תיאור הבעיה:

ישנם רכבים בתוך מבוך שאינו ידוע מראש.  
כל רכב מתחיל בנקודת התחלה שונה, ומקבל נקודת סוף שונה, אליה הוא רוצה להגיע.  
כל רכב סורק את המבוך, ומנסה לעדכן בזמן אמת את תכנון המסלול שלו בהתחשבות במכשולים חדשים.  
לרכב יש מידע מדויק אודות איפה הוא נמצא ביחס לנקודת ההתחלה שלו.  
כאשר 2 רכבים קרובים מספיק זה לזה, הם יכולים לשתף מידע שהם צברו אודות המבוך.  
המטרה היא לכתוב אלגוריתם שמביא כל מכונית לנקודת הסוף, בזמן מינימלי.

מדריך למשתמש:

ניתן להכניס מבוך כרשימת קווים פוליגונליים בתוך קובץ,  
ניתן לבחור את מספר המכוניות, ונקודות התחלה וסיום של כל אחת מהמכוניות  
התוכנה בונה סימולציה של תנועת המכוניות במבוך והנסיון שלהן להגיע לנקודת הסיום.  
ניתן לבקש מהתוכנה לבחור באופן אקראי את אחד מהארגומנטים, או לבחור מצב דיפולטי נתון.

מדריך למתכנת:

הפרויקט בנוי ממספר קבצים:  
הקובץ consts מכיל את כל הקבועים של האלגוריתם, מחולקים לפי קטגוריות.  
הקובץ helper מכיל פונקציות עזר, בעיקר לחישובים גיאומטרים ולדיסקרטיזציה של הסביבה.  
הקובץ המרכזי PRM שפותר את הבעיה לרכב אחד במבוך ידוע מראש.  
הקובץ mazes מכיל מספר מבוכים דיפולטים, עם נקודות התחלה וסיום מוגדרות מראש.  
הקובץ scan\_to\_map בונה מפה שהיא קירוב של המבוך – בעזרת הסריקות שהמכונית סורקת.

הקובץ :

תפקיד הקובץ לטפל בבניית המפה – ייצוג המבוך ב"מוח" של המכונית, בעזרתו מחושב גרף הroad\_map ובהתאם לגרף - המסלול המינימלי למטרה.  
אובייקט מסוג מכיל 2 תכונות:

1. מפה: ייצוג של המבוך בעזרת רשימה של מסילות פוליגונליות.  
   המסילה הפוליגונלית המיוצגת על ידי רשימת הנקודות

שקולה לרשימת הקטעים:

בפועל, למכשולים יש עובי שמוגדר בתור בקובץ .  
לכן בפועל כל ישר מתורגם למלבן עם עובי ואורך .

1. גודל: מימדי הסביבה, ערך הx המקסימלי (אנו מניחים שכל סביבה היא ריבועית וממורכזת ב)

בעת ריצת התוכנה, כאשר מכונית תפגוש במכשול, עליה לעדכן את המסלול שלה בהתאם.  
על מנת לעשות זאת, היא תעדכן את המפה שלה, באמצעותה מחושב מחדש הגרף, ובאמצעותו מחושב מחדש המסלול.  
סריקות של המכשול החדש יכולות להיות מאוד רועשות, מרכבות מספר קטעים שונים, ומגלמות בתוכן טעויות מדידה וקירובים.  
על מנת למצות מהמידע הזה את המידע הרלוונטי, אנו משתמשים באלגוריתם (RDP).  
האלגוריתם מקבל רשימת נקודות והופך אותן למסילה פוליגונלית כך שהמלבנים המתאימים לה מכילים את כל נקודות הסריקה.  
  
הפונקציה העיקרית שבעזרתה מוסיפים נקודות למפה היא .  
היא מסננת נקודות לא רלוונטיות, שכבר מוכלות במלבן ששייך למפה  
מפרקת את רשימת הנקודות הנותרת לנקודות שנבעו ממכשולים נפרדים, בעזרת קבוע מרחק בין 2 נקודות סמוכות – אם 2 סריקות סמוכות (בזמן) רחוקות זו מזו , הקוד יפרק את הישרים ויניח שנבעו מ2 מכשולים שונים.  
כעת לכל קבוצת נקודות, מחושבת המסילה הפוליגונלית בעזרת .

הקובץ road\_map.py:

תפקיד הקובץ הוא לשמור ולייצג את הקשתות והקודקודים של גרף המצבים שלנו.

אובייקט מסוג Road\_Map מכיל :

1. נקודת הסיום של הגרף.
2. גודל המפה
3. זווית הסיבוב המקסימלית של המכונית
4. האורך המקסימלי לחיבור קשתות בין שני קודקודים
5. קצה השינוי המקסימלי בזווית המכונית בזמן סיבוב
6. מופע של d\_star לצורך חישוב תנועה
7. מופע של גרף המכיל מידע על הקשתות והקודקודים
8. רשימה של רשימות של רשימות של קודקודים המייצגים מיקום של מכונית בזווית מסוימת, כאשר verteices[x][y][angle] הוא הקודקוד הx בסדרו במאוזן, y במאונך, והזווית הangle.

בזמן יצירת האובייקט הוא ייצור את הקודקודים ויאתחל את מבנה הנתונים, יש לקרוא לgenerate\_graph כדי ליצור את הקשתות בין הקודקודים, set\_end יגדיר את נקודת הסיום.

בזמן תפעול הקוד נקרא לget\_closest\_vertex, אשר תקבל מיקום של מכונית ותחזיק את הקודקוד הקרוב ביותר המייצג את מיקומו.