תכן מפעלים

פרויקט גמר

קבוצה 22

גיא ליפשיץ 313328981

תומר וולפסון 204420103

רותם לוי 316381862

תוכן עניינים

1. אלגוריתמים מתמטיים

1.1. הנחות

1.1.1. במחסן 8 קומות, בכל קומה 32 תאי אחסון במתכונת של 16 תאים בצד ימין ו16 תאים בצד שמאל.

1.1.2. בכל קומה מעבר יחיד ובסופו מעלית.

1.1.3. במחסן מעלית יחידה.

1.1.4. נקודת הפריקה (I/O) נמצאת בצמוד למעלית בקומת הקרקע.

1.1.5. בכל תא ניתן לאחסון משטח יחיד.

1.1.6. בכל קומה שאטל בודד שאינו עובר בין קומות.

1.1.7. כל תא יכול לאחסן משטח אחד בלבד.

1.1.8. הזמנה מכילה 40 משטחים לשליפה.

1.1.9. הנחת אי תלות בין הגרלת מוצרים שונים בכל הזמנה.

1.1.10.

1.1.11. בעת שליפה, השאטל תמיד מתחיל מתחילת המסדרון (למעט הפעמים שמגיע מהחזרת פריט) ומסיים בתחילת המסדרון והמעלית תמיד תתחיל בקומת הפריקה ותחזור לקומת הפריקה.

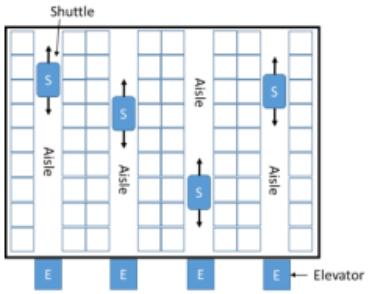
1.1.12. כאשר שאטל מחזיר פריט לתא (פירוט על פעולת ההחזרה בהמשך) הוא ממשיך ישר לאיסוף מוצר אחר, במידה ויש בקומה עוד פריט שצריך לספק להזמנה.

1.1.13. השאטלים נעים במקביל ואינם תלויים אחד בשני.

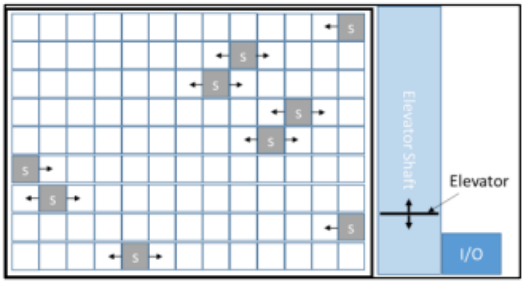
1.1.14. השאטלים נעים ללא תלות בפעולות המעלית, אלא דואגים שבכל רגע נתון יחכה פריט אחד בסמוך למעלית.

1.1.15. בין כל הזמנה להזמנה מלאי המחסן חוזר לכמותו ההתחלתית.

מבנה המחסן:



2 המחסן במבט עילי



1 המחסן במבט צידי

1.2. פרמטרים

1.2.1. זמן העמסת משטח ע״י השאטל – 5 שניות.

1.2.2. זמן העמסת משטח מהשאטל למעלית – 5 שניות.

1.2.3. זמן פריקת משטח מהמעלית לנקודת הפריקה – 3 שניות.

1.2.4. זמן תנועת השאטל בין 2 תאים סמוכים – 2 שניות.

1.2.5. זמן תנועת המעלית בין 2 קומות סמוכות – 2 שניות.

1.3. <u>אלגוריתם האחסון</u>

ראשית, עוד בטרם ניסוח אלגוריתם האחסון, מיפינו את המחסן והבנו את תכונותיו השונות בהתבסס על הנתונים. זאת, במטרה לזהות דפוסים ומאפיינים של המחסן, מבחינת התנועה בו, מהירות וזמן שליפת המשטחים מהתאים השונים וכו׳. מטרתנו היתה, ליישם אלגוריתם אחסון שיאפשר בסוף שליפה אפקטיבית שתמזער את זמני אספקת ההזמנות שיגיעו אל המחסן.
ראינו, כי זמני ההובלה מסודרים מהתא הקרוב ביותר לנקודת I/O ועד לקצה השני של המפעל בצורה אלכסונית בזמן שינוע עולה. כלומר, אם נמפה את המחסן כמלבן בעל 8 שורות (שורה = קומה) ובכל שורה ישנן 16 משבצות, המשבצות שעל כל אלכסון הן בזמן שינוע זהה לנקודת I/O כל משבצת מייצגת למעשה 2 תאים – אחד מימין ואחד משמאל. זמני השינוע ביניהם תמיד זהים. (מיפוי התאים בנספחים).

ההזמנות אינן צפויות ולא ניתן לחזות אותן, אך התבססנו על טבלת ההסתברויות של המוצרים השונים בעת יישום הפתרון.
אינסטינקטיבית – נטינו לחשוב שכדאי לאחסן את המוצרים נטו לפי הסתברותם לצאת בהזמנה, כלומר, כל היחידות של המוצר הכי שכיח ימוקמו הכי קרוב לנקודת I/O וכך הלאה בסדר הסתברות יורד.
הפער בפתרון אחסון שכזה, הוא התעלמות ממבנה המפעל בקומות ובעובדה שהשאטלים של כל קומה יכולים לנוע במקביל, בזמן שמעלית יש רק אחת. לכן, אין חשיבות בלהביא את המוצרים כמה שיותר מהר לתחילת כל קומה כל עוד הם יחכו למעלית שתגיע. לעומת זאת, אם נבצע איזשהו פיזור ״חכם״ של מוצרים בעלי הסתברות גבוהה ברחבי המחסן, כך :

א. נצמצם זמנים ״מתים״ בהם המטען מחכה למעלית שתגיע.
ב. נשאיר יותר מוצרים פנויים לאיסוף אשר מצויים בקרבת המעלית, ככה שבמקרים בהם המעלית תהיה פנויה יותר, ניתן יהיה להשתמש בהם ולמזער זמני הובלה.

מצד שני, אם נפזר לחלוטין את כל המשטחים מכל הסוגים באופן שווה ללא איזושהי התחשבות בהסתברות של כל לצאת בהזמנה, אנחנו נעלה משמעותית את הסיכויים של השאטלים לנוע רחוק מהמעליות בתדירות גבוהה יותר.

לכן, החלטנו לחלק את רשימת האחסון לבאצ׳ים (batches), כאשר כל באץ׳ מכיל קבוצת מוצרים בעלי הסתברות עוקבת. נמיין את הבאצ׳ים ונרוץ עליהם לפי סדר הסתברות יורד.
כלומר, הבאץ׳ הראשון יכיל את המוצרים בעלי ההסתברות הגבוהה ביותר לצאת בהזמנה. בעת ריצת האלגוריתם, נעבור על תאי המחסן בסדר אלכסוני (כפי שמתואר למעלה. כלומר נמלא את התאים לפי זמני השינוע בסדר עולה, כאשר את סידור המוצרים נעשה ב״סבבים״ לפי דירוג הבאץ׳. נתחיל עם באץ׳ בעל הדירוג הגבוה ביותר להופיע בהזמנה (המסודר באופן דומה גם בתוך הבאץ׳), ונשבץ אחד מכל סוג פריט. כאשר נסיים את הסבב נצרף גם את הבאץ׳ הבא בתור בעדיפות וכעת נשבץ אחד מכל סוג פריט מ-2 הבאצ׳ים (הראשון והשני) וכך הלאה עד שנסיים סיבוב של כלל הדירוגים (5 קבוצות דירוג). כאשר מסיימים סבב מתחילים מההתחלה (באץ׳ אחד בלבד, שניים וכו׳) עד לסיום החלוקה הכללית. בנוסף, במידה ונגמרים הפריטים לסדר מבאץ׳ מסוים הוא יצא מהסבב.

בסיום ריצת האלגוריתם, נקבל מיפוי מדויק של מיקומי הפריטים לפי סוג, בכל תא.

1.4. <u>אלגוריתם השליפה</u>

בעת ניסוח אלגוריתם השליפה, התלבטנו בין שתי אלטרנטיבות למימוש: לוגיקה, וסימולציה. עבור לוגיקה, ניסחנו חישובים שונים במטרה למצוא את החישוב המדויק ביותר של זמני העמסת הסחורה מהשאטל למעלית, ומהמעלית לנקודת ה-I/O. אמנם, לא הצלחנו למצוא נוסחה מספיק מדויקת אשר מתחשבת בחפיפת זמני הריצה של המעלית והשאטלים. אחת ההנחות בבעיה היא, שהמעלית והשאטלים רצים במקביל, ולכן במקרה בו המעלית עולה לקומה מסוימת כדי לקחת ממנה משטח, ובמקביל רץ השאטל באותה הקומה, שכבר סוחב איתו את המשטח בדרכו למעלית, מימוש אלגוריתם לוגי לא יכיל חישוב אשר מתחשב בזמן החופף (ולמעשה יספור את הזמן פעמיים, וזה לא חישוב נכון).

ליבת הרעיון שעומד מאחורי אלגוריתם השליפה שלנו מתרכז במספר נקודות עיקריות:

1.4.1. סידור המידע שתאפשר לגשת לפריטים לפי קרבה: בתהליך Pre-Processing שאנו מבצעים (הרחבה לגביו בסעיף 2.4) מתקבלת בין היתר עבור כל קומה, רשימה של פריטים אשר נמצאים בהזמנה המסודרים בסדר לא יורד של זמני השליפה (המרחק מהמעלית). כאשר שאטל יוצא לאסוף פריט, התיעדוף שלו כבר מונח לפניו: הוא יגש לפריטים שקרובים אליו קודם.

1.4.2. כל השאטלים רצים במקביל, וללא תלות בתנועת המעלית. אנחנו מנצלים את תכונת "עצמאות" זו של השאטל כדי למזער ואף לבטל לחלוטין את זמני ההמתנה של המעלית. אנו יוצרים מצב שבו, בכל קומה בכל רגע נתון ימתין פריט לאיסופה של המעלית, או לכל הפחות השאטל בדרכו לספק פריט. ככה תמיד יש למעלית לאן ללכת כדי לאסוף פריטים.

1.4.3. הגדרנו כי שאטל יכול להחזיר פריט. במידה והועמס הפריט האחרון בהזמנה מסוג הפריט בו מחזיק השאטל על ידי שאטל אחר, אנחנו נחזיר את הפריט לתא הקרוב ביותר הפנוי ונמשיך באיסוף פריט אחר שעדיין מופיע בהזמנה. בקונספט זה, הנחנו כי הזמנים ה"עודפים" כביכול של שאטל כשהוא יוצא להחזיר מוצר אינם מוסיפים זמן רב לזמני האיסוף. זאת כי בכל רגע נתון קיימים פריטים בקומות אחרות שממתינים לאיסוף המעלית, ובזמן שהמעלית תאסוף אותם השאטל יחזיר את הפריט ויקח את הבא בתור להזמנה.

2. <u>מימוש</u>

2.1. חבילות פייתון – pandas,numpy.

2.2. מחלקות – בנינו שתי מחלקות: מחלקת Allocation ומחלקת Creat_Order. פירוט בסעיפים הבאים.

2.3. <u>מימוש אלגוריתם האחסון</u>

2.3.1. ראשית סידרנו את הדאטה שיתאים לאלגוריתם הרעיוני שהצגנו בסעיף 1.4.

2.3.1.1. הערה: עדיפות "1" היא הגבוהה ביותר ועדיפות "5" היא הנמוכה ביותר.

	item	item_prob	quantity_to_allocate	expected_value_in_order	priority
0	100	0.004831	1	0.193237	5
1	101	0.014493	4	0.579710	4
2	102	0.033816	7	1.352657	3

2.3.2. פונקציות מחלקת Allocation:

2.3.2.1. Init – אתחול אובייקט במחלקה מכיל:

2.3.2.1.1. דאטה פריים מטיפוס פנדס ובו המשטחים לאחסון

2.3.2.1.2. מספר הקומות במחסן

2.3.2.1.3. מספר התאים בכל קומה

2.3.2.1.4. רשימת העדיפויות של המשטחים לאחסון

2.3.2.1.5. אתחול המחסן

2.3.2.1.6. אתחול משתנה allocation_result המחזיק במיקומים עבור כל סוג משטח.

2.3.2.2. Setup – פונקצייה המבצעת הכנות לקראת פעולה החלוקה ובהן:

2.3.2.2.1. הגדרת זמן הגעה מכל תא לנקודת האיסוף I/O

2.3.2.2.2. סידור טבלת המשטחים לאחסון בסדר יורד על פי משתנה expected_value_in_order

2.3.2.2.3. סידור משתנה available_cells המתעד תאים פנויים לאחסון בכל שלב

2.3.2.3. Allocation – פונקצייה האקלוציה. כפי שתואר באלגוריתם, כל עוד היו משטחים לאחסון ביצענו 5 סבבי אלוקציה בסדר עדיפות יורד (משטחי "עדיפות 1" ואז משטחי "עדיפות 1/2" וכן הלאה). בכל סבב השתמשנו בלולאת while לשיבוץ כל המשטחים תוך עדכון משתנים דינאמיים למעקב אחר החלוקה (available_cells,current_items_to_allocate).

2.3.3. אופן השימוש במחלקה:

```
allocate = Allocation(items,8,16)
allocate.setup()
allocate.allocation()
```

2.4. <u>מימוש אלגוריתם השליפה</u>

מימוש הקוד באמצעות סימולציה (וניהול יומן אירועים), לעומת זאת, מאפשר לפרק את הפעולות המבוצעות בזמן השליפה לאירועים נפרדים. אלגוריתם כזה, "יעניק" לנו חופש יותר גדול בתיעוד הזמנים ולדייק ככל הניתן את זמני הפריקה. לכן, החלטנו לממש את הקוד בשיטה זו.

האינפוט של האלגוריתם הינו הזמנה בת 40 פריטים. את ההזמנות ייצרנו באמצעות מחלקת Order (פירוט בהמשך).

ראשית, נבצע תהליך של Pre-processing אשר ישרת אותנו בזמן ריצת האלגוריתם:

- נכין מילון המונה את כמויות הפריטים לפי סוג ממה שקיים בהזמנה. בכל פעם שנעמיס פריט על המעלית, נחסיר אותו מהמונה של הפריטים מרשימה זו. כך נוכל לנטר אחר כמות הפריטים שנותרו מכל סוג שנותר להעמיס.
- נכין רשימה אשר מייצגת את הפריטים שיש בכל קומה ומופיעים בהזמנה. הרשימה תסודר בסדר עולה של קרבת הפריטים למעלית. מתחשב במספר הפריטים מכל סוג

שקיימים בהזמנה. לדוגמה: פריט מסוג x שמופיע פעמיים בהזמנה, יכול להופיע עד פעמיים בכל קומה. כל פעם שנשלוף פריט מסוג מסוים מתא נוריד אותו מהרשימה.

- נייצר שמונה אירועים מסוג סיום העמסת שאטל (פירוט אירועים מטה).
- נייצר אירוע אחד מסוג העמסת מעלית לקומה הראשונה בזמן השווה לזמן הגעת השאטל עם הפריט הראשון ברשימה.

2.4.1. הגדרנו אירועים מן הסוגים הבאים:

2.4.1.1. **העמסת מעלית:** שליחת מעלית לקומה להעמסת פריט. זהו האירוע הראשון שרץ בתחילת התהליך.

קורא לאירועים מסוג "סיום פריקת מעלית" ו"סיום העמסת שאטל". במידה והועמס על שאטל פריט שהמכסה שלו מולאה בהזמנה, אז נקרא לאירוע מסוג "החזרת פריט".

חישובי זמנים ליצירת העמסת מעלית: זמן העליה של המעלית לקומה (2 שניות לקומה).

2.4.1.2. **סיום פריקת מעלית:** נבדוק באיזו קומה יש פריט שמחכה לאיסוף המעלית. את הבדיקה נבצע בסדר עולה כדי לתעדף את הקומות הנמוכות. ברגע שמוצא את הקומה הראשונה שממתינה – קורא לאירוע מסוג "העמסת מעלית".

במידה ואין אף קומה עם פריט שממתין לאיסוף ולא סיימנו עדיין להעמיס את כל ההזמנה (סיכוי שואף לאפס) – נבצע בדיקה נוספת בעוד 10 שניות (קורא לעצמו בזמן של 10 שניות מאוחר יותר).

חישובי זמנים ליצירת סיום פריקה: 5 שניות העמסת המעלית, 3 שניות פריקה שלה וזמן הירידה של המעלית (2 שניות לקומה).

2.4.1.3. **סיום העמסת שאטל:** במידה ונותרו בקומה של השאטל פריטים השייכים להזמנה, נשלוף מהרשימה את הפריט הקרוב ביותר ואותו נעמיס על השאטל.

חישובי זמנים ליצירת סיום העמסת שאטל: 5 שניות העמסה לשאטל, 5 שניות העמסה למעלית (במידה ואנו לא נקראים מאירוע "החזרת פריט"), זמן ריצת השאטל (4 שניות כפול מיקום התא – שתי שניות למרחק התא פעמיים, הלוך וחזור).

במידה ומדובר בהעמסה שלאחר החזרה של פריט זמן ריצת השאטל יחושב בהתאם (2 שניות כפול מרחק השאטל מהתא הפנוי הקרוב ועוד 2 שניות כפול המרחק מהתא למעלית)

2.4.1.4. **החזרת פריט:** באירוע זה אנחנו מחזירים את הפריט שאינו דרוש יותר במשלוח, לתא הפנוי הקרוב ביותר למעלית ונחשב את זמני הפעולות. האירוע קורא בסופו לאירוע מסוג "סיום העמסת שאטל" – כי אנחנו ממשיכים לפריט אחר שברצוננו להעמיס.

חישובי זמנים ליצירת החזרת פריט: 2 שניות כפול מרחק השאטל מהתא הפנוי הקרוב ביותר.

*** עבור כל אירוע, אנו מחשבים את זמניו בהתאם לפעולות שנעשות בו ולזמנים הנתונים של כל פעולה.

2.4.2. הגדרנו שלוש מחלקות, כאשר כל אחת מהן מחזיקה את הנתונים הבאים:

2.4.2.1. מחלקת Event:

2.4.2.1.1. סוג האירוע

2.4.2.1.2. הזמן בו מתרחש

2.4.2.1.3. איזו קומה התרחש

2.4.2.2. מחלקת Shuttle:

2.4.2.2.1. מיקומו (קומה)

2.4.2.2.2. סטטוס שלו (האם הוא מוכן להעמיס את הפריט למעלית)

2.4.2.2.3. הפריט שכרגע בטיפול שלו

2.4.2.2.4. ההחזרה האחרונה שלו (במידה והחזיר פריט לתא)

2.4.2.2.5. רשימת הפריטים שיש בקומה וקיימים בהזמנה

2.4.2.3. מחלקת Order:

2.4.2.3.1. Init – אתחול אובייקט במחלקה המכיל: סידור המחסן בשתי תצורות שונות, תכולת ההזמנה, זמן ההזמנה.

2.4.2.3.2. Is_Feasible – בדיקת תקינות ההזמנה.

2.4.2.3.3. Prepare_Items_For_Gathering – סידור ההזמנה בכל קומה לפי אלגוריתם השליפה שפירטנו לעיל.

2.5. במהלך מימוש האלגוריתם בפייתון ניסינו לשמור ככל האפשר על הדמיון אל האלגוריתם המתמטי. השינויים שביצענו היו בעיקר שינויי מיקום שתרחישים באירועים וסידור כך שיבצע את הרעיון המקורי (פסאודו קוד ראשוני מצורף בנספחים).

3. תוצאות

לאחר יישום האלגוריתמים, ייצרנו 10 הזמנות באמצעות המחלקה Order שהגדרנו.
לצורך מדידת והערכת ביצועי המודל שלנו, יצרנו את המשתנים הבאים אותם נאמוד עבור כל 10 ההזמנות:

3.1. Order Times: זמן הוצאת הפריטים הכולל (C_{max}). לפי הגדרת התרגיל אנו שואפים למזערו.

3.2. Order Waits: זמני ההמתנה של המעלית (נספר רק לאחר ההעמסה הראשונה). לפי אלגוריתם השליפה שהגדרנו וההנחות שהנחנו, נצפה לראות שלא קיימים זמני המתנה, או זמנים מעטים מאוד לכל היותר.

3.3. Orders Returns: מספר הפעמים הכולל שבוצעו החזרות פריט – מתעדכן כשאר יוצרים אירוע מסוג החזרת פריט עבור קומה ספציפית. מאפשר לנו לנתח את יעילות השליפה ה"מהירה" שהגדרנו ורואים כמה פעמים בפועל "בזבזו" השאטלים זמן על החזרת פריט.

3.4. Orders Floor Daily: רשימה המכילה את כמות הפריטים שנמצאים בהזמנה עבור כל קומה. נצפה למספרים גבוהים יותר בקומות הנמוכות ומספרים נמוכים בקומות הגבוהות.

3.5. Orders Floor Gathered: מספר הפריטים שנאספו בפועל מכל קומה עבור כל הזמנה. גם כאן, נצפה לראות ערכים בסדר יורד מהקומות הנמוכות ועד לגבוהות.

3.6. Avg_order_time : זמן ממוצע לכל ההזמנה (נגזר מכל זמני ההזמנות מסעיף 3.1).

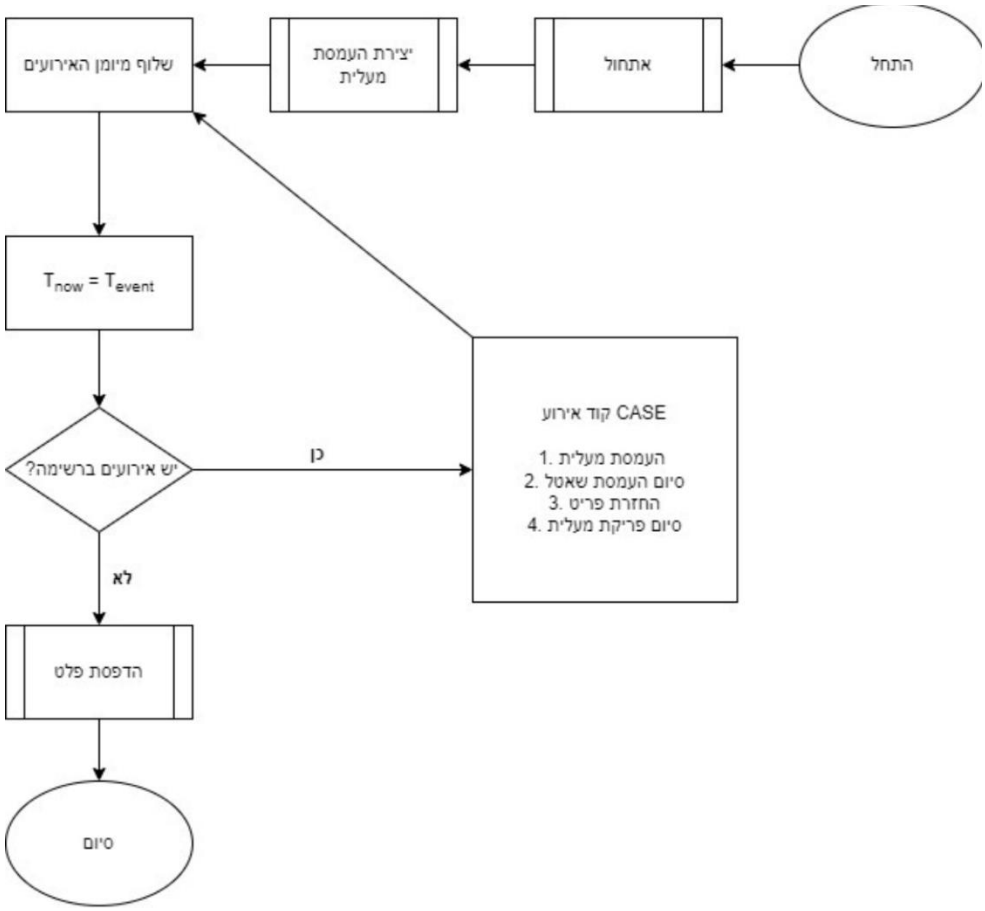
לאחר הרצת המודל, קיבלנו את התוצאות הבאות :

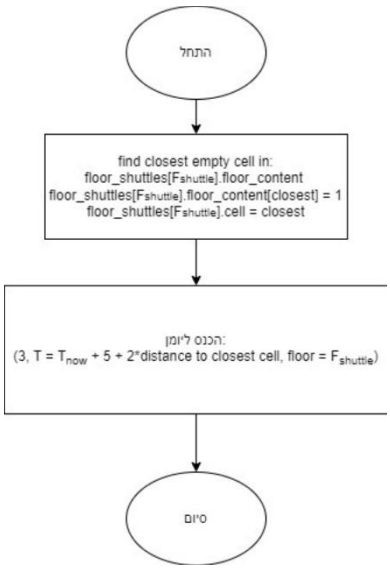
```
order times: [565, 549, 565, 565, 577, 597, 621, 621, 603, 549]
orders waits: [0, 0, 0, 0, 0, 2, 0, 0, 0, 0]
orders returns: [10, 7, 10, 8, 14, 7, 8, 11, 13, 3]
orders floor daily: [[23, 18, 21, 19, 19, 16, 17, 16], [24, 24, 25, 21, 22, 20, 16, 13], [25, 26, 24, 21, 20, 19, 20, 16], [28,
22, 20, 20, 17, 15, 11, 8], [23, 24, 20, 15, 19, 15, 17, 14], [22, 23, 19, 17, 14, 15, 19, 11], [21, 21, 22, 16, 17, 15, 13, 1
2], [20, 20, 19, 20, 17, 14, 11, 9], [21, 22, 24, 16, 17, 16, 13, 8], [21, 21, 22, 15, 16, 12, 17, 10]]
orders floor gathered: [[12, 11, 9, 4, 2, 1, 1, 0], [13, 11, 8, 5, 2, 0, 1, 0], [13, 12, 5, 6, 1, 2, 1, 0], [13, 11, 8, 5, 2,
0, 1, 0], [11, 12, 7, 5, 4, 0, 1, 0], [11, 12, 7, 5, 3, 2, 0, 0], [9, 9, 10, 6, 4, 1, 1, 0], [10, 10, 9, 5, 4, 1, 1, 0], [10, 1
1, 8, 6, 2, 2, 1, 0], [13, 12, 7, 5, 1, 1, 1, 0]]
avg_order_time: 581.2
```

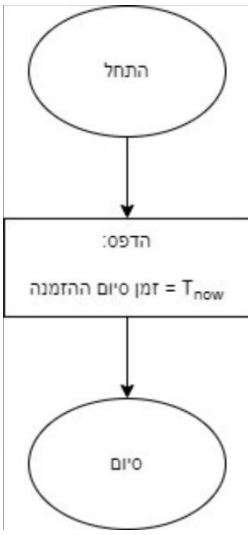
- ראשית, אנו רואים כי הזמן הממוצע של כל ההזמנה הינו 581 שניות, כלומר כ-14.5 שניות לפריט. מדובר בזמן קצר מאוד, שכן לוקחות 8 שניות סה"כ כדי להעמיס ולפרוק את המעלית (פעולות שנספרות <u>תמיד</u> בחישוב הזמנים).
- כפי שהנחנו, ניתן לראות כי המעלית כמעט ולא ממתינה כלל לשאטלים. בהחלט תואם את ציפיותינו מאלגוריתם השליפה.
- ניתן לראות כי אכן בוצעו החזרות של פריטים בכל הזמנה, אם כי לפני הנתונים לא ניתן להסיק האם החזרות הפריטים משפיעות על הזמן הכולל.
- אנו רואים כי ישנו דפוס חוזר במספר הפריטים שנמצאים בכל קומה ומופיעים בהזמנה : באופן כללי המגמה היא כמות יורדת של פריטים ככל שעולים בקומות וזה אכן תואם את הנחתנו ומאשש את יעילותו של אלגוריתם האחסון שלנו.
- אנו יכולים לראות כי עבור כל הזמנה, הרוב המוחלט של הפריטים נאסף מהקומות הראשונות (קומות 1-3 בעיקר) וזה נותן גיבוי לזמני השליפה הכוללים הקצרים יחסית שהתקבלו.

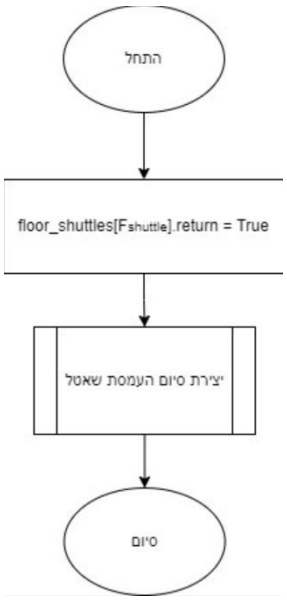
4. <u>נספחים</u>

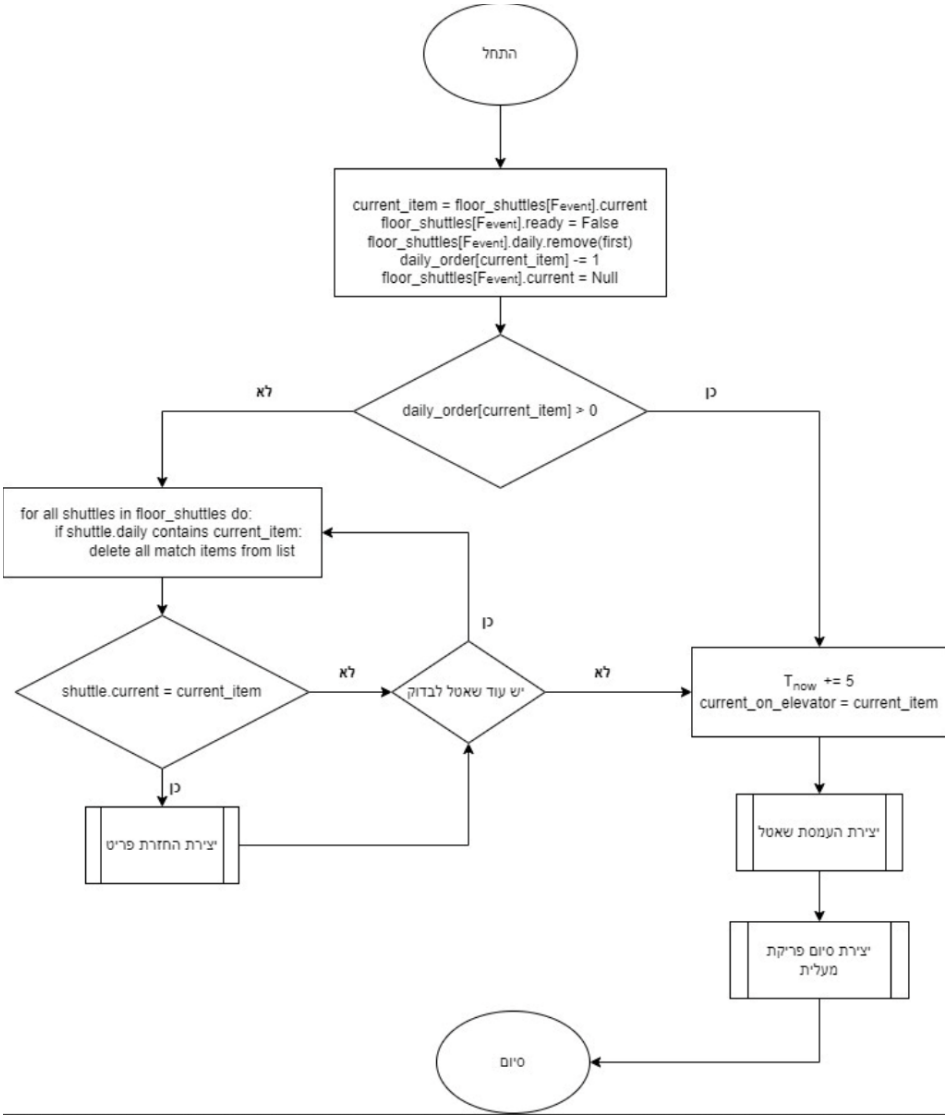
4.1. לצורך מימוש אלגוריתם מבוסס סימולציה, בנינו תרשימים שסייעו לנו בהבנת התהליכים בצורה הטובה ביותר:

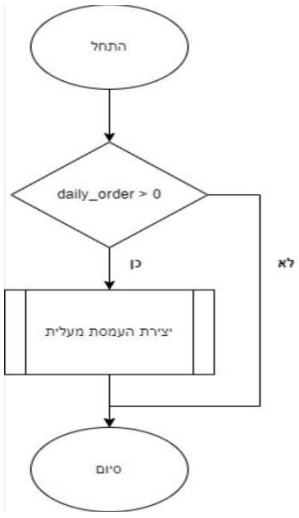


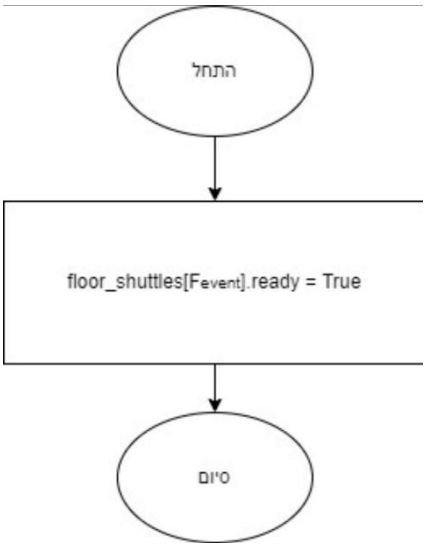


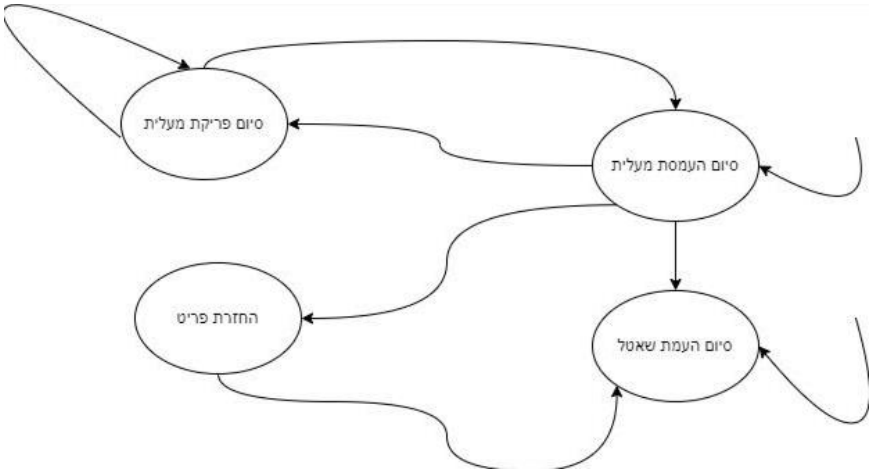





התחל
current_item = floor_shuttles[Fevent].current
floor_shuttles[Fevent].ready = False
floor_shuttles[Fevent].daily.remove(first)
daily_order[current_item] -= 1
floor_shuttles[Fevent].current = Null
daily_order[current_item] > 0
לא
כן
for all shuttles in floor_shuttles do:
if shuttle.daily contains current_item:
delete all match items from list
shuttle.current = current_item
לא
יש עוד שאטל לבדוק
כן
לא
כן
יצירת החזרת פריט
Tnow += 5
current_on_elevator = current_item
יצירת העמסת שאטל
יצירת סיום פריקת מעלית
סיום


התחל
daily_order > 0
כן
לא
יצירת העמסת מעלית
סיום


התחל
floor_shuttles[Fevent].ready = True
סיום



4.2. מיפוי התאים:

44	42	40	38	36	34	32	30	28	26	24	22	20	18	16	14	
42	40	38	36	34	32	30	28	26	24	22	20	18	16	14	12	
40	38	36	34	32	30	28	26	24	22	20	18	16	14	12	10	
38	36	34	32	30	28	26	24	22	20	18	16	14	12	10	8	
36	34	32	30	28	26	24	22	20	18	16	14	12	10	8	6	
34	32	30	28	26	24	22	20	18	16	14	12	10	8	6	4	
32	30	28	26	24	22	20	18	16	14	12	10	8	6	4	2	
30	28	26	24	22	20	18	16	14	12	10	8	6	4	2	0	I/O