נגדיר את הערכים הבאים:*

*– העמודה של המצב הנוכחי שבו אנו נמצאים – .*

*– השורה של המצב הנוכחי שבו אנו נמצאים – .*

*– העמודה של מצב היעד .*

*– השורה של מצב היעד .*

*על מנת להגיע ליעד כך שהמחיר שלו הוא הזול ביותר נבצע:*

*צעדים ימינה ו- צעדים למטה.*

*כלומר בסה"כ: צעדים ולפי הגדרה זהו מרחק מנהטן.*

*A yellow square with black letters and numbers

Description automatically generatedולכן בסה"כ:*

1. *נפריך את הטענה:*

*עבור המפה הבאה וצומת , מתקיים:*

* *המסלול הסגול הוא מרחק מנהטן מ- ולצומת היעד*

*לכן מתקיים:*

*שאלה 6:*

1. *האלגוריתם שלם מאחר ומרחב המצבים סופי. כפי שראינו בתרגול ההיוריסטיקה רק מחליטה על הסדר. במצב הגרוע ביותר נעבור על כל הצמתים עד למציאת הפתרון.*

*האלגוריתם אינו קביל מאחר והיוריסטיקה יכולה להטעות אותנו.*

1. *יתרון של* Greedy Best First Search *לעומת* Beam Search*:*

*שלם, כלומר ימצא פתרון אם הפתרון קיים, טיב הפתרון יהיה טוב יותר.*

*A diagram of a diagram

Description automatically generatedיתכן מצב ש-*Beam Search *לא ימצא פתרון למשל עבור החיפוש הבא:*

*אם נקבע האלגוריתם עלול לבחור*

*את צמתים* A *ו-*B *ל-*open *ואילו לזרוק את צומת* C *ובכך להפסיד*

*את המסלול לצומת* target *שהוא צומת היעד.*

*אלגוריתם* Greedy Best First Search *ימצא את צומת*

Target*.*

*חסרון של* Greedy Best First Search *לעומת* Beam Search*:*

*זמן הריצה של אלגוריתם* Greedy Best First Search *עלול להיות ארוך מזה*

*של* Beam Search *מאחר ו-* Beam Search*הוא חיפוש ממוקד יותר. בפרט, כמות הזיכרון שאלגוריתם* Greedy Best First Search *ישתמש יהיה גבוה יותר מ-*Beam Search *מאחר ואין הגבלה על גודל ה-*open *ב-*Greedy Best First Search.

*שאלה 7:*

1. *מומש במחברת ה-*jupyter*.*
2. *איימי צודקת מאחר והכפלת הפונקציה פי חצי כדי לקבל את שומרת על הסדר של פיתוח הצמתים.*

*למשל, אם בקבוצת ה-*open*, שני צמתים מקיימים אזי בפרט:*

*(תחת ההנחה שהמחירים אי שליליים)*

*לכן, המסלול שיחזור יהיה שקול למסלול עבור וכך גם עלות המסלול מאחר ועלות המסלול היא הסכום של ה--ים של הצמתים שבהם עברנו והם אינם משתנים.*

1. *מומש במחברת ה-*jupyter*.*
2. *יתרון של ביחס ל-:*

*בהתאם למה שראינו בהרצאה הוא בעל חתימת זיכרון נמוכה מזו של .*

*עבור צריכת הזיכרון היא ואילו עבור צריכת הזיכרון היא .*

*לכן, משיקולי זיכרון נעדיף להשתמש ב-.*

*חיסרון של ביחס ל-:*

*הריצה שלו תיקח יותר זמן מאחר ובכל פעם אנו מעדכנים את ה-*limit *כדי לאפשר פיתוח של צמתים נוספים שעד ערך ה-*f *שלהן היה גבוה מה-*limit *ואז אנחנו מתחילים את הריצה שוב ומעבר על הצמתים ופיתוח שלהם שוב למרות שכבר פותחו בעבר.*

*במקרים בהם אין לנו הגבלת זיכרון נעדיף להשתמש ב-.*

1. *יתרון של ביחס ל-:*

*אלגוריתם זה מאפשר בחירה לא אופטימלית של הצומת הבא לפיתוח וזאת במטרה לנסות למצוא פתרון מהר יותר מ- כאשר לנו בעיה להתפשר במעט על טיב הפתרון (עד כדי פקטור של מהפתרון האופטימלי).*

*חסרון של ביחס ל-:*

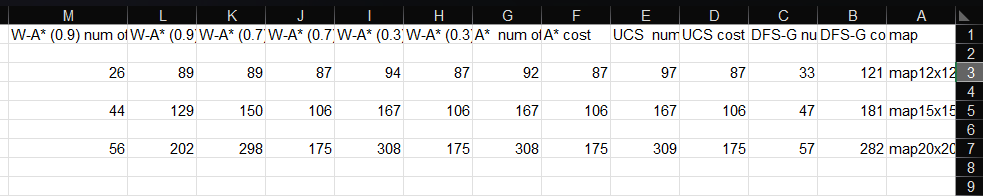
*הפתרון לא בהכרח אופטימלי שכן יש סטייה מהאופטימליות על פי בחירת המשתמש (פקטור של ).*

*מתי נשתמש בזה ומתי בזה?*

*אם אנו מוכנים להתפשר על סטייה מסוימת מהפתרון האופטימלי על מנת לשפר את הזמן למציאת פתרון, נשתמש ב-. מצד שני, אם חשוב לנו למצוא את הפתרון האופטימלי כאשר אנו מבינים שזה יכול לקחת זמן נשתמש ב-.*

*שאלה 8:*

1. *נצרף את התוצאות המוצגות בקובץ:*

**

1. *ננתח את התוצאות:*

* *תחילה נבחין כי האלגוריתמים עם המשקלים: 0.7, 0.3, אלגוריתם ו- מחזירים את עלות המסלול הנמוך ביותר וזה מתאים לציפיות מאחר וככל שאנו יורדים במשקל שאנו מעבירים ל- אנו מתקרבים יותר ויותר לפתרון של . את ההתנהגות הזו רואים גם כאשר מסתכלים על כמות הצמתים שפותחו בכל אחד ואחד מהאלגוריתמים.*
* *דבר נוסף שמתאים לציפיות הוא שככל שהמשקל של גדל, כמות הצמתים המפותחים קטנה (כלומר האצנו את ההגעה לפתרון), אך איכות הפתרון נפגעת (עלות המסלול המוחזר גדלה).*
* *עבור אלגוריתם קיבלנו שמספר הצמתים שפותחו יחסית קטן ביחס לרוב האלגוריתמים וזה הגיוני מאחר שאין שימוש ב-"חוש ריח" אלא פשוט להעמיק קדימה כמה שניתן עד להגעה לפתרון. מאחר והאסטרטגיה היא כזו הציפיה היא למחיר גבוה וזה אכן מה שקרה ביחס לשאר האלגוריתמים שניסו להשתמש בידע על המסלול.*
* *שימוש בהיוריסטיקה מיודעת יותר הייתה משנה את התוצאות בכך ש...*

*שאלה 9:*

1. *נגדיר את המצבים כווקטור בגודל* n *כאשר הערך שבמיקום ה- בווקטור מציין את מקומה של המילה ה- במסמך.*
2. *מספר המצבים הוא שכן יש לסדר מילים בשורה (ווקטור).*
3. *נבחין בהתנהגות הבעיה – החלפה בין זוג מילים יכולה להביא למצבים הבאים:*
   1. *החלפת שתי מילים ככה שאין שיפור בסידור המסמך.*
   2. *החלפת שתי מילים ככה ששיפרנו את מצב המסמך במילה אחת.*
   3. *החלפת שתי מילים ככה ששיפרנו את מצב המסמך בשתי המילים שביניהן החלפנו.*

*מאחר ואלגוריתם* SAHC *יבחר תמיד במסלול משפר אנחנו נהיה במצבים b או c עד שנגיע למצב שאין אפשרות לשפר, כלומר הגענו לסידור הקובץ הנכון. מכאן שהאלגוריתם בהכרח ימצא פתרון.*

1. *1) מאחר ותמיד יהיו מצבים עוקבים משפרים עד להגעה לפתרון (קובץ מסודר), אלגוריתם*

*ה-*SAHC with sideways steps *יתנהג כמו* SHAC *ולכן גם הוא בהכרח ימצא פתרון.*

*2) אלון טועה מאחר וההתנהגות של האלגוריתמים תהיה זהה.  
אלגוריתם* SAHC with sideways steps *פונה ל-*sideways steps *כאשר אין לו אפשרות לשיפור. כאן זה לא המקרה כפי שצוין בסעיף 3.*

1. *אלגוריתם זה בהכרח ימצא פתרון מאחר והוא בוחר בצורה הסתברותית מבין המצבים המשפרים. יתכן מצב שישפר את הקובץ ב-2 (כלומר* U=2, *וזה אומר ששיפרנו עבור שתי המילים שהחלפנו ביניהם) והאלגוריתם יבחר מצב שישפר את הקובץ ב-1 (הבחירה במצב היא הסתברותית). עדיין, האלגוריתם יתקדם וישפר את מצב המסמך, ובצעדים הבאים יגיע לצעד שיחליף בין שתי המילים שדילג עליהן בשלב הנוכחי.*