הטכניון – מכון טכנולוגי לישראל

Introduction to Artificial Intelligence

תרגיל בית 1

מגישים :

שי יחזקאל - 205917883

דורין רואינסקי – 315691410

14.05.22

תוכן עניינים

שאלה 1 - מבוא 2
שאלה 2 4
שאלה3 5
שאלה 4 6
שאלה5 8
שאלה 6 11
שאלת סגנון מבחן 12

שאלה 1 - מבוא

1.1 S יהיה קבוצת כל המצבים על גבי הלוח, סך כל המצבים האפשריים:
$destination \cdot PassngersPlace \cdot TaxisPlace = 4 \cdot (1+4) \cdot 25 = 500$
הוספת 1 עבור המיקומים של הנוסע נועדה להצגת המצב בו הנוסע על גבי המונית.

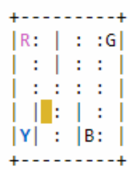
כלומר מצב $s \in S$ מקיים: $s = (D, P, T)$ כאשר D היעדים האפשריים4, P מיקום הנוסע (4 יעדים + 1 על המונית), T מיקום המונית מתוך 25 המשבצות:

$$P, D \in \{(0,0), (4,0), (0,4), (4,3)\}, T \in \{0,1,2,3,4\}^{[5]\times[5]}$$

O יהיה כל האופרטורים האפשריים: $O = \{South,\ North,\ East,\ West,\ Pickup,\ Dropoff\}$

I מצב התחלתי: $I = 328$, כאשר 328 הוא מספר מצב מקודד המתאים ללוח המבוקש בבעיה

:$I = (D = (0,0), P = (4,0), T = (3,1))$



G היא קבוצת המצבים הסופיים $G = \{s \in S \mid s.done = true\}$ כלומר כל המצבים בהם הנוסע נמצא על המונית, המונית נמצאת על משבצת היעד ומבצעת פעולת *Dropoff*.

1.2 הפונקציה תחזיר את כל המצבים מלבד המקרים בהם המונית ממוקמת בשורה העליונה במרחב המצב שבו יש קיר בחלק העליון.
בפועל נציין כי הסביבה **אינה** חוסמת פעולה צפונה גם במקרה שהמונית ממוקמת בשורה העליונה, ובמקרה זה היא מחזירה *Reward=-1* ומותירה את הסביבה באותו מצב. אם נתייחס למקרה שהסביבה מתירה את הפעולה צפונה גם בשורה העליונה, הרי ש$Domain(\{north\}) = S$.

1.3 כל המצבים העוקבים למצב ההתחלתי $Succ(328) = \{s \in S \mid \exists o \in O, [s_1 \in Domain(o) \wedge o(s_1) = s_2]\}$
$o \in \{South, North, East,\} \rightarrow Succ_1(328) = \{428, 228, 348\}$
$o \in \{West, Pickup, Dropoff\} \rightarrow Succ_2(328) = \{328\}$
הפעולות הנ"ל שמובילות אל $Succ_2$ הן לא חוקיות ומובילות את הסוכן לאותו מצב.
וסה"כ:

$$Succ(328) = Succ_1 \cup Succ_2 = \{428, 228, 348\}$$

1.4 נבחין כי במימוש המוצע במחברת, הסוכן מחזיק רשימת *Visited*, ולכן לא יבקר במצבים בהם היה בעבר. בנוסף, בחנו מקרים (ווידאנו בתיעוד של הסביבה) כי ניתן להוריד את הנוסע באחת משלוש היעדים **שאינם** היעד שנבחר. מכאן שמספר הצעדים הגרוע ביותר שיקח לסוכן להוריד את הנוסע ביעד **הנכון** הוא:

$$\underbrace{25 \times 3}_{3\ wrong\ destination\ for\ passenger} + \underbrace{25}_{states\ traveling\ while\ passenger\ onboard} - 1 = 99$$

נציין כי הסוכן יבקר ב100 מצבים סה"כ, אך מכיוון שנשאלנו על כמות פעולות – התשובה היא 99.

בהנחה ונספור כל פעולה כביצוע:

```
action = env.action_space.sample()
state, reward, done, info = env.step(action)
```

אז ייתכנו אינסוף פעולות של הסוכן הרנדומלי שכן הוא יכול במקרה הגרוע בכל פעם להגריל את אותה פעולה שמובילו אותו למצב שכבר ביקרנו בו, וכך להיתקע בלולאה אינסופית שבודקת האם ביקרנו כבר במצב.

1.5 מספר הפעולות הוא (המסלול מפורט בסעיף הבא)

$$\underbrace{4}_{states\ traveling\ while\ passenger\ waiting} + \underbrace{1}_{Pickup} + \underbrace{4}_{states\ traveling\ while\ passenger\ onboard} + \underbrace{1}_{Dropoff} = 10$$

1.6 המסלול הטוב ביותר מתקבל ע"י האופרטורים הבאים:

$$[N, W, S, S, P, N, N, N, N, D], N - North, W - West\ , S - South, P - Pickup, D - Dropoff$$

בכל צעד שאינו הורדת הנוסע ביעד הנכון נקבל קנס של -1 ובהורדה נקבל 20 נקודות, ועל כל העלות עבור הפתרון הטוב ביותר היא 11 .

1.7 בהחלט ייתכנו מעגלים במרחב החיפוש שלנו שכן ניתן לבצע פעולות חזרה לאותה משבצת שמייצגת את אותו מצב. כך למשל ביצוע *North-South*.

אם נחסום ביקור במצבים שביקרנו בעבר (עם *Close list*) אז לא ייתכנו מעגלים.

שאלה 2

2.1 מימוש האלגוריתם

2.2 מספר המצבים שפותחו הם 31. רשימת הClose:

[328, 428, 228, 348, 448, 128, 248, 208, 28, 108, 148, 268, 308, 8, 48, 168, 368, 288, 408, 68, 188, 468, 388, 416, 88, 488, 316, 216, 116, 236, 16]

2.3 מספר הצמתים השונים שפותחו עבור מצב התחלתי 328 הוא עדיין 31 משום שמימשנו את הBFS על הגרף, ולכן לא ייתכן שנפתח צומת שכבר פותח בעבר.

2.4

<u>יתרון</u> – אלגוריתם BFS הינו קביל ושלם, כלומר הוא ימצא לנו את הפתרון האופטימלי (במידה שקיים) לעומת DFS שעלול למצוא פתרון שאינו אופטימלי (כמו עבור מצב התחלתי 328).

<u>חסרון</u> – אלגוריתם BFS עשוי לפתח יותר מצבים בעת מציאת הפתרון לעומת DFS שעשוי למצוא את פתרון (לאו דווקא אופטימלי) במספר צמתים נמוך יותר, כלומר עשוי לרוץ פחות זמן.

כך למשל – עבור $I = 209$ מצאנו כי BFS מפתח 56 מצבים לעומת DFS שמצליח להגיע אל פתרון בפיתוח 28 צמתים:

מצב 209:

```
+---------+
|R: | : :G|
| : | : : |
| : : : : |
| | : | : |
|Y| : |B: |
+---------+
```

נציין כי מספר הצמתים שיפותחו תלוי בסדר הפעולות בעת הצמתים. במקרה שלנו *Dropoff* היא הפעולה **האחרונה**, והתשובה תשנה אילו היא הייתה **הראשונה**.

2.5 הטענה נכונה, כל עוד מחירי הקשתות בבעיה זו הן תמיד 1-.

נשים לב כי בבעיה שאנו פתרנו, אנו לא מאפשרים חזרה למצב שכבר פותח.

הורדה והעלאת נוסע בצורה לא חוקית שנותנת Reward שונה מ--1 גורמת לכך שבגרף המצבים היינו חוזרים לאותו מצב, אך מכיוון שחסמנו זאת – לא קיימות קשתות עם מחירים שאינם 1-.

סה"כ כל הקשתות בעלי משקל זהה, ולכן לפי טענה שלמדנו מתרגול – אלגוריתם BFS הוא שלם ויחזיר את המסלול האופטימלי. (וגם תחת ההנחה שהמצב ההתחלתי חוקי, כלומר לא מתחיל ב-Dropoff).

שאלה 3

3.1 מימוש האלגוריתם

3.2 מספר הצמתים שפותחו הוא 100:

Close = [328, 428, 448, 348, 248, 148, 48, 68, 168, 268, 368, 468, 488, 388, 288, 188, 88, 228, 128, 28, 8, 108, 208, 308, 408, 416, 316, 216, 116, 16, 36, 136, 236, 336, 436, 456, 356, 256, 156, 56, 76, 176, 276, 376, 476, 496, 396, 296, 196, 96, 84, 184, 284, 384, 484, 464, 364, 264, 164, 64, 44, 144, 244, 344, 444, 424, 324, 224, 124, 24, 4, 104, 204, 304, 404, 472, 372, 272, 172, 72, 92, 192, 292, 392, 492, 52, 152, 252, 352, 452, 432, 332, 232, 132, 32, 12, 112, 212, 312, 412]

3.3 מכיוון שנתבקשנו לשמר מצבים שפתחו, גם כאן מספר הצמתים שיפתחו לא שונה ועומד על 100.

שאלה 4

4.1 מימוש האלגוריתם.

הערה: מכיוון שנתבקשנו לחפש על הגרף, תחזקנו רשימת CLOSE.

רשימת הClose פועלת מעט שונה - כעת אנחנו לא ניצור צומת חדש אם מצאנו אותו כבר ב-CLOSE עם depth גבוה יותר מה-depth שכרגע הוא נמצא בו בחיפוש, וזאת מכיוון שהחיפוש בdepth נמוך יותר מוכל בחיפוש שב-depth גבוה יותר עבור אותו צומת.

כלומר התנאי בעת יצירת השכנים הוא:

```
for s in get_neigbours(node):
    # If the neighbour was visited in a much higher depth - skip
    if depth-1<=CLOSE[s.state] :
        continue
```

4.2 נציין כי שוב נתבקשנו לממש את האלגוריתם על הגרף, ועל כן בכל Depth, נשמרת רשימת CLOSE ולא נבקר במצבים שפותחו.
להלן מספר הצמתים שפותחו בכל שלב עומק עד עומק 10 בו נמצא פתרון:

Depth	0	1	2	3	4	**5**	6	7	8	9	10	**Sum**
Expanded Nodes	0	1	4	8	14	**22**	32	42	52	63	70	**308**

ולכן פותחו סה"כ 308 צמתים. אם נתבקשנו לחפש כמה צמתים באופן גלובאלי, התשובה היא **70**.

בהינתן עומק 5, מס' הצמתים הוא 22.

4.3 נוכל לספור את כמות הNodes ב-Close List בכל איטרציה:

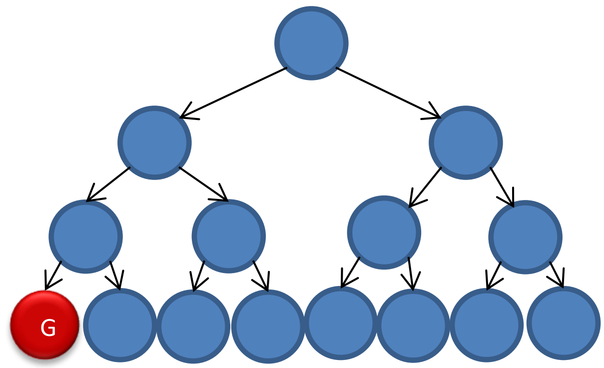
Depth	0	1	2	3	4	**5**	6	7	8	9	10	**Sum**
Expanded Nodes	0	1	4	8	13	**19**	24	27	28	30	30	**184**

התשובה היא 184 סה"כ אם נסכום את כל האיטרציות, ו**30** Nodes באפן גלובלי. (הסיבה שאין הפרש בין האיטרציה 9 וה10 היא מכיוון שאת צומת המטרה אנחנו לא מפתחים, אך כן מגיעים אליו באיטרציה ה10).

בהינתן עומק 5, מס' הצמתים הוא 19.

4.4 יתרון $ID-DFS$ מוצא פתרון שלם וקביל לעומת DFS.

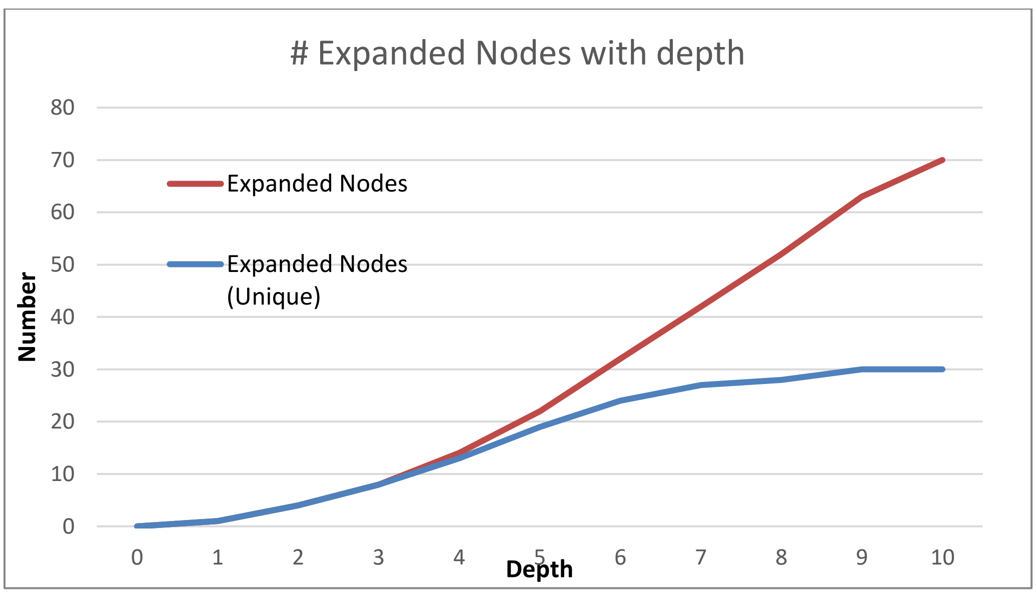
חסרון ה- $ID-DFS$ הוא שאנו מפתחים צמתים שביקרנו בהם בעומקים שונים פעמים נוספות, ועל כן יש בזבוז כלשהו של מקום וזמן. כך למשל בגרף המצבים הבא, הDFS יחזיר פתרון הרבה יותר מהר מאשר ID-DFS:



4.5 יתרון ה $ID - DFS$ היא שסיבוכיות המקום קטנה בהשוואה לאלגוריתם הBFS .

החסרון הוא שיתכן ויהיו מספר איטרציות (עבור עומקים שונים) באלגוריתם ה $ID - DFS$ ובמקרה זה יפתח יותר צמתים מאשר באלגוריתם הBFS.

5.1



נבחין כיצד מספר הצמתים שמפותחים עולה ככל שהעומק המקסימלי גדל.

נציין שאם לא היינו מתבקשים לשמור צמתים שביקרנו בהם, היינו מצפים לראות מגמה אקספוננציאלית, שכמות הצמתים גדל בצורה מערכית (בפקטור מקדם הסיעוף = 6).

שאלה 5

נגדיר משקול חדש על הקשתות בגרף, שפרופורציונאלי לערך ה-Reward של סביבה.

מכיוון שנרצה שכל מחירי הקשתות יהיו חיוביות וחסומות ע"י $\delta > 0$, והמחיר יפעל כך:

$$Cost(e_{destination}) = 1$$

$$Cost(e_{normal}) = 1$$

$$Cost(e_{illegal}) = 10$$

כלומר מחיר הקשתות חסום ע"י $\delta = 1$ ומחירי הקשתות חיוביים.

5.2 מימשנו את האלגוריתם

5.3 בהחלט ייתכן, מכיוון שבאלגוריתם אנחנו בודקים לראות האם יש לעדכן מסלולים של צמתים במידה שמצאנו ערך g נמוך יותר עבורם. את הבדיקה אנחנו מבצעים גם לצמתים שנוצרו וגם שפותחו. הנה קטע הקוד מההרצאה בו מתבצע עדכון הסלול לצומת שכבר פותח:

```
○ Else // s∈ CLOSED
    ❖ n_curr <- node in CLOSED with state s
    ❖ If new_g < n_curr.g ()                          //found better path to s
        • n_curr <- update_node(s, n, new_g , new_g + h(s))
        • OPEN.insert(n_curr)
        • CLOSED.remove(n_curr)
```

5.4 מימשנו.

5.5

1) *GreedyHeuristic*

נשים לב כי מתקיים:

$$h_1(s) = GreedyHeuristic(s) = \begin{cases} 0 & s \in goal\ state \\ 1 & o.w \end{cases} \leq \min\{1,10\}$$

כאשר 10 הוא המחיר על העלאת\הורדת נוסע שלא במקום חוקי, ואילו 1 על כל שאר הפעולות. מכאן שסה"כ היורסטיקה מקיימת:

$$h_1(s) - h_1(s') \leq 1 \ \ \forall s', s' \in SUCC(s)$$

ומכאן שהיוריסטיקה עקבית, וזה גורר קבילות.

2) המסלול האופטימלי לפתרון הבעיה הוא זה שמבצע את הפתרון במספר מינימלי של צעדים, מכיוון שמחיר הקשתות הוא זהה (ומחיר 10 של פעולת העלאה והורדת נוסע בהכרח לא יהוו פתרון טוב יותר מאותו מסלול שמחסיר פעולה זו).

ולכן h^* תמיד שווה למספר הפעולות שיש לבצע על מנת להוריד את הנוסע ביעד.

מכיוון שהמונית תופסת משבצת אחת בלבד, בהכרח המרחק מנהטן ייתן ערך קטן או שווה למרחק האמיתי של המונית מהנוסע/יעד הורדה, ובאופן ויזואלי ניתן לחשוב על יוריסטיקה זו כסכום מרחקים אוויריים של המונית מהנוסע + נוסע מיעדו.

במקרה שלנו:

$$h_2(s) = ManhatanSumHeuristic(s) = \left|loc_{taxi} - loc_{passenger}\right| + \left|loc_{passenger} - loc_{drop\ off}\right| \\ \leq \#N\ operations\ for\ pickup\ from\ s + \#M\ operations\ for\ dropoff\ from\ s = h^*(s)$$

ולכן סה"כ היוריסטיקה קבילה.

3+4) היורסטיקות אינן קבילות. נסתכל על מצב שבו המונית **אספה** את הנוסע וכבר ממקומת ביעדה, ומבצעת פעולה של הורדת נוסע. במקרה זה $h^* = 0$ שכן אנחנו הגענו למצב סופי.

ארם נקח דוגמא שב היעד הורדת נוסע ומיקום העלאת הנוסע **שונים**, ואז מתקיים:

$$s = loc_{drop\ off}$$

$$loc_{pick\ up} \neq loc_{drop\ off}$$

$$MD(s, loc_{pick\ up}) > 0\ \ , MD(sloc_{drop\ off}, loc_{pick\ up}) > 0$$

וסה"כ:

$$\frac{MD(s, loc_{pick\ up}) +\ MD(sloc_{drop\ off}, loc_{pick\ up})}{25} > 0 = h^*$$

וגם:

$$\frac{MD(s, loc_{pick\ up}) *\ MD(sloc_{drop\ off}, loc_{pick\ up})}{25} > 0 = h^*$$

ולכן יוריסטיקות 3+4 אינן קבילות לפי הגדרה.

5.6 מימשנו.

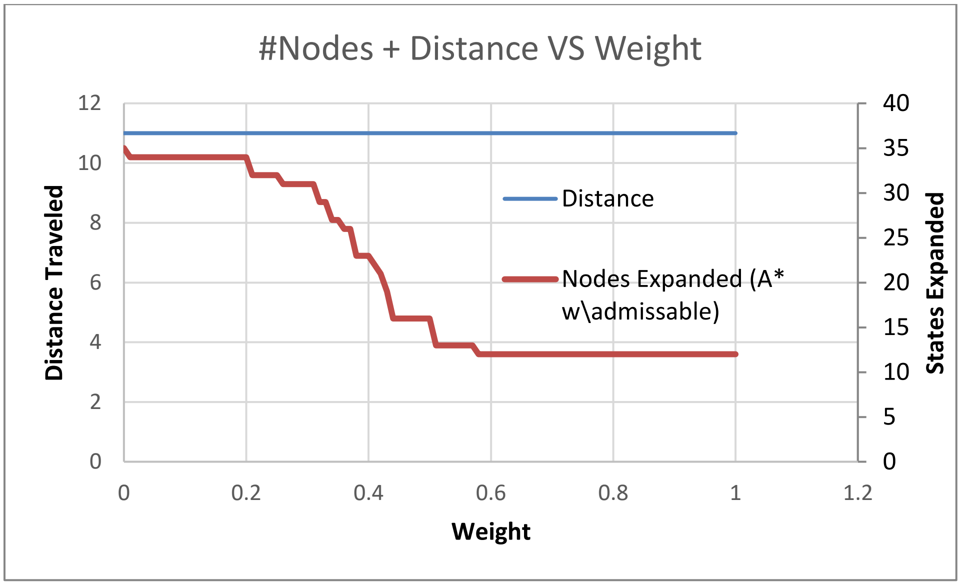
5.7 בריצה העיוורת פיתחנו 35 מצבים. חסם תחתון למספר זה הוא 31 שאלו מספר המצבים שפותחו ב*BFS*. השוני נובע מהעובדה שבעת מיון על פי ערך *F* ייתכן והסדר בתור ישתנה.

בריצת ה-*MD* פיתחנו 16 מצבים, ולכן חסכנו 19 מצבים.

שתי הריצות בוצעו עם משקל $w = \frac{1}{2}$, כלומר A^* רגיל.

5.8 מימשנו

5.9



5.10

נשים לב כי טיב הפתרון נותר זהה מכיוון שהיוריסטיקה היא קבילה, והאלגוריתם ימצא פתרון אופטימלי בכל המשקלים.

ככל שהמשקל גדל, מספר הצמתים שמפותחים קטן כצפוי, וזה מתאים לנקודה שנלמדה בכיתה – שככל שהאלגוריתם "מיודע" יותר (משקל גבוה יותר ליוריסטיקה) כך אנו עשויים לבצע פחות צעדים אל הפתרון.

בכיתה נלמד כי ככל שהאלגוריתם מיודע יותר, אנו עשויים לפגוע בטיב הפתרון, אך לא רואים זאת כאן מכיוון שהיורסטיקה שלנו קבילה והפתרון שמתקבל אופטימלי. מאותה סיבה זו אנחנו לא רואים את התופעה שככל שהמשקל קטן יותר, הפתרון משתפר שכן הוא אופטימלי בכל משקל.

שאלה 6

נגדיר את משקלי הקשתות בגרף המצבים באופן זהה לזו שהגדרנו בשאלה 5.

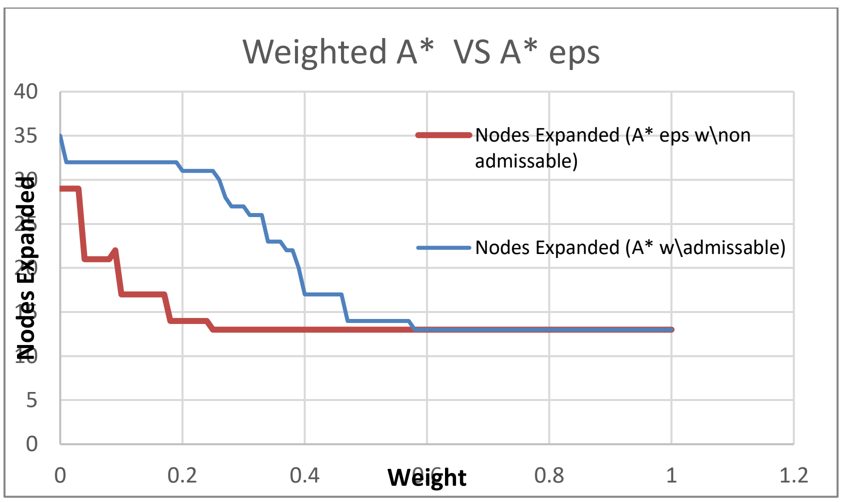
6.1+6.2 מימשנו.

6.3 מימשנו יוריסטיקה קבילה – סכום MD. היוריסטיקה הוכחה כקבילה בשאלה קודמת.

ליוריסטיקה לא קבילה – בחרנו את סכום MD כפול פקטור:

$$h_{non\ admissable}(s) = ManhatanSumHeuristic(s) \times 1.5$$

כאשר הרצנו אותה עם $\epsilon = 1$ על משקלים משתנים, עדיין קיבלנו תמיד את הפתרון האופטימלי, אך מעניין לראות כי בהשוואה אל סעיף קודם שבו הרצנו את $A*$ עם יוריסטיקה קבילה, מספר הצמתים שפותח תמיד יותר קטן, כלומר הגענו אל הפתרון האופטימלי מהר יותר מאשר $A*$:



ועל כן היוריסטיקה הלא קבילה שבחרנו חוסכת פתיחת צמתים, והרבה במקרים במשקלים שקטנים מ-0.5, חסכנו ביותר מ10 צמתים שפתחו.

כפי שנלמד, השימוש באלגוריתם $A^*\epsilon$ נותן לנו חסם על טיב הפתרון: $(1+\epsilon)C^*$ כאשר C^* הוא טיב הפתרון הטוב ביותר. בשילוב עובדה זו, נוכל להיעזר ביורסטיקה שגם אינה קבילה, שכן כעת ההבטחה על טיב המסלול אינו קשור ליורסטיקה הנבחרת.

הגמישות מגיעה לידי ביטוי בכך שנבחר טווח של ערכי f המינימליים מהצמתים שפותחו, ומתוך קבוצה זו נבחר בפיתוח צומת בעלת היוריסטיקה המינימלית ביותר. ככה אנחנו גורמים לאלגוריתם להיות מאוד מיודע (וזהו הפוטנציאל לשיפור במספר הצמתים שמפותחים), ועם זאת לתת חסם על טיב הפתרון.

נציין שהבחירה של היוריסטיקה שלנו ב-*MD* היא בחירה נכונה לטעמנו מכיוון שהסוכן יכול לנוע בצורה אנכית ואופקית. אם כן העובדה שאנחנו מאפשרים בחירה של צמתים בצורה גמישה יותר, מאפשר לנו לתת יותר משקל דווקא ליוריסטיקה, ואנחנו מאפשרים חיסכון בפיתוח צמתים. במקרה שלנו תמיד הגענו לפתרון הטוב ביותר, ובפחות צמתים שפותחו.

שאלת סגנון מבחן

<u>מימוש I:</u>

לא קביל.

ייתכן ועדיין קיימים פועלי רקע שמעבדים צומת שבו מסלול יותר טוב לצומת המטרה (שב*A הרגיל היינו ממתינים כי לא נחזיר את המטרה עד שהצומת מטרה יצאה מה-OPEN). המסלול הטוב יותר לא ייכנס לשקלול, ונחזיר בהכרח את המסלול האשון שעובד כלשהו מצא (ולא אופטימלי).

כן שלם.

נתון כי מרחב המצב סופי, ולכן בהכרח לא נתקע בחיפוש אין סופי, ובמקרה הגרוע נחפש את כל המסלולים בגרף, ולכן בהכרח נמצא פתרון.

<u>מימוש II</u>

לא קביל.

ייתכן מקרה ובו הגענו אל צומת המטרה, וכל העובדים סיימו, אך קיים מסלול טוב יותר אל צמת המטרה עם צמתים שנמצאים בOPEN. במקרה זה נחזיר את הפתרון הלא אופטימלי.

לא שלם – יתכן והעובדים לא סיימו ולכן הצומת לא תוחזר והיא לא נשמרת בשום מקום אחר.

<u>מימוש III</u>

לא קביל.

אותו מקרה כמו בII יכול לקרות גם כאן.

כן שלם – מאותם שיקולים של מימוש I+II, אנחנו בהכרח נמצא את צומת המטרה אך לא בהכרח במסלול האופטימלי.

<u>מימוש IV</u>

כן קביל.

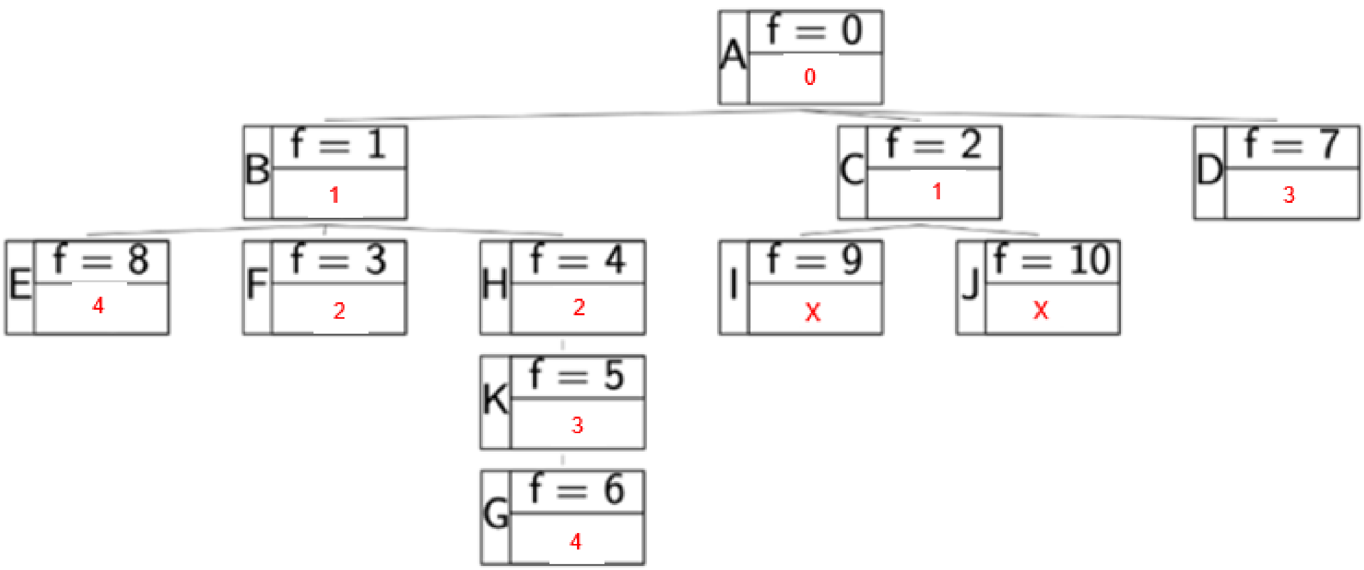
ראשית אנו ממתינים כי כל העובדים יסיימו, ולכן רשימת הOPEN תהיה מעודכנת לאחר שורת ה-Busy waiting בצמתים נוספים בעלי פוטנציאל שיפור.

מכיוון שנתון שהיורסטיקה **עקבית**, אז בהכרח ערכי הf של צמתים הם מונוטוניים. מכאן שאם מצאנו שערך הf של צומת המטרה הוא הכי נמוך (או שווה לערך f של צומת אחר ב-Open) אז בהכרח מצאנו את המסלול האופטימלי, שכן כל צומת אחר ב-Open רק יכול להגדיל את ערכי הf (וגם את הf הסופי ששווה ל-g, המחיר בפעל של המסלול).

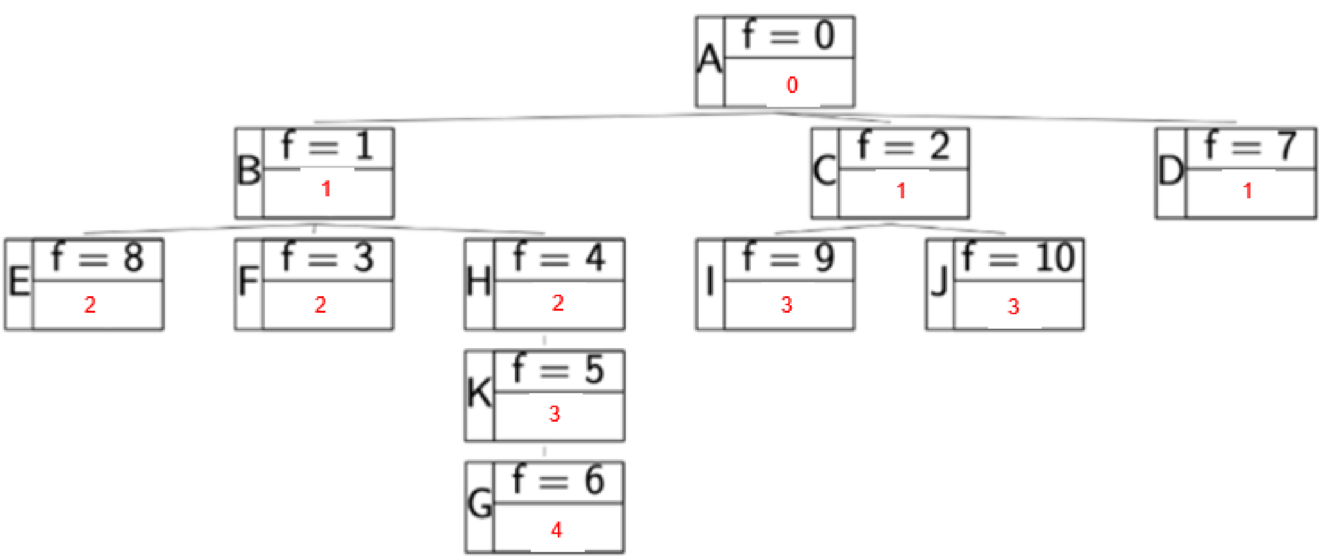
כן שלם – כי האלג' קביל.

חלק ב':

.1


A f = 0
0
B f = 1
1
C f = 2
1
D f = 7
3
E f = 8
4
F f = 3
2
H f = 4
2
I f = 9
X
J f = 10
X
K f = 5
3
G f = 6
4

.2


A f = 0
0
B f = 1
1
C f = 2
1
D f = 7
1
E f = 8
2
F f = 3
2
H f = 4
2
I f = 9
3
J f = 10
3
K f = 5
3
G f = 6
4