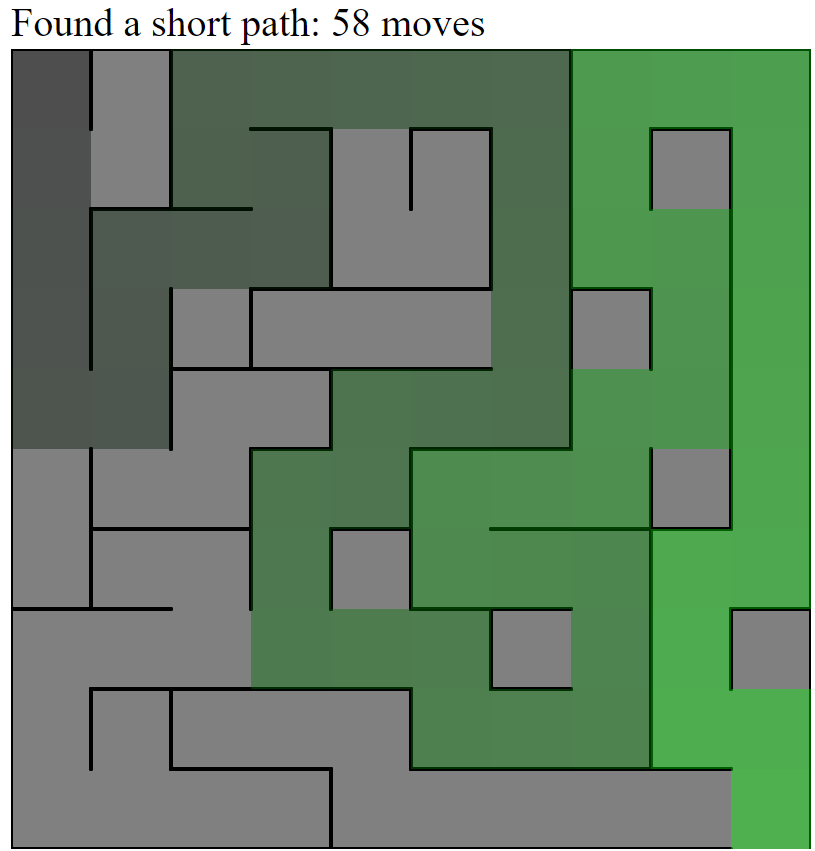
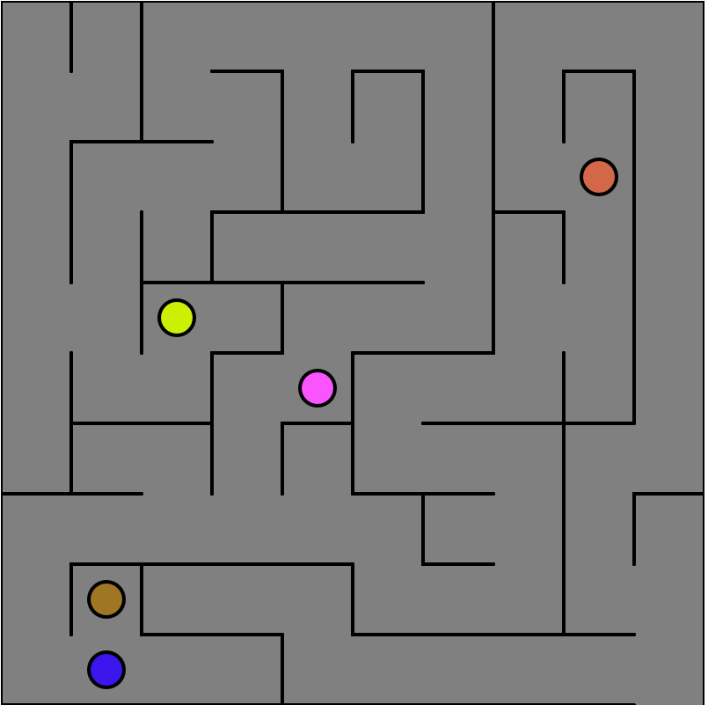
**פרויקטון בינה מלאכותית-**

**מגישים:**

ולדיסלב בלאוסוב 314599531

ריאהם ...

**פותר מבוך מרובה סוכנים**



* **תקציר**

בפרויקטון שלנו החטלטנו להשתמשProcessing.js היא סיפריית JavaScript, הספרייה מיועדת קוד אשר מאפשר ויזואליזציות, תמונות ותוכן אינטראקטיבי. אפשר מאפשר לדפדפני אינטרנט להציג אנימציות, יישומים חזותיים, משחקים ותוכן עשיר גרפי אחר ללא צורך ביישומון Java או תוסף פלאש.

בהסבר שלנו לבעיה הנתונה תחילה נדון בטכנולגיה נסביר על ספריית הגאווה סקריפט,לאחר מיכן נסביר על המבוך והסוכן, על העיבוד של כל סוכן, ונסיים בפירוט על האלגוריתם.

לרוב הקוד של הספריה נראה כך.

function setup() {

createCanvas(400, 400);///פוקציה של הגדרת מסך בגודל 400 על 400

}//פונקציה אשר רצה פעם אחת בעליית התוכנית אשר לרוב עושים בה איתחולים

function draw() {

background(255,0,0); ציור של צבע אדום על המסך

}// פונקציה אשר רצה בלופים שבא נעשה השלב של הציור ושינוי התוכן

* **מבוך-**

תחליה בחרנו במבנה נתונים של מטריצה בשביל לאפיין את המבוך, כאשר 4 סיביות לכל נקודה במובך מציונות את האפשריות לאן ללכת בכל נקודה.

זאת אומרת

|  |  |  |  |
| --- | --- | --- | --- |
| Up | Down | Right | Left |

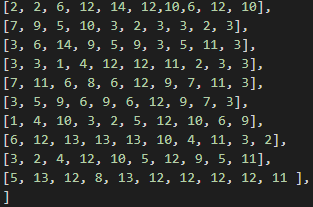
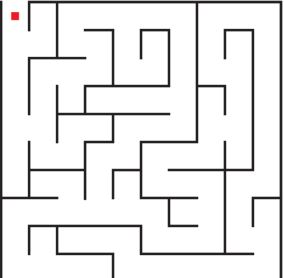
כאשר למשל סוכן נימצא בנוקדה מסוימת במפה הוא רואה שתוכן הנקודה במטריצה שווה בבסיס 2 ל-1011

|  |  |  |  |
| --- | --- | --- | --- |
| 1 | 1 | 0 | 1 |

זאת אומרת סכון

שהסוכן יכול לזוז מאותה נקודה למעלה למטה ושמאלה ולא יכול לזוז ימינה.

בדוגמה זו ניתן לראות את היצוג של המבוך בבסיס 10.



* **סוכן-**
  + **מבנה מחלקה סוכן-**

class Agent {

  static where\_agents\_be;// מטריצה סטסטית אשר מסמן את כמות הפעמים שסוכן ביקר בנקודה מסוימת

  constructor(x, y, maze) {

    this.stack\_way=[[x,y]]//נקודות אשר ביקר הסוכן

this.color=color(random (0,255),random(0,255),random (0,255))//צבע סוכן

    this.x = x;//מיקום סוכן בציר האיקס

    this.y = y;//מיקום הסוכן בציר הויי

    this.position\_to\_draw\_x;//מיקום הסוכן לציור בציר האיקס

    this.position\_to\_draw\_y;//מקום הציור בציר הויי

    this.wall\_size = maze.get\_Size\_wall() / 2;//גודל הקיר במבוך

    this.size = size / 4; //גודל לשם הציור של המבוך

    this.maze = maze; // אובייקט המבוך

    this.last\_visit=[this.x,this.y]//ביקור אחרון

* + **עיבוד של כל סוכן -**

כמו שצינו קודם סיפרית הגוואה סקריפט של פרוססינג קוראת למתודה

function draw(){

}

כך שלאחר היצירת האוביקט של הסוכן או מספר הסוכנים.

נקרה למתודה של סוכן הi

agents[i].start\_move\_randomly();

וכך נעשה כל פריים עד אשר אחד הסוכנים יגיע לנקודת הסוף.

* + **הסבר האלגוריתם-**

האולגוריתם הוא עיבוד של סוכנים הסימולציה פועלת בשלבים נפרדים. במהלך כל שלב, כל הסוכנים עוקבים אחר כל תוכנית העיבוד המתוארת להלן. העיבוד נעשה סוכן אחד בכל פעם, וזה מגבלה במקרה האמיתי שבו כל סוכן מסוגל לעבד את תנועתו במקביל לסוכנים האחרים. כל סוכן עוקב אחר ערכת העיבוד הבאה: הוא מקבל תחילה תקשורת מהסוכנים האחרים, ו- התקשורת, בשילוב מפת המבוך, מעובדות ליצירת מיפוי של הסביבה. באמצעות את המיפוי, הסוכן קורא לאלגוריתם של לתזוזה רנדומלית וקובע את המעבר שלו לתא הבא. המהלך הזה מבוצע על ידי הסוכן, אשר מעדכן את עמדתו, בהתבסס על קלט ממפת המבוך והסוכנים האחרים, והמיפוי התאים שעברו ביקורולא. לבסוף, הסוכן מעביר את ממצאיו ואת הסביבה לסוכנים אחרים. ניתן להגדיר את הבעיה ככניסת קבוצת סוכנים למבוך מנקודה אחת, ומצאנו היציאה בצד השני של המבוך. לכן, חיפושי המבוך האלגוריתם צריך להתמודד עם שני השלבים: למצוא את היציאה ולסמן את הדרך במהירות הקצרה ביותר. במהלך החיפוש אחר היציאה, כל סוכן עוקב אחר הכללה של האלגוריתם שלנו, אך ההבדל הוא בכך הידע באיזה תא ביקר משותף לכל הסוכנים. כל סוכן מקיים את הכללים הבאים:

1. בדיקה לאן קיימת אפשרות לזוז(אילוצי קירות)
2. ניקוד דרכי הליכה
   * + - כשאר הניקוד שווה למספר הביקורים בכך שהעלות היא-מינוס מספר הביקורים שביצעו הסוכנים באותה נקודה.
3. נבדוק עלות מינמלית.
4. נעלה את התא במטריצה ששיך לנקודה שכרגע נימצע הסוכן ב-1
   * + - נעלה המונה של המטריצה "איפה סוכנים היו" בנקודה הביקור ב-1
5. נבצע תזוזה.
6. נוסיף את הביקור למחסנית
   * + - אם זה מקום שביקרנו במהלך הקודם נוציא את הנקודה האחרונה.

**הפניות-**

**Multi-Agent Maze Exploration**

https://www.researchgate.net/publication/257655885\_Multi-Agent\_Maze\_Exploration