תרגיל מס׳ 1: פתרון בעיית N-Puzzle בעזרת אלגוריתם ID

מצורפים הקבצים הבאים:

stack.py – מימוש מבנה הנתונים: מחסנית.

frontier.py – מימוש מבנה נתונים המתאים ל-ID: מחסנית ״משוכללת״ – מחסנית שמאחסנת מצבים עד עומק חיפוש d; כאשר היא מתרוקנת היא טוענת שוב את המצב ההתחלתי ומגדילה את d ב – 1.

state.py – מימוש מצב של N-Puzzle בגודל nXn.

עליכם לעבור על הקבצים המצורפים, להבין אותם ולבצע את המשימות הבאות:

1. כתבו את הקובץ search.py המממש את החיפוש, כפי שנלמד בהרצאה\*.   
   הקובץ יכיל פונקציה אחת: search(n:int) שפותרת בעיה בגודל n.  
   הפונקציה תדפיס את המצב ההתחלתי (האקראי) ותחזיר את מצב המטרה.   
   הפונקציה תבדוק האם מצב נתון הוא מצב מטרה לאחר שהוא יישלף ממבנה הנתונים.
2. שנו את מבנה הנתונים כך שישמור את מספר המצבים שאוחסנו במבנה. כתוצאה מזה, frontier יכלול 5 פריטים ולא 4:

]#stack, max. depth, init. state, try next level?, total items pushed]

1. כתוב תוכנה שיפלוט את העומק המקסימאלי ואת מספר הבדיקות (items) בממוצע מ100 ריצות. הפלט יכול להיות עבור search(2):

Average depth 1.83

Average number 6.39

1. הריצו search(2) , search(3) ו-search(4), מה מסקנתכם?

\* תזכורת מההרצאה:

search(n)

f←frontier.create(state.create(n))

while not frontier.isEmpty(f) do

s←frontier.delNext(f)

if state.isTarget(s)

then return s

ns←state.getNext(s)

for i←1 to length(ns) do

frontier.insert(f,ns[i])

return null