**AI – HW1**

Nadav Levinton 203906425 , Or Asraf 302761895

חלק א'

|  |  |
| --- | --- |
| **פרמוטציות** | **K** |
| 1 | 1 |
| 6 | 2 |
| 90 | 3 |
| 2520 | 4 |
| 113400 | 5 |
| 7484400 | 6 |
| 681080400 | 7 |
| 81729648000 | 8 |
| 1.25046E+13 | 9 |
| 2.37588E+15 | 10 |

1. הטבלה:

חלק ג'

1. מקדם הסיעוף המינימלי הינו 0 כאשר המצבים שעבורם מקדם הסיעוף הינו 0 הם מצבי מטרה שעבורם מתקיים כאשר v הינה צומת על המפה, זאת משום שכאשר אין אופרטור חוקי המאפשר מעבר למצב אחר.

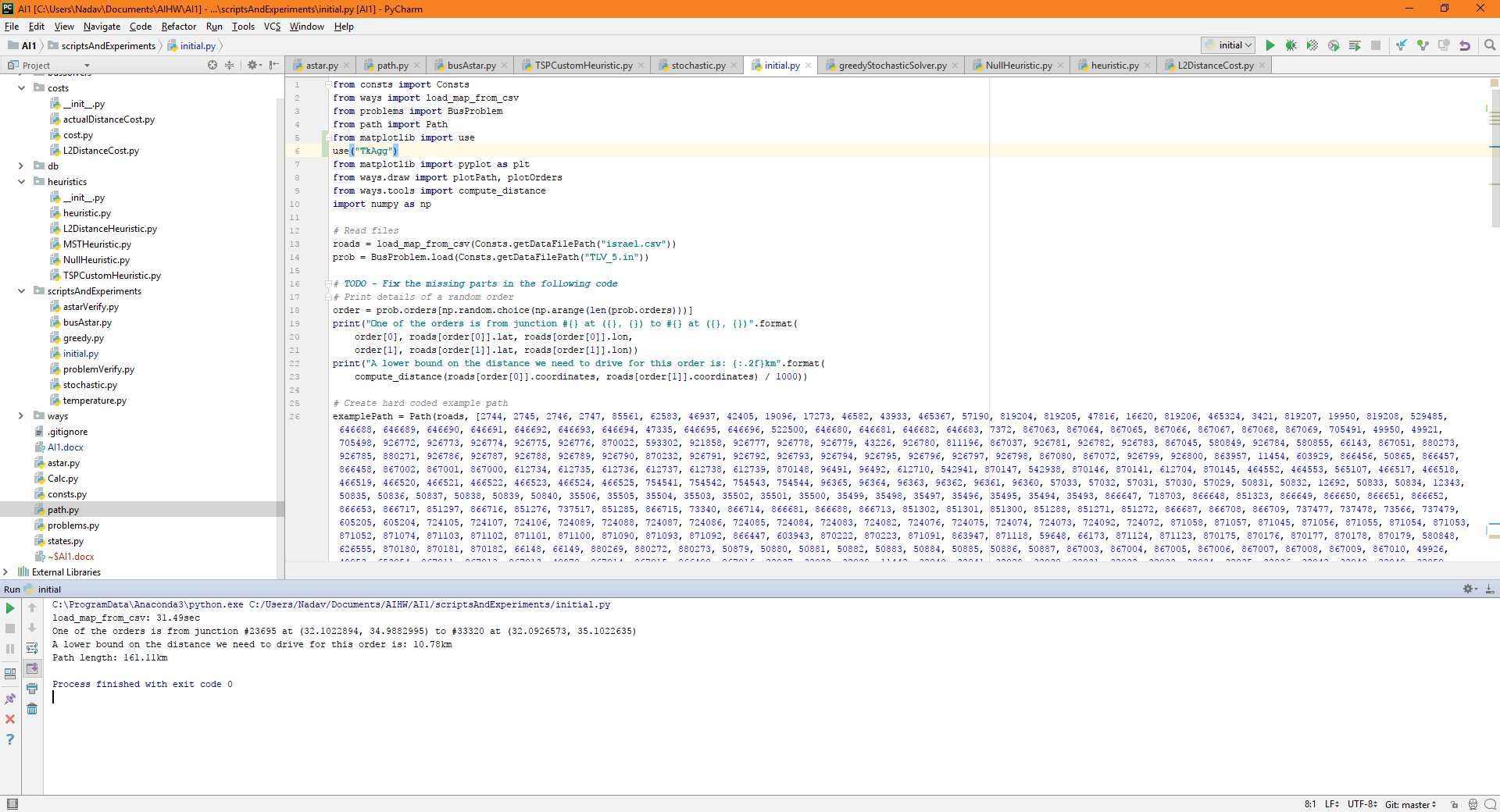
מקדם הסיעוף המקסימלי הינו k והוא ייתכן כאשר כל הזמנה הינה מצומת אחר במפה וכן קיים מסלול ישיר מצומת ההתחלה לכל אחד מצמתים אלו – כלומר, האוטובוס יכול לבחור כל אחד מההזמנות לאיסוף ראשון.

1. לא יכולים להיות מעגלים במרחב החיפוש משום שלפי האופרטורים החוקיים קבוצת ההזמנות הממתינות יורדת מונוטונית וכן קבוצת ההזמנות שבוצעו עולה מונוטונית ולפי הגדרת האופרטורים לפחות אחת מהן משתנה בעקבות הפעלת אופרטור ולכן אין מצב שבו יתקיים עבור שני מצבים כך וגם וכן נמצאת אחרי על אותו מסלול.
2. G הם קבוצת המצבים כך שלא נותרו על האוטובוס הזמנות וכן כל ההזמנות ברשימה המקורית בוצעו. הגדרה פורמלית:
3. לפי הגדרת האופרטורים והנתון בשאלה לפיו אין שתי הזמנות שמתחילות ונגמרות במיקומים זהים נסיק כי בכל מעבר בין מצבים מתקיים אחד מהמקרים הבאים: או שאדם אחד עולה על האוטובוס, או שאדם אחד יורד מהאוטובוס או שאדם אחד עולה ואדם אחד יורד. לכן, ב-G יש מצב עבור כל הזמנה כך שאדם זה הינו האחרון שיורד מהאוטובוס (לא יכולים להיות שני אנשים שונים היורדים באותה תחנה) => הביטוי המתאים הינו:
4. לא יתכנו בורות שאינם מצבי מטרה משום שנניח שאנו במצב S כלשהו שאינו מצב מטרה לכן בהכרח קיים מצב S’ שונה מ-S כך שבמצב זה מורידים או מעלים לפחות נוסע אחד וכן קיים מסלול במפה בין ו- (ההנחה לגבי המסלולים במפה ניתנת ב-FAQ ).
5. פונקציית העוקב:
6. הערך המקסימלי הינו מצב זה יקרה כאשר בכל תחנה או נאסוף נוסע או נוריד נוסע כלומר מספר האנשים על האוטובוס בכל רגע נתון הוא לכל היותר 1, כלומר מספר הנוסעים על האוטובוס יהיה : 0, 1, 0, 1 וכו'.

הערך המינימלי הוא k+1 במקרה זה בכל תחנה נאסוף נוסע ונוריד נוסע ולכן מלבד במצב ההתחלה ובמצב הסופי על האוטובוס יש בדיוק נוסע 1 ולכן נבצע k מעברים בכדי לאסוף k נוסעים וכן נעשה מעבר נוסף בכדי להוריד את הנוסע האחרון.

חלק ד'

1. *הקוד טוען את קובץ Israel.csv, טוען את ההזמנות מ-TLV\_5.in, בוחר בצורה רנדומלית הזמנה,* *מדפיס את המרחק המינימלי שצריך לעבור על מנת לבצע את ההזמנה, מדפיס את המרחק המינימלי ומדפיס את אורך המסלול.*



1. *הפלט שהתקבל:*

*Junction idx: #851288*

*waiting for bus: [(23695, 33320)]*

*orders on bus: [(851288, 533396), (32056, 834603)]*

*finished orders: [(47521, 606430), (466524, 29249)]*

*Is goal? False*

*חלק ה'*

1. *הערך שהתקבל הינו :77.35 ק"מ.*

**

1. *התשובה היא C.  
   לא ניתן לקבוע באופן כללי משום שהמספר המתקבל הינו סכום כל המסלולים על כל ההזמנות בנפרד, בסכום זה אין התחשבות במסלולים חופפים (לכן זה אינו חסם תחתון) וכן אין התחשבות במרחק בין הזמנה להזמנה (למשל המרחק בין S4 ל-S3 בגרף לא מחושב – ולכן זה אינו חסם עליון).*

*חלק ו'*



|  |  |
| --- | --- |
| Solution cost (total distance in KM) | Input file |
| *135.06km* | TLV\_5 |
| *539.59km* | SDEROT\_50 |
| *565.68km* | HAIFA\_100 |
| *1329.21km* | BEER\_SHEVA\_100 |

1. *חסם תחתון : 2 צמתים אם כל ההזמנות יוצאות מV ומגיעות ל U .*

*מ* I *יפותח רק המצב X = וממנו יפותח רק המצב Y=*

*חסם עליון : 2k, נשים לב שבמצב בו כל הזמנה נאספת מצומת אחת שאינה משותפת לאף צומת אחרת (הן באיסוף והן ובהורדה) וכנ"ל לגבי נקודת ההורדה נקבל כי בכל צעד נטפל רק בהזמנה אחת ובגלל שיש k הזמנות ולכל הזמנה 2 נקודות נקבל כי החסם הינו 2k.*

1. *סיבוכיות המקום של האלגוריתם היא O(k) כאשר k הוא מספר ההזמנות. פלט האלגוריתם הינו סידור של צמתי ההזמנות=> רשימה של 2k נקודות הורדה/ איסוף במקרה הגרוע כפי שכתבנו בסעיף הקודם. בנוסף, פונקציית Succ יכולה להחזיר לכל היותר O(k) צמתים במקרה הגרוע שבוע יש k אפשרויות למצב הבא (או יש k הזמנות שניתן לאסוף או שיש k הזמנות על האוטובוס שיש להוריד).*

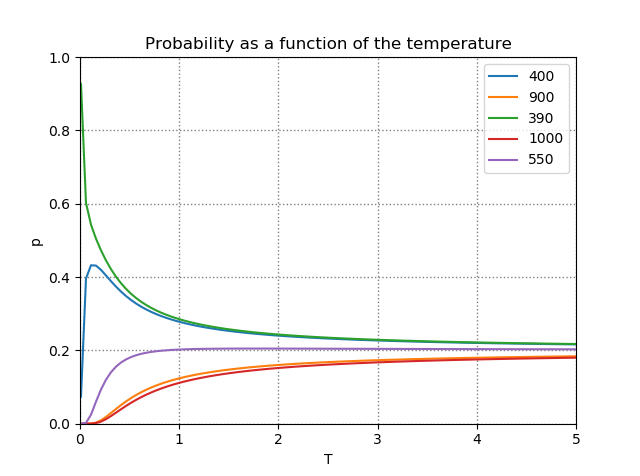
*חלק ז'*

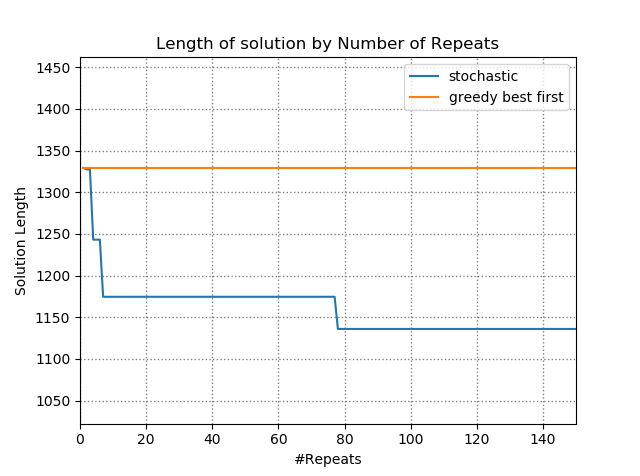
1. *נוכיח ששינוי הסקאלה אינו משנה את התפלגות הווקטור, כלומר נראה שעבור ווקטור נתון מתקיים:*

*תהי נפריד למקרים:*

1. =>
2. =>

נשים לב כי לעולם לא יהיה 0 כי פונקציות ה-Score שלנו מחשבת מרחק אווירי ולעולם לא נקבל מרחק אווירי 0 עבור שני מצבים עוקבים – כי אז הם היו אותו מצב.

1. הגרף המתקבל:  
   
2. על פי התבוננות בגרף נראה כי כאשר נקבל כי ההסתברות של הנקודה הטובה ביותר לפי פונקציית העלות (נסמנה ) תקיים . זאת משום שכפי שצויין בחלק ז' במסמך ככל ש-T קטן יותר נבחר אפשרויות "בטוחות יותר" ובמקרה שלנו הנקודה הקרובה ביותר אווירית הינה האפשרות הבטוחה ביותר שהיא גם הקרובה ביותר בפועל.
3. על פי התבוננות בגרף נראה כי כאשר נקבל כי ההסתברות של כל אחת מ-N הנקודות הטובות יותר לפי פונקציית העלות תקיים . זאת משום שכפי שצויין בחלק ז' במסמך ככל ש-T יגדל יותר ההסתברות של כל אחת מהנקודות האפשרויות תראה לנו שווה ולכן לכל נקודה ההסתברות תשאף ל-



* + ממוצע מדגמי: 1298.53744667

סטיית תקן מדגמית:

P-Value:

* + על פי מה שנלמד בתרגול נדחה את השערת ה-0 משום שקיבלנו .
  + נתבונן בנתונים שקיבלנו עבור הנקודה הראשונה, נשים לב שתוצאת האלגוריתם החמדני הינה 1329.21 וכי ממוצע הרצות האלגוריתם הסטוכאסטי הינה כ-1298.  
    ולכן נסיק שרוב הרצות האלגוריתם הסטוכאסטי יפיקו תוצאה נמוכה מהתוצאה של האלגוריתם החמדן (כלומר התוחלת שלהן קטנה באופן משמעותי מתוצאת האלגו' החמדן) ולכן השערת האפס שגויה.

1. *בdataset שלנו יש 5 הזמנות עם מוצאים ויעדים שונים כלומר 10 נקודות שונות על המפה וכולן שונות מנקודת ההתחלה של האוטובוס. מספר המסלולים עד למצב מקבל כלשהו שקול למספר הפרמוטציות החוקיות על איסוף והורדת ההזמנות כולן. בשלב ראשון ניתן לאסוף כל אחד מ5 ההזמנות. לאחר מכן ניתן לאסוף כל אחד מ4 ההזמנות האחרות או להוריד את ההזמנה שעל האוטובוס. וכן הלאה . נחשב בעזרת רקורסיה. נגדיר:*

*numberOfRouts(n,k) =*

*כאשר n הוא מספר ההזמנות שמחכות וk מספר ההזמנות שנמצאות על האוטובוס. נוסחא זו נכונה רק עבור מצבים בהם אין חפיפה בין נקודות העלאה והורדה בהזמנות.*

*אז: numberOfRouts (5,0) =113400*

1. *התוצאות שהתקבלו:*

*Greedy result: 135.06km*

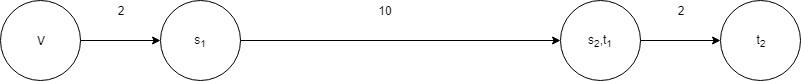
*Stochastic (200 repetitions): 132.682km*

*A\* (null heuristic): g(G)=127.29km, h(I)=0.00km, developed: 1640 states*

1. **נשתמש בהגדרת היוריסטיקה מיודעת מההרצאה.**
2. **קבילה**: הפונקציה מחזירה את המרחק האווירי הגדול ביותר שיש לעבור עבור הזמנה שעדיין לא עלתה על האוטובוס (מרחק אווירי מקסימלי מנק' האיסוף של ההזמנה עד נק' ההורדה של אותה הזמנה), נשים לב שלא יכול להיות שנבצע בפועל מרחק קטן יותר מהמרחק האווירי הנ"ל עד למצב המטרה משום שעלינו לאסוף את ההזמנה ולאחר מכן להגיע אל יעדה וכפי שצוין בתרגול לא יכול להיות כביש קצר מהמרחק האווירי. בנוסף, במידה ויש יותר מהזמנה אחת יתכן ונצטרך לעבור מרחק יותר גדול מהמרחק שהפונקציה חישבה. לכן, פונקציה זו תמיד תעריך בצורה אופטימית את המרחק מהמצב הנוכחי למצב המטרה.
3. **קבילה**: הפונקציה מחזירה את המרחק האווירי הגדול ביותר שיש לעבור עבור בין נקודת איסוף ונקודת הורדה מבין הזמנות שעוד לא אספנו (מרחק אווירי מקסימלי מנק' איסוף של הזמנה כלשהי עד נק' הורדה של הזמנה כלשהי), באופן דומה לפונקציה a נשים לב שלא יכול להיות שנבצע בפועל מרחק קטן יותר מהמרחק האווירי הנ"ל עד למצב המטרה משום שעלינו להגיע אל שתי הנק' הנ"ל, למרות שכעט הסדר אינו נאכף עדיין עלינו לעשות את הדרך בין שתי הנק' וכפי שצוין בתרגול לא יכול להיות כביש קצר מהמרחק האווירי. לכן, פונקציה זו תמיד תעריך בצורה אופטימית את המרחק מהמצב הנוכחי למצב המטרה.
4. **אינה קבילה**: הפונקציה מחזירה את המרחק האווירי הגדול ביותר שיש לעבור עבור בין נקודת איסוף ונקודת הורדה של הזמנה אשר כבר אספנו. נראה דוגמה:

יהיו שתי הזמנות כאשר המרחק האווירי עבור ההזמנה הראשונה הינו 100 והמרחק האווירי עבור ההזמנה השנייה הינו 1 ונניח כי שתי ההזמנות יורדות באותה נק'. נניח שראשית אספנו את ההזמנה הראשונה וכרגע אנו בנק' בה אוספים את ההזמנה השנייה => על המצב הנוכחי שלנו הינו 100 למרות שמצב היעד במרחק אווירי של 1 => אינה קבילה. **מיודעות:** ההגדרה של היוריסטיקה מיודעת רלוונטי רק להיוריסטיקות קבילות.

1. **קבילה**: הפונקציה מחזירה את המרחק האווירי הגדול ביותר שיש לעבור מהנק' הנוכחית עד נק' האיסוף של הזמנה שעדיין לא עלתה על האוטובוס, נשים לב שלא יכול להיות שנבצע בפועל מרחק קטן יותר מהמרחק האווירי הנ"ל עד למצב המטרה משום שעלינו להגיע לנק' האיסוף של ההזמנה ולאחר מכן להגיע אל יעדה וכפי שצוין בתרגול לא יכול להיות כביש קצר מהמרחק האווירי. לכן, פונקציה זו תמיד תעריך בצורה אופטימית את המרחק מהמצב הנוכחי למצב המטרה.
2. **לא קבילה**: נראה דוגמה נגדית:



נשים לב כי:

*אינה קבילה לפי הגדרה.*

1. **קבילה**: הפונקציה מחזירה את המרחק האמיתי הגדול ביותר שיש לעבור עבור הזמנה שעדיין לא עלתה על האוטובוס (מרחק אמיתי מקסימלי מנק' האיסוף של ההזמנה עד נק' ההורדה של אותה הזמנה), נשים לב שלא יכול להיות שנבצע בפועל מרחק קטן יותר מהמרחק האמיתי הנ"ל עד למצב המטרה משום שעלינו לאסוף את ההזמנה ולאחר מכן להגיע אל יעדה. לכן, פונקציה זו תמיד תעריך בצורה אופטימית את המרחק מהמצב הנוכחי למצב המטרה.
2. **אינה קבילה**: הפונקציה מחזירה את המרחק האמיתי הגדול ביותר שיש לעבור עבור בין נקודת איסוף ונקודת הורדה מבין הזמנות שעוד לא אספנו, נשים לב כי אנו לא מוכרחים לבצע את סדר איסוף והורדת ההזמנות לפי הסדר אותו חישבה הפונקציה כלומר: נניח כי ישנן שתי הזמנות שעוד לא אספנו וקיימים הכבישים הבאים:

נקבל במצב זה: משום שהמרחק המקסימלי הינו מ- ל- למרות שנוכל לבצע את המסלול הבא באורך קצר יותר : .

**מיודעות:**

בגלל שאנו משתמשים בהגדרה מההרצאה וכן כל הפונקציות הקבילות לעיל הן פונקציות על הקבוצה W נקבל שעבור כל מצב s כך ש- נקבל כלומר עבור כל מצב בו כבר אספנו את כל ההזמנות אך יש יותר מהזמנה אחת על האוטובוס נקבל כי פונקציית ההיוריסטיקה מחזירה את הערך 0 כך שלכל שתי פונקציות לעיל קיים מצב שאינו מצב מטרה וכן מתקיים .

ולכן אף אחת מהיוריסטיקות אינה מיודעת משאר היוריסטיקות.

1. הפלט שהתקבל:

A\* (Custom heuristic): g(G)=127.29km, h(I)=20.37km, developed: 1672 states

1. הפלט שהתקבל:

A\* (MST heuristic): g(G)=127.29km, h(I)=81.34km, developed: 475 states

**פרק שני**

1. יהי מצב s, נפרק למקרים:
   * => , מהנתון h קבילה => מתקבל כי .
   * => => מתקבל כי .

בשני המקרים נקבל כי => קבילה לפי הגדרה.

1. בסעיף זה נשתמש בהגדרת המיודעות מהתרגול.

נגדיר את פונקציית היוריסטיקה:

הסבר: פונקציית היוריסטיקה תחזיר את הערך הגדול בין למחיר המינימלי מ-s לאחד מבניו, נשים לב שבגלל שמרחב המצבים הינו עץ לכל מצב v יש רק אב אחד (s) ולכן רק ערך עלות אחד ולכן לא נתקל במצב בו הגענו לצומת מסוים פעם שניה ופונקציית היוריסטיקה תחזיר לנו ערך חדש.

אם למצב s אין בנים .

נראה כי הפונקציה קבילה, *נפרק למקרים:*

* : כבר הוכחנו כי קבילה.
* אחרת: מתקיים משום שcost(s,v) הינו המחיר לעבור מ-s לבן של s עם המחיר המינימלי שיש לעבור, כלומר, במידה ו-v מצב מטרה זהו גם ערך , במידה ו-v אינו מצב מטרה אזי משום שאם ל-v יש מצב מטרה בתת העץ שלו שהעלות להגיע אליו מ-s בהכרח גדולה מ- cost(s,v).

נראה מיודעות: לפי הגדרת :

פסודו קוד:

H1(state,H0,parent):

Min = max\_val

For v in succ(s):

If (min> cost(s,v)):

Min = cost(s,v)

If( H0(state)> Min):

Return H0(state)

Else:

Return Min

1. גם בסעיף זה אנו משתמשים בהגדרה מהתרגול למיודעות. נגדיר:

פסאודו קוד :

H2(state):

If Applicable\_h(state):

Return H(state)

Sons = get\_all\_succesors(state)

For son in sons:

If applicable\_h(son) == False:

Return 0

Min = infinity

For son in sons:

If H(son) + cost(state,son) < min: // cost is the weight of the edge between the args

Min = H(son) + cost(state,son)

Return min

הסבר -

היוריסטיקה החדשה מתנהגת כמו הבסיסית בשינוי אחד- אם ההיוריסטיקה h אינה מוגדרת על s כלשהו, ננסה להעריך את s ע"י כך שננסה להפעיל את h על כל אחד מהבנים שלו , נבחר את h(s’) המינימלי מבניהם ונוסיף את המשקל האמתי של הקשת s->s’ .

נראה קבילות:

אם אחד או יותר מהבנים של s אינו מקיים ש h מוגדרת עליו אז התנהגות בדיוק כמו .

אם h מוגדרת על כל הבנים של s אז משערכת את s להיות סכום מחיר הקשת s->s’

ו- h(s’) כאשר סכום זה הוא המינימלי על האופציות האפשריות. מכיוון ש h אופטימית מתקיים ש:

לכן:

את הטענה הבאה נסמן (\*):

טענה זו נכונה מפני שלכל צומת s’ מתקיים ש ולכן גם

עבור כלשהי, בפרט cost .

אי השוויון נשמר לכל s’ בקבוצת הבנים ולכן גם אי השוויון על המינימום נשמר.

כעט נסיים את ההוכחה:

כאשר המעבר הראשון נובע מההגדרה שלנו, השני מהטענה (\*) והאחרון נובע מכך שבמסלול הזול ביותר מs אל צומת מטרה העובר דרך s’ , בהכרח יש לשלם את cost(s,s’) ועוד משקל שבהכרח שווה ל .

מכך נובע:

נראה מיודעות:

מיודעת יותר מ מכיוון שבמקרה שכל הבנים של צומת s (עליה לא מוגדרת h ) מקיימים ש h מוגדרת עליהם יתקיים

עבור מצבים שזה לא מתקיים עבורם נקבל את הערך 0 שהינו שווה לערך של וזה עונה להגדרת המיודעות לפי התרגול.

1. **הוכחה**: נראה אלגוריתם B המהווה שיפור לזמן ריצת A\* עם .

תחילה נשים לב ש A\* שקול ל uniform cost search כי בשום שלב הוא אינו בוחר את המצב הבא על פי היוריסטיקה שאינה היוריסטיקת האפס. גם עבור המצב ההתחלתי – המצב ההתחלתי בכל מקרה נכנס ל OPEN ומפותח ללא תלות בהיוריסטיקה. הדבר היחיד ש h’ תורמת הוא ידיעה מראש על עומק הפתרון אך A\* אינו עושה בו שימוש.

יהיה סדר בחירת הצומת הבא לפיתוח על צמתים עם F זהות כפי שמוגדר בA\*

כלומר אם ל s1,s2 אותה F אז A\* בוחר בינהן על פי סדר כלשהו ידוע מראש (למשל לפי סדר לקסיקוגרפי)

אז B יעבוד כך:

B(init-state,Goal-test , h’ )

Return DFS-L(init-state,Goal-test, h’(init-state) )

כאשר סדר בחירת הצומת הבא לפיתוח מתוך צמתים עם F זהות הוא כשל A\* (לקסיקוגרפי נניח)

במילים – B מריץ את DFS חסום בעומק הפיתרון הטוב ביותר הנתון על פי h’(init-state) שמחושב בתחילת B .

מכיוון שבחרנו את אותו סדר הבחירה של הצומת הבא לפיתוח כאשר יש כמה צמתים שקולות, B אינו יכול לפתח יותר צמתים מ A\*.

בכל ריצת A\* תמיד יפותח עץ מלא בגובה D-1 ועוד N צמתים כאשר D הוא עומק הפתרון הכי טוב.

המקרה היחיד בו B יפתח אותה כמות צמתים כמו A\* היא במקרה שA\* פיתח את כל עץ החיפוש. אם A\* נעצר לפני שפיתח את כל העץ אז B היה "חוסך" גזרה שלמה מהעץ.

B A\*

B קביל מפני ש DFS-L עם L=עומק הפתרון (במקרה הזה עומק הפתרון הכי טוב) הוא אלגוריתם קביל לפי ההרצאה.