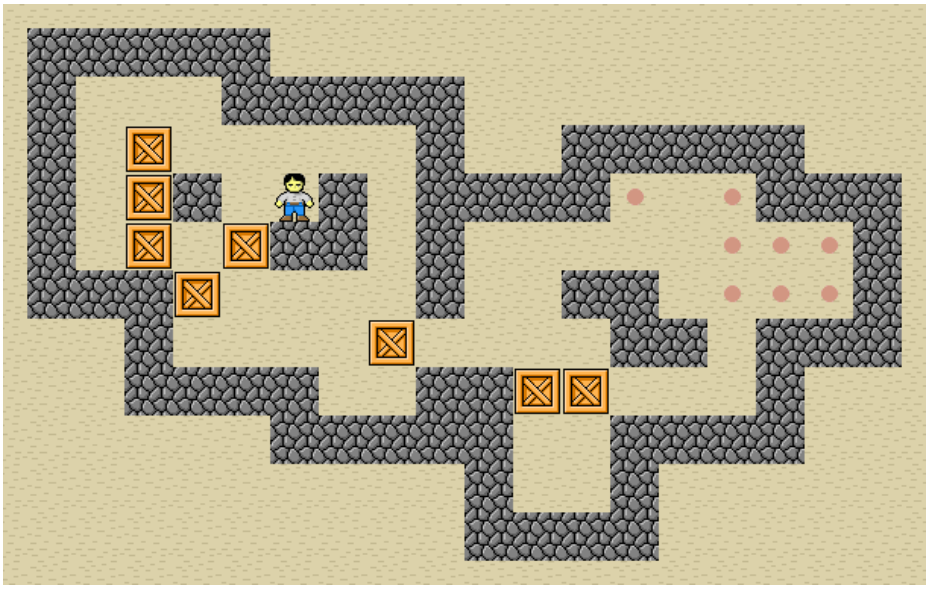
תרגיל בית 1- פתרון בעיות על ידי חיפוש

מבוא

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



פירוש המילה סוקובאן ביפנית היא "מנהל מחסן". העיקרון של כל שלב במשחק הוא זהה:- השחקן חייב לדחוף את כל הארגזים ליעדים שונים במבוך (הנקודות הורודות המסומנות). השחקן יכול לנוע לכל היותר בארבעה כיוונים וישנו מספר אפשרויות מוגבל להזזת הארגזים.

הכללים קובעים שניתן להזיז רק קופסה אחת בכל פעם; כלומר, אם יש שתי קופסאות ברצף, השחקן לא יכול לדחוף אותן. אפשר לדחוף קופסאות רק על ידי שחקן ולא למשוך קופסא; שחקנים וקופסאות לא יכולים לעבור דרך מכשולים (קירות או קופסאות אחרות).
לוח המשחק מכיל 3 סוגים של אובייקטים: קיר, שחקן וארגז, ושני סוגי משבצות: משבצות רגילות ומשבצות מטרה, כאשר יש בהכרח שחקן אחד על הלוח, ומספר הארגזים שווה למספר משבצות מטרה.
לוח המשחק מורכב מ- N × M משבצות, כאשר בכל משבצת יכול להימצא לכל היותר אובייקט אחד (וקיר לא יופיע במשבצת המטרה).

הבעיה

כקלט תקבלו את המצב ההתחלתי של המשחק. זוהי מטריצה (game) בגודל NxM שמייצגת את לוח המשחק, הכניסות במטריצה מסמנות את המצב ההתחלתי של האובייקטים בו. שחקן יכול לזוז ימינה, שמאלה, מעלה ומטה, בתנאי שהמשבצת (הקיימת) שאליה הוא זז היא ריקה, או שיש בה ארגז שאפשר להזיז אותו למשבצת הריקה הסמוכה בכיוון התנועה.

מטרת המשחק היא להביא את כל הארגזים למשבצות מטרה (אין חשיבות לאיזה ארגז מגיע לאיזו משבצת מטרה).

המשחק יכול להיגמר ב 2 אופנים:

- **הצלחה:** כל הקופסאות נמצאות במקומות האחסון.
- **כשלון:** קופסה שלא ניתן יותר להזיזה ממקומה ואינה במקום.

תיאור המשימה

לתרגיל זה מצורף קוד הממש אלגוריתמי חיפוש שלמדנו. עליכם לממש את המחלקה המייצגת משחק Sokoban 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 לממש פונקציות נוספות):

1. פונקציית successor המקבלת מצב ומחזירה tuple הכולל את כל הפעולות המותרות מאותו המצב
2. פונקציית result המקבלת מצב ופעולה ומחזירה מצב חדש הנוצר כתוצאה מהפעלת פעולה נתונה במצב נתון
3. פונקציית goal_test המקבלת מצב ומחזירה true אם המצב הנתון הוא מצב מטרה (מה שמוגדר כהצלחה בסעיף הקודם), ו-false אחרת
4. פונקציית היוריסטיקה (h) המקבלת node (שימו לב, לא מצב), ומחזירה הערכה של המרחק למטרה. אתם יכולים לממש איזו יוריסטיקה שאתם רוצים עבור בעיה זו ובלבד שתהיה אדמיסבילית (קבילה).
5. 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_sokoban_problem(Game)
```

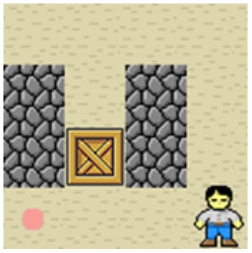
כאשר:

- `Game` – מטריצה N × M שמייצגת את לוח המשחק. כל כניסה במטריצה מקבלת את אחד הערכים הבאים:
- `8` – קיר.
- `5` – ארגז שנמצה במשבצת המטרה.
- `4` – ארגז.
- `3` – שחקן שנמצה במשבצת המטרה.
- `2` – שחקן.
- `1` – משבצת המטרה.
- `0` – משבצת ריקה.
-

דוגמה: הקריאה

```
create_sokoban_problem())0,0,0,0(,
                    (8,0,8,0),
                    (8,4,8,0),
                    (1,0,0,2)))
```

מתארת משחק Sokoban בגודל 4 × 4, שיש בו 4 קירות, שחקן, וזוג ארגז + מטרה, המסודרים באופן הבא:



כל פעולה מיוצגת ע"י string (מחרוזת) המכיל את שם הפעולה. הפעולות החוקיות הן:

- הזזת שחקן למלה – מיוצג ע"י 'U'
- הזזת שחקן ימינה – מיוצג ע"י 'R'
- הזזת שחקן למטה– מיוצג ע"י 'D'
- הזזת שחקן שמאלה – מיוצג ע"י 'L'

כל פעולה כזאת גורמת לשחקן לזוז משבצת אחת בכיוון הרצוי, חוץ מתזוזה לכיוון קיר.

את השחקן ניתן להזיז ע"פ החוקים.

לדוגמה, פתרון אפשרי (שהוא גם הפתרון האופטימאלי) עבור הבעיה המתוארת למעלה הוא:

['U', 'U', 'U', 'L', 'L', 'D', 'D', 'U', 'U', 'R', 'R', 'D', 'D', 'D', 'L', 'L']

אם שחקן יעבור למיקום של קופסה, הקופסה תזוז ריבוע אחד לאותו כיוון. עם זאת, קופסאות ושחקנים לא יכולים לעבור דרך קירות או מכשולים.

שחקן לא יכולים לדחוף יותר מקופסה אחת בכל פעם; אם שתי קופסאות נמצאות ברצף השחקן לא יוכל להזיז אוֹתָם.

המטרה מושגת כאשר כל קופסא ממוקם באזור אחסון בלוח.

<u>הערה</u>: מצב כישלון במשחק (קופסא נמצאת במיקום ללא יכולת הזזה) 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 לייצג 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 מאוד, וזה יהיה חבל.

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 את כל 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
```

שימו לב שהבדיקה מריצה כל חיפוש עם timeout של שניות בודדות, ולכן ייתכן שתקבלו תשובה שמייצגת timeout.

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 `SokobanProblem` 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 **16.6.24**
- 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)
- **שימו לב בהגשה לבחור בקבוצה 02.** הגשות לקבוצה 01 לא יתקבלו ולא יבדקו! שם התרגיל להגשה הוא **ex-1**