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 שבו ביקר וכך חוסך בזמן ריצה כי נמנע מלחזור ולחקור צמתים יותר מפעם אחת. בכל איטרציה האלגוריתם בוחר את הצומת האופטימלי ביותר מתוך רשימה שצבר ובכך חוסך עוד בזמן ריצה כי אינו משקיע משאבים בחקירת צמתים מיותרים או פחות אופטימליים. כדי לאפשר את הפעולות האלו שמטרתן לחסוך בזמני ריצה, לאלגוריתם דרישות זיכרון גבוהות.

אלגוריתם Iterative deepening A* (IDA*) - הוא אלגוריתם חיפוש בגרף שנועד לאפשר חיפוש לרוחב בדומה לאלגוריתם A*, אך ללא דרישות הזיכרון הגבוהות שלו על ידי שימוש חוזר באלגוריתם חיפוש לעומק שגדל בכל איטרציה עד למציאת המטרה. הוא פותר את הצורך בזיכרון שמצריך האלגוריתם A*. העומק המוגדר בכל איטרציה הוא העומק הכי נמוך של בן של צומת שפותח באיטרציה הקודמת והיה עמוק יותר מהעומק שנבחר באותה איטרציה.

מטרת הניסוי

מטרת הניסוי היא להשוואת את ביצועי האלגוריתמים ולקבל אינדיקציה על ההפרשים בין זמני הריצה, סיבוכיות מקום ומספר הקודקודים שהאלגוריתם מפתח.
אנו משערים כי אלגוריתם A* בעל ביצועים טובים יותר מבחינת זמן ריצה ושיפתח פחות מצבים מאשר מספר המצבים שיפותחו ע"י אלגוריתם IDA*. אנו משערים כי הרצת האלגוריתם IDA* בכל איטרציה, שמתחילה מאותה נקודת המוצא, מאריכה משמעותית את זמן הריצה עד למציאת הפיתרון.

תיאור מהלך הניסוי

מימשנו את האלגוריתמים עבור בעיית החיפוש N-puzzle כאשר בחרנו ב N=8.

תיאור הבעיה:
N-Puzzle הוא פאזל המורכב מ-N אריחים (N יכול להיות 8, 15, 24 וכן הלאה) וחלל אחד ריק שמאפשר בכל מהלך להזיז אריח בודד. ניתן לפתור את החידה על ידי הזזת האריחים בזה אחר זה אל החלל הריק היחיד ובכך להשיג את תצורת המטרה, הנראת באופן הבא:

(כאשר 0 מסמל את התא ריק ללא אריח)

1	2	3
4	5	6
7	8	0

מימשנו את שני האלגוריתמים בקוד 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 אופטימלי, בעל ערך f מינימלי וכאשר האלגוריתם נתקל בכמה כאלה הוא יבחר את הState עם הערך g הנמוך יותר.
הרשימה closed היא מסוג set ששומר כל צומת שנחקר לאחר שהתגלו הצמתים השכנים שלו, כדי לחסוך במקום שמרנו עבור כל צומת את המזהה שלה לפי פונקצית hash שהופעלה על הפאזל שלה.

- מימוש אלגוריתם *IDA שכלל מבנה נתונים Stack שאכסן רשימה מתעדכנת של מצבים לפיתוח (חקירת השכנים).

- Puzzle Generator – זוהי מחלקה שאחראית ליצור start state, שממנו ניתן להגיע למצב המטרה. יצירת state זה, נוצרת תחילה ממצב המטרה ומבצעת הרצה של 1000 איטרציות, כך שבכל איטרציה מתבצעת הזזה של האריח הריק לכיוון רנדומלי חוקי (כלומר, בגבולות הלוח).

בהתחלה הרצנו את האלגוריתמים עם פונקציה היוריסטית של סכום של 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 התחלתי זהה עבור שני האלגוריתמים. ובכל סיום ריצה ביצענו כתיבה של התוצאות אל קובץ results.csv המצורף לקבצי ההגשה.

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
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 שפותחו ע"י הרצת אלגוריתם *A
7. זמן ריצה עבור אלגוריתם *IDA
8. מספר הקודקודים שפותחו ע"י הרצת אלגוריתם *IDA

בכל 50 ההרצות שבוצעו במהלך הניסוי, זמן הריצה של אלגוריתם *IDA היה גדול משמעותית מהרצת האלגוריתם *A ובכל הרצה מספר הקודקודים ש *IDA פיתח, היה גדול/שווה למספר הקודקודים שפיתח אלגוריתם *A.

<u>להלן ממוצעים של התוצאות שהתקבלו מ50 ההרצות:</u>

#Wrong indexes	**Manhattan distance**	**Path length**	**A* time** (sec)	**A* nodes**	**IDA* time** (sec)	**IDA* nodes**
7.22	14.04	22.04	0.209799495	821.3	31.03584949	251278.98

מסקנות הניסוי

ראינו כי ההבדלים בביצועים של שני האלגוריתמים גדולים מאוד ומורגשים. ריצת *IDA נמשכת זמן רב משמעותית יותר מהרצתו של *A ובהרצת *IDA נעשה פיתוח למספר גדול בהרבה של קודקודים.
בנוסף ראינו כי לא ניתן לדעת את רמת 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 אורך המסלול של הפתרון, אך אותו אנחנו מגלים רק לאחר הרצת אחד האלגוריתמים לפתרון.