# תרגיל ריצה - חיפוש

## הבעיה

עליכם לממש מנוע חיפוש התומך במספר אלגוריתמי חיפוש כדי לפתור את משחק ה- 2-holes NxM-tile puzzle (הכללה של ה- 15 puzzle שראינו בשיעור).
במשחק נתון לוח בגודל NxM המכיל NxM-2 בלוקים הממוספרים מ- 1 ועד NxM-2, ושני בלוקים ריקים. הבלוקים מסודרים בסדר התחלתי נתון כלשהו, והמטרה היא למצוא את מספר הפעולות הזול ביותר מהסידור ההתחלתי למצב הסופי.
במצב הסופי כל הבלוקים מסודרים מ- 1 ועד NxM-2 משמאל לימין ומלמעלה למטה, כאשר הבלוקים הריקים נמצאים בפינה הימנית תחתונה ושמאלה ממנה. לדוגמה, אם הלוח הוא 3x4 אז המצב הסופי הוא:

| | | | |
|---|---|---|---|
| 1 | 2 | 3 | 4 |
| 5 | 6 | 7 | 8 |
| 9 | 10 | | |

**הפעולות**
ניתן להזיז כל בלוק שנמצא בסמוך לבלוק הריק. בנוסף, נגדיר במשחק זה פעולה נוספת- הזזה של 2 בלוקים במקביל. פעולה זו מותרת רק במקרה בו 2 הבלוקים הריקים צמודים אחד לשני, במאונך או במאוזן. בניגוד למשחק ה - 15 puzzle הרגיל שראינו, בו כל הזזה נחשבת צעד אחד, במשחק זה ישנן עלויות שונות לפעולות. הזזה של בלוק אחד עולה 5, הזזה של 2 בלוקים במקביל במאוזן עולה 6 והזזה של 2 בלוקים במקביל במאונך עולה 7. לדוגמה, אם הלוח במצב הזה:

| | | | |
|---|---|---|---|
| 1 | 2 | 3 | 4 |
| 5 | | 6 | |
| 9 | 10 | 7 | 8 |

נוכל להזיז את 6 שמאלה ואז את 7 ו- 8 ביחד למעלה, כדי להגיע למצב הסופי. עלות המסלול המתואר תהיה 5+7=12.

## מימוש

**קלט**
התוכנית תקרא את כל הקלט שלה מקובץ יחיד- input.txt. השורה הראשונה בקובץ תקבע באיזה אלגוריתם להשתמש: BFS, DFID, A*, IDA*, או DFBnB. השורה השנייה בקובץ תקבע האם להדפיס את זמן הריצה (with time) או לא (no time). השורה השלישית תכיל את גודל הלוח בפורמט הבא: NxM, ז"א לוח המכיל N שורות ו- M עמודות. לאחר מכן יופיע הסידור ההתחלתי של הלוח לפי שורות, כאשר יש פסיקים בין מספרי הבלוקים. הבלוק הריק יסומן כ- "_".

**פלט**
הפלט ייכתב לקובץ output.txt. בשורה הראשונה בקובץ יש לכתוב את סדרת הפעולות שנמצאה ע"י האלגוריתם. בשורה השנייה יש לכתוב " Num:" ואח"כ את מספר הקדקודים שפותחו (שהוצאו מה-open list). בשורה השלישית יש לכתוב " Cost:" ואח"כ את עלות הפתרון שנמצא. אם בקובץ הקלט נכתב שיש להדפיס גם את זמן הריצה, בשורה הרביעית יש לכתוב את הזמן שלקח לאלגוריתם למצוא את הפתרון (בשניות).
הפעולות יסומנו על ידי מספר הבלוק שזז וכיוון ההזזה: R (ימינה), D (למטה), L (שמאלה), U (למעלה). במידה ושני בלוקים זזו יחד יש להפריד בניהם באמצעות &. הפעולות עצמם יופרדו ע"י מקף. לדוגמה, המסלול המתואר קודם ייכתב בקובץ הפלט כ- 6L-7&8U.

על מנת לקבל פלט אחיד ככל שניתן, נחיל יחס סדר על הוצאת קדקודים בעלי אותה עדיפות. סדר פיתוח הקדקודים (סדר ההוצאה מה-open-list) באלגוריתמים השונים הוא על פי זמן ייצורם. קדקודים שנוצרו באותו זמן (בעלי אב משותף) יסודרו על פי האופרטור שייצר אותם לפי הסדר הבא: 2 בלוקים שמאלה, 2 בלוקים למעלה, 2 בלוקים ימינה, 2 בלוקים למטה. אם לא ניתן להזיז 2 בלוקים הסדר יהיה: שמאלה, למעלה, ימינה, למטה, אל הבלוק הריק שנמצא קרוב יותר לשורה הראשונה ואז אל הבלוק הריק השני. אם שני הבלוקים באותה שורה תהיה עדיפות לפעולות אל הבלוק השמאלי יותר. סדר זה רלוונטי גם ל-A* במידה ויש מספר קדקודים בעלי ערך זהה בפונקציית ההערכה f(n). שימו לב שאלגוריתמים שמשתמשים במחסנית צריכים להכניס את הקדקודים בעלי אותה עדיפות בסדר הפוך, כדי שיוכלו להוציא אותם לפי סדר העדיפות שהוגדר.
לסיכום: הקדקודים יסודרו על פי 1) דרישות האלגוריתם 2) זמן ייצור 3) האופרטור שייצר אותם כאשר תזוזה של 2 בלוקים בעדיפות גבוהה יותר מתזוזה של בלוק אחד, ובכל סדר תזוזות שמאלה בעדיפות הגבוהה ביותר והעדיפות הולכת ויורדת בשאר הכיוונים נגד כיוון השעון. לדוגמה, אם הלוח במצב כזה:

בס"ד

| | | | |
|---|---|---|---|
| 1 | 2 | 3 | 4 |
| 5 | | 6 | 10 |
| 9 | | 7 | 8 |

הפעולה הראשונה שתצא מה- open-list תהיה 6&7L, אחריה 5&9R, 6L, 5R, 2D, 7L, ולבסוף 9R.
אם הלוח במצב כזה:

| | | | |
|---|---|---|---|
| 1 | 2 | 3 | 4 |
| 5 | | 6 | |
| 9 | 10 | 7 | 8 |

הפעולה הראשונה שתצא מה- open-list תהיה 6L, אחריה 10U, 5R, 2D, 8U, 6R, ולבסוף 4D.

בנוסף יש להגיש קובץ וורד details.docx. בתחילת הקובץ יש לכתוב את פרטי המגיש (שם ות.ז.). לאחר מכן יש לתאר במילים את הפונקציה היוריסטית בה בחרתם להשתמש ולהוכיח מדוע היא admissible ו- consistent.

**דגשים**

- BFS ו- A* ימומשו עם closed list. יש להשתמש ב- hash-table גם עבור ה- open list כמו שלמדנו.
- IDA* ו- DFBnB ימומשו עם מחסנית וללא closed-list אך עם loop-avoidance, ז"א בדיקה האם הקדקוד המפותח נמצא על הענף שעליו אנחנו עובדים או כבר במחסנית. יש לממש לפי הפסאודו-קוד שניתן בכיתה.
- DFID ימומש בצורה רקורסיבית, ללא closed-list אך עם loop-avoidance.
- אם לא נמצא מסלול יש לכתוב: "no path" בשורה הראשונה של קובץ הפלט. שאר השורות ללא שינוי.
- ב- DFID האיטרציה הראשונה היא כאשר l=1, כי ברור שהמצב ההתחלתי אינו המצב הסופי.
- למרות שהמטרה שלנו היא מציאת המסלול הזול ביותר, BFS ו- DFID לא ימצאו בהכרח את המסלול הזול ביותר אלא את המסלול הקצר ביותר (=עם הכי פחות פעולות הזזה).
- יש לממש את האלגוריתמים לפי מה שלמדנו בכיתה.
- כדי לא לאבד סתם נקודות, הקפידו על פלט בדיוק לפי ההוראות: רווחים, אותיות גדולות, 4x5 ולא 4X5, וכו'.

## *אופן הניקוד*

- קוד נכון, שמממש את האלגוריתמים כמו שנלמדו בכיתה, ומחזיר את התוצאה המבוקשת על כל הקלטים החוקיים.
- איכות הפונקציה היוריסטית בה בחרתם להשתמש ב-A*, IDA* ו- DFBnB (זו כמובן אותה הפונקציה), ונכונות ההוכחה שהפונקציה היא admissible ו- consistent.
- קוד מתועד וקריא (שמות משתנים ופונקציות משמעותיים).
- הגשה בזמן.

## *פרטי ההגשה*

- ההגשה ביחידים בלבד. תתבצע בדיקת העתקות.
- ניתן לכתוב את התוכנית ב- Java בלבד, והיא צריכה להתקמפל ולרוץ בגרסת 1.8. שם המחלקה בה נמצאת פונקציית ה- main יהיה Ex1. יש להשתמש ב default-package בלבד (ללא תתי תיקיות). חובה להגיש את קבצי המקור.
- אין לממש GUI.
- עליכם להניח שקובץ ה- input.txt (שאתם מקבלים כקלט) נמצא באותה ספריה בה נמצאת התוכנית, ולכן אין לקרוא את המיקום שלו כארגומנט או לציין ספריה ספציפית בקוד שאתם מגישים (במידה וכן, ירדו על כך נקודות).
- קובץ ה- output.txt (שאתם מוציאים כפלט) צריך להיכתב באותה ספריה בה נמצאת התוכנית, ולכן אין לקרוא את המיקום שלו כארגומנט או לציין ספריה ספציפית בקוד שאתם מגישים (במידה וכן, ירדו על כך נקודות).
- יינתן קלט ופלט לדוגמה. ודאו שתוכנתכם עובדת אתו כמו שצריך, אך זהו לא הקלט היחיד אתו תיבדק התוכנית.
- התוכנית תיבדק דרך ה- command line ולא ב- eclipse. לכן, כדי לוודא שהתוכנית שלכם עובדת עליכם להעתיק את קבצי המקור ואת הקובץ input.txt שניתן כדוגמה לאחת הספריות במחשב, לפתוח command line ולהריץ javac *.java ואז java Ex1. התוכנית תיצור את הקובץ output.txt באותה ספריה והוא צריך להיות זהה לקובץ output.txt שניתן כדוגמה.
- ההגשה נעשית דרך מערכת הגשות. פרטים בהמשך. תאריך הגשת התרגיל- 16.05.19

בהצלחה!