תרגיל בית 1: שימוש באלגוריתמי חיפוש מיודעים לתכנון מסלולים

מטרות התרגיל

- נתמודד עם בעיות פרקטיות ותיאורטיות של חיפוש במרחבי מצבים.
- נתרגל את הנלמד בהרצאות ובתרגולים.
- נתנסה בתכנות ב- python לפתרון בעיות פרקטיות.

הנחיות כלליות

- **תאריך הגשה:** מוצאי שבת, 7.05.2022, בשעה 23:59.
- את המטלה יש להגיש **בזוגות בלבד**.
- יש להגיש מטלות מוקלדות בלבד. פתרונות בכתב יד לא ייבדקו.
- התשובות צריכות להיות כתובות בשפה העברית. ניתן לבקש אישור פרטני להגשה באנגלית מהמתרגל האחראי על הקורס.
- ניתן לשלוח שאלות בנוגע לתרגיל **בפיאצה בלבד**.
- המתרגל האחראית על תרגיל זה: רועי עוגן.
- בקשות דחיה **מוצדקות** (מילואים, אשפוז וכו') יש לשלוח למתרגל האחראי (ספיר טובול) בלבד.
- במהלך התרגיל ייתכן שנעלה עדכונים, למסמך הנ"ל – תפורסם הודעה בהתאם.
- העדכונים הינם **מחייבים**, ועליכם להתעדכן עד מועד הגשת התרגיל.
- שימו לב, התרגיל מהווה כ- 10% מהציון הסופי במקצוע ולכן העתקות תטופלנה בחומרה.
- ציון המטלה יורכב מהגורמים הבאים:
  - **60% - המסמך היבש.** מעבר לתשובות הנכונות, אתם נבחנים גם על הצגת הנתונים והתוצאות בצורה קריאה ומסודרת במקומות בהם התבקשתם לכך. הניקוד המפורט בסעיפים של מסמך זה הינו מתוך הציון היבש בלבד.
  - **40% - הקוד המוגש.** הקוד שלכם ייבדק באופן מקיף ע"י מערכת בדיקות אוטומטיות. המערכת תבדוק את התוצאות שלכם לעומת התוצאות המתקבלות במימוש שלנו. אנו מצפים שתקבלו את אותם הערכים בדיוק. נבדוק בין היתר את המסלול המתקבל, את עלותו ואת מס' הפיתוחים. לכן עליכם להיצמד להוראות בתרגיל זה. הבדיקות יהיו כמובן מוגבלות בזמן ריצה. יינתן לכם זמן סביר ביותר להרצת כל טסט. אם תעקבו אחר ההוראות במסמך זה ובקוד אין סיבה שלא תעמדו בזמנים אלו. בנוסף, יש להקפיד על הגשת קוד מסודרת בהתאם להנחיות. יש לכתוב הערות במקומות חשובים בקוד כדי שיהיה קריא וקל לבדיקה ידנית.
- אנו יודעים שעבור חלקכם זו התנסות ראשונה בכתיבת קוד בפיתון ותרגיל זה מתוכנן בהתאם לכך.
- שימו לב שלא יענו שאלות בסגנון: "איך מוצאים את עלות הפתרון שהוחזר?" / "איך ניגשים למפות הכבישים מתוך המימוש של הפונק' ההיא?" / "באיזה שדה שמור ה...?" וכדומה.
- אנחנו רוצים לעודד אתכם לעיין בקוד ולמצוא פרטים אלו בכוחות עצמכם. הכרת סביבת העבודה שסיפקנו לכם והתמצאות בה הן למעשה חלק מהתרגיל.
- בתרגילי הבית בקורס הרצת הניסויים עשויה לקחת זמן רב. לכן מומלץ מאוד להימנע מדחיית העבודה על התרגיל ו/או כתיבת הדו"ח לרגע האחרון. לא תינתנה דחיות על רקע זה.
- מסמך זה כתוב בלשון זכר מטעמי נוחות בלבד, אך מתייחס לנשים וגברים כאחד.

אנחנו קשובים לפניות שלכם במהלך התרגיל ומעדכנים את המסמך הזה בהתאם. גרסאות עדכניות של המסמך יועלו לאתר. הבהרות ועדכונים שנוספים אחרי הפרסום הראשוני יסומנו כאן בצהוב. בנוסף, לכל עדכון יהיה מספר גרסה כדי שתוכלו לעקוב. ייתכן שתפורסמנה גרסאות רבות – אל תיבהלו מכך. השינויים בכל גרסה יכולים להיות קטנים.

הבהרות ועדכונים: כל השינויים במסמך בצהוב, ישנם כמה שינוים אך כאן המרכזיים:

- שימו לב כי עקב מסלולים אינסופיים שונתה הדרישה של DFS ו- DFS-L להיות חיפוש בגרף – כלומר כעת אין פיתוח חוזר של צמתים.
- לשם שלמות, במידה ואלגוריתם כלשהו לא מוצא פתרון החזרו (frames=[],reward=0)
- בקוד השתמשו ב- env.unwrapped.s במקום env.s למשל:

```
env = gym.make('Taxi-v3')
state1 = env.reset(seed=42)
env.render()
env.unwrapped.s = 42
env.render()
```

חלק א' – מבוא

במטלה זו נעסוק בהפעלת אלגוריתמי חיפוש על מרחבי מצבים לבעיות ניווט. מומלץ לחזור על שקפי ההרצאות והתרגולים הרלוונטיים לפני תחילת העבודה על התרגיל.

במהלך התרגיל תתבקשו להריץ מספר ניסויים ולדווח על תוצאותיהם. אתם נדרשים לבצע ניתוח של התוצאות, כפי שיוסבר בהמשך.

מוטיבציה

אפליקציות מוניות פופולריות (Gett-Taxi, Uber) משתמשות באלגוריתמים לתכנון המסלול החוקי והמהיר ביותר לנסיעה בכבישים, ונעזרות בנתונים קיימים על דרכים, על היסטורית נסיעה בדרכים, ועל נתונים בזמן-אמת.

יותם, סטודנט תפרן, מעוניין להרוויח כסף מהצד ולכן החל לעבוד בחברת Uber. יותם הוא נהג מוכשר ואחראי שמחפש לנסוע במסלול הקצר ביותר בין לקוחותיו.

חברים של יותם (אתם!) לוקחים הסמסטר את הקורס "מבוא לבינה מלאכותית". יותם מבקש מכם לעזור לו לתכנן את המסלול הקצר ביותר ממיקום ההתחלתי דרך נקודת האיסוף ולבסוף ליעד.

חלק ב' – מתחילים לתכנת (40 נקודות)

משימה – קריאה מקדימה

מומלץ לעיין בתיעוד של הסביבות השונות ב https://gym.openai.com/ בפרט ניתן להתבונן ב-GIT של בעיית המונית: https://github.com/openai/gym

משימה – רטוב

כעת נעבור לחלק הרטוב של התרגיל. אנו נעבוד בopenai בסביבת Taxi-v3.

מבוא לסביבה, דוגמאות ושלד הקוד מסופקים לכם באתר הקורס בקובץ: "236501_HW1.ipynb"

אנא מלאו אחר ההוראות בקובץ בכדי לפתור את התרגיל.

מומלץ לעבור על הקוד בnotebook במקביל לקובץ הנוכחי וכך שלב שלב לענות על השאלות השונות.

אנא עקבו אחר ההוראות הנתונות, **חריגה מהן תוריד נקודות !**

חשוב לשים לב – (מפורט יותר בnotebook):

1. עליכם להריץ כל סוכן ממצב ההתחלתי 328
2. כל סוכן צריך להחזיר טאפל: (frames,reward)
3. frames: רשימה של מסגרות שניתן יהיה להדפיס באמצעות הפונקציה "print_frames" שקיבלת.
4. reward: מספר שלם שמכיל את התגמול הכולל של הנתיב האופטימלי (הקצר ביותר).

לדוגמא

```
frames = [{'frame': '+---------+\n|\x1b[35mR\x1b[0m: | : :G|\n| : | :
: |\n| : : : : |\n| |\x1b[43m \x1b[0m: | :
|\n|\x1b[34;1mY\x1b[0m| : |B: |\n+---------+\n  (Dropoff)\n',
'state': 328, 'action': -1, 'reward': 0}, {'frame': '+---------
+\n|\x1b[35mR\x1b[0m: | : :G|\n| : | : : |\n| : : : : |\n| | : |
: |\n|\x1b[34;1mY\x1b[0m|\x1b[43m \x1b[0m: |B: |\n+---------+\n
(South)\n', 'state': 428, 'action': 0, 'reward': -1}]
```

reward = -440

- לשם שלמות, במידה ואלגוריתם כלשהו לא מוצא פתרון החזרו (frames=[],reward=0)
- בקוד השתמשו ב- env.unwrapped.s במקום env.s למשל:

```
env = gym.make('Taxi-v3')
state1 = env.reset(seed=42)
env.render()
env.unwrapped.s = 42
env.render()
```

חלק ג' – שאלות יבשות על הרטוב (48 נקודות)

שאלה 1 – מבוא:

1.1 יבש (2 נק'): תחילה נרצה להגדיר את מרחב החיפוש כפי שנלמד בתרגול. הגדר את $\{S, O, I, G\}$ עבור סביבת המוניות (יש להגדיר בדומה לאופן שהסביבה הגדירה). מה גודל מרחב המצבים s? הסברו.

1.2 יבש (1 נק'): מה תחזיר לנו הפונקציה Domain על אופרטור North?

1.3 יבש (1 נק'): מה תחזיר לנו הפונקציה Succ על המצב ההתחלתי 328?

1.4 יבש (1 נק'): במקרה הגרוע ביותר, כמה פעולות ידרשו לסוכן random (כפי שממומש במחברת) לפתור את הבעיה, עבור מצב התחלתי 328?

1.5 יבש (1 נק'): במקרה הטוב ביותר, כמה פעולות ידרשו לסוכן random (כפי שממומש במחברת) לפתור את הבעיה, עבור מצב התחלתי 328?

1.6 יבש (2 נק'): מהי העלות המסלול המתקבלת עבור פתרון הבעיה במקרה הטוב ביותר, עבור מצב התחלתי 328 ומהו המסלול (הראה את המסלול כרשימה של אופרטורים)?

1.7 יבש (2 נק'): האם ייתכנו מעגלים במרחב החיפוש שלנו? אם כן תנו דוגמה למעגל כזה, אחרת נמקו.

שאלה 2 – BFS-G:

2.1 רטוב: ממשו את אלג' BFS-G (על גרף) בקובץ ע"פ ההנחיות המופיעות שם.

2.2 יבש (2 נק'): מהו מספר הצמתים שפותחו עבור מצב התחלתי 328?

2.3 יבש (2 נק'): מהו מספר הצמתים **השונים** שפותחו עבור מצב התחלתי 328?

2.4 יבש (2 נק'): מנו יתרון וחסרון של BFS לעומת DFS, בהתייחס לבעיית המוניות.

2.5 יבש (2 נק'): האם חיפוש BFS תמיד ימצא את הפתרון האופטימלי (בעלות הנמוכה ביותר) בבעיית המוניות? הוכח או הבא דוגמא נגדית

שאלה 3 – DFS:

3.1 רטוב: ממשו את אלג' DFS (על גרף – שומר מצבים שפותחו) בקובץ ע"פ ההנחיות המופיעות שם.

3.2 יבש (2 נק'): מהו מספר הצמתים שפותחו עבור מצב התחלתי 328?

3.3 יבש (2 נק'): מהו מספר הצמתים **השונים** שפותחו עבור מצב התחלתי 328?

שאלה 4 – ID-DFS: על גרף – שומר מצבים שפותחו

4.1 רטוב: ממשו את החלקים החסרים של אלג' ID-DFS בקובץ ע"פ ההנחיות המופיעות שם.

4.2 יבש (2 נק'): מהו מספר הצמתים שפותחו עבור מצב התחלתי 328?

4.3 יבש (2 נק'): מהו מספר הצמתים **השונים** שפותחו עבור מצב התחלתי 328?

4.4 יבש (2 נק'): מנו יתרון וחסרון של ID-DFS לעומת DFS, בהתייחס לבעיית המוניות.

4.5 יבש (2 נק'): מנו יתרון וחסרון של ID-DFS לעומת BFS, בהתייחס לבעיית המוניות.

5.1 יבש (2 נק'): הציגו גרף המראה את השפעת D (לפחות 5 ערכים שונים) על ריצת ID-DFS, אשר מתארת את מספר המצבים שפותחו כפונקציה של עומק החיפוש. הסבירו בקצרה את המגמה בגרף.

שאלה 5 – W-A*:

5.2 רטוב: עליכם לממש את W-A*. מומלץ לממש את A* ולאחר מכן להרחיב אותו ל-W-A*.

5.3 יבש (1 נק'): האם ייתכן באלג' A* שנפגוש מצב בשנית גם לאחר שפיתחנו אותו? אם כן – ציין את השורה באלג' A* מההרצאה שבה זה קורה, אם לא - נמק.

5.4 רטוב: כידוע, לצורך הרצת A* יש צורך בהיוריסטיקה. לצורך בדיקת שפיות, הפעילו את ה- A* עם Null Heuristic (תחזיר 0 לכל מצב) - הריצה העיוורת - תחת null_h.

5.5 יבש (4 נק'): לפניכם מספר יוריסטיקות, הוכיחו או הפריכו בעזרת דוגמא נגדית את קבילותן של היוריסטיקות. מהי היוריסטיקה המיודעת ביותר?

1) $GreedyHeurisitc(s) = \begin{cases} 0 & if\ s \in goal\ state \\ 1 & otherwize \end{cases}$

2) $ManhatanSumHeurisitc(s) = MD\left(s, loc_{passenger}\right) + MD\left(loc_{passenger}, loc_{drop-off}\right)$

3) $PickupSumHeurisitc(s) = \frac{MD\left(s, loc_{pick-up}\right) + MD\left(loc_{pick-up}, loc_{drop-off}\right)}{5 \cdot 5}$

4) $PickupMultHeurisitc(s) = \frac{MD\,(s, loc_{pick-up}) * MD\,(loc_{pick-up}, loc_{drop-off})}{5 \cdot 5}$

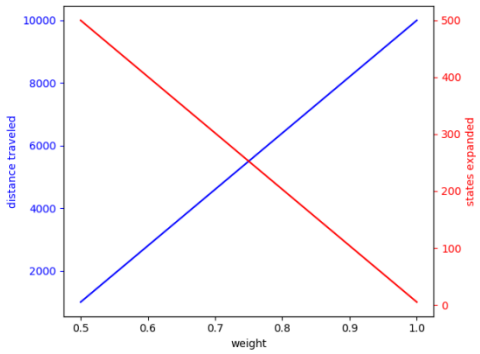
כאשר $loc_{passenger}$ הוא מיקום הנוסע, $loc_{drop-off}$ הוא מיקום ההורדה של הנוסע, $loc_{pick-up}$ זה המיקום ההתחלתי של הנוסע (ממנו הוא עלה למונית) ו-MD זה מרחק מנהטן.

5.6 רטוב: בחרו אחת מהיוריסטיקות מהסעיף הקודם וממשו אותה תחת הפונקציה chosen_h.

5.7 יבש (1 נק'): כתוב בדו"ח את מס' פיתוחי המצבים היחסי שחסכנו בריצה עם היוריסטיקה שבחרתם בסעיף קודם לעומת הריצה העיוורת.

5.8 רטוב: כעת נרצה לבחון את השפעת המשקל w על ריצת W-A*. מלאו בקובץ את מימוש האלגוריתם.

5.9 יבש (2 נק'): הריצו את W-A* עם ערכי w שונים כדי לצייר גרף שמציג את מגמת מחיר הפתרון מגמת מס' הפיתוחים כאשר w משתנה בתחום [0,1] וצרו גרף אשר בו מופיעות 2 עקומות: אחת מהעקומות (הכחולה) מתארת את טיב הפתרונות (בציר y) כפונק' של המשקל (עלות המסלול במקרה של בעיית המפה הבסיסית). העקומה השנייה (האדומה) מתארת את מספר המצבים שפותחו כפונק' של המשקל. (מוזמנים ליצור או דרך פייתון או ידנית בוורד מה שנוח לכם, רק כמובן שיתאים לתוצאות אשר יבדקו – כלומר לא להגיש קוד אשר יוצר גרף)



5.10 יבש (4 נק'): הסבירו את הגרף שהתקבל. ציינו אילו אזורים בגרף הם יותר כדאיים ואילו פחות – ציינו למה. בכיתה למדתם כלל אצבע לפיו "ככל ש- w קטן יותר כך הפתרון איכותי יותר ומס' הפיתוחים גדול יותר". הכלל הנ"ל מצביע על מגמה כללית, אך איננו נכון באופן גורף (כלומר ייתכנו זוג ערכים w_1<w_2 עבורם הפתרון המתקבל עם w_1 פחות טוב מאשר הפתרון המתקבל עם w_2 ו/או מס' הפיתוחים עם w_2 גדול יותר ממס' הפיתוחים עם w_1). כיצד הכלל שהוזכר והדגש הנ"ל באים לידי ביטוי בתרשים שקיבלתם?

שאלה 6 – A*-epsilon:

6.1 רטוב: ממשו את אלג' A*ε בקובץ ע"פ ההנחיות המופיעות שם.

6.2 רטוב: ממשו יוריסטיקה קבילה (admissible_h) ויוריסטיקה לא קבילה (non_admissible_h) שצפויה לתת תוצאות טובות.
מוטיבציה: הבעיה היא שאין לנו אף הבטחה על איכות הפתרון שמניב A* עם היוריסטיקה שאינה קבילה. נרצה לנצל את הבטחת איכות הפתרון של A*ε כדי לעשות שימוש מועיל בהיוריסטיקה שאינה קבילה במטרה לחסוך במספר הפיתוחים מבלי לפגוע באופן דרסטי באיכות הפתרון.

6.3 יבש (4 נק'): תארו את היוריסטיות שמימשתם וצרפו לדו"ח את התוצאות שקיבלתם בסעיף הקודם (אל תצרפו את המסלולים עצמם). האם חסכנו בפיתוחים? אם כן, בכמה? הסבירו למה בכלל ציפינו מראש ש- A*ε יוכל לחסוך במס' הפיתוחים בתצורה שבה הרצנו אותו. לא מספיק לטעון ש- A*ε גמיש יותר בבחירה של הצומת הבא לפיתוח. נסו להסביר למה בעצם אנחנו מצפים שהגמישות הזאת של A*ε אכן תעזור לנו במקרה הזה לבחור מ- open צומת לפיתוח שיקדם אותנו מהר יותר למטרה. מה בעצם הוספנו לאלג' החיפוש?

חלק ד' – שאלה בסגנון מבחן (12 נקודות)

נזכיר שניתן לבצע חיפוש בגרף *A בעזרת הפסאודו קוד הבא:

```
1: function A*-Graph-Search(problem, open)
2:      closed ← an empty set
3:      open ← Insert(Make-Node(Initial-State[problem]), open)
4:      loop do:
5:              if open is empty then return failure
6:              node ← Remove-Front(open)
7:              if Goal-Test(problem, State[node]) then return node
8:              if State[node] is not in closed then
9:                      add State[node] to closed
10:                     child-nodes ← Expand(node, problem)
11:                     open ← Insert-All(child-nodes, open)
```

נשים לב שריצת פונקציית ה-successor (Expand) לוקחת הרבה מאוד זמן לחישוב ומשך הזמן יכול להשתנות מאוד מצומת לצומת, אז נרצה לזרז דברים באמצעות תכנות במקביל (parallelization).

לצורך כך **נגדיר את A*-Parallel** (מימוש בעמוד הבא):
המשתמשת בתהליך "מאסטר" (master thread) שמריץ את A*-Parallel ומשתמשת בקבוצה של $n \geq 1$ "עובדים", (workers) שהם תהליכים נפרדים שמריצים את הפונקציה Worker-Expand שמבצעת הרחבת צומת וכתיבת תוצאות בחזרה לopen המשותף לכולם.

תהליך ה"מאסטר" מוציא קריאות לא חוסמות[1] ל-Worker-Expand אשר שולח לעובד נתון פקודה להתחיל להרחיב צומת מסוים.

הפונקציה Wait שנקראת מהשרשור הראשי מפסיקה את ההרצה (sleep) של תהליך ה"מאסטר" לפרק זמן קצר, למשל, 20 אלפיות השנייה.

הערה: ה-open של פונקציות אלה נמצאים בזיכרון המשותף ותמיד מועבר by reference. כמו כן נניח שניתן לשנות בבטחה את הopen המשותף מכל worker.

מומלץ לחשוב על A*-Parallel כשינוי של A*-Graph-Search .
בשורות 5-9 A*-Parallel מחכה לעובד כלשהו שיהיה פנוי, לאחר מכן (במידת הצורך) ממתין עד שה-open לא ריק, כך שניתן יהיה להקצות לעובד את הצומת הבא שיורחב מopen.
אם כל העובדים פנויים בזמן ש open ריק, זה אומר שלא תתבצע יותר הכנסה לopen, מה שאומר שאין מסלול למצב למטרה ולכן החיפוש נכשל ויחזיר- כישלון.(זה מתאים לשורה 5 ב- A*-Graph-Search)

שורה 16 ב A*-Parallel-מקצה תהליך worker פנוי שיבצע את Worker-Expand בשורות 17-19.
(זה מתאים לשורות 10-11 של A*-Graph-Search)

לבסוף, שורות 11-13 ב– A*-Parallel (המקביל לשורה 7 ב- A*-Graph-Search) הוא המקום שבו **העבודה שלכם מתחילה.**

מכיוון שהעובדים פועלים במקביל, זו משימה לא פשוטה לקבוע מתי ניתן להחזיר מטרה מהקוד. זאת כי אולי אחד מהעובדים בדיוק עמד להוסיף צומת מטרה ממש טוב ל-open ואולי נרצה להשתמש במידע החדש.

1. קריאה לא חוסמת אומרת שהשרשור הראשי ממשיך לבצע את הקוד שלו מבלי לחכות שהעובד יחזור מהפעילות שלו.

מימוש A*-Parallel:

```
function A*-Parallel(problem, open, workers)
    closed ← an empty set
    open ← Insert(Make-Node(Initial-State[problem]), open)
    loop do
        while All-Busy(workers) do Wait
        while open is empty do
            if All-Idle(workers) and open is empty then
                return failure
            else Wait
        node ← Remove-Front(open)
        if Goal-Test(problem, State[node]) then
            if Should-Return(node, workers, open) then
                return node
        if State[node] is not in closed then
            add State[node] to closed
            Get-Idle-Worker(workers).Worker-Expand(node, problem, open)

function Worker-Expand(node, problem, open)
    child-nodes ← Expand(node, problem)
    open ← Insert-All(child-nodes, open)
```

כעת, התבוננו במימושים הבאים של Should-Return הנקראת לפני שמחזירים צמת מטרה (goal node) ב-A*-Parallel:

I.
```
function Should-Return(node, workers, open)
    return true
```

II.
```
function Should-Return(node, workers, open)
    return All-Idle(workers)
```

III.
```
function Should-Return(node, workers, open)
    open ← Insert(node, open)
    return All-Idle(workers)
```

IV.
```
function Should-Return(node, workers, open)
    while not All-Idle(workers) do Wait
    open ← Insert(node, open)
    return F-Cost[node] == F-Cost[Get-Front(open)]
```

עבור כל אחד מהמימושים, ציינו האם הוא מביא לאלגוריתם חיפוש שלם, והאם הוא מביא לאלגוריתם אופטימלי
לחיפוש (קביל). תנו נימוק קצר לתשובתכם (תשובות ללא נימוק יזכו לניקוד אפסי). (כל תת סעיף 1 נק)
נניח שמרחב המצב סופי, וההיוריסטיקה עקבית.

- ייתכן ואותו node יוכנס מחדש לopen עם אב אחר, כך שה"אב" משמש כמזהה

1. מימוש I
   a. קביל? כן / לא - נמק:

b. שלם? כן / לא - נמק:

2. מימוש II
   a. קביל? כן / לא - נמק:

   b. שלם? כן / לא - נמק:

3. מימוש III – מקביליות ? אולי לשנות – מצב המטרה לא אופטימלי נכנס לopen. בדיוק כולם סיימו לעבוד בזמן שנכנסנו לפונקציה ובגלל זה נכניס את הלא אופטימלי
   a. קביל? כן / לא - נמק:

   b. שלם? כן / לא - נמק:

4. מימוש IV
   a. קביל? כן / לא - נמק:

   b. שלם? כן / לא - נמק:

חלק ב

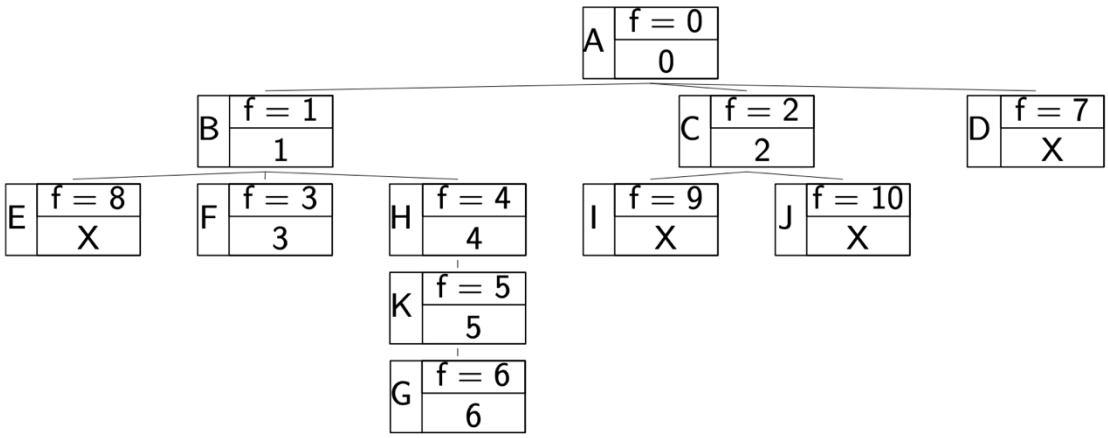
נניח שאנו משתמשים ב- A*-Parallel עם מימוש IV של הפונקציה Should-Return.

כעת, נוסיף הנחה חדשה נוספת לגבי זמן ביצוע: לכל עובד לוקח בדיוק O(1) זמן כדי להרחיב צומת ולדחוף את כל הצמתים הבנים (succesors) אל *open*, ללא תלות במספר הבנים (כולל אם יש אפס בנים).
כל חישוב אחר הוא מיידי (ללא זמן נוסף).

A*-Parallel עם מאפייני התזמון לעיל הופעל עם **עובד בודד** בבעיית חיפוש עם עץ החיפוש בתרשים למטה.

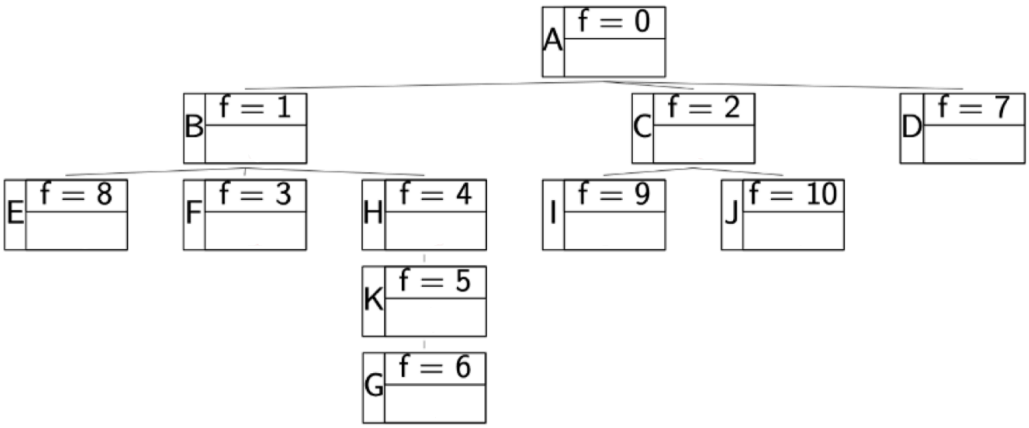
כל צומת מצויר עם המצב בצד שמאל, ערך f בצד ימין למעלה (f(n) = g(n) + h(n)) והזמן שבו עובד הרחיב את הצומת בפינה הימנית התחתונה (הסימון X הוא עבור צמת שלא הורחב). G הוא צומת המטרה.

בתרשים שלהלן, אנו יכולים לראות שצומת A הורחב על ידי העובד בשלב זמן 0, ואז צומת B הורחב בשלב זמן 1, צומת C הורחב בשלב זמן 2, צומת F הורחב בשלב זמן 3, צומת H הורחב בשלב זמן 4, צומת K הורחב בשלב זמן 5, וצומת G הורחב בשלב זמן 6. צמתים D, E, I, J מעולם לא הורחבו.

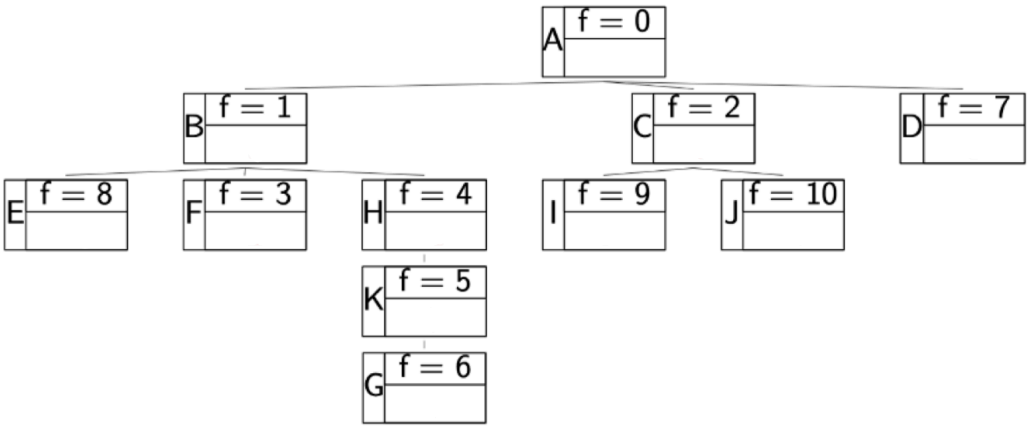


בשאלה זו תשלימו דיאגרמות דומות על ידי מילוי זמני ההתרחבות של הצומת במקרה של שנים ושלושה עובדים. שימו לב שכעת ניתן להרחיב מספר צמתים בכל זמן נתון.

1. השלם את זמני הרחבת הצומת במקרה של **שני עובדים** ומלא 'X' עבור כל צומת שאינו פותח. (2 נק)



2. השלם את זמני הרחבת הצומת במקרה של **שלושה עובדים** ומלא 'X' עבור כל צומת שאינו פותח. (2 נק)



חלק ה' – הגשת המטלה

מעבר למימוש ולדו"ח, ציונכם מורכב גם מהגשה תקינה של המטלה לפי הכללים הבאים.

- **יש לכתוב קוד ברור**:
  - קטעי קוד מסובכים או לא קריאים יש לתעד עם הערות.
  - לתת שמות משמעותיים למשתנים.
- **הדו"ח**:
  - יש לכתוב בדו"ח את תעודות הזהות של **שני** המגישים.
  - הדו"ח צריך להיות מוקלד במחשב ולא בכתב יד. הדו"ח צריך להיות מוגש בפורמט PDF (לא נקבל דוחות שהוגשו בפורמט וורד או אחרים).
  - יש לשמור על סדר וקריאות גם בתוך הדו"ח.
  - אלא אם נכתב אחרת, תשובות ללא נימוק לא יתקבלו.
  - יש לענות על השאלות לפי הסדר ומספרי הסעיפים שלהם.
- **ההגשה**:
  - יש להעלות לאתר קובץ zip בשם AI1_123456789_987654321.zip (עם תעודות הזהות שלכם במקום המספרים).
  - בתוך ה- zip צריכים להיות זה לצד זה:
    - הדו"ח הסופי בפורמט PDF בשם: AI1_123456789_987654321.pdf.
    - קובץ הnotebook בפורמט ipynb. בשם: AI1_123456789_987654321.ipynb.

חריגה מכללים אלו עלולה לגרור הורדה של עד 5 נקודות!!

בנוסף, קוד לא ברור עלול לגרור הורדת נקודות <u>נוספת</u> בפרק בו הוא נכתב.

שימו לב: הקוד שלכם ייבדק ע"י מערכת בדיקות אוטומטיות תחת מגבלות זמני ריצה. במידה וחלק מהבדיקות יכשלו (או לא יעצרו תוך זמן סביר), הניקוד עבורן יורד באופן אוטומטי. לא תינתן הזדמנות להגשות חוזרות. אנא דאגו לעקוב באדיקות אחר הוראות ההגשה. שימו לב כי במהלך חלק מהבדיקות ייתכן שחלק מהקבצים שלכם יוחלפו במימושים שלנו. אם עקבתם אחר כל הדגשים שפורטו במסמך זה - עניין זה לא אמור להוות בעיה.

לא תתאפשרנה הגשות חוזרות, גם לא בגלל טעות טכנית קטנה ככל שתהיה. אחריותכם לוודא טרם ההגשה שהתרגיל רץ בסביבה שהגדרנו ושהקוד עומד בכל הדרישות שפירטנו.

אנא עברו בשנית על ההערות שפורסמו בתחילת מסמך זה. וודאו שאתם עומדים בהם.

שימו לב: **העתקות תטופלנה בחומרה.** אנא הימנעו מאי-נעימויות.

מקווים שתיהנו מהתרגיל!