**מבוא לבינה מלאכותית- תרגיל 1**

**מגישות: גל קסטן ודנית כהן**

**חלק א:**

סידור ביקורים בדירות

בחירת מעבדה בה יבקר אחרי הדירה האחרונה

פריסת ביקורים במעבדות לפני ביקור בדירה (m מעבדות אפשריות + האופציה לא לבקר)

3.

.

|  |  |  |  |
| --- | --- | --- | --- |
| estimated calc time | #possiblePaths | M | K |
| [secs] 18.47787201 | 2.204\* 10^7 | 2 | 7 |
| 3.845214844 [mins] | 2.477 \*10^8 | 3 | 7 |
| 2.255859375 [hours] | 7.927 \*10^9 | 3 | 8 |
| 19.55777407 [hours] | 6.300 \* 10^10 | 4 | 8 |
| 3.69140625 [days] | 2.854 \* 10^11 | 3 | 9 |
| 5.33203125 [months] | 1.142 \* 10^13 | 3 | 10 |
| 21.0546875 [ years] | 5.023 \* 10^14 | 3 | 11 |
| 1.0828125 [thousand years] | 2.411 \* 10^16 | 3 | 12 |
| 22.40994945 [thousand years] | 4.678 \*10^17 | 4 | 12 |
| 1.547687134 [million years] | 3.041 \* 10^19 | 4 | 13 |

חלק ג:

4.

**מקרה מקסימלי** :m+k

עבור מצב התחלתי v0 בו canVisit(s,)=True לכל i מ 1-k canVisit(s,)=True לכל i מ1-m

**מקרה מינימלי**:0

עבור מצב מטרה מתקיים כי ולכן canVisit(s,)=False ו  *לכן* canVisit(s,)=False*.*

*5. לא ייתן כי קיים מעגל בגרף. נניח בשלילה שקיים מעגל- לא ייתכן כי אחד הצמתים במעגל יהיה דירה כי* s.Taken∪s.Transferred *יגדל ב1 בכל הצמתים שבאים אחרי צומת זה מהגדרת האופרטור (ולכן לעולם לא נוכל לחזור למצב ראשוני). לכן המסלול המעגלי יכול לכלול רק מעבדות- אבל מכיוון שכל צומת s במסלול היא מעבדה, מהגדרת צומת מעבדה* s.Taken=∅ . *ומהגדרת אופרטור , ותחום ההגדרה של אופרטור זה , חייב להיות מעבר בין 2 מעבדות שונות במסלול דבר שיוביל להגדלת הקבוצה visited labs בכל המשך המסלול, בסתירה לכך שצריך לחזור למצב בו קבוצת visited labs הייתה קטנה יותר.*

*6. צריך לבדוק!*

*7. כן ייתכנו בורות.*

*מרחב חיפוש של דירה אחת ומעבדה אחת כאשר מס' מטושים התחלתי+ מס' המטושים במעבדה קטן ממס' הדיירים בדירה נתחיל ממצב התחלתי ונעבור למעבדה והיא תהיה מצב בור ע"פ חוקי האופרטורים.*

*8. טווח האורכים האפשרי הוא בין k+1 ל2k+m*

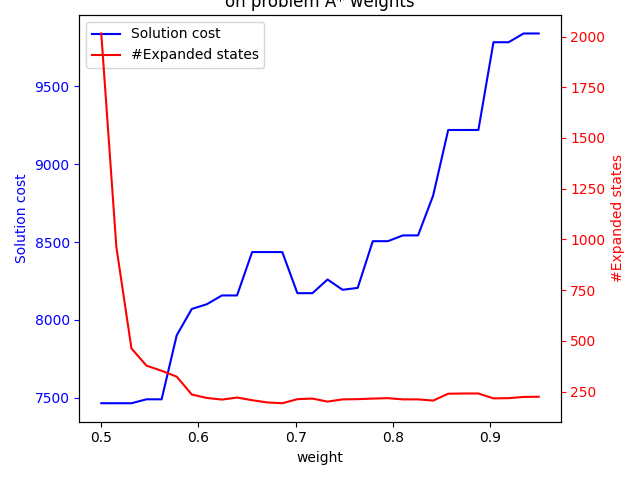
*הסבר-* ***מקרה מינימלי****: k+1. מסלול חוקי יכלול לפחות את כל הדירות ומעבדה בסיום המסלול. לכן אם* AmbulanceTestsCapacity , *מספיק כדי להכיל את כל הבדיקות מכל הדירות ו-* InitialMatoshimAmb *מספיק גדול כדי לבדוק את כל הדיירים בכל הדירות, נוכל לבנות מסלול שעובר בכל הדירות (k קשתות) ומסתיים במעבדה( קשת נוספת).*

***מקרה מקסימלי****: 2k+m. במקרה זה נעבור ראשית בכל המעבדות כדי לאסוף את כל המטושים (m קשתות) ולאחר מכן נעבור בין כל הדירות, כאשר בין כל 2 דירות עוברים במעבדה על מנת לפרוק את הבדיקות ( סה"כ k קשתות לדירות ומכל דירה קשת למעבדה ולכן k קשתות נוספות). מקרה זה הוא המקסימלי כיוון שלכל היותר נוכל לפרוק את הבדיקות k פעמים (לאחר ביקור כל דירה) ונרצה לפצל את פריקת הבדיקות מלקיחת מטושים חדשים על מנת למקסם ביקורים במעבדה.*

*9.*

*דו"ח ריצה:*

*14. סה"כ בריצה עיוורת פיתחנו 17354 צמתים ובריצה עם הירוסטיקה פיתחנו 2015 צמתים לכן יחס החיסכון הוא 0.88.*

*15.*

*האזורים שיותר כדאים בגרף הם משקלים בין 0.55-0.6 כיוון שהמחיר שלהם קרוב למחיר האופטימלי (במשקל 0.6) אבל עם זא מס' הצמתים שפותחו יחסית נמוך (מאות צמתים).*

*הכלל שהוזכר בא לידי ביטוי בכך שניתן לראות שבאופן כללי ככל שהמשקל נמוך יותר כך מחיר הפתרון נמוך יותר (באופן כללי יש מגמת ירידה אם בגרף הכחול נסתכל מימין לשמאל) וכי מס' הצמתים שפותחו גבוה יותר . עם זאת, הדגש בא לידי ביטוי בכך שגרף המחירים אינו עולה או יורד מבחינת הגדרה . עבור המשקל w=0.648 וw=0.7 למשל ניתן לראות כי המחיר של הפתרון עבור w=0.7 נמוך יותר מw=0.648 למרות שהמשקל הראשון גדול מהשני (כלומר הדוגמה הספציפית הזו נוגדת את הכלל).*

*19. החיסרון בגישה זו מבחינת יעילות הפתרון היא חוסר יכולת להשתמש בcache הפנימי השומר את המרחק בין 2 צמתים ע"י פתרון של תת בעיה במפה. זאת מכיוון, שכעת לא ניתן פשוט לשמור את המרחק בין 2 דירות למשל שמצאנו בהרצה קודמת, אלא יש ממש למצוא מסלול במרחב העל של המצבים שכולל את כל הנקודות במפה בכל הרצה (גם אם נשתמש בזיכרון המהיר עדיין נצטרך לשמור שם את המסלול כולו וזו עלות גדולה מבחינת זיכרון כך שלא נשמור הרבה מסלולים)*

*20.*

1. *@dataclass(frozen=True)*
2. *שורה זה לא מספיקה כיוון שהיא מבטיחה שלא ניתן יהיה לשנות את המשתנים שהוגדרו בתוך הdataclass אבל עבור טיפוסים מורכבים למשל כמו סטים או רשימות היא יכולה רק להבטיח שהמשתנה יצביע על אותו מקום בזיכרון(על אותה רשימה או על אותו סט) , אך עדיין יהיה ניתן לבצע שינויים בטיפוסים המורכבים. לכן, גם תכונות של אובייקט שהן מטיפוס מורכב בתוך מחלקה זו הוגדרו כfrozen כלומר כטיפוסים שלא ניתן לשנותם.*
3. *כן ייתכן שנפגוש בצומת שכבר פיתחנו כלומר צומת בclose. להלן השורה מהאלגוריתם שנלמד בהרצאה שמוכיחה זאת :*

*old\_node ← find-state(s,CLOSED)*

*if old\_node: ; A node with state s exists in CLOSED*

*20. באג: יצירת צומת עוקב חדש ע"י שימוש בחלק מהשדות של הצומת הנוכחית (state to expand ), כאשר שדות אלו ניתנים לשינוי. נניח שניצור….,state\_to\_expand.visited\_labs…) succ=MDAState( עבור צומת עוקב של מעבדה, מתוך מחשבה שהסט מועתק. ולאחר מכן נבצע succ.visited\_labs.add(new\_lab) . בגלל שsucc והצומת הנוכחית מצביעים לאותו סט אז שינינו גם את הסט של המעבדות שביקרנו בהם אצל state to expand . הצומת הזה ייכנס לאחר מכן לclose ובעתיד יכול להיות שנפתח אותו שוב. הבעיה היא שכעת הוא למעשה צומת שונה מהמקורית – כי הvisited\_labs שלו השתנה וזה יגרום לבאג.*

.