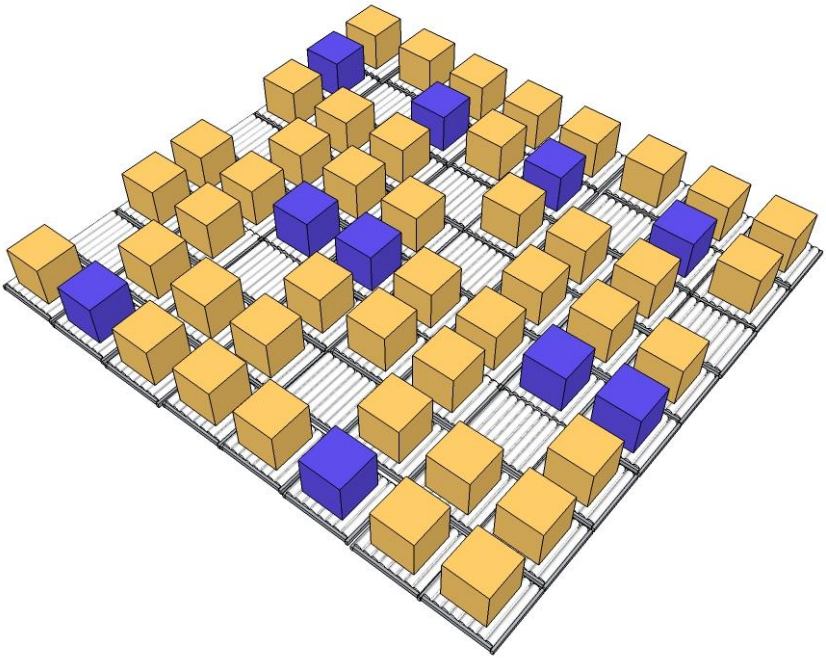
פרויקט ליקוט פריטים במערכת אחסון המבוססת פאזל

קורס תכן מפעלים תשפ"א



מרצה: פרופ' יוסי בוקצ'ין

מתרגל: אלון בלוך

<u>מגישים</u>:

טל אילון

אביהו מנחם

עמיחי כלב

תוכן עניינים

תיאור כללי של הפרויקט

הפרויקט עוסק במחסן בתצורת פאזל, תצורה המאפשרת ניצול גבוה של שטח המחסן באמצעות ניצול שטחי המעבר הקיימים במחסנים רגילים לאחסון.

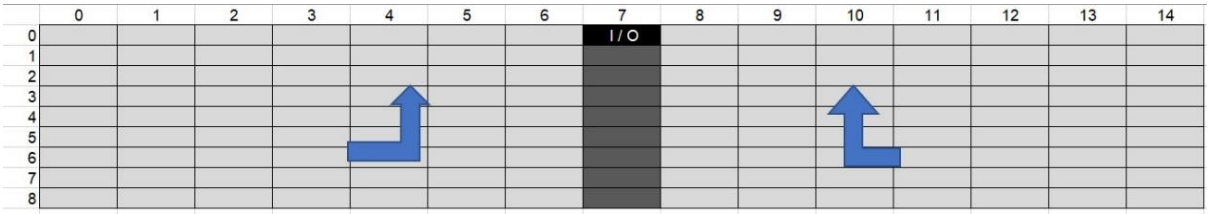
בפרויקט זה, במחסן נתון בגודל של 15X9 שורות על 15 עמודות רובוטים שתפקידם הזזת פריטים ואסקורטים, שטחים ריקים, המאפשר מזרון מעבר לפריטים עד לנקודת ה I/O, הנמצאת במיקום (0,7), שבה יוצאים ונכנסים הפריטים. ביחידת זמן אחת הרובוטים יכולים לנוע צעד אחד, כאשר הצעדים המוגדרים הם למעלה, למטה, שמאלה וימינה, ללא תנועה אלכסונית. הרובוטים יכולים לנוע מתחת לפריטים ללא הגבלה, אך תנועת פריטים תעשה ע"י רובוטים במידה ויש אסקורט פנוי בצמוד לפריט.מטרת הפרויקט היא למצוא היוריסטיקה עבור בעיית מינימיזציה של מזעור את הזמן הכולל עד להוצאת רשימה של פריטים:

$$Min\ Makespan$$

פייתון ההיוריסטיקה המוצעת בפרויקט זה נכתבה בשפת התכנות פייתון ונעשה שימוש בחבילת *pandas* ובחיבלת *numpy*.

המודל

המודל המוצע כאן מתבסס על הפתרון שהוצג במאמר של Gue & Kim משנת 2008. בתחילה, המחסן חולק לשני ריבועים 8X8 כאשר עמודה 7 היא עמודה המשותפת לשני המחסנים. חלוקה זו נבחרה כדי להתמודד באופן טוב יותר עם תנועות בכיוונים שונים במחסן.



איור 1: אופן חלוקת המחסן

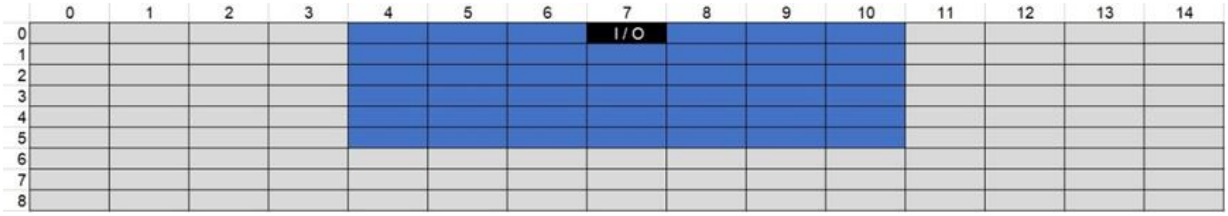
בחלוקה זו, לכל מחסן יש נק' I/O פינתית אליה יכולים להתקדם הרובוטים בתנועות שמאלה ולמעלה בלבד (ביחס לאיור 1 לעיל) – ופתרון זה אופטימלי (Gue & Kim, 2008).

הרובוטים הפזורים במחסן משובצים תחילה לשני המחסנים באופן שרירותי – 3 רובוטים למחסן השמאלי, ו2 רובוטים למחסן הימני. לאחר שיבוצם, נבחרים פריטים אותם הם צריכים להוציא מהמחסן דרך נק' ה I/O. את דרכם אל הפריט הם מבצעים בצעדי מנהטן, ומרגע ההגעה לפריט, מתחילים הרובוטים לבצע "צעדי 3" ו"צעדי 5" (ראה נספח 1) עד להגעת הפריטים אל דפנות המחסן, ומשם אל נק' היציאה. כאשר רובוט מסיים להוציא פריט, הוא יחל את דרכו אל הפריט הבא, עד שכל הפריטים הנמצאים ברשימת הפריטים להוצאה, יצאו מהמחסן.

הנחות

הנחות אלו הינן בתוספת להנחות הפרויקט שניתנו בהנחיותיו.

- לכל רובוט יוגדר אסקורט יחיד.
- אסקורט יכול להיות שייך לרובוט אחד בלבד.
- יוגדר אזור סטרילי בטווח [0,5] שורות וטווח [4,10] עמודות בסביבת נקודת ה I/O. (ראה איור 2). איזור סטרילי זה רלוונטי רק לרובוטים שנמצאים בתהליך הוצאת פריטים, ואינו רלוונטי לרובוטים שנמצאים בדרכם לפריט. באיזור זה, מבין הרובוטים שנמצאים בדרך לנקודת הI/O יחד עם פריט להוצאה, רק הרובוט שמרחק הפריט מנק' הI/O הוא הקצר ביותר (מרחק מנהטן).
- עמודה 7 תוגדר כדופן משותפת עבור שני חלקי המחסן. פריטים הנמצאים בעמודה זו ישויכו באופן שרירותי למחסן הימני.
- אסקורט יכול להיות רחוק מהרובוט המשוייך אליו בתא אחד בלבד.
- לאחר שרובוט מוציא פריט, הפריט הבא אותו הוא צריך לבחור נבחר באופן אקראי מתוך הפריטים הנותרים להוצאה, תוך התחשבות במיקומו היחסי במחסן (כלומר במחסן השמאלי או הימני).
- רובוט המוציא פריט ואין יותר פריטים להוצאה יצעד בצעדי מנהטן, באופן אקראי (עמודות ואז שורות או שורות ואז עמודות), עד לנק' במחסן שתיקבע מראש.
- רובוט ופריט יכולים לנוע למעלה, למטה, ימינה ושמאלה בלבד.
- אסקורט יזוהה כפריט בעל המספר 0.
- לכל רובוט יש רשימת צעדים מתוכננת לביצוע שלפיהם הוא מבצע את תנועותיו במחסן.
- בחלק מקביעת מסלול הרובוט נעשה שימוש בהחלטות אקראיות על מנת להתגבר על התנגשויות רובוטים, עליהן תהיה הרחבה בחלק הפונקציות שבדו"ח זה.
- לאחר שידוך רובוט לפריט לא תתבצע החלפה עם רובוטים או פריטים אחרים, אלא אם כן הפריט נמצא במרחק של לכל היותר תא אחד מתא הIO.
- לאחר הבאת פריט ל I/O הרובוטים ינועו על הצלע הארוכה אל עבר הפריט הבא, ומשם בצלע הקצרה, עד לשורה אחת מתחת אל הפריט אליו הם אמורים להגיע.
- בעקבות שינוי אפשרי במיקום הפריט אליו הרובוט אמור להגיע כתוצאה מתנועת רובוטים אחרים, קיימת האפשרות לחשב מסלול מחדש אל עבר הפריט המיועד ליציאה.



איור 2: האזור הסטרילי במחסן

מחלקות

לצורך המודל הנ״ל, נעשה שימוש בתכנות מונחה עצמים ובמסגרתו נבנו המחלקות הבאות:

1. מחלקת פריט

מחלקה זו מאפיינת פריט, ותכונותיה הן:

א. Number – מזהה הפריט

ב. Exit – מכילה את הערך True אם יש צורך בהוצאת הפריט, False אחרת

ג. Side – מציינת את הצד אליו שייך הפריט. 1 – שייך למחסן השמאלי, 2 – שייך למחסן הימני, 3 – משותף ל2 המחסנים.

2. מחלקת אסקורט

מחלקה זו מאפיינת אסקורט, ותכונותיה הן:

א. Robot_id - מזהה הרובוט שאליו שייך האסקורט

ב. Number – מזהה האסקורט. מוגדר 0 כברירת מחדל.

ג. Exit – מוגדר כ0 כברירת מחדל. תכונה זו נוצרה כדי לאפשר להתייחס למחלקה זו כאל פריט במהלך ריצת התוכנית.

3. מחלקת רובוט

מחלקה זו מאפיינת רובוט, ובעלת התכונות הבאות:

א. Id – מזהה הרובוט

ב. Item_to_take – שידוך מספר הפריט לרובוט שאליו הוא צריך להגיע

ג. Currently_taking – מספר הפריט שהרובוט לוקח אל נק׳ ה IO

ד. Path – רשימת הצעדים שהרובוט צפוי לבצע

ה. Side – המחסן אליו הרובוט שייך.

4. מחלקת תא

מחלקה זו מאפיינת תא במחסן, המורכב מפריט/אסקורט ומרובוט. היא בעלת התכונות הבאות:

א. Item – אובייקט מסוג פריט

ב. Robot – אובייקט מסוג רובוט

5. מחלקת מחסן

המחלקה המרכזית בפרויקט זה. מחלקת המחסן מורכבת מהתכונות הבאות:

א. Distances_left – רשימת צמדים של פריט להוצאה והמרחק (בערך מוחלט) שלו מנק׳ הIO בצעדי מנהטן במחסן השמאלי. רשימה זו תהיה ריקה כאשר לא יישארו פריטים להוצאה.

ב. Distances_right – רשימת צמדים של פריט להוצאה והמרחק (בערך מוחלט) שלו מנק׳ הIO בצעדי מנהטן במחסן הימני. רשימה זו תהיה ריקה כאשר לא יישארו פריטים להוצאה.

ג. Robot_side – מילון המכיל את מספר הרובוט כמפתחות שלו, ואת המחסן אליו שייך הרובוט כערך.

ד. Robot_positions – רשימה המכילה את מיקומי הרובוטים בפורמט (X,Y) כאשר X שורות וY עמודות.

ה. Items_to_exit_positions – מיקומי הפריטים שיש להוציא. רשימה זו תהיה ריקה כאשר לא יישארו פריטים להוצאה.

ו. Robots_moves – היסטוריית הצעדים של כל רובוט. היסטוריה זו נשמרת במילון שהמפתח שלו הוא הrobot_id, והערך שלו הוא רשימת היסטוריית הצעדים שביצע.

ז. Exited_items – מילון המכיל את הפריטים שיצאו מהמחסן כמפתחות, ואת הזמן בו הם יצאו כערך.

ח. Robot_final_positions_left - מיקומי החניה של הרובוטים הסיימו להוציא פריטים בצד השמאלי של המחסן.

ט. Robot_final_positions_right - מיקומי החניה של הרובוטים הסיימו להוציא פריטים בצד הימני של המחסן.

פונקציות

להלן הפונקציות שנבנו במסגרת המחלקות הנ"ל:

Class Robot:

- **robot_will_take (self, item_number):**
הפונקציה מקבלת מספר פריט ומבצעת עדכון לשני תכונות של רובוט: item_to_take למספר הפריט שקיבלה הפונקציה, currently_taking לnone, כלומר הרובוט בדרך לפריט ועדיין לא לוקח אותו ל I\O.
- **robot_is_taking(self, item_number):**
הפונקציה מקבלת מספר פריט ומבצעת עדכון לשני תכונות של רובוט: item_to_take לnone, currently_taking למספר הפריט, כלומר הרובוט לא בדרך לפריט וכרגע לוקח פריט ל I\O.
- reset(self):
הפונקציה מאפסת את רשימת הצעדים של הרובוט והפריט המשויך לרובוט, כלומר במידה והיה בדרך לפריט או בדרך ל I\O עם פריט. פונקציה זו מסייעת להתמודד עם מקרי קצה של איבוד פריט.

Class Warehouse:

- **init (self, warehouse_filename, items_to_exit_filename):**
פונקציה מאתחלת את המחסן ע"י טעינת כל הפריטים, הרובוטים והאסקורטים לתאים המתאימים, ומגדירה:

- רשימת מרחקי פריטים מהI\O השייכים לצד שמאל של המחסן וימין בהתאמה
- מילון המגדיר לאיזה מחסן (ימין או שמאל) שייך כל רובוט, רשימת מיקומי רובוטים שתעדכן בכל יח' זמן
- רשימת מיקומי פריטים להוצאה שתעדכן בכל יח' זמן
- רשימת היסטוריית צעדי רובוטים
- מילון שיכיל את הפריטים שהוצאו והזמן שבו הוצאו

הפונקציה קוראת לשתי פונקציות (עליהן תהיה הרחבה בהמשך):

calculate_positions() – מחשבת את מיקומי הרובוטים והפריטים להוצאה,

calculate_distance_from_IO() – מחשבת את מרחקי הפריטים להוצאה מהI\O .

- **define_robot_path(self, robot_id, steps, overwrite=True):**

הפונקציה מקבלת מספר רובוט ורשימת צעדים. הפונקציה משייכת רשימת צעדים (מסלול) לפריט לרובוט. במידה והמשתנה overwrite = true הפונקציה תדרוס את רשימת הצעדים הקיימים לרובוט ותעדכן ברשימת הצעדים המתקבלת כקלט.

- **exit_item(self, robot_id, current_time):**

הפונקציה מקבלת מספר רובוט ויחידת זמן נוכחית. לאחר שהפריט הגיע לנקודת ה- I\O, הפונקציה מסירה את הפריט מרשימת פריטים להוצאה, משנה את קוד הפריט ל999 ומוסיפה את הפריט לרשימת הפריטים שהוצאו ביחד עם הזמן שבו יצא.

- **calculate_positions(self):**

הפונקציה מחשבת את מיקומים הרובוטים במחסן ומיקומי הפריטים להוצאה.

- **calculate_distance_from_IO(self):**

הפונקציה מחשבת לכל פריט ששייך לרשימת הפריטים להוצאה את המרחק בצעדי מנהטן מה-I\O. הפונקציה נעזרת בפונקציה find_item_location בכדי למצוא לכל פריט שצריך לצאת את מיקומו הנוכחי (יכול להשתנות בכל יח' זמן כתוצאה מהזזה של רובוטים אחרים). כמו כן הפונקציה ממיינת את הפריטים להוצאה עפ"י מרחקם מנקודת ה-I\O עבור כל צד של המחסן (ימין ושמאל).

- **find_closest_robot(self, position):**

הפונקציה מקבלת מיקום פריט ומחשבת את הרובוט הקרוב ביותר ומחזירה את המרחק בינהם, המרחק מחושב כהפרש בערך מוחלט של קורדינאטות הX והY של הפריט והרובוט.

- **find_item_location(self, item_number):**

פונקציה מקבלת מספר פריט ומחזירה את מיקום הפריט בפורמט (X,Y) במחסן.

- **manhattan_journey_to_item(self, robot_id, item, overwrite=True, final=False, final_loc=0):**

הפונקציה מקבלת כקלט את מספר הרובוט ומספר הפריט להוצאה ומחשבת את המסלול, האופטמלי אל הפריט במרחק מנהטן – המסלול מורכב מהתאים במחסן שיצטרך הרובוט לעבור בצעדי מנהטן אל הפריט, הפונקציה מתחשבת 2 תרחישים:

- מיקום הפריט להוצאה באותה שורה של הרובוט – כאשר הם במצב זה הרובוט ינוע בעמודות המחסן
- מיקום הפריט להוצאה באותו טור של הרובוט – כאשר הם במצב זה הרובוט ינוע בשורות המחסן

במידה והרובוט לא בעמודה/שורה של הפריט, נבחר רנדומלית באיזה דרך נתחיל את המסע (תחילה בשורות או תחילה בעמודות). הפונקציה מוודאת כי הרובוט והאסקורט יחדיו, במידה ולא מאחדת בינהם ע"י הצעדים הנדרשים. כמו כן, בסיום הוצאת פריטיו של הרובוט הספציפי, הפונקציה מחשבת את המסלול בצעדי מנהטן אל נקודת הסוף (תא במחסן) המיועדת של הרובוט.

- **rows_steps(self, current_loc, target):**

הפונקציה מקבלת כקלט את המיקום הנוכחי של הרובוט ומיקום המטרה בציר Y (מספר השורה אליו הרובוט רוצה להגיע) ומחזירה את רשימת הצעדים שאליו לעשות בהתחשב במיקומו והאם זהו הצעד הראשון שעושה הרובוט או לא, בסוף הצעד הרובוט והאסקורט יהיו יחדיו.

<u>הסבר</u>: כאשר זהו הצעד הראשון הרובוט והאסקורט ביחד, בעוד כאשר זוהי תזוזה שאינה מתחילה צעד, הרובוט והאסקורט אינם יחדיו ועל כן יש צורך בצעד נוסף של הרובוט.

- **columns_steps(self, current_loc, y_target):**

הפונקציה מקבלת כקלט את המיקום הנוכחי של הרובוט ומיקום המטרה בציר X (מספר העמודה אליו הרובוט רוצה להגיע) ומחזירה את רשימת הצעדים שאליו לעשות בהתחשב במיקומו והאם זהו הצעד הראשון שעושה הרובוט או לא, בסוף הצעד הרובוט והאסקורט יהיו יחדיו.

<u>הסבר</u>: כאשר זהו הצעד הראשון הרובוט והאסקורט ביחד, בעוד כאשר זוהי תזוזה שאינה מתחילה צעד הרובוט והאסקורט אינם יחדיו ועל כן יש צורך בצעד נוסף של הרובוט.

- **three_step_horizontal(self,current_loc):**

הפונקציה מקבלת כקלט את מיקום הרובוט הנוכחי ומבצעת "צעד 3 אופקי" (ראה נספח), הפונקציה מחשבת את הצעדים הנדרשים לביצוע ה"צעד 3 אופקי". הפונקציה קוראת לפונקציות: rows_steps ו-columns_steps לחישוב תוכנית הצעדים. הפונקציה מחזירה את המיקום הסופי שבו ימצא הרובוט בסיום הצעד ורשימת הצעדים הדרושים לביצוע.

<u>הנחה</u>: הרובוט והאסקורט יחדיו בתחילת הקריאה לפונקציה.

- **three_step_vertical(self,current_loc):**

הפונקציה מקבלת כקלט את מיקום הרובוט הנוכחי ומבצעת "צעד 3 אנכי" (ראה נספח), הפונקציה מחשבת את הצעדים הנדרשים לביצוע ה"צעד 3 אנכי". הפונקציה קוראת לפונקציות: rows_steps ו-columns_steps לחישוב תוכנית הצעדים. הפונקציה מחזירה את המיקום הסופי שבו ימצא הרובוט בסיום הצעד ורשימת הצעדים הדרושים לביצוע. <u>הנחה</u>: הרובוט והאסקורט יחדיו בתחילת הקריאה לפונקציה.

- **three_step(self,robot_id):**

זוהי הפונקציה שמאגדת את "צעדי ה3" אופקי ואנכי, היא מקבלת כקלטת את מספר הרובוט שצריך לבצע "צעד 3" באופן הבא: בהתחשב במיקום האחרון המאומת של הרובוט, כל עוד הרובוט יכול לבצע צעדי 3 אנכיים ואופקיים, כלומר לא הגענו לצלע הארוכה של המחסן או לעמודת ה-I\O (מהווה בעצם דופן כי המחסן שלנו מחולק ל-2 כאשר המשותף בינהם הוא עמודת ה- I\O) הרובוט יעשה זאת. לאחר מכן קוראת לפונקציה five_step לביצוע "צעדי ה-5" הנדרשים. הפונקציה מוודא כי בצעד ה3 הראשון שנבצע ניהיה קרובים יותר לI\O מאשר הפריט, כלומר מתחתיו ומימינו (אם הוא בצד שמאל של המחסן) או משמאלו (אם הוא בצד ימין של המחסן) במידה ולא תחשב מסלול מחדש לפריט. כמו כן, בודקת הפונקציה האם מיקום הפריט השתנה, כתוצאה מהזזה של רובוטים אחרים תוך כדי ההגעה של הרובוט לפריט, אם כן מחשבת מסלול מחדש עבור הרובוט לפריט. בנוסף מוודא כי הרובוט והאסקורט יחדיו.

- **five_step_horizontal(self,current_loc):**

הפונקציה מקבלת כקלט את מיקום הרובוט הנוכחי ומבצעת "צעד 5 אופקי" (ראה נספח), הפונקציה מחשבת את הצעדים הנדרשים לביצוע ה"צעד 5 אופקי". הפונקציה קוראת לפונקציות: rows_steps ו-columns_steps לחישוב תוכנית הצעדים. הפונקציה מחזירה את המיקום הסופי שבו ימצא הרובוט בסיום הצעד ורשימת הצעדים הדרושים לביצוע. <u>הנחה</u>: הרובוט והאסקורט יחדיו בתחילת הקריאה לפונקציה.

- **five_step_vertical(self,current_loc,side):**

הפונקציה מקבלת כקלט את מיקום הרובוט הנוכחי ומבצעת "צעד5 אנכי" (ראה נספח), הפונקציה מחשבת את הצעדים הנדרשים לביצוע ה"צעד 5 אנכי". הפונקציה קוראת לפונקציות: rows_steps ו-columns_steps לחישוב תוכנית הצעדים. הפונקציה מחזירה את המיקום הסופי שבו ימצא הרובוט בסיום הצעד ורשימת הצעדים הדרושים לביצוע. הנחה: הרובוט והאסקורט יחדיו בתחילת הקריאה לפונקציה.

- **five_step(self,robot_id):**

הפונקציה מקבלת כקלטת את מספר הרובוט שצריך לבצע "צעד 5" . זוהי הפונקציה שמאגדת את "צעדי ה-5" אופקי ואנכי, "צעד 5" יכול להתרחש בעמודת ה-I\O או בשורות 0 או 1 של המחסן באופן הבא: בהתחשב במיקום האחרון המאומת של הרובוט, כל עוד הרובוט יכול לבצע צעדי 5 אנכיים או אופקיים, כלומר לא הגענו לנקודת ה-I\O הרובוט יעשה זאת.
הפונקציה בודקת האם מיקום הפריט השתנה, כתוצאה מהזזה של רובוטים אחרים תוך כדי ביצוע הצעדים, או לחילופין הרובוט "איבד" מסיבה כלשהי את הפריט בדרך – וזאת על מנת למנוע מצב שבו הרובוט מבצע את צעדיו לכיוון ה-I\O ללא הפריט. אם כן מחשבת מסלול מחדש עבור הרובוט לפריט.

- **to_next_item(self,robot_id,item_number):**

הפונקציה מקבלת כקלט את מספר הרובוט ומספר פריט, מטרת הפונקציה היא לחשב מסלול לפריט הבא להוצאה לרובוט שזה עתה סיים להביא פריט ל-I\O. היא מחשבת עבורו את המסלול לפריט הבא להוצאה מרשימת הפריטים להוצאה של הרובוט הספציפי, החל מהפריט השני שרובוט מוציא המסלול המחושב יהיה על דפנות המחסן. תחילה תנועה על הצלע הארוכה עד קצה המחסן ובמידה ונדרש מעלה אל עבר השורה שמתחת לפריט וחזרה במידת הצורך אל עמודת הפריט. הפונקציה מוודא כי הרובוט בתחילת החישוב ביחד עם האסקורט, במידה ולא מחברת בינהם ומחשבת את המסלול לפריט הבא כמתואר לעיל.

- **new_route(self, robot_id):**

הפונקציה מקבלת כקלט את מספר הרובוט. הפונקציה מסייעת במקרים של התנגשויות בין רובוטים ומטרתה להגדיר מסלול חדש לפריט אחר, מרשימת הפריטים להוצאה לרובוט הספציפי. במידה ולרובוט ישנם פריטים נוספים שעליו להוציא – נשנה את הסטטוס של הפריט הקודם (שלפני הקריאה לפונקציה היינו בדרכו אליו) לממתין להוצאה, כלומר item_to_take. ולפריט החדש שאליו יוגדר מסלול הגעה לכרגע בדרך ל-I\O, כלומר currently_taking. הפונקציה קוראת לפונקציה manhattan_journey_to_item בכדי לחשב את מסלול המנהטן לפריט החדש שנבחר להוצאה. במידה ולא קיימים פריטים נוספים או שלא הצלחנו להגדיר מסלול הפונקציה תחזיר False.

- **escort_in_target(self, location, robot_id):**

הפונקציה מקבלת כקלט מיקום (X,Y) במחסן ומספר רובוט. מטרת הפונקצייה למנוע "גניבה" של אסקורט של רובוט מסויים על ידי רובוט אחר. הפונקציה בודקת האם האסקורט הנכון, ששיך לרובוט הספציפי ביעד הנכון. במידה ואסקורט של רובוט אחר ביעד הפונקציה תקרא לפונקציה new_route לחישוב מסלול חלופי, במידה ולא תחזיר False.

- **apply_robot_step(self, robot_id, fictitious=False):**

הפונקציה מקבלת כקלט את מספר הרובוט וכערך ברירת מחדל fictitious=False מטרתו להגדיר האם מדובר בצעד פקטיבי של הרובוט או בצעד אמיתי של הרובוט – עם או בלי פריט. מטרת הפונקציה היא לבצע בפועל את צעדי הרובוט, עד עכשיו בעצם בדקנו ובנינו את הדרך והצעדים שהרובוט רוצה לבצע כעת אנחנו נבדוק אם כל התנאים לביצוע צעד מתקיימים ואפשריים, במידה וכן נבצע את הצעד. זהו בעצם השלב שבו אנו ממשים את התכנון אל מול הביצוע בפועל של הצעד.

ראשית, נבדוק האם הצעד הוא פיקטיבי או לא, אם פקטיבי – נוסיף לרשימת הצעדים של הרובוט צעד פקטיבי ונמחוק צעד זה מרשימת הצעדים לביצוע של הרובוט. אם הצעד לא פקטיבי – נבדוק שאכן הרובוט במיקום הנכון לביצוע צעד 3 או 5, במידה ולא נחשב מסלול מחדש לפריט בעזרת קריאה לפונקציה new_route, אם במיקום הנכון נבדוק שאכן במיקום היעד יש אסקורט, אם אין – הרובוט ימתין כדי להימנע מהתנגשות ברובוט אחר. אם יש אסקורט – נוודא כי זהו האסקורט של הרובוט הספציפי ואיננו "גונבים" אסקורט של רובוט אחר באמצעות קריאה לפונקציה escort_in_target, במידה וזהו אסקורט של רובוט אחר נמתין 3 יח' זמן בכדי שהרובוט שהאסקורט שלו נמצא בדרכנו יזוז ונוכל להמשיך במסלול המתוכנן. במידה והגענו למצב ששני רובוטים לא זזים כי כל אחד מפריע לשני במסלולו, כלומר קריאה לפונקציית

new_route מחזירה False – לא הצלחנו ליצור מסלול חדש לפריט, נקרא לפונקציה escape, שתסייע בהתנגשות (פירוט שלה בהמשך), במידה והכל מתאפשר הצעד ימחק מרשימת הצעדים לביצוע של הרובוט, ויכתב כצעד האחרון שהרובוט ביצע.

- **running_first_time(self):**

הפונקציה מגדירה את הריצה הראשונה של התוכנית, כלומר מתבצעת השמה של הרובוטים למחסנים השונים, ולאחר מכן את הפריט הראשון להוצאה לחמשת הרובוטים במתודולגיה הבאה לרובוט הראשון פריט קרוב ל-I\O מרשימת הפריטים לבא אחריו פריט רחוק מה-I\O מרשימת הפריטים וחוזר חלילה. לאחר מכן הפונקציה קוראת לפונקציה manhattan_journey_to_item בכדי לחשב את הדרך האופטימלית לפריט בצעדי מנהטן עבור כל רובוט.

- **can_proceed(self, robot_id):**

הפונקציה מקבלת כקלט מספר רובוט. מטרת הפונקציה למנוע התנגשויות בין רובוטים בסביבת ה-I\O, לכן הוגדר אזור בגודל 6X7 מסביב לנקודת ה-I\O שבו נזהה את כל הרובוטים שנכנסו לאזור ונושאים פריטים להוצאה ועבור כל אחד מהם נחשב את המרחק מנקודת ה-I\O, הרובוט עם המרחק הקצר ביותר יוכל להתקדם בעוד שהשאר ימתינו עד שיסיים. כמו כן, רובוטים שעוברים באזור זה אך נמצאים בדרך ללקיחת פריט רשאים להיכנס לאזור ולהמשיך במסלולם.

- **final(self,robot_id):**

הפונקציה מקבלת כקלט את מספר הרובוט. לאחר שרובוט סיים להוציא את כל הפריטים שהושמו לו להוצאה, פונקציה זו באה לידי ביטוי. מטרתה להביא את הרובוט לאחר שסיים את עבודותו לנקודה מוגדרת מראש (המוגדרת לכל רובוט בהינתן תת המחסן שאליו שוייך) בכדי למנוע התנגשויות עם רובוטים אחרים בסביבת ה- I\O בפרט ובשאר המחסן בכלל (הסבר לבחירת נקודות אלו מפורט בהנחות המודל).

פונקציות להתמודדות עם התנגשויות

מכיוון שהרובוטים נעים במקביל במחסן, היה צורך הכרחי להתמודד עם התנגשויות אפשריות במחסן. התנגשויות יכולות להתרחש בין רובוטים כאשר הם בדרכם לקחת פריט וכאשר הם לוקחים עמם פריט אל עבר נקודת ה IO. על מנת לנסות למנוע את ההתנגשויות האלו, נבנו פונקציות המאפשרות את המצבים הבאים:

1. בדיקת תכנון מול ביצוע: מניעת התנגשות ע"י בדיקה בכל יחידת זמן האם הרובוט יכול לבצע את הצעד המתכונן שלו.
2. האם יש אסקורט בנק' היעד: התחשבות בעובדה שייתכן שבצעד הבא ממוקם אסקורט של רובוט אחר, וע"י כך למנוע "גניבה" של האסקורט וכפועל יוצא למנוע התנגשות אפשרית עם הרובוט שאמור לחזור על האסקורט.
3. בדיקה האם יש 2 רובוטים שנעים בכיוונים מנוגדים על אותה עמודה או על אותה שורה: באופן הזה נקבע שרובוט יזוז לשורה אחרת/עמודה אחרת וימתין עד שהרובוט יחלוף.
4. איזור סטרילי סביב נק' הIO: איזור זה מבטיח שרובוטים שנמצאים בתהליך הוצאת הפריט ימשיכו בתנועותיהם מבלי שרובוטים אחרים שנעים לעבר נק' הIO יפריעו להם.

כדי להביא לידי ביטוי את הנ״ל, נעשה שימוש בפונקציות הבאות במסגרת מחלקת המחסן בנוסף לאלגוריתם המפורט בהמשך:

- **escape(self, robot_id, other_robot_next_loc=0):**

הפונקציה מקבלת כקלט את מספר הרובוט ואת המיקום של הרובוט האחר, כברירת מחדל מוגדר המשתנה other_robot_next_loc=0, בכדי לאפשר התחמקות לרובוט מבלי שהוא תלוי ברובוט אחר. מטרת הפונקציה להתמודד עם התנגשויות, הפונקציה מבדילה בין שלושה סוגי התנגשויות (ראה נספח):

1) כאשר הרובוטים רוצים לנוע זה לכיוון זה באותה שורה.
2) כאשר הרובוטים רוצים לנוע זה לכיוון זה באותה עמודה
3) כאשר רובוטים רוצים לגשת לאותו תא במחסן מכיוונים שונים – אחד מעמודה מסויימת ואחר משורה מסויימת.

כאשר אנו במקרה 1, הרובוט ינוע שורה אחת הצידה - בהתחשב במיקומו במחסן,יוסיף צעד זה לרשימת צעדיו ויכניס את הצעד לרשימת הצעדים לביצוע בכדי שידע לחזור למיקומו התחילי לפני ביצוע הescape וימשיך בצעדיו. לאחר התזוזה ימתין 3 יחידות זמן, בכדי שהרובוט השני יוכל לסיים את הצעד שהוא ניסה לבצע ואז יחזור למיקום התחילי וימשיך את מסלולו כמתואר לעיל. מקרה 2 בעל אופן טיפול דומה רק כאשר התזוזה היא לעמודה ליד בהתאם למיקומו.

כאשר אנו במקרה 3 הרובוט לא יבצע תזוזה אלא ימתין במקום 3 יחידות זמן בכדי שהרובוט השני יבצע את צעדו וימשיך בדרכו.

- **reroute(self,robot_id):**

הפונקציה מקבלת כקלט מספר רובוט. מטרת הפונקציה היא לסייע בהתאוששות ממצבים שבהם מסיבה כלשהי הרובוט מסיים צעד והפריט שלקח איננו לידו, הן אם בגלל רובוט אחר או הן אם בגלל צעד לא תקין שקרה. הפונקציה מוצאת את מיקום הפריט במחסן וקוראת לפונקציה manhattan_journey_to_item בכדי לחשב מחדש את המסלול בצעדי מנהטן לפריט.

- **location_check(self,robot_id):**

הפונקציה מקבלת כקלט את מספר הרובוט. הפונקציה בודקת אם הרובוט נמצא במיקום תקין בכדי לבצע צעדי 3 או 5. במידה וכן תחזיר True ובמידה ולא תקרא לפונקציה reroute לחישוב מסלול מחדש לפריט.

- **return_to_escort(self, location, robot_id):**

הפונקציה מקבלת כקלט מיקום (X,Y) במחסן ומספר רובוט. הפונקציה בודקת האם האסקורט נמצא מסביב למיקום הרובוט (מעל, מתחת, מימין או משמאל) במידה וכן מחזירה את מיקום האסקורט, במידה ולא – האסקורט והרובוט יחדיו ומחזירה False.

- **around_robot(self, robot_id):**

הפונקציה מקבלת כקלט את מספר הרובוט. הפונקציה בודקת האם סביב הרובוט הנוכחי (בכיוונים מעל, מתחת, משמאל או מימין) נמצאים רובוטים נוספים אם כן תחזיר את מיקומי הרובוט/ים אחרת תחזיר False.

- **around_IO(self):**

הפונקציה בודקת האם מסביב ל-I\O ישנם פריטים להוצאה, במידה וכן תחזיר את מיקומי הפריטים.

האלגוריתם

להלן פסאודו קוד המתאר את המתרחש במחסן :

- **ראשית, אחתל את המחסן:**
  - מקם את : הפריטים, הרובוטים והאסקורטים
  - קבל רשימת פריטים להוצאה
  - חלק את הפריטים לתתי מחסנים (שמאל וימין), והגדר לכל תת מחסן רשימת פריטים להוצאה.
  - מצא את המיקומים של כלל הרובוטים והפריטים להוצאה
- **כל עוד קיימים פריטים להוצאה** :
  - **בדוק האם קיים פריט להוצאה בתא הIO.**
    - אם כן, הוצא את הפריט, ושמור את זמן ההוצאה שלו.
    - אם קיימים עוד פריטים להוצאה, בחר בפריט בעל המרחק הגדול ביותר מנק׳ הIO וחשב אליו מסלול הגעה רלוונטי, תוך התחשבות במיקום היחסי של הפריט ושל הרובוט במחסן.
      - אחרת, הרובוט לא צריך להוציא פריטים יותר ושלח אותו לנקודת הסיום המוגדרת מראש לרובוט בצד זה של המחסן.
  - **עבור כל רובוט במחסן** :
    - **בדוק האם יש צעדים מתוכננים אותם הרובוט צריך לבצע.**
      - **בדוק האם הצעד המתוכנן הוא צעד 3 או צעד 5**
        - אם המיקום של הרובוט לא תקין, עבור לרובוט הבא
      - **בדוק האם המיקום הנוכחי של הרובוט שווה למיקום היעד שלו**
        - אם כן – מדובר בצעד פיקטיבי
      - **בדוק האם יש רובוט אחר בתא היעד**
        - אם היעד של הרובוט האחר זהה ליעד של הרובוט בריצה זו אנו לפני התנגשות, לכן בצע התחמקות (באמצעות פונקצית escape)
        - אחרת, עצור במקום למשך 3 יח׳ זמן
      - **בדוק האם יש אסקורט בתא היעד**
        - אם האסקורט שבמיקום הבא שייך לרובוט בריצה זו המשך בצעד
        - אחרת, בצע התחמקות באמצעות פונקצית escape
      - **בדוק האם הרובוט נושא איתו פריט ונכנס אל האיזור הסטרילי**
        - אם הפריט שהוא נושא הוא הקרוב ביותר מבין הפריטים שבדרך לנק׳ הIO, המשך בצעד. אחרת, עצור במקום עד שירד העומס מהאיזור הסטרילי.
    - **אחרת, אין לרובוט צעדים לבצע, ולכן**
      - **בדוק האם הרובוט סיים עכשיו את מסלולו אל הפריט**
        - אם כן, תכנן עבורו מסלול צעדי 3 וצעדי 5 אל עבר נק׳ הIO ועדכן שהרובוט הוא זה שלוקח את הפריט

- **בצע את כל הצעדים המתוכננים (פיקטיבים ובפועל)**
- **עדכן את כל המיקומים של הרובוטים והפריטים להוצאה**
- **עדכן ליחידת הזמן הבאה**

דוגמא להוצאת פריט

יהיה פריט שנמצא במיקום (5,9) ורובוט הנמצא במיקום (3,8), נקבל כי על הרובוט להגיע לפריט והמסלול אליו יהיה:

```
[((3, 8), (4, 8), False), ((4, 8), (3, 8), True), ((3, 8), (4,
8), False), ((4, 8), (4, 9), False), ((4, 9), (4, 8), True), ((4,
8), (4, 9), False)]
```

הרובוט מתחיל ממיקום (3,8) ונע לפריט שנמצא ב (5,9) כך שבכל שלב הרובוט זז לפריט כלשהו מזיז אותו לאסקורט וחוזר חזרה לאסקורט במיקום החדש היכן שהיה הפריט שהוזז, כך הוא מפנה את דרכו עד לפריט ומתמקם מתחתיו (4,9) ולאחר מכן יתחיל בצעדי 3 מאחר והפריט לא נמצא בדפנות המחסן.

לדוגמא עבור ((3, 8), (4, 8), False) הרובוט זז ל (4,8) אל הפריט (לא המיועד) שנמצא צמוד אליו לכן נקבל False ולאחר מכן מזיז את הפריט ל (3,8) בכדי לאפשר מעבר לאסקורט ((4, 8), (3, 8), True), לכן נקבל True.

סיכום תוצאות הקלט

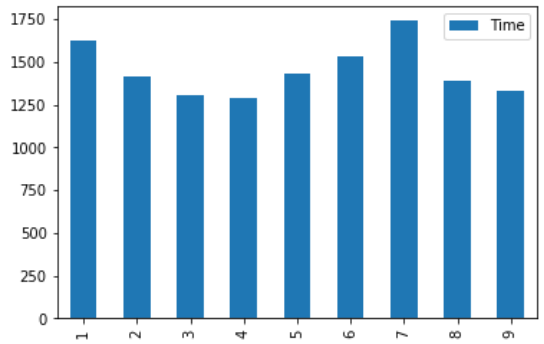
להלן תוצאות ההיוריסטיקה שהתקבלו עבור הקלטים השונים :

מדדים / קלטים	**1**	**2**	**3**	**4**	**5**	**6**	**7**	**8**	**9**
זמן הוצאה כולל ביח' זמן	1623	1414	1310	1288	1433	1529	1749	1388	1333
זמן ממוצע להוצאת פריטים ביח' זמן	64.92	56.56	52.4	51.52	57.32	61.16	69.6	55.52	53.32

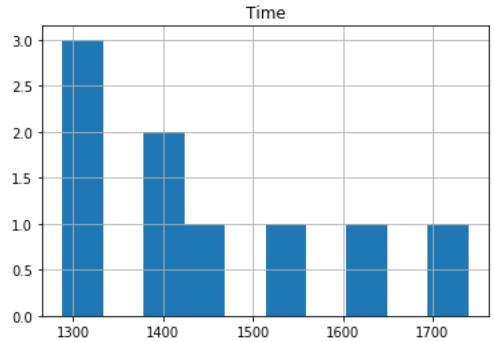
טבלה 1 : סיכום תוצאות ההרצה עבור הקלטים השונים

זמן ממוצע להוצאת כלל הפריטים – 1450.89 יח' זמן

זמן ממוצע להוצאת פריט בודד – 58.035 יח' זמן



איור 3 : יח' הזמן הכוללות עד להוצאת כלל הפריטים



איור 4 : היסטוגרמה לזמני ההוצאה הכוללים

פערים

- שימוש ברנדומיזציה על מנת להתגבר על התנגשויות בין רובוטים, מה שעושי שלהביא לזמני ריצה שונים עבור seed שונים, זאת לעומת האלגוריתם המקורי בו אין אקראיות כלל.
- מספר צעדי הרובוט שונה ממספר צעדי האסקורט , עבור שינוי אסקורט יש לבצע 3 צעדי רובוט, באלגוריתם המקורי אין התייחסות לרובוטים.
- אין איזור סטרילי באזור ה I/O מה שמעכבר את קצב הוצאת הפריטים מהמחסן.
- אין הגבלה על מסלולי הרובוטים, אנו הגדרנו כי רובוט שהוציא פריט חייב לעבור דרך הדופן של המחסן.

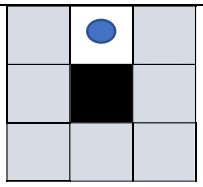
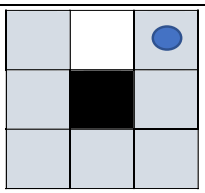
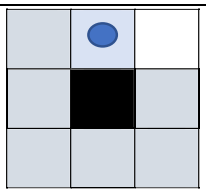
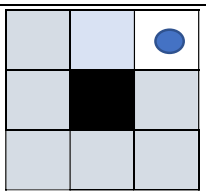
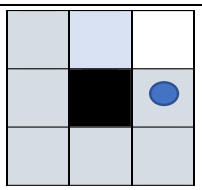
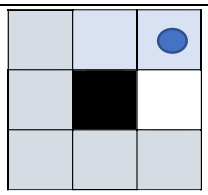
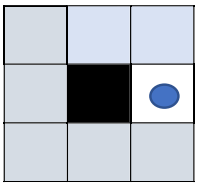
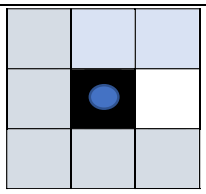
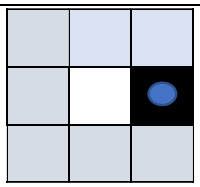
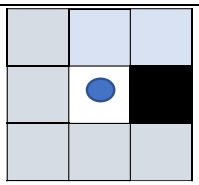
- חלוקה של הרובוטים ביחס של 2 ל 3 לשני החלקים במחסן באופן קבוע, כך שתת מחסן עם יותר פריטים עשוי לקבל אליו רק 2 מתוך 5 הרובוטים.

סיכום ומסקנות

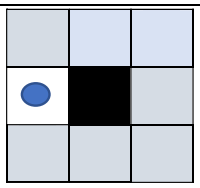
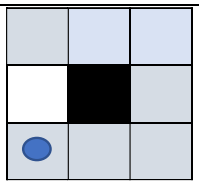
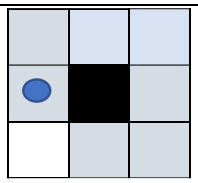
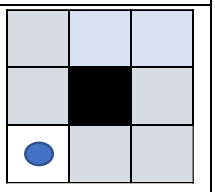
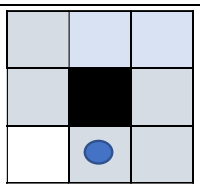
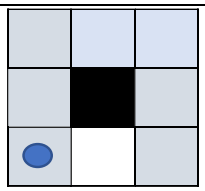
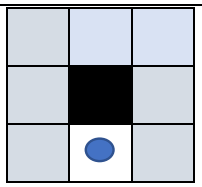
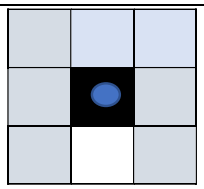
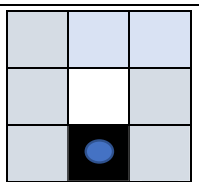
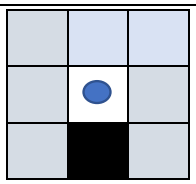
מהנתונים לעיל ניתן לראות כי הזמן המינימלי שהתקבל להוצאת כלל הפריטים הינו 1288 יח' זמן, והזמן המקסימלי שהתקבל הינו 1749. כפי שניתן לראות באיור 4, רוב הקלטים הגיעו לזמן הוצאה כולל של הפריטים בפחות מ1500 יח' זמן. אנו סבורים כי במסגרת מגבלות הזמן להכנת הפרויקט, התוצאה שהתקבלה מעידה על יציבות של תוצאות האלגוריתם ועל היעדר תוצאות חריגות. כמו כן על מנת לקבל ממוצע וסטיית תקן נכונים יותר ההינו שואפים להריץ את האלגוריתם על לפחות 30 מדגמים ולקבל זמני בקרה לתהליך. ההינו רוצים גם לבדוק חלופות אחרות לאלגוריתם כמו שינוי גודל האזור הסטרילי או חלוקה לוגית אחרת למחסן ולהשוות את התוצאות לאגוריתם הראשון על מנת לקבל חוות דעת לטיב האלגוריתם שהצענו.

נספחים

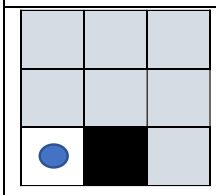
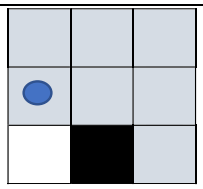
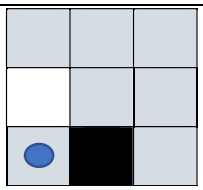
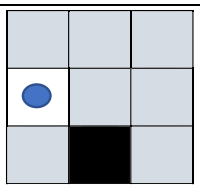
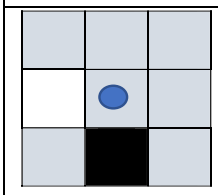
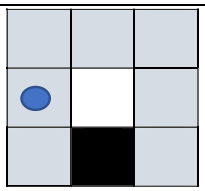
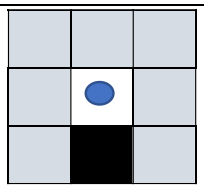
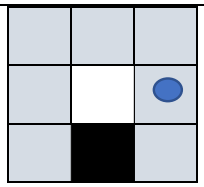
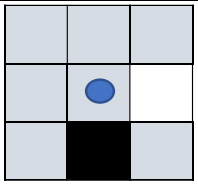
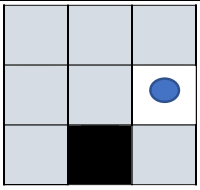
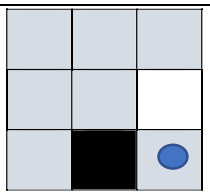
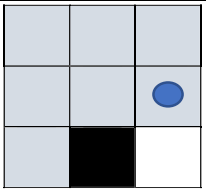
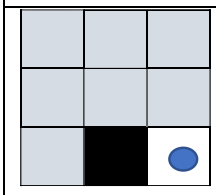
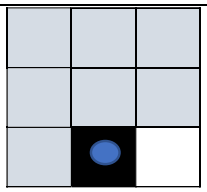
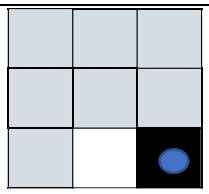
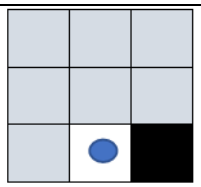
צעדי 3 אופקיים :

1		2		3	4
					
5		6		7	8
					
		9		10	
					

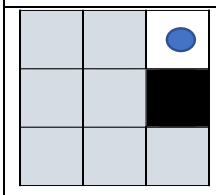
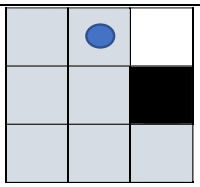
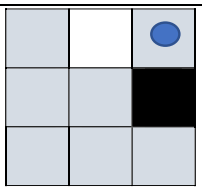
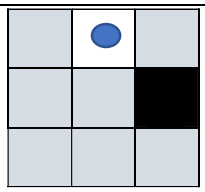
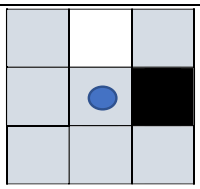
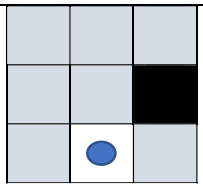
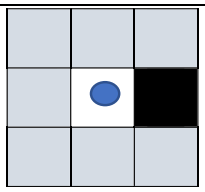
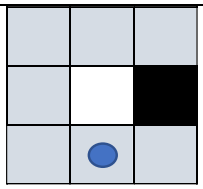
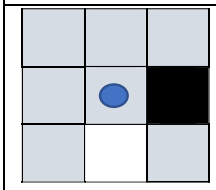
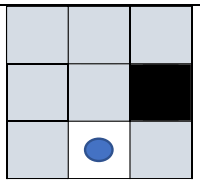
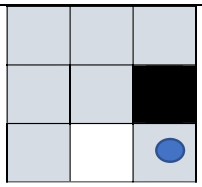
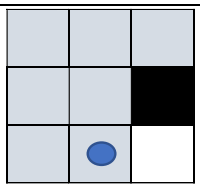
צעדי 3 אנכיים

1		2		3	4
					
5		6		7	8
					
		9		10	
					

צעדי 5 אופקיים :

1		2		3		4
						
5		6		7		8
						
9		10		11		12
						
13		14		15		16
						

צעדי 5 אנכיים :

1		2		3		4
						
5		6		7		8
						
9		10		11		12
						
13		14		15		16