דו"ח מעבדה

נושא הפרויקט: IDA* מול A*

הקדמה

בפרויקט זה יצרנו השוואה בין שני אלגוריתמי חיפוש: A* ו-Iterative deepening A* (IDA*) .

אלגוריתם A* - הוא אלגוריתם למציאת מסלול אופטימלי בגרף כאשר ההיוריסטיקה אדמיסיבילית. הוא שלם, כלומר הרצת האלגוריתם מבטיחה מציאת מסלול בין צומת המקור לצומת היעד אם קיים מסלול כזה. התכונה המרכזית של אלגוריתם זה היא שהוא עוקב אחר כל צומת שביקר, מה שעוזר בהתעלמות מהצמתים שכבר ביקרו בהם, וחוסך בזמן ריצה. יש לו גם רשימה מתעדכנת שמכילה את כל הצמתים שנותרו לחקור והוא בוחר את הצומת האופטימלי ביותר מתוך רשימה זו, בכך חוסך זמן כי לא משקיע משאבים בחקירת צמתים מיותרים או פחות אופטימליים.

אלגוריתם Iterative deepening A* (IDA*) - הוא אלגוריתם חיפוש בגרף שנועד לאפשר חיפוש לרוחב בדומה לאלגוריתם A*, אך ללא דרישות הזיכרון הגבוהות שלו. כדי לפתור את הצורך בזיכרון שמצריך האלגוריתם A*, משתמשים באלגוריתם חיפוש לעומק איטרטיבי מעמיק IDA*, כשהעומק המוגדר בכל איטרציה הוא העומק הכי נמוך של בן שפותח שהיה עמוק יותר מהעומק שנבחר באיטרציה הקודמת.

מטרת הניסוי

מטרת הניסוי היא להשוות את ביצועי האלגוריתמים ולקבל אינדיקציה על ההפרשים בין זמני הריצה, סיבוכיות מקום ומספר הקודקודים המפותחים במהלך הריצה ע"י שני האלגוריתמים.
אנו משערים כי אלגוריתם A* בעל ביצועים טובים יותר מבחינת סיבוכיות מקום וזמן מאשר אלגוריתם IDA*. אנו משערים כי הרצת האלגוריתם IDA* בכל איטרציה, שמתחילה מאותה נקודת המוצא, מעכבת משמעותית את זמן הריצה.

תיאור מהלך הניסוי

מימשנו את האלגוריתמים עבור בעיית החיפוש 8-puzzle.

תיאור הבעיה:
N-Puzzle הוא פאזל מ-N אריחים (N יכול להיות 8, 15, 24 וכן הלאה) ובניסוים שלנו הגדרנו פאזל בגודל N = 8. הפאזל מורכב מ-N אריחים וחלל אחד ריק שבו ניתן להזיז את האריחים. ניתן לפתור את החידה על ידי הזזת האריחים בזה אחר זה אל החלל הריק היחיד ובכך להשיג את תצורת המטרה.

מימשנו את שני האלגוריתמים בשפת Python, לפי התיאור הבא:

- מימוש אלגוריתם A* שכולל שתי רשימות, open ו – closed. הרשימה open היא מבנה נתונים מסוג תור עדיפויות (min-heap) וכך מובטח שבכל איטרציה יבחר להיחקר State בעל ערך f הנמוך וכאשר האלגוריתם נתקל בכמה כאלה הוא יבחר את זה עם ה ערך g הנמוך יותר.
הרשימה closed היא מסוג set ששומר כל צומת שנחקר לאחר שהתגלו הצמתים השכנים שלו.

- מימוש אלגוריתם *IDA שכלל מבנה נתונים Stack.
- Puzzle Generator – אחראי ליצור start state שממנו ניתן להגיע למצב המטרה. יצירת מצב התחלתי זה, נוצרת תחילה ממצב המטרה ומבצעת הרצה של 1000 איטרציות, כך שבכל איטרציה מתבצעת הזזה של האריח הריק לכיוון רנדומלי.

בהתחלה הרצנו את האלגוריתמים עם פונקציה היוריסטית של חישוב מספר האריחים שלא במקומם, אך זמני הריצה עבור שני האלגוריתמים היו גבוהים ובעיקר עבור *IDA שהתקשה להגיע לפתרון בזמן סביר. לבסוף, עבור כל אחד משני האלגוריתמים קבענו את הפונקציה ההיוריסטית להיות: סכום מרחקי מנהטן של כל האריחים עבור state הנוכחי בהשוואה ל- goal state.

ביצענו 50 הרצות שהתחילו ממצב זהה עבור שני האלגוריתמים. בכל סיום הרצת start state על שני האלגוריתמים מתבצעת כתיבה של התוצאות לקובץ results.csv.

תוצאות הניסוי

תוצאות הניסוי מפורטות בטבלת תוצאות קובץ results.csv.

בתוצאות הניסוי החלטנו למדוד עבור כל הרצה את הפרמטרים הבאים:

1. המצב ההתחלתי ממנו התחיל האלגוריתם את הריצה
2. מספר האריחים שלא במקומם במצב ההתחלתי
3. סכום מרחקים אוקלידים בין כל אריח במצב ההתחלתי לבין מיקומו במצב המטרה שהוגדר להיות:

1	2	3
4	5	6
7	8	0

(כאשר 0 מסמל תא ריק ללא אריח)

4. אורך המסלול של הפתרון (מספר הזזות של אריחים כדי להשיג את סידור האריחים במצב המטרה)
5. זמן ריצה עבור אלגוריתם *A
6. מספר הקודקודים שאלגוריתם *A פיתח
7. זמן ריצה עבור אלגוריתם *IDA
8. מספר הקודקודים שאלגוריתם *IDA פיתח

בכל 50 ההרצות שבוצעו במהלך הניסוי, זמן הריצה של אלגוריתם *IDA היה גדול משל *A ובכל הרצה מספר הקודקודים של *IDA היה גדול שווה למספר הקודקודים שפיתח אלגוריתם *A.

ממוצעים של התוצאות שהתקבלו מ50 ההרצות:

A* time (sec)	A* nodes	IDA* time (sec)	IDA* nodes
0.209799495	821.3	31.03584949	251278.98

מסקנות הניסוי

ראינו כי אלגוריתם *A אכן בעל ביצועים טובים יותר מבחינת סיבוכיות מקום וזמן מאשר אלגוריתם *IDA. ריצת אלגוריתם *IDA נמשכת זמן רב משמעותית יותר מ*A ובריצת *IDA האלגוריתם מפתח הרבה יותר קודקודים.

בנוסף ראינו כי לא ניתן לחזות את הקושי של בעיית החיפוש ב n-puzzle על בסיס כמות הקודקודים שבמקום שגוי במצב ההתחלתי לעומת מצב המטרה או על בסיס חישוב מרחק מנהטן בין המצב ההתחלתי למצב המטרה. מה שיכול לתת אינדיקציה לקושי הבעיה הוא רק אורך המסלול של הפתרון, אך אותו אנחנו מגלים רק לאחר הרצת אחד האלגוריתמים לפתרון.