

מבוא לבינה מלאכותית

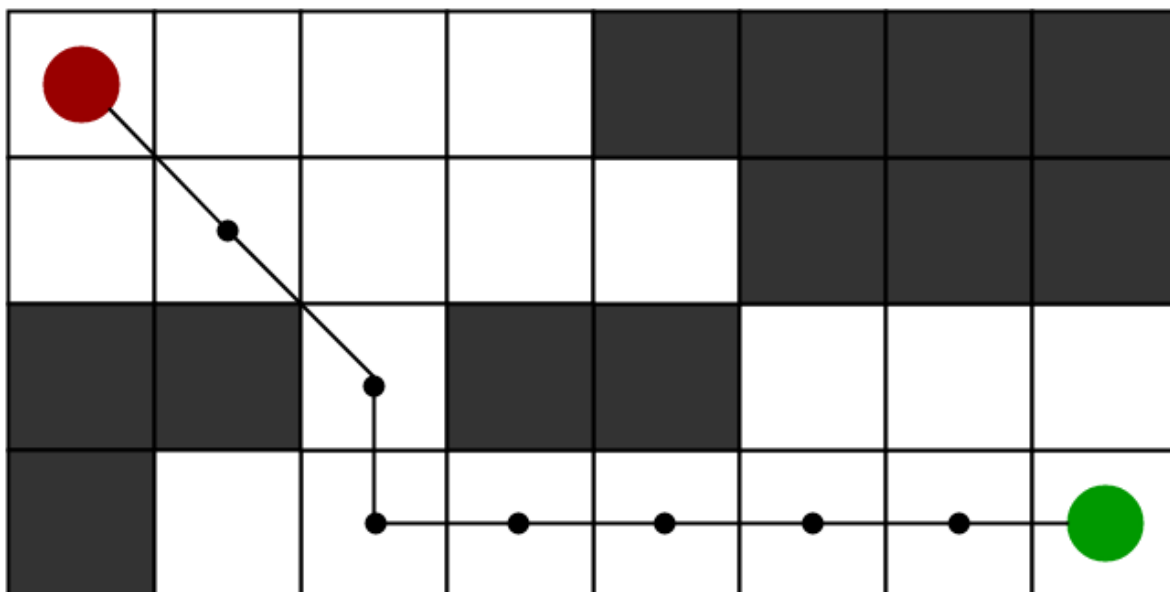
פרויקט באלגוריתמי חיפוש

שם : שאדי חילף

ת.ז. : 316328525

שפת מימוש : C++

סביבת מימוש : Visual Studio 2017



הסבר על קבצי הקוד :

:AISearchProject.cpp

הקובץ הראשי של התוכנית שאחראי על קליטת הנתונים, הרצת אלגוריתמי החיפוש והדפסת תוצאותיהם.

: Graph.cpp & Graph.h

קבצים אלה אחראים על הגדרת אובייקט גרף שמבצעים עליו חיפוש. הגרף יכול שדות של מחירי הצמתים במבוך וגם את הקשתות המוגדרות בין צמתי המבוך.

:Algorithms.cpp & Algorithms.h

קבצים אלה אחראים על מימוש אלגוריתמי החיפוש שאנו משתמשים בהם בפרויקט זה.

: Utils.cpp & Utils.h

קבצים אלה אחראים על עיבוד הנתונים מקובץ הקלט של התוכנית ולהשיג את שם אלגוריתם החיפוש, נקודות ההתחלה והיעד ואת הגרף שמייצג מבוך שמחפשים בו.

:pch.cpp & pch.h

קבצי מערכת שהם pre-compiled header שמטרתם להפחית מזמן הקומפילציה של התוכנית.

מימוש האלגוריתמים של החיפוש :

מימוש IDS :

IDS ממומש כלולאה אינסופית סופית שבה עומק החיפוש המקסימאלי מאותחל ל 0 ובכל איטרציה אנו מגדילים אותו ב 1 ומבצעים DLS (Depth Limited Search) שמחפש את צומת היעד עד עומק החיפוש המקסימאלי הנוכחי. DLS ממומש עם פרמטר שהוא עומק החיפוש המקסימאלי ובאמצעות טור רגיל ולא של עדיפויות (כמו מינימום) כך שכל פעם אנו מוציאים את האיבר הראשון בתור, מסמנים אותו כצומת שביקרנו בו ובודקים אם העומק שלו תקין, אם כן אז ממשיכים ואם לא אז לא. אם העומק שלו תקין אז נבדוק אם הוא צומת היעד, אם כן אז עוצרים את האלגוריתם ומחזירים פתרון ואם לא אז עוברים על כל שכניו שלא ביקרנו בהם ומוסיפים אותם לטור. תהליך זה יכול להימשך עד שהטור מתרוקן.

מימוש UCS :

UCS ממומש כאלגוריתם ASTAR עם היוריסטיקת האפס.

מימוש ASTAR :

ASTAR ממומש באמצעות טור עדיפויות מינימום עבור פונקציית ה f , כך שכל פעם אנו מוציאים את האיבר עם מינימום ערך f בתור, מסמנים אותו כצומת שביקרנו בו ובודקים אם הוא צומת היעד, אם כן אז עוצרים את האלגוריתם ומחזירים פתרון ואם לא אז עוברים על כל שכניו שלא ביקרנו בהם, אם שכן מסוים שלא ביקרנו בו לא נמצא בטור אז נוסיף אותו לטור ונעדכן את ערך ה g עבורו ואם שכן זה כבר נמצא בטור אז רק נדעכן עבורו את פונקציית ה g . תהליך זה יכול להימשך עד שהטור מתרוקן.

מימוש IDASTAR :

IDASTAR ממומש כלולאה אינסופית סופית שבה ערך הסף threshold מאותחל ל FLT_MIN ובכל איטרציה אנו מבצעים ASTAR שמחפש את צומת היעד עד ערך ה threshold הנוכחי כלומר ASTAR עוצר את החיפוש שלו אם מצאנו צומת שנמחק מטור העדיפויות וערך ה f שלו גדול ממש מ threshold. בכל איטרצית חיפוש של ASTAR אנו מעדכנים את הערך של threshold להיות ערך ה f המינימלי הראשון שגדול ממש מ threshold וששייך לצומת שנמחק מטור העדיפויות.

מימוש BIASTAR :

BIASTAR ממומש באמצעות שני טורי עדיפויות מינימום עבור פונקציית ה f כך שהטור אחד שייך לריצת ה forwards והטור השני שייך לריצת ה backwards. נשים לב שגם פונקציית ה g וגם פונקציית היוריסטיקה h שונות עבור ריצות ה forwards וה backwards כי ריצת ה forwards מתחילה מצומת ההתחלה וריצת ה backwards מתחילה מצומת היעד. כל פעם שאנו מוציאים את הצומת עם מינימום ערך f בטור ה forwards מסמנים אותו כצומת שביקרנו בו בריצת ה forwards וכל פעם שאנו מוציאים את האיבר עם מינימום ערך f בטור ה backwards מסמנים אותו כצומת שביקרנו בו בריצת ה backwards. צומת שמוציאים אותו מהטור בריצת ה forwards עוברים על כל שכניו שלא ביקרנו בהם בריצת ה forwards ואם שכן מסוים שלא ביקרנו בו בריצת ה forwards לא נמצא בטור של ריצת ה forwards אז נוסיף אותו לטור של ריצת ה forwards ונעדכן את ערך ה g עבורו בריצת ה forwards ואם שכן זה כבר נמצא בטור של ריצת ה forwards אז רק נדעכן עבורו את פונקציית ה g בריצת ה forwards. צומת שמוציאים אותו מהטור בריצת ה backwards עוברים על כל שכניו שלא ביקרנו בהם בריצת ה backwards ואם שכן מסוים שלא

ביקרנו בו בריצת ה backwards לא נמצא בטור של ריצת ה backwards אז נוסיף אותו לטור של ריצת ה backwards ונעדכן את ערך ה g עבורו בריצת ה backwards ואם שכן זה כבר נמצא בטור של ריצת ה backwards אז רק נדעכן עבורו את פונקציית ה g בריצת ה backwards. תהליך זה ימשיך כל עוד הטור של ריצת ה forwards וגם הטור של ריצת ה backwards לא מתרוקנים. אנו נמצא פתרון כאשר הצומת שמוציאים אותו מהטור בריצת ה forwards והצומת שמוציאים אותו מהטור בריצת ה backwards הם זהים ובמקרה זה אנו גם מחפשים בטורים של ריצות ה forwards וה backwards וגם ב closed list בשתי ההרצות (צמתים שנמחקו בשתי ההרצות מהטורים) אם יש פתרון טוב יותר ומחזירים אותו.

שימוש בהיוריסטיקות :

Chebyshev Distance :

פסאודו קוד :

ChebyshevDistance (Vertex, Goal)

```
[
    dx = |Vertex.x - Goal.x|
    dy = |Vertex.y - Goal.y|
    return max(dx, dy)
]
```

Vertex : צומת מסוים במבוך עם קואורדינטות (Vertex.x, Vertex.y)

Goal : צומת יעד במבוך עם קואורדינטות (Goal.x, Goal.y)

הוכחת אדמיסביליות :

ההיוריסטיקה פה היא בעצם המקסימום בין מרחק הצומת הנוכחי לצומת היעד בציר ה x לבין מרחק הצומת הנוכחי לצומת היעד בציר ה y . לפי מבנה והגדרת המחירים במבוך שלנו העלות המינימלית להגיע מהצומת הנוכחי לצומת היעד היא לפחות המקסימום בין מרחק הצומת הנוכחי לצומת היעד בציר ה x לבין מרחק הצומת הנוכחי לצומת היעד בציר ה y ולכן ההיוריסטיקה הזו לא תבצע הערכת יתר לעלות ההגעה המינימלית מהצומת הנוכחי לצומת היעד ומכאן מסיקים ש Chebyshev Distance היא היוריסטיקה אדמיסבילית.

הוכחת קונסיסטנטיות :

לפי ההגדרה של Chebyshev Distance ערך ההיוריסטיקה של צומת היעד הוא 0 וההפרש בין ערך ההיוריסטיקה של צומת במבוך לערך ההיוריסטיקה של שכן שלו הוא 0 או 1 או -1 ובגלל הגדרת המבוך שלנו עלות ההגעה מצומת לשכן שלו היא לפחות 1 ולכן מתקיים שההפרש בין ערך ההיוריסטיקה של צומת במבוך לערך ההיוריסטיקה של שכן שלו קטן או שווה לעלות ההגעה מצומת לשכן שלו ולכן Chebyshev Distance היא היוריסטיקה קונסיסטנטית.

Octile Distance :

פסאודו קוד :

OctileDistance (Vertex, Goal)

```
[  
    dx = |Vertex.x - Goal.x|  
    dy = |Vertex.y - Goal.y|  
    return dx + dy + (sqrt(2) - 2) * min(dx, dy)  
]
```

Vertex : צומת מסוים במבוך עם קואורדינטות (Vertex.x, Vertex.y)

Goal : צומת יעד במבוך עם קואורדינטות (Goal.x, Goal.y)

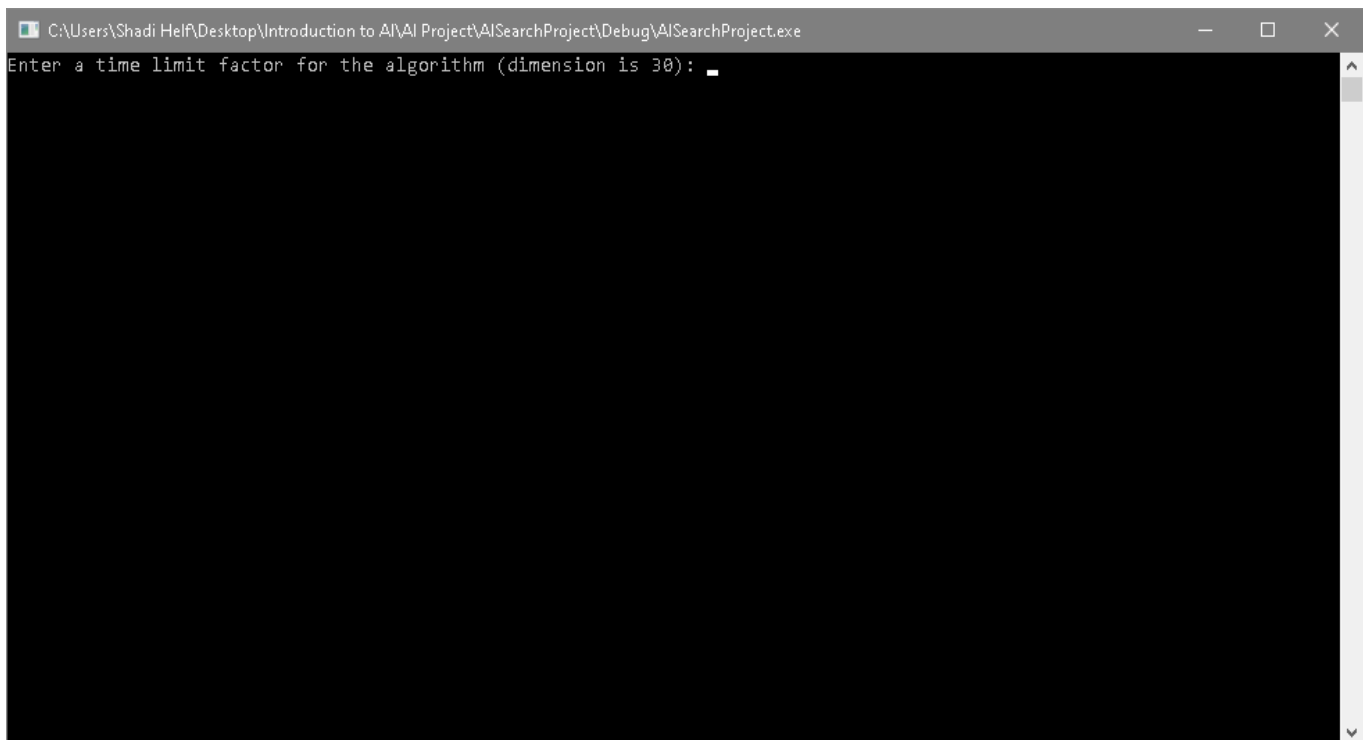
הוכחת אדמיסביליות :

הערך שההיוריסטיקה פה מחזירה חסום על ידי המקסימום בין מרחק הצומת הנוכחי לצומת היעד בציר ה x לבין מרחק הצומת הנוכחי לצומת היעד בציר ה y. לפי מבנה והגדרת המחירים במבוך שלנו העלות המינימלית להגיע מהצומת הנוכחי לצומת היעד היא לפחות המקסימום בין מרחק הצומת הנוכחי לצומת היעד בציר ה x לבין מרחק הצומת הנוכחי לצומת היעד בציר ה y ולכן ההיוריסטיקה הזו לא תבצע הערכת יתר לעלות ההגעה המינימלית מהצומת הנוכחי לצומת היעד ומכאן מסיקים ש Octile Distance היא היוריסטיקה אדמיסבילית.

הערה : התוצאות וקובץ ה EXE משתמשים ב Chebyshev Distance

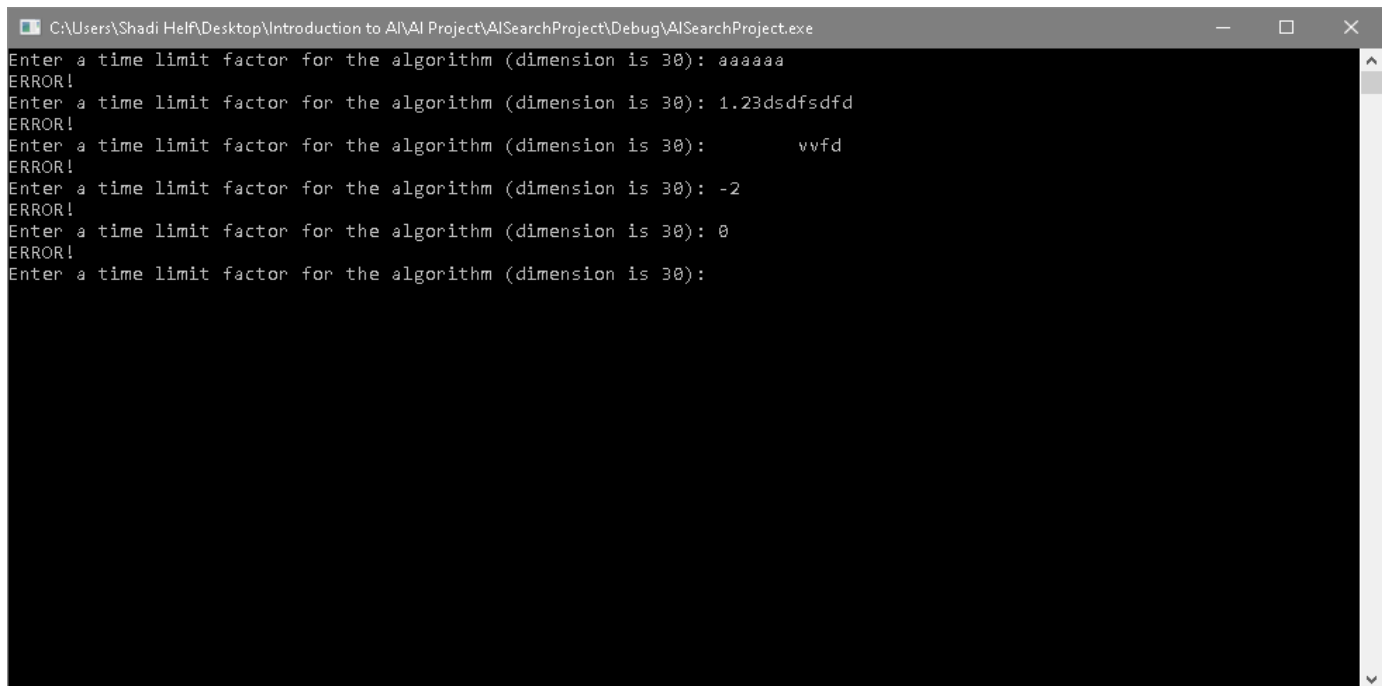
אופן עבודת התוכנית וקלט פלט :

התוכנית מתחילה עם קריאת הנתיב של קובץ הקלט ועיבוד הנתונים בו ולאחר מכן מוצג המסך ההתחלתי הבא :



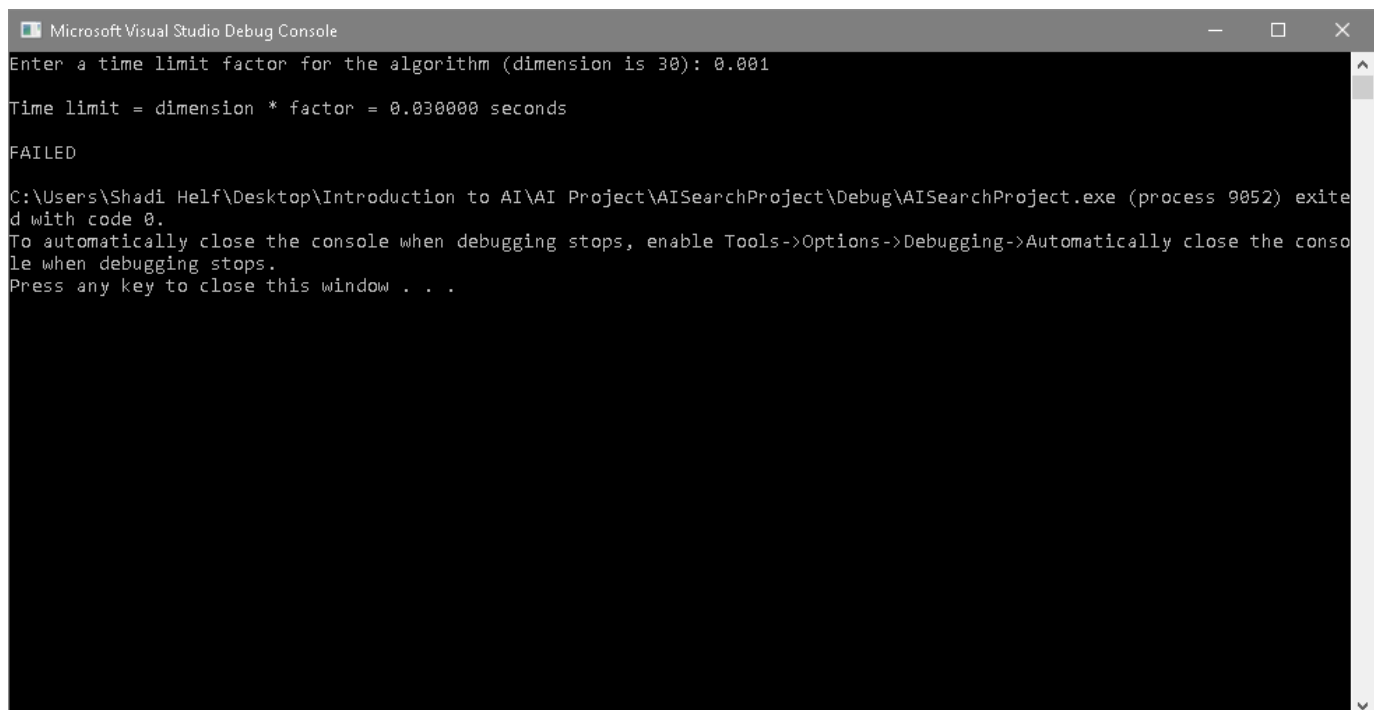
בשורה למעלה התוכנית מציגה את המימד של שורות ועמודות המבוך שהוא שווה ל 30 בדוגמה זה. שורה התחלתית זו גם מבקשת גם להזין מספר חיובי שהוא פקטור מסוים כך שחסם הזמן יהיה הפקטור שהזנו בפול המימד של שורות ועמודות המבוך.

אם הזנו מספר לא חיובי או תווים שלא מהווים מספר אז נקבל את המסך הבא ונצטרך להמשיך להזין קלט עד שנקלוט מספר ממשי חיובי :



```
C:\Users\Shadi Helf\Desktop\Introduction to AI\AI Project\AISearchProject\Debug\AISearchProject.exe
Enter a time limit factor for the algorithm (dimension is 30): aaaaaa
ERROR!
Enter a time limit factor for the algorithm (dimension is 30): 1.23dsdfsdfd
ERROR!
Enter a time limit factor for the algorithm (dimension is 30): vvfd
ERROR!
Enter a time limit factor for the algorithm (dimension is 30): -2
ERROR!
Enter a time limit factor for the algorithm (dimension is 30): 0
ERROR!
Enter a time limit factor for the algorithm (dimension is 30):
```

אם לאלגוריתם אין פתרון בכלל או שהוא נכשל במציאת פתרון תחת חסם הזמן אז יוצג המסך הבא :



```
Microsoft Visual Studio Debug Console
Enter a time limit factor for the algorithm (dimension is 30): 0.001
Time limit = dimension * factor = 0.030000 seconds
FAILED
C:\Users\Shadi Helf\Desktop\Introduction to AI\AI Project\AISearchProject\Debug\AISearchProject.exe (process 9052) exited with code 0.
To automatically close the console when debugging stops, enable Tools->Options->Debugging->Automatically close the console when debugging stops.
Press any key to close this window . . .
```

התוצאות עבור UCS, IDS בעת הצלחה יראה ככה :

[illegible]

התוצאות עבור IDASTAR, ASTAR בעת הצלחה יראה ככה :

```

Microsoft Visual Studio Debug Console
Enter a time limit factor for the algorithm (dimension is 30): 0.2
Time limit = dimension * factor = 6.000000 seconds

SOLUTION DETAILS :

*****
Path : RD-RD-RD-RD-R-RD-RD-RD-R-R-RD-RD-RD-RU-R-RD-RD-RD-R-RD-RD-RD-RD-R-RD-RD-D-D-RD-D-RD-LD-RD-RD-D-D-D
Cost : 119.000000
Number Of Expanded Vertices : 812
*****

Solution Length : 38
Execution Time : 0.181000 seconds
Penetrance : 0.046798
Heuristic Average : 19.381773
Effective Branching Factor Is Approximately : 1.192799
Minimum Search Tree Depth : 1
Maximum Search Tree Depth : 40
Average Of Search Tree Depths : 21.911535

```

התוצאות עבור BIASTAR בעת הצלחה יראה ככה :

```
Microsoft Visual Studio Debug Console
Enter a time limit factor for the algorithm (dimension is 30): 0.3

Time limit = dimension * factor = 9.000000 seconds

SOLUTION DETAILS :

*****

Path : RD-RD-RD-RD-R-RD-RD-RD-R-RD-RD-RD-RU-R-RD-RD-RD-R-RD-RD-RD-RD-RD-D-D-RD-D-RD-LD-RD-RD-D-D-D

Cost : 119.000000

Number Of Expanded Vertices : 612

*****

Solution Length : 37

Execution Time : 0.242000 seconds
```

המשך פתרון יכלול את תוצאות ריצת ה forwards ויראה ככה :

```
FORWARDS RUN DETAILS :

Path Forwards: RD-RD-RD-RD-R-RD-RD-RD-R-RD-RD-RD-RU-R (Path begins at 0,0 and ends at 9,15)

Cost Forwards: 55.000000

Number Of Expanded Vertices Forwards: 306

Solution Length Forwards: 15

Penetrance Forwards: 0.049020

Heuristic Average Forwards: 23.267973

Effective Branching Factor Forwards Is Approximately : 1.464586

Minimum Search Tree Depth Forwards: 1

Maximum Search Tree Depth Forwards: 26

Average Of Search Tree Depths Forwards: 13.683185
```

המשך פתרון יכלול את תוצאות ריצת ה backwards ויראה ככה :

```
BACKWARDS RUN DETAILS :  
Path Backwards: RD-RD-RD-R-RD-RD-RD-RD-RD-RD-D-D-RD-D-RD-LD-RD-RD-D-D-D (Path begins at 9,15 and ends at 29,29)  
Cost Backwards: 64.000000  
Number Of Expanded Vertices Backwards: 306  
Solution Length Backwards: 23  
Penetrance Backwards: 0.075163  
Heuristic Average Backwards: 23.696079  
Effective Branching Factor Backwards Is Approximately : 1.282552  
Minimum Search Tree Depth Backwards: 1  
Maximum Search Tree Depth Backwards: 24  
Average Of Search Tree Depths Backwards: 14.431062
```

הסבר תוצאות לאלגוריתמים :

Path : מסלול הפתרון שהאלגוריתם מחזיר.

Cost : עלות מסלול הפתרון שהאלגוריתם מחזיר .

Number Of Expanded Vertices : מספר הצמתים שנפרשו במהלך ביצוע האלגוריתם.

Solution Length : מספר הצעדים בפתרון שמוחזר על ידי האלגוריתם.

Execution Time : זמן כללי שלקח לתוכנית להתבצע וזה כולל את הזמן לקליטת ועיבוד נתוני האלגוריתם והגרף והזמן לביצוע האלגוריתם אבל זה לא כולל את הזמן של קליטת הפקטור שקובע את מגבלת הזמן כי זמן זה תלוי במשתמש ולא במהירות המימוש של האלגוריתם ובמבני הנתונים.

Penetrance : מדד חדירות קרי היחס.

Heuristic Average : הממוצע של ערכי פונקציית ההיוריסטיקה עבור הצמתים שנפרשו לאורך ריצת האלגוריתם.

Effective Branching Factor Is Approximately : ערך קירוב ל EBF.

Minimum Search Tree Depth : המינימום של עומק עץ החיפוש שהאלגוריתם יצר עד עצירתו.

Maximum Search Tree Depth : המקסימום של עומק עץ החיפוש שהאלגוריתם יצר עד עצירתו.

Average Of Search Tree Depths : הממוצע של עומקי עץ החיפוש שהאלגוריתם יצר עד עצירתו.

הערה לגבי תוצאות אלגוריתם ה BIASTAR :

בתוצאות של אלגוריתם ה BIASTAR יש לנו 3 תוצאות ראשיות שהן מסלול הפתרון, מחיר הפתרון ומספר הצמתים הנפרשים בפתרון. שאר התוצאות שאנו מקבלים הן עבור ריצות ה forwards וה backwards של האלגוריתם. תוצאות ריצות אלו יש להן את אותם נתונים סטטיסטיים כמו הנתונים סטטיסטיים שמוסברים למעלה וגם רשומה נקודת המפגש בין הרצות ה backwards וה forwards שקובעת את מסלול הפתרון.

טסטים, תוצאות וסטטיסטיקה :

ביצעתי 8 טסטים ובכל טסט אנו בודקים את הפלט של כל אלגוריתמי החיפוש על המבוך של הטסט. הקבצים בכל תיקיית טסט הם :

in1.txt : קובץ קלט של הגרף שבו מפעילים את אלגוריתם ה BIASTAR

in2.txt : קובץ קלט של הגרף שבו מפעילים את אלגוריתם ה IDASTAR

in3.txt : קובץ קלט של הגרף שבו מפעילים את אלגוריתם ה ASTAR

in4.txt : קובץ קלט של הגרף שבו מפעילים את אלגוריתם ה UCS

in5.txt : קובץ קלט של הגרף שבו מפעילים את אלגוריתם ה IDS

out1.txt : קובץ פלט של הגרף שבו מפעילים את אלגוריתם ה BIASTAR

out2.txt : קובץ פלט של הגרף שבו מפעילים את אלגוריתם ה IDASTAR

out3.txt : קובץ פלט של הגרף שבו מפעילים את אלגוריתם ה ASTAR

out4.txt : קובץ פלט של הגרף שבו מפעילים את אלגוריתם ה UCS

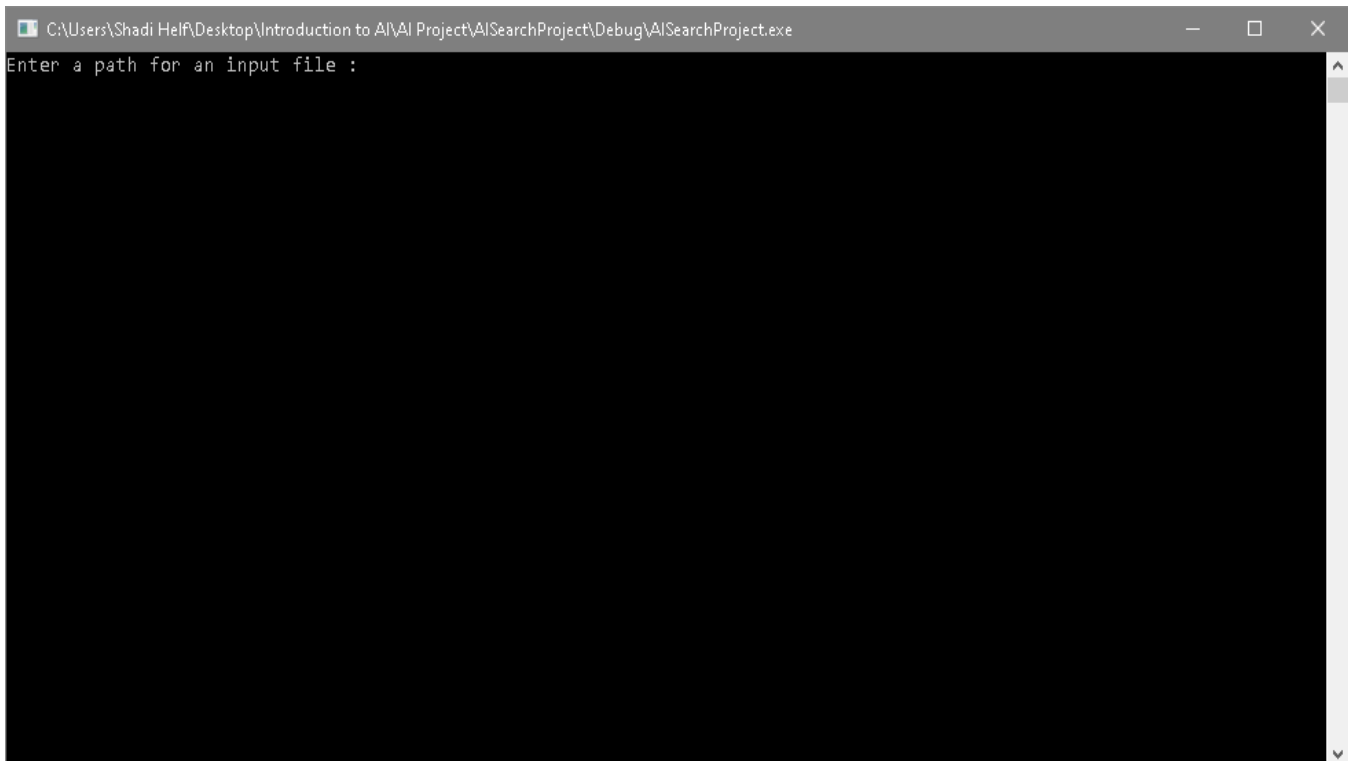
out5.txt : קובץ פלט של הגרף שבו מפעילים את אלגוריתם ה IDS

בקבצים out1,out2,out3,out4,out5 ניתן לראות את תוצאות הסטטיסטיקה שנדרשו עבור הפרויקט. בגלל שיש 8 טסטים ובכל טסט אנו מריצים את התוכנית 5 פעמים כך שבכל פעם אנו מריצים עם אלגוריתם חיפוש שונה אז סה"כ יש לנו 40 הרצות ומתוכם יש 37 הרצות שהצליחו למצוא פתרון במסגרת הזמן שנקבעה להם (במקרים שלא מצאנו פתרון אנו הרצנו את IDASTAR שלוקח לו הרבה זמן למצוא פתרון) ולכן אחוז ההצלחה הוא :

$$37/40 = \underline{\underline{92.5\%}}$$

הסבר על קובץ ה EXE :

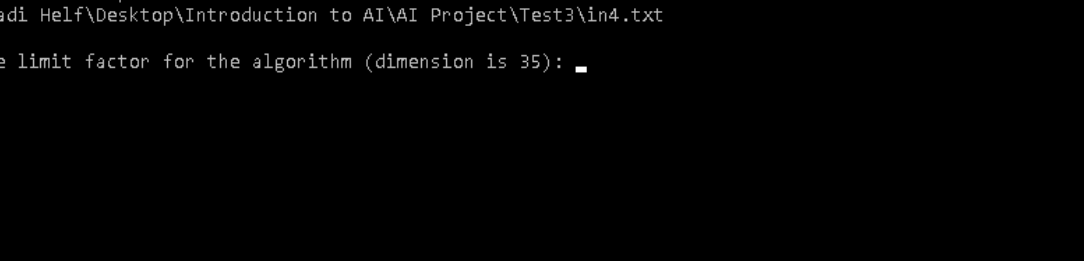
קובץ ה EXE יהיה תחת השם AI Search Project עם מסך התחלתי :



תוכנית ה EXE תמתין עד שיוזן נתיב קובץ קלט תקין ואם לא יוזן נתיב תקין אז נקבל שגיאות מהצורה :

[illegible]

אחרי קבלת נתיב קובץ קלט תקין נקבל את המסך הבא :



```
C:\Users\Shadi Helf\Desktop\Introduction to AI\AI Project\AI Search Project\Debug\AI Search Project.exe
Enter a path for an input file :
C:\Users\Shadi Helf\Desktop\Introduction to AI\AI Project\Test3\in4.txt
Enter a time limit factor for the algorithm (dimension is 35): 35
```


וכמו ההרצה הרגילה מה Visual Studio אנו נתבקש להזין מספר חיובי שהוא פקטור שקובע את חסם הזמן בתלות מימד המבוך. בעת הזנת פקטור תקין האלגוריתם ירוץ ויצג את מסך הפתרון הבא :

```
C:\Users\Shadi Helf\Desktop\Introduction to AI\AI Project\AISearchProject\Debug\AISearchProject.exe
Enter a path for an input file :
C:\Users\Shadi Helf\Desktop\Introduction to AI\AI Project\Test3\in4.txt
Enter a time limit factor for the algorithm (dimension is 35): 0.2
Time limit = dimension * factor = 7.000000 seconds
SOLUTION DETAILS (UCS) :
*****
Path : RD-RD-RD-RD-R-RD-RD-RD-R-RD-RD-RD-RU-R-RD-RD-RD-R-RD-RD-RD-RD-R-R-R-RD-R-R-RD-RD-D-D-D-D-D-D-D-D-D-LD-RD-RD-RD
Cost : 133.000000
Number Of Expanded Vertices : 1173
*****
Solution Length : 47
Execution Time : 0.216000 seconds
Penetrance : 0.040068
Effective Branching Factor Is Approximately : 1.162262
Minimum Search Tree Depth : 1
Maximum Search Tree Depth : 47
Average Of Search Tree Depths : 24.858067
Press 0 To Continue...
```

בגלל שאנו מפעילים קובץ EXE אז כדי שמסך התוכנית לא ייסגר אוטומטית בעת סיום ביצוע האלגוריתם, המשתמש יתבקש להזין את ה 0 ידנית כדי שהתוכנית תיסגר, אחרת אם לא יוזן 0 אז התוכנית לא תיסגר.

פורמט הגשה :

1. קובץ **Info.txt** שמכיל את השם שלי ומספר תעודת הזהות שלי

2. תיקייה בשם **Headers** שמכילה את כל קבצי ה headers (.h) של התוכנית

3. תיקייה בשם **Sources** שמכילה את כל קבצי ה sources (.cpp) של התוכנית

4. תיקיות הטסטים Test1, Test2, Test3, Test4, Test5, Test6, Test7, Test8 שמכילות את תוצאות וסטטיסטיקת ההרצות של אלגוריתמי החיפוש על המבוכים של הטסטים

5. קובץ ה EXE של התוכנית בשם **AIsearchProject**

6. קובץ זה שמכיל מידע על מימוש האלגוריתמים, היוריסטיקות ועוד.