דו"ח מעבדה מס' 1

תאריך הגשה :

10.04.2020

מגישים:

דניאל שלם – 205745052

אלבסיי אלייב - 322162330

מידע כללי:

שפת תכנות: python .

אינטרפרטר: השתמשנו בanaconda 3.7 אבל אמור לעבוד עם כל גרסה של פייטון.

הכנסת הקלט להגבלת הזמן לפתרון תינתן בשניות.

טבלת הסטטיסטיקות המלאה של הפתרונות השונים נמצא ב**statistics.csv**.

```
heuristics = [1, 2, 3, 4]
```

על מנת לבחור היוריסטיקה יחידה להרצה ניתן לשנות את השורה בהתאם להיורסטיקה שרוצים להציג.

ארכיטקטורת פתרון:

השתמשנו **ב*A** על מנת לפתור את הלוח.

את **הopen list** מימשנו תחילה באמצעות ערמת פיבונאצ'י בשביל לייעל חלק מהפעולות (הוצאת מינימום) + hash בשביל בדיקה יעילה של האם צומת נמצא **בopen list אך** לאחר מס' הרצות גילינו כי ערימה בינארית אשר מיושמת בpython מהירה יותר ולכן בחרנו בה.

את ה**closed list** מימשו באמצעות **hash .**

השתמשנו ב**Multi-threading** כדי ליצור **חיפוש מקבילי** בצורה הבאה:

ללוח ההתחלתי שיצרנו מהpuzzle הנתון, יצרנו מהלכים מהתזוזות האפשריות של מכוניות אנכיות בנפרד ואופקיות בנפרד. כל Thread חיפש פתרון מהבנים האלו והלאה. ברגע שחיפוש אחד הסתיים וקיבלנו תוצאה נכונה הפסקנו את החיפושים. את הנתונים לסטטיסטיקות לקחנו מ2 החיפוש (משתנים גלובליים).

לחיפוש Un-informed השתמשנו בA* עם היוריסטיקה = 0, שזה בעצם כמו לבצע חיפוש בעזרת דייקסטרה.

היוריסטיקות:

Advanced blocking heuristic: אדמיסבילית. השימוש בה הוא חיבור של 2 היוריסטיקות, מס' הרכבים שחוסמים את המכונית האדומה + מס' הרכבים הללו שחסומים בעצמם (מ2 הכיוונים).

Advanced double blocking heuristic: לא אדמיסבילית. סופרים את מס' הרכבים שחוסמים את המכונית האדומה, לכל רכב כזה ניתן ערך של 2 על מנת לתת משקל גדול יותר להוצאה של מכוניות חוסמות + מס' הרכבים הללו שחסומים בעצמם גם אם רק מכיוון אחד.

Vertical from right heuristic: לא אדמיסבילית. מס' המכוניות האנכיות שנמצאות מצד ימין למכונית האדומה. היוריסטיקה זו בעצם מסמלת את העומס על הצד הימני מכיוון שלא ניתן להזיז מכונית כזו לצד השמאלי של הלוח, ולכן בשביל להקטין אותה נצטרך להתקדם עם המכונית האדומה.

את N יצרנו כמס' הצמתים שבדקנו בכללי ולא את מס' הצמתים שבחרנו, ולכן המס' עשוי להיות גדול.

סטטיסטיקות ממוצעים:

For Advanced blocking heuristic

Number of solved puzzles: 40

Average number of expended nodes: 4107.871794871795

Average rate of penetrability: 0.015672411920546685

Average EBF value: 1.5362222034443311

Average heuristic value: 4.048917665888676

Average max depth: 26.205128205128204

Average min depth: 1.0256410256410255

Average of depth Average: 15.888964260227327

Average solving time: 5.379974294320131

For Advanced double blocking heuristic

Number of solved puzzles: 40

Average number of expended nodes: 4302.7692307692305

Average rate of penetrability: 0.016419829336968672

Average EBF value: 1.4855887851610416

Average heuristic value: 9.07352529246231

Average max depth: 27.846153846153847

Average min depth: 1.0512820512820513

Average of depth Average: 16.009125727724488

Average solving time: 5.241530326696543

For Vertical from right heuristic

Number of solved puzzles: 40

Average number of expended nodes: 3854.641025641026

Average rate of penetrability: 0.0154693186877067

Average EBF value: 1.5319767645373255

Average heuristic value: 4.084042965927313

Average max depth: 25.923076923076923

Average min depth: 1.0512820512820513

Average of depth Average: 16.192256300497334

Average solving time: 5.86490875024062

For Uninformed search heuristic

Number of solved puzzles: 40

Average number of expended nodes: 4329.7692307692305

Average rate of penetrability: 0.013129726391057498

Average EBF value: 1.6183019781818713

Average heuristic value: 0.0

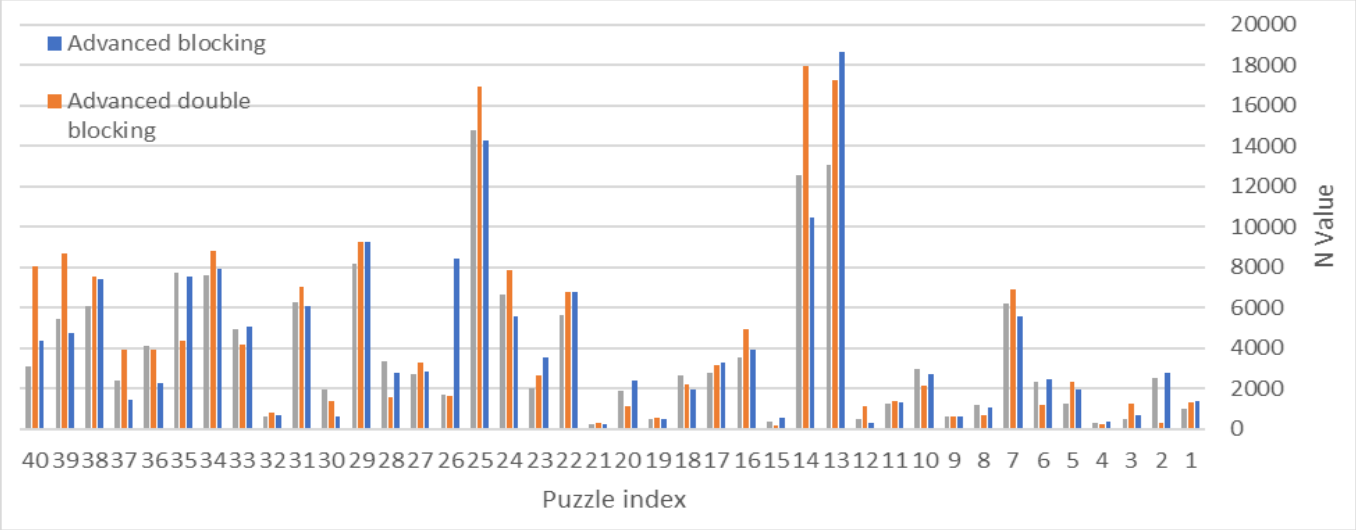
Average max depth: 25.256410256410255

Average min depth: 1.0256410256410255

Average of depth Average: 16.506185000349397

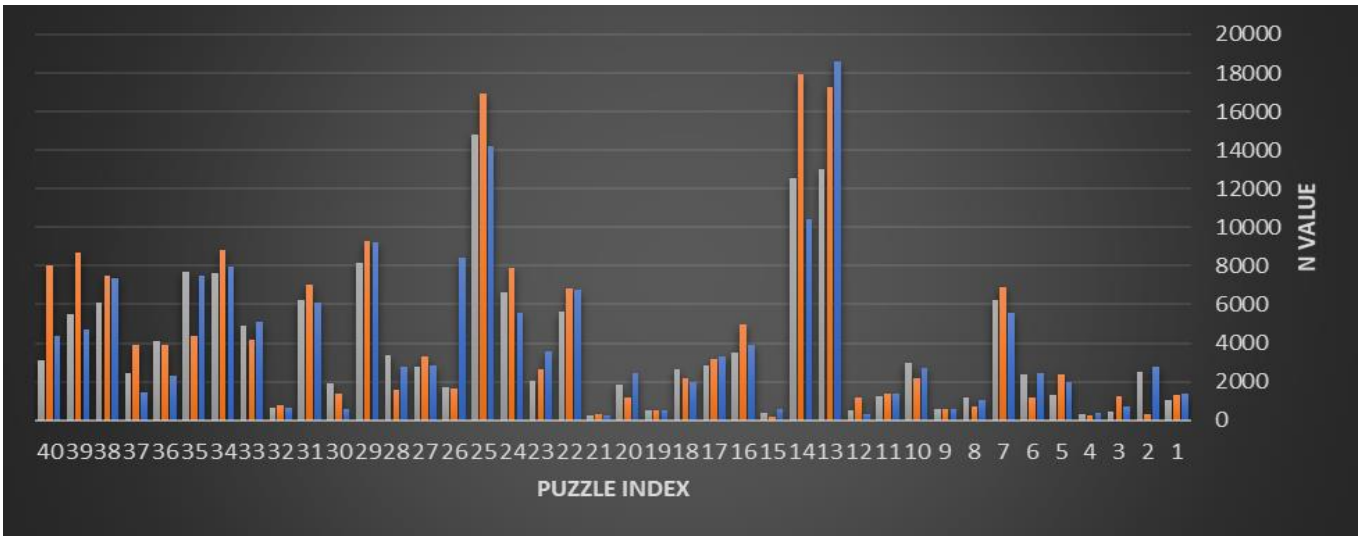
Average solving time: 6.2804020429268865

השוואה בין היוריסטיקות מבחינת זמנים: (הצבע האפור הוא לVertical from right)



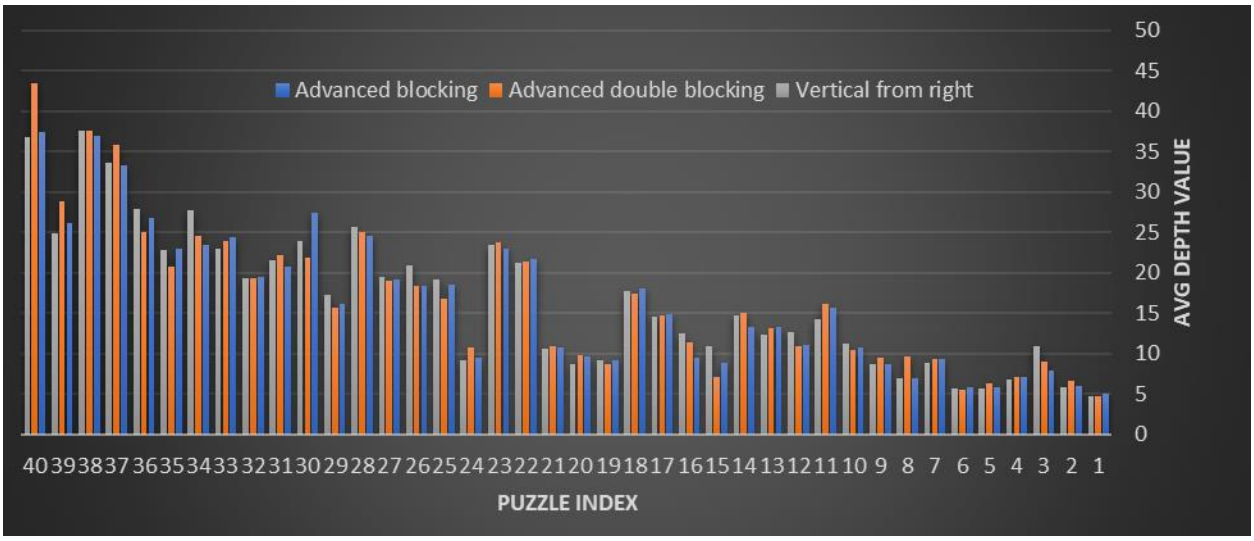
לפי הגרף, ניתן לראות שברוב המקרים הפתרון המהיר ביותר יהיה של ההיוריסטיקה הלא אדמיסבילית Advanced double blocking, אך קיימים פאזלים בהם היא גבוהה משמעותית משאר ההיוריסטיקות, מה שמביע על חוסר איזון אצל פונקציות לא אדמיסביליות. מבחינת ממוצעים, Advanced double blocking המהירה ביותר עם ממוצע של 5.24153032669654, אך בהרבה בדיקות שעשינו Advanced blocking יוצאת מהירה יותר ויורדת מה5 שניות בממוצע לפתרון.

השוואה בין היוריסטיקות מבחינת מס' הצמתים שנבדקו: (אפור - Vertical from right, כחול – Advanced blocking, כתום –Advanced double blocking)



ניתן לראות כי קיימת התאמה כמעט מלאה בין מס' הצמתים שנבדקו (N), לבין הזמן לפתרון. מבחינת הממוצעים, Vertical from right היא בעלת המס' הנמוך ביותר של צמתים שנבדקו, וזה הגיוני מכיוון שהיא פונקציה **לא אדמיסבילית** ולכן מטבעה עשויה לבדוק פחות צמתים (פגיעה בשלמות הפתרון).

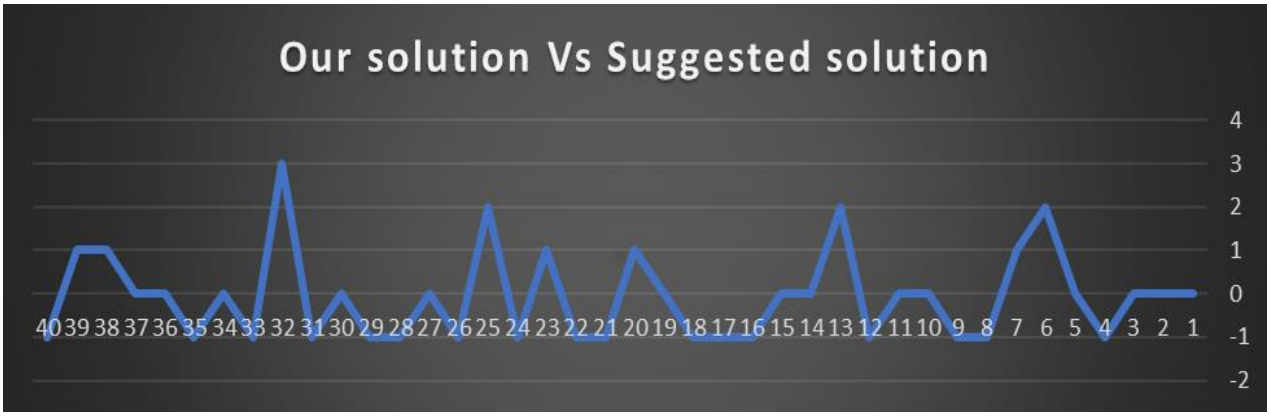
השוואה בין היוריסטיקות מבחינת עומק ממוצע לחיפוש:



כאן ניתן לראות כצפוי, שככל שרמת הקושי עולה עומקי החיפוש עולים בהתאמה.

מבחינת הממוצעים, היוריסטיקה האדמיסבילית חיפשה בממוצע בעומקים נמוכים יותר אך לא באופן משמעותי.

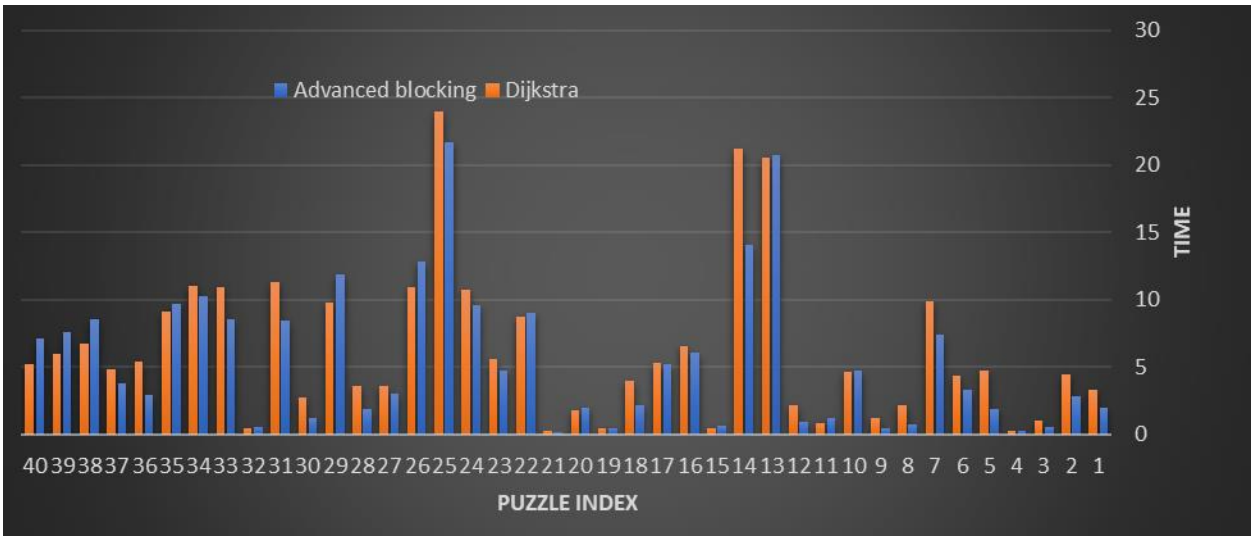
השוואה בין מס' המהלכים הנדרשים לפי הקובץ שקיבלנו לעומת עומק הפתרון שמצאנו:



קבענו את מס' המהלכים הנדרשים מכל תשובה בקובץ כמס' התזוזות שצריך לבצע והשוונו מול עומק הפתרון שמצאנו. כאשר הערך שלילי זה אומר שהתשובה בקובץ בעומק נמוך יותר, אם הערך שלילי זה אומר שהפתרון שמצאנו הוא בעומק נמוך יותר.

לפי הגרף ניתן לראות כי המרחק הגדול ביותר בין הפתרונות הוא בעומק 3 (3 מהלכים) וגם יש אחד כזה, ברוב המקרים המרחק הוא של מהלך אחד. כמעט בחצי מהפאזלים הפתרון שלנו טוב יותר (16 מהם) , מה שמעיד על אופטימליות בחיפוש.

השוואה בין הפתרון הנבחר לעומת חיפוש Uninformed (דייקסטרה):



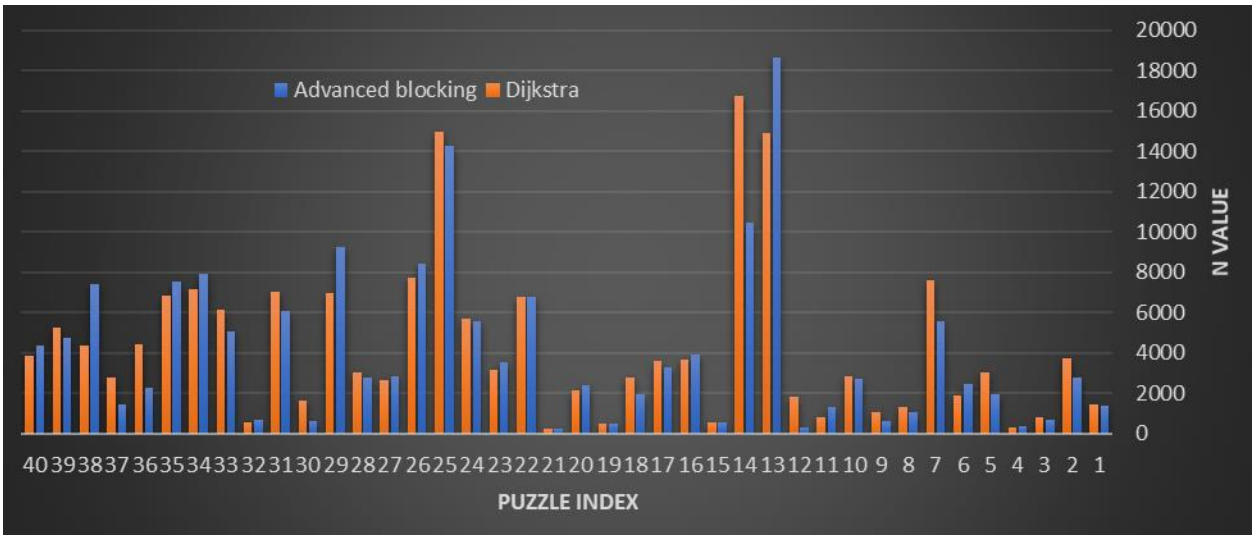
מימשנו את אלגוריתם דייקסטרה כ*A עם היורסטיקה = 0. ניתן לראות כי בחלק מהמקרים הפתרון שלנו מפסיד מבחינת זמן לדייקסטרה, אך ברוב המקרים האלגוריתם שלנו יעיל יותר. מבחינת ממוצעי זמן:

ממוצע הזמן של הפתרון שלנו: (ברוב המקרים שבדקנו ירד מ5 שניות)

5.379974294320131

ממוצע הזמן של דייקסטרה:

6.2804020429268865



מבחינת מס' הצמתים שנבדקו, שוב, ברוב המקרים האלגוריתם שלנו חיפש במס' קטן יותר משמעותית של צמתים. מבחינת ממוצעים, ניתן לראות שבאלגוריתם שלנו הממוצע נמוך ביותר מ200 צמתים, מס' משמעותית נמוך יותר ומביע על החשיבות של היוריסטיקות טובות בחיפוש.

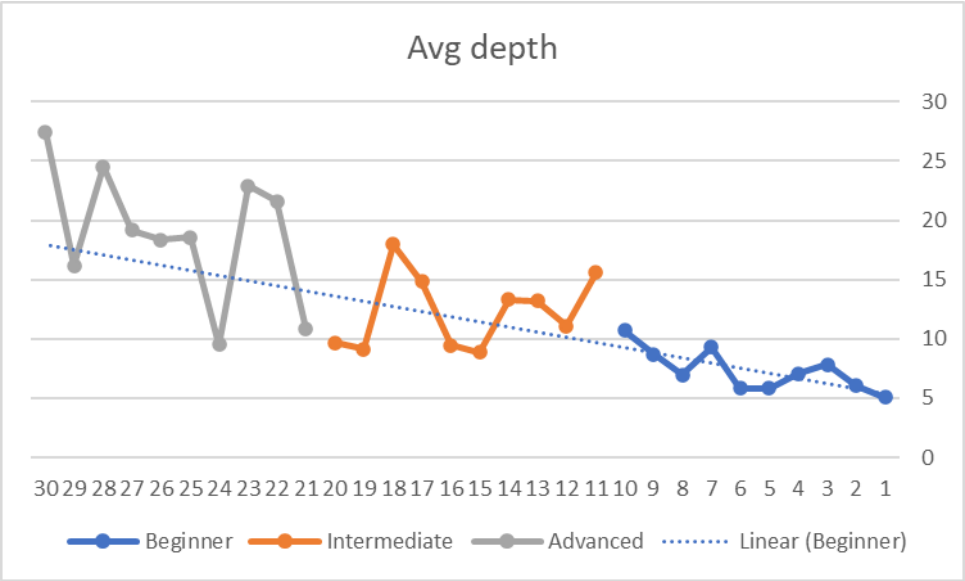
ממוצע הצמתים שנבדקו של הפתרון שלנו:

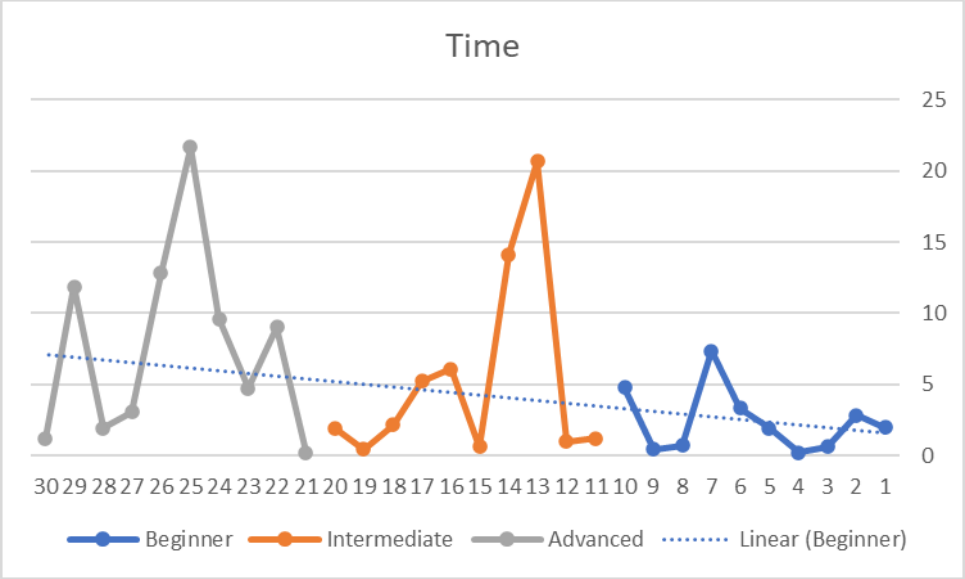
4107.871794871795

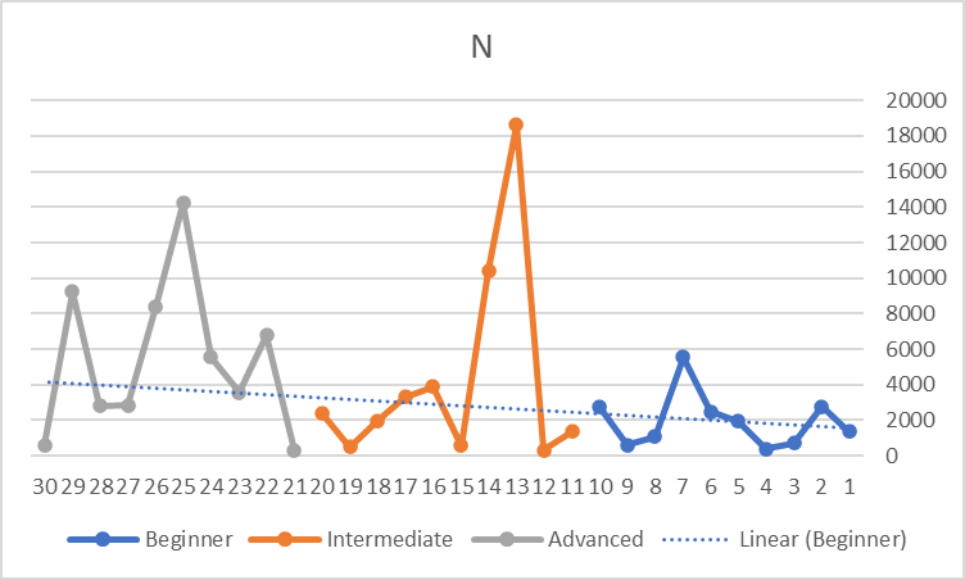
ממוצע הצמתים שנבדקו של דייקסטרה:

4329.7692307692305

השוואה בין דרגות הקושי השונות:







מהגרפים המוצגים למעלה ניתן ללמוד כי ככל שרשמת הקושי גבוהה יותר החיפוש ארוך יותר ועמוק יותר. בגרף המציג את **העומק הממוצע של החיפוש** (Avg depth) ניתן לראות כי השינוי הוא המשמעותי ביותר, העלייה היא חדה והחיתוך בין הקבוצות כמעט ולא קיים.

מבחינת **זמנים**, ניתן לראות כי קיים סף שאותו גם החידות המתקדמות לא עוברות, אך מס' החידות שמתקרבות אליו גבוהה יותר בAdvanced מאשר בשאר רמות הקושי.

בגרף המציג את מס' הצמתים שנבדקו, ניתן לראות כי למעט חידה 14, מס' הצמתים כמעט זהה בחידות Beginner וIntermediate אבל Advanced המספרים עולים באופן משמעותי.

מחולל בעיות:

פיתחנו 2 מחוללי בעיות:

PuzzleGenerator:

עובד הכי טוב. עובד בצורה הבא: לקחנו את ה40 פזאלים שניתנו, פתרנו כל אחד מהם וקיבלנו את העומק של הפתרון כאשר משתמשים ביוריסטיקה מסוג advanced blocking. כל זה precompile ואז הארדקודד ולא בזמן ריצה. כל זה כדי שהמכולל יעבוד בצורה מהירה ככל שניתן.

סיננו את כל הפאזלים שעומק הפתרון שלהם נמוך יותר מהעומק המבוקש ויצרנו רשימה חדשה הכוללת רק פאזלים שלהם עומק פתרון גדול יותר מהקלט.

לאחר מכן בחרנו אקראית פאזל מתוך הקבוצה ובעזרת A* פתרנו אותו וקיבלנו את המסלול (רשימה ממוינת) של כל הצמתים מהשורש עד העלה שהוא הפתרון. אחרי זה החזרנו את הצומת שאורך המסלול ממנו עד העלה הוא כעומק הקלט. (עומק הפתרון הוא מספר הצעדים שנעשים עד שלמכונית האדומה יש קו נקי ליציאה)

ProblemGenerator :

עובד בצורה הבאה: לוקח לוח ריק ומייצר פאזל לפי השלבים הבאים:

<u>שלב אחד:</u> מקם מכונית אדומה באופן אקראי על המסילה שלה לפחות במרחק אחד מהיציאה, ותכניס את הקצה לרשימת נקודות חסימה עתידיות.
<u>שלב שתיים:</u> מקם מכוניות כל עוד עומק הפתרון (נבדק עם) קטן יותר מהקלט לפי האופן הבא: בהסתברות של 2/3 בחר נקודת חסימה אקראית מתוך רשימת נקודות החסימה ואז בחר אקראית ובאופן אחיד כיוון וסוג מכונית, מקם את המכונית (רק אם אפשר למקם והפאזל עדיין פתיר), תוסיף את קצוותיה לרשימת נקודות החסימה ומחק את נקודת החסימה שממנה באנו. בהסתברות של 1/3 בחר נקודה אקראית על הלוח ואז בחר באופן אחיד כיוון וסוג מכונית, מקם אותה ותוסיף את הקצוות שלה לנקודות החסימה.