מטלה 0 תכנות מונחה עצמים

מגישים:319645735,208640326

https://github.com/Daniel-Ros/ex0

חלק ראשון - סקירת ספרות:

https://elevation.fandom.com/wiki/Destination_dispatch

סיקור על האלגוריתם של Destination dispatch. אלגוריתם של מעליות חכמות רבות בימנו.

https://github.com/do-ryan/destination-dispatch-elevator-simulation

מימוש של האלגוריתם בפייתון. עם הסברים מלאים ב pdf

https://www.youtube.com/watch?v=xOayymoIl8U

סרטון הסבר קצת על איך מעליות עובדות וכל מיני שיטות ושיפורים שניתן לעשות על מנת לשפר אותם עוד יותר, והצגת התופעה שהמעליות היותר פשוטות בדרך כלל גם יביאו את התוצאה היותר טובה

חלק שני - ההבדל בין אלגוריתם online לאלגוריתם offline:

ההבדל בין שני סוגי האלגוריתמים האלו הוא בזמינות הקלט. באלגוריתם offline המידע ניתן לנו מראש, לדוגמא אלגוריתם שפותר קובייה הונגרית, צריך להזמין את המצב של הקוביה לפני הרצת האלגוריתם ולא משנים אותו בסוף. לעומת אלגוריתם online שמקבל קלט בזמן הריצה ומגיב לפיו, לדוגמא אלגוריתם של נהג אוטומטי, חוץ מיעד האלגוריתם צריך לקרוא את תנאי הכביש בזמן אמת ולקחת החלטות לפיהם.

אלגוריתם offline פשוט:

מכיוון שכל המידע שלנו נתון מראש נוכל דבר ראשון לעבור על המידע על מנת לחשב את המסלול האופטימלי.

נוכל לראות לאיזה כיוון יש יותר דרישה בתחילת ההרצה, ונשלח את המעליות לנוסע הראשון, וניקח את כל הנוסעים בדרך מכיוון שאנחנו כבר יודעים באיזה תחנות צריך לעצור, נוכל לבדוק (בתנאי שיש לנו מספיק מעליות) אם נוכל לחלק את הנסיעה לכמה חלקים על מנת שנעבוד במקביל. ברגע שמעלית תסיים את הנסיעה, היא תכין את עצמה לנסיעה הבאה,כלומר תחנה בקומה של הנוסע הבא.

בהנחה ויש לנו כמות אנשים עולים באותה קומה בהפרשי זמני שקטנים מהזמן שבו מעלית תוכל להגיע בתנאי שהמעלית לקחה את הנוסע, המעלית תחכה עד שהנוסע הבא יעלה

חלק שלישי - אלגוריתם online:

נאחסן את כל הבקשות שלנו בתור של קריאות (CallForElevator).

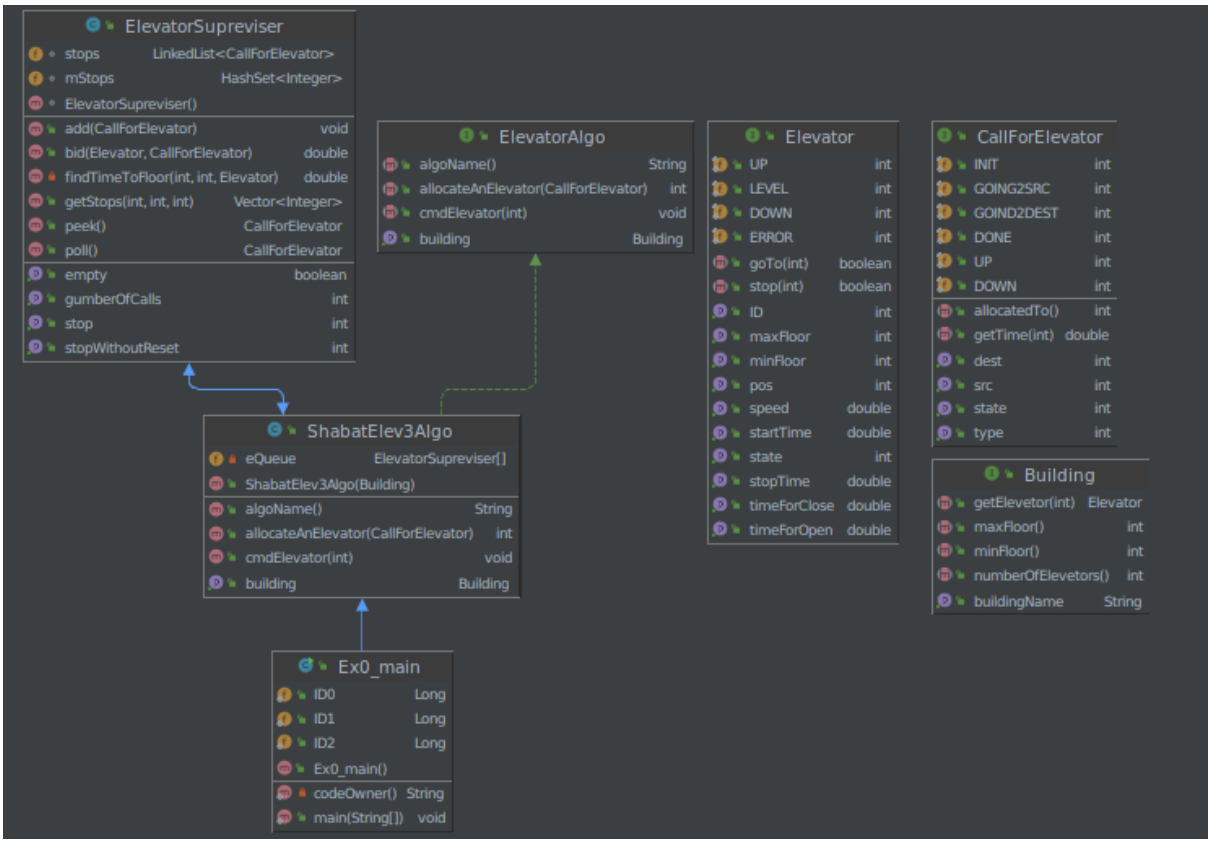
כל פעם שנצתרך להביא פקודה למעלית:
נמחוק את הפעולות שבצענו
-נציץ באיבר הראשון בתור
- נבדוק האם הקריאה נאספה (כלומר בן אדם עלה למעלית)
    - אם לא ניסע לאסוף אותו
    - אחרת ניסע להוריד אותו

כאשר מעלית תהיה בתנועה נוכל לעבור על כל הקריאות שלנו ונבדוק:
- האם הקריאה נאספה:
    - האם קומת המקור של הקריאה בין מיקום המעלית לבין קומת היעד של המעלית
    -אם כן הכנס את הקומה למחסנית
-אחרת:
    - האם קומת היעד של הקריאה בין מיקום המעלית לבין קומת היעד של המעלית
    -אם כן הכנס את הקומה למחסנית

בעת קבלת קריאה חדשה: נעבור על כל המעליות, וכל מעלית תחשב כמה זמן יקח לה לסיים את כל המשימות הנוכחיות כולל הקריאה החדשה, המעלית שתעשה זאת בהכי פחות זמן תקבל את המשימה.

לפני תחילת החישוב נאתחל משתנה צבירה ב0, ומשתנה של קומה אחרונה במיקום המעלית.
החישוב יתבצע בכך שנעבור על כל המשימות שלנו ועבור כל קריאה:
- נבדוק האם הקריאה נאספה או לא
-אם היא לא נאספה, נוסיף את הזמן בשניות שיקח לנו להגיע מהקומה האחרונה למתשנה הצבירה שלנו, ובנוסף נתייג את הקומה הזאתי כך שנדע שכל האנשים מהקומה הזאתי נאספו, ונכניס למשתנה של הקומה האחרונה שלנו את הערך של קומת המקור
-לאחר מכן נוסיף לאיבר הצבירה שלנו את הזמן בשניות שיקח לנו לעבוד מהקומה האחרונה לקומת היעד
-בכל חישוב כזה נקרא לפונקציה של getStops עם הערכים של המסלול שלנו, וכך נוכל לקבל עצירות בניים, עבור כל עצירת בניים נוסיף את הזמן שיקח לנו לעצור בה
-

חלק רביעי - דיאגרמת מחלקות:



חלק חמישי - JUINT:

הבדיקות שלנו יתחלקו ל2.

בדיקות על ElevatorSuperviser:

-נבצא בדיקה שבודקת האם המעלית מחשבת את הזמן כראוי, כשאר נשים לה קריאות ונבדוק שהמעלית אכן מביאה תוצאה תיקנית לגבי כמה זמן לוקח לה לבצא את הפעולות

-בדיקה על הadd שאכן האיברים נכנסים בסדר הנכון.

-נבדוק את פונקציה getStops שמביאה לנו את כל תחנות הבניים כמו שצריך ומאפסת את הcache הפנימי

-נבדוק את פונקציה getSop בכך שנכניס קריאות ונדמה קריאות שנאספו ונבדוק שאכן האגוריתם מביאה לנו את היעד הנכון

בדיקות על ShabatElev3Algo:

-נבדוק את הפונקציה AllocateAnElevator בכך שניצור בניין עם מעליות , ונתחיל לשלוח לו קריאות ונראה שאכן המעלית הכי טוב היא המעלית שמוקצת לקריאה

-נבדוק את הפונקציה cmdElevator בכך שניצור בניין עם מעליות , ונתחיל לשלוח לו קריאות ונראה שאכן המעליות נשלחות ליעד התקין שלהם.