תרגיל בית 1- פתרון בעיות על ידי חיפוש

מבוא

מטרת התרגיל היא להתנסות בפתרון אחת הגרסאות של משחק zuma בעזרת שיטות חיפוש שונות, אותן למדנו בקורס.

עקרון המשחק הוא שהשחקן (הצפרדע) צריך לירות כדורים כך שמצטמצם השורה של הכדורים. השחקן מקבל צבע כדור לירות לפי מה שיש במחסנית. כאשר יש 3 או יותר כדורים אחד אחרי השני בשורה מאותו הצבע אז הם נעלמים.

המשחק מורכב מ:

- מחסנית של כדורים שהשחקן יכול לירות
- שורה/מערך של כדורים עליו השחקן לצמצם

הבעיה

כקלט תקבלו את המצב ההתחלתי של המשחק. זוהי זוג סדור של שני רשימות, הראשון מסמל את השורה של כדורים עליו השחקן לצמצם והשני מסמל את המחסנית של כדורים שהשחקן יכול לירות, כלומר הסדר צבעים של כדורים. לאומת המשחק האמיתי, שחקן יכול לירות לכל מקום בשורה ויכול גם לדלג הכדור הנוכחי במחסנית, כלומר אנחנו מתעלמים מאלמנט המעגלי, ואנחנו מקבלים את כל השורה בבת אחת ואין אלמנט של התקדמות לצפרדע.

זאת אומרת מטרת המשחק היא אך ורק לצמצם את השורה בכמות הכדורים המינימלית. המשחק יכול להיגמר ב 2 אופנים:

- **הצלחה:** השחקן צמצם את השורה
- **כשלון:** השחקן לא הצליח לצמצם את השורה עם המחסנית הנתונה לו

תיאור המשימה

לתרגיל זה מצורף קוד הממש אלגוריתמי חיפוש שלמדנו. עליכם לממש את המחלקה המייצגת משחק Zuma כך שניתן יהיה לפתור משחק נתון על ידי שימוש בקוד החיפוש המצורף.

קובץ ex1.py המצורף מכיל את חתימות הפונקציות שעליכם לממש. (ניתן כמובן גם למשש פונקציות נוספות):

1. פונקציית successor המקבלת מצב ומחזירה רשימה של זוגות סדורים, tuple, של פעולה והמצב הנובע מהפעולה.
2. פונקציית goal_test המקבלת מצב ומחזירה true אם המצב הנתון הוא מצב מטרה (מה שמוגדר כהצלחה בסעיף קודם), ו-false אחרת

3. פונקציית היוריסטיקה (h) המקבלת קודקוד (לעומת מצב כאשר קודקוד מכיל מצב) ומחזירה את העלות המשוערת להגעה מהמצב הנתון למטרה. שימו לב כי טיב ההיוריסטיקה ישפיע על הביצועים של אלגוריתמי החיפוש.
4. עליכם גם לכתוב את מספרי תעודת זהות שלכם במשתנה id מחוץ למחלקה

<u>אין לשנות את חתימות הפונקציות כלל!</u>

ייצוג קלט

הקלט שמתאר את הבעיה מועבר לפונקציה

```
create_zuma_problem(game)
```

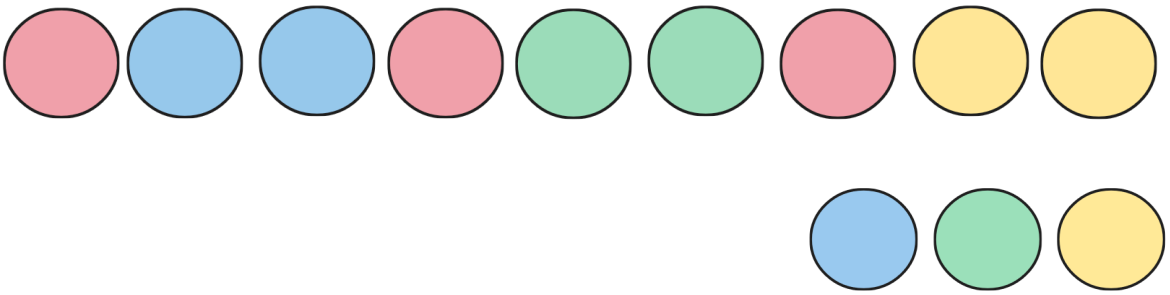
כאשר:

- הזוג הראשון הוא השורה עליו השחקן לצמצם והשני הוא המחסנית
- 1 - מסמל כדור אדום
- 2 - מסמל כדור כחול
- 3 - מסמל כדור ירוק
- 4 - מסמל כדור צהוב

לדוגמא הקריאה:

create_zuma_problem ((1,2,2,1,3,3,1,4,4), (2,3,4))

מתארת את המשחק הבא:



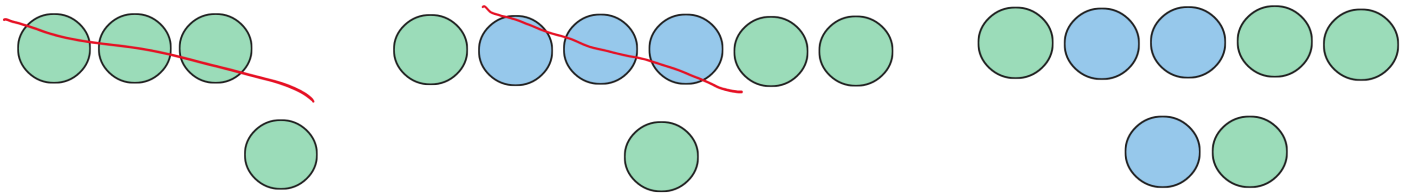
כאשר התמונה הראשונה מתארת את השורה שעליו השחקן לצמצם והתמונה השנייה מתארת את המחסנית, את סדר הכדורים שהשחקן מקבל כדי לירות על השורה (משמאל לימין).

אופן וחוקיות התזוזה

לרשותכם n+2 פעולות כאשר n מייצג אורך השורה:

- יש n+1 פעולות של להכניס את הכדור שבראש המחסנית בין כל שני כדורים בשורה או בקצוות השורה
- יש פעולה אחת של לדלג על הכדור שבראש המחסנית לעבור לכדור הבא (נזרק ולא משומש יותר)

כל פעולה יכולה לגרום לשורה להצטמצם ויכולה לגרום לתגובת שרשרת של צמצום, כלומר אם אחרי שהכנסנו כדור ונוצר 3 או יותר ברצף מאותו הצבע נוציא את הרצף ואז נצטרך לעשות את הבדיקה שוב פעם כי יכולה להיווצר מצב שבו קיים רצף נוסף לאחר הוצאת רצף. לדוגמה מצב משחק כזה:



נכניס את הכדור הכחול ככה שיווצר רצף של 3 כדורים כחולים, הכדורים מצטמצמים ואז אחר בדיקה נוספת קיימת רצף חדש של 3 כדורים ירוקים ומצמצמים אותם גם, לכן לאחר פעולה אחת צמצמנו את השורה.

שימו לב, שהמצב ההתחלתי לא יכול להתחיל עם רצפים גדולים מ2.
<u>הערה</u>: מצב כישלון במשחק (נגמר המחסנית ושורה לא ריקה) אין להמשיך לפתח את המצבים על ענף זה.

דגשים

- שימו לב שכדי לקבל את המצב מתוך קודקוד החיפוש (node), ניתן לגשת ל-node.state
- אתם יכולים לייצג מצב של הבעיה איך שאתם רוצים, אבל שימו לב שהקוד של החיפוש דורש שהמצב יהיה hashable , ולכן לא ניתן להשתמש ברשימה או במילון (או ב – tuple המכיל רשימה או מילון) על מנת ליצג מצב. ניתן לממש מחלקה המתארת מצב ויש לה פונקצית hash או להשתמש ב – tuple
- אמנם ייצוג המצב הוא בחירה שלכם אולם יש להקפיד על ייצוג הפעולות בדיוק כפי שכתוב – אותיות גדולות/קטנות, קווים תחתונים, ייצוג הפעולות נכון וכו. הבדיקה אוטומטית, ולכן אם תטעו כאן תקבלו ציון נמוך מאד, וחבל.

בדיקת התרגיל

התרגיל המוגש ייבדק באופן אוטומטי, ולכן חשוב להקפיד על שמות מדויקים של קבצים, מחלקות ופעולות. הבדיקה האוטומטית תשתמש באלגוריתמי חיפוש שונים (A* וGBFS) על מנת לפתור אוסף קלטים בגדלים שונים. לאחר מכן הפתרון ייבדק צעד אחר צעד על מנת לוודא שהפתרון אכן תקין.

הקובץ ex1_check.py המצורף מריץ כל את אלגוריתמי החיפוש שנבדוק על מספר קלטים. קובץ זה לא מבצע בדיקת נכונות של הפתרון המוחזר, ולכן זוהי אחריות שלכם לוודא שהפתרונות שלכם נכונים.

הרצת הקובץ באמצעות הפקודה:

```
python3 ex1_check.py
```

הסבר קצר על הקוד המצורף

הקוד מורכב מחמישה קבצי פייתון:

1. Ex1.py – קובץ שבו נעשית העבודה העיקרית שלכם, והקובץ היחיד שעליכם לשנות. אמור להכיל את ה-class של ZumaProblem המכיל את כל הפונקציות כפי שמתואר בסעיף "תיאור משימה". הקובץ כולל את חתימות של הפונקציות שעליכם לממש. **התרגיל יבדק עם קובץ ex1.py שלכם, ושאר הקבצים כפי שהם מופיעים במצבם המקורי ולכן אין טעם לשנות קבצים אחרים** (למעט הבעיות השונות שנבדוק עליהם את הקוד).
2. ex1_check.py – קובץ המכיל פונקציות מעטפת המנסות לפתור את הבעיה, ומכיל בעיה קטנה לדוגמא שאפשר לפתור. זה הקובץ שעליכם להריץ לבדיקת הפתרון שלכם.
3. Search.py – מכיל את אלגוריתמי החיפוש, ומכיל את מחלקת node.
4. שאר הקבצים – קבצי עזר

הגשה

- תאריך ההגשה עד ה 27.11.24
- הגשה ביחידים בלבד
- את הת.ז. של המגיש יש לרשום במשתנה id בקובץ ex1.py וכן לצרף קובץ details.txt המכיל את שם ות.ז של המגיש.

- שאלות על התרגיל יש לשאול בפורום שאלות ותשובות יעודי ב"למדה". שאלות יענו אחת ליומיים - שלושה במרוכז.
- הגשת התרגיל תהיה דרך מערכת ה submit. יש להגיש רק את הקובץ ex1.py וקובץ details.txt. אין להגיש קבצי עזר המצורפים לתרגיל. אין להגיש קובץ בפורמט אחר (לדוגמא zip או rar)