חלק א – ImprovedGreedy

1. (יבש: 4 נק') כפי שלמדתם, אנו מגדירים בעיה במרחב בתור רבעיה.(S, O, I, G) הגדירו פורמלית את המשחק המתואר לכם ע"פ הנתונים שאתם מקבלים מהסביבה.

- $S = \{Robot_i = [position, battery, credit, package, NumOfSteps],$
  $Packages_i = [position, destination],$
  $Charging_Station_i = [position]\}$

  *$i \in 0,1$

- $O = \{north, south, east, west, picck\ up, drop\ off, charge\}$

- $I = \{Robot_i = [position_{init}, battery_{init}, credit = 0, package = 0, NumOfSteps = 0],$
  $Packages_i = [position_{init}, destination_{init}],$
  $Charging_Station_i = [position_{init}]\}$

  *All initial(init) values are random

- $G = \{(Robot_0[NumOfSteps = max] \cap Robot_1[NumOfSteps = max]) \cup (Robot_0[battery = 0] \cap Robot_1[battery = 0])\}$

2. (יבש: 4 נק') הגדירו היוריסטיקה משלכם להערכת מצבי המשחק. עליכם לתעד אותה בנוסחה מפורשת ועלייה לכלול לפחות שלושה מאפיינים של הסביבה. בחרו שמות ברורים בנוסחה שלכם.

כדי להגדיר את היוריסטיקה נשתמש בפרמטרים הבאים:

א. מרחק מנהטן בין הרובוט לחבילה הקרובה ביותר אליו:
$D_{R\backslash P} = \min(MD_{R\to P})$

ב. מרחק מנהטן בין החבילה הקרובה ביותר ליעד שלה:
$D_{P.pos\backslash p.dest} = \min(MD_{p.pos\to p.dest})$

ג. מרחק מנהטן בין חבילה שנאספה ליעד שלה:

ד. מרחק מנהטן של הרובוט המחזיק חבילה ליעד של החבילה:
$D_{R\backslash p.dest} = \min(MD_{R\to p.dest})$

ה. מרחק מנהטן בין הרובוט לתחנת טעינה הקרובה ביותר אליו:
$D_{R\backslash C} = \min(MD_{R\to C})$

ו. כמות יחידות הבטרייה של הרובוט: R_B

ז. כמות יחידות הניקוד של הרובוט: R_C

כעת נגדיר את הפרמטרים הבאים:

$$I = \begin{cases} 1, & robot\ hold\ a\ package \\ 0, & else \end{cases}$$

$$F = \begin{cases} 1, & 0 < robot\ has\ credit \\ 0, & else \end{cases}$$

$$G = \begin{cases} 1, & R_B > D_{R\backslash P} + D_{P.pos\backslash p.dest} \\ 0, & else \end{cases}$$

$$E = \begin{cases} 1, & R_B > D_{R\backslash P} + D_{R\backslash p.dest} \\ 0, & else \end{cases}$$

כאשר:

$$A = R_B - D_{R\backslash P}$$

$$B = R_C + R_B + D_{P.pos\backslash p.dest} - D_{R\backslash p.dest}$$

$$C = R_B - D_{R\backslash C}$$

$$\boldsymbol{H_{robot} = [A \cdot \bar{I} \cdot G + B \cdot I \cdot E + C \cdot F(\bar{I} \cdot \bar{E} + \bar{I} \cdot \bar{G})] + \alpha \cdot R_C + \beta \cdot R_B}$$

על ידי שינוי של הפרמטרים *alpha* ו- *beta*, ניתן להשפיעה על האופן שבו ישאף להתנהל במשחק. כאשר נגדיל את *beta* הרובוט ינסה להחזיק כמה שיותר זמן במשחק לפני שתגמר לו הבטרייה. אם נגדיל *alpha*, הוא ינסה לצבור ויתר נקודות במשחק.

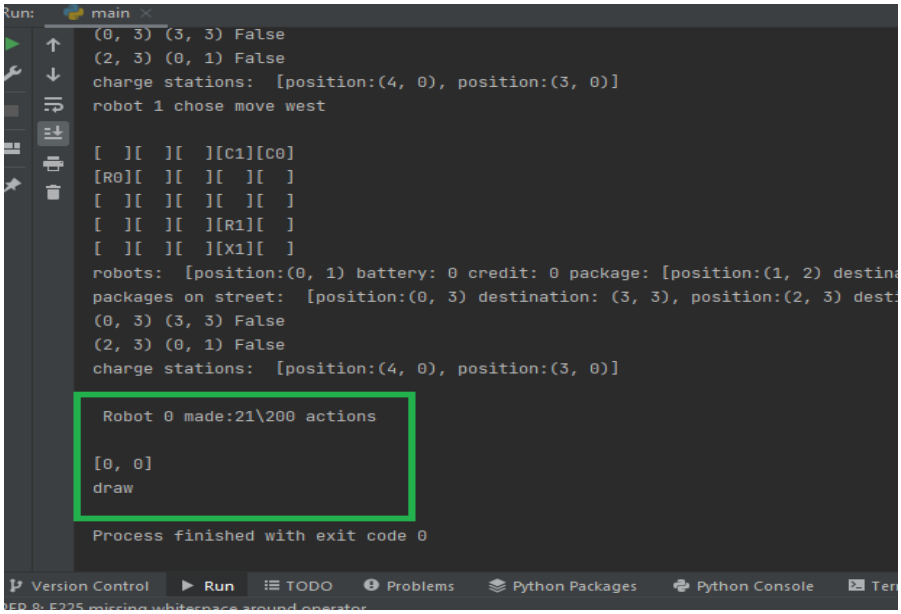
היוריסטיקה הסופית:

$$\boldsymbol{H = H_{robot_0} - H_{robot_1}}$$

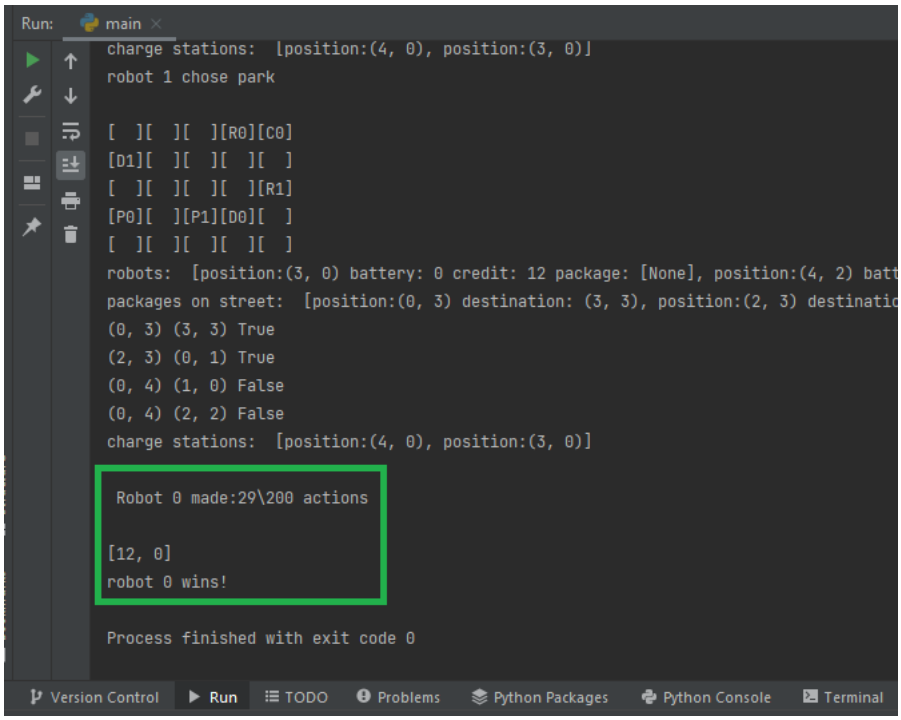
3. (רטוב: 10 נק') ממשו בקובץ py.submission את הפונקציה smart_heuristic שבה משתמש הסוכן AgentGreedyImproved.

במהלך הרצת הקוד בדקנו ערכים שונים של $\boldsymbol{\alpha}$ ו- $\boldsymbol{\beta}$ כדי לבחון את האופן שבו יתנהל הסוכן במהלך המשחק. הפרמטרים שבדקנו הם כמות הצעדים במשחק והניקוד הסופי.

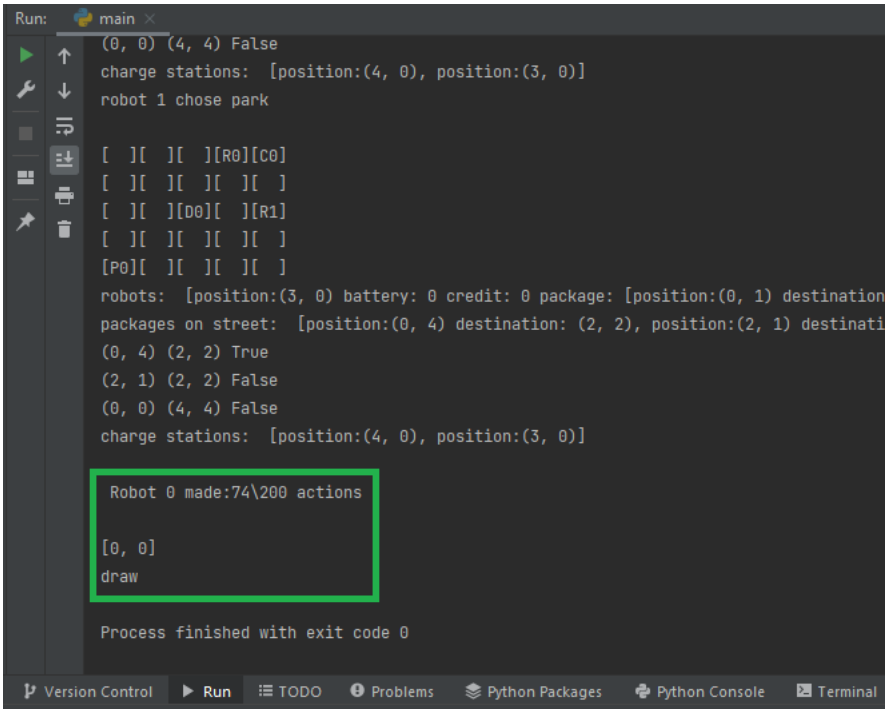
עבור ערכים של $\boldsymbol{\alpha = 0}$ ו- $\boldsymbol{\beta = 0}$: הסוכן השיג 0 נקודות והצליח לבצע 21 מהלכים במשחק



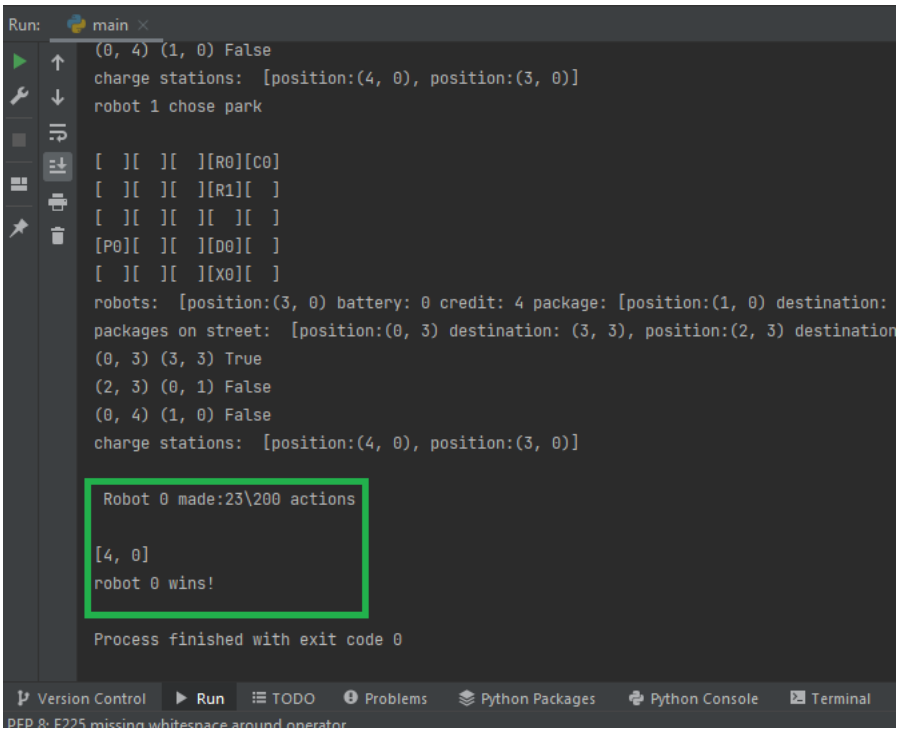
עבור ערכים של $\alpha = 2$ ו- $\beta = 2$: הסוכן השיג 12 נקודות והצליח לבצע 29 מהלכים במשחק



עבור ערכים של $\alpha = 2$ ו- $\beta = 5$: הסוכן השיג 0 נקודות והצליח לבצע 74 מהלכים במשחק



עבור ערכים של $\alpha = 5$ ו- $\beta = 2$: הסוכן השיג 4 נקודות והצליח לבצע 23 מהלכים במשחק



** מתוך התוצאות שהתקבלו בחרנו להישאר עם $\alpha = 2$ ו- $\beta = 2$

4. (יבש: 2 נק') מהו החיסרון העיקרי של האלגוריתם? (לעומת minimax)

החיסרון העיקרי של האלגוריתם הוא בעובדה שהוא איננו מסתכל על מגוון רחב של מהלכים ואופציות מתוך המהלכים הצפויים במשחק. בנוסף הוא איננו יכול להבטיח לנו שום פתרון,לעומת minimax שיכול להבטיח לנו פתרון אשר ממנו נוכל רק להשתפר.

חלק ב – RB-Minimax

1. (יבש: 3 נק') מה היתרונות והחסרונות של שימוש בהיוריסטיקה קלה לחישוב לעומת היוריסטיקה קשה לחישוב בהינתן שהיוריסטיקה הקשה לחישוב יותר מיודעת מהקלה לחישוב ? בהינתן שאנו ב max-minמוגבל משאבים.

היתרון של יוריסטיקה קשה לחישוב הוא בעובדה שנוכל לקבל תמונת מצב איכותית יותר עבור המצב של הסוכן במהלך המשחק, מה שיכול לסייע לסוכן לבצע צעדים טובים יותר. לעומת זאת יוריסטיקה קשה לחישוב תדרוש יותר משאבים ותגדיל את זמן החיפוש של הסוכן בכל שכבה, מה שיגרום לכך שנוכל להגיע לשכבות רדודות יותר לעומת יוריסטיקה קלה לחישוב.

2. (יבש: 4 נק') חברתכם לקורס דנה מימשה סוכן minimax, היא שמה לב כי לעיתים הסוכן יכול לנצח בצעד אחד אך הוא בוחר בצעד אחר. האם יש לה באג באלגוריתם ?אם אין באג הסבירו מה באלגוריתם גורם להתנהגות שכזו. אם יש באג מה הוא יכול להיות?

כאשר אנחנו מסתכלים על צעדי המשחק של אלגוריתם Minimax , לעיתים אנחנו מזהים מצב שבו הסוכן היה יכול לנצח בתור מסוים אולם הוא בחר בצעד אחר. התנהגות זו אינה באג באלגוריתם. בשיטת ה -minimax ההנחה של הסוכן שלנו היא שהיריב שמולו הוא יריב אופטימלי שתמיד יבחר בצעד שימקסם את התועלת וימזער את התועלת שלנו. ישנם פעמים שהריב שלנו אינו אופטימלי, ובמצבים אלה האסטרטגיה של הסוכן שלנו תיפגע, משום שהוא לא יבחר בצעד הפשוט שיביא אותו לניצחון כיוון שהוא מאמין שהיריב יבחר בצעד שימנע מאיתנו ניצחון. צעדים אלה יראו לנו כלא הגיוניים או כמו "באג באלגוריתם" אך זהו למעשה התנהגות צפויה מהאלגוריתם.

3. (רטוב: 10 נק') עליכם לממש את המחלקה AgentMinimax בקובץ .py.submissionשימו לב! הסוכן מוגבל משאבים, כאשר המשתנה limit_time מגביל את מספר השניות שהסוכן יכול לרוץ לפני שיחזיר תשובה.)הגבלת הזמן עלייה אתם נבדקים הינה שנייה כלומר t.1

4. (יבש: 2 נק') נניח שבסביבה היו K שחקנים במקום 2 (תחשבו על משחק כללי לוא דווקא המשחק שלנו, אך עדיין משחק סכום אפס). אילו שינויים יהיה צריך לעשות במימוש סוכן Minimax?
   a. בהינתן שכל סוכן רוצה לנצח ולא אכפת לו רק ממכם.
   b. בהנחה והדבר היחיד שכל סוכן רוצה הוא שלא תנצחו.
   c. בהנחה שכל סוכן רוצה שהסוכן שאחריו בתור ינצח.

בסעיף זה נדגים את כל אחת מהאופציות באמצעות החלק העיקרי של הפסודו-קוד, הדומה לזה שהוצג בתירגול. בהנחה שבסביבה יהיו k שחקנים במקום 2, נצטרך לחלק את האלגוריתם ל k- שלבים כאשר $Turn = 0$ מייצג את הסוכן שלנו, ו- $Turn \neq 0$ מייצג את שאר הסוכנים.

a) כשהתור הוא לא של הסוכן שלנו נניח היריב יבחר את הפעולה הכי טובה עבורו לפי יוריסטיקה

```
For Turn in length(k):
        If Turn = 0 then:
                CurMax←  - ∞
                Loop for c in Children
                        v ← Minimax(c, Agent=0)
                        CurMax ← Max(v, CurMax)
                Return(CurMax)
        else: (Turn ≠ 0)
                CurMax←  0
                Loop for c in Children:
                        v ← heuristic (c, Agent= Turn)
                        CurMax ← Min(v, CurMax)
        Return(CurMax)
```

b)

כשהתור הוא לא של הסוכן שלנו נניח שכל יריב ינסה למזער את התועלת של הסוכן שלנו:

```
For Turn in length(k):
        If Turn = 0 then:
                CurMax←  - ∞
                Loop for c in Children
                        v ← Minimax(c, Agent=0)
                        CurMax ← Max(v, CurMax)
                Return(CurMax)
        else: (Turn ≠ 0)
                CurMin←  ∞
                Loop for c in Children:
                        v ← Minimax(c, Agent=0)
                        CurMin ← Min(v, CurMin)
        Return(CurMin)
```

c) כשהתור הוא לא של הסוכן שלנו נניח היריב יבחר את הפעולה הכי טובה עבור הסוכן שמשחק אחריו לפי יוריסטיקה.

```
For Turn in length(k):
        If Turn = 0 then:
                CurMax←  - ∞
                Loop for c in Children
```

```
                v ← Minimax(c, Agent=0)
                CurMax ← Max(v, CurMax)
        Return(CurMax)
else: (Turn ≠ 0)

        if Turn<k:   NextTurn= Turn+1
        else:    NextTurn=0

        CurMax←  0
        Loop for c in Children:
                v ← heuristic (c, Agent= NextTurn)
                CurMax ← Min(v, CurMax)
Return(CurMax)
```

חלק ג – Alpha-Beta

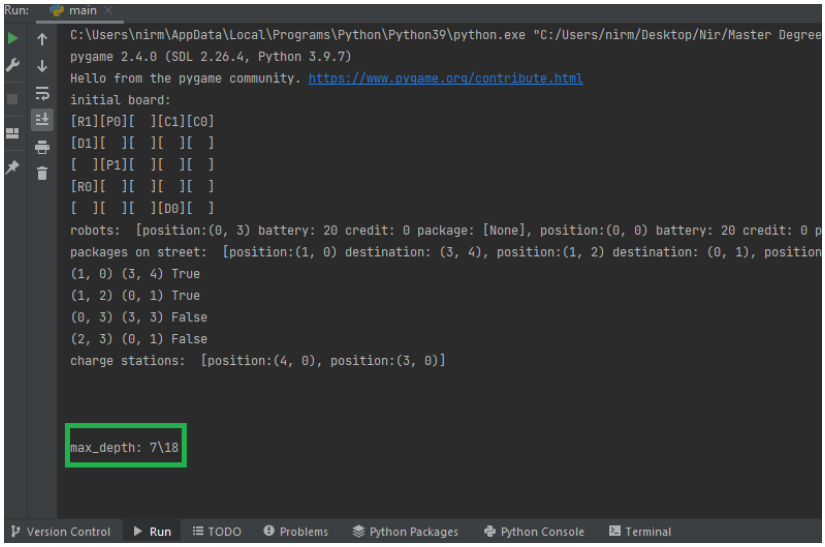
1. (יבש: 10 נק') ממשו שחקן אלפא - בטא מוגבל משאבים במחלקה AgentAlphaBeta בקובץ py.submission, כך שיתבצע גיזום כפי שנלמד בהרצאות ובתרגולים.

2. (יבש: 3 נק') האם הסוכן שמימשתם בחלק זה יתנהג שונה מהסוכן שמימשתם בחלק ב מבחינת זמן ריצה ובחירת מהלכים ? הסבירו.

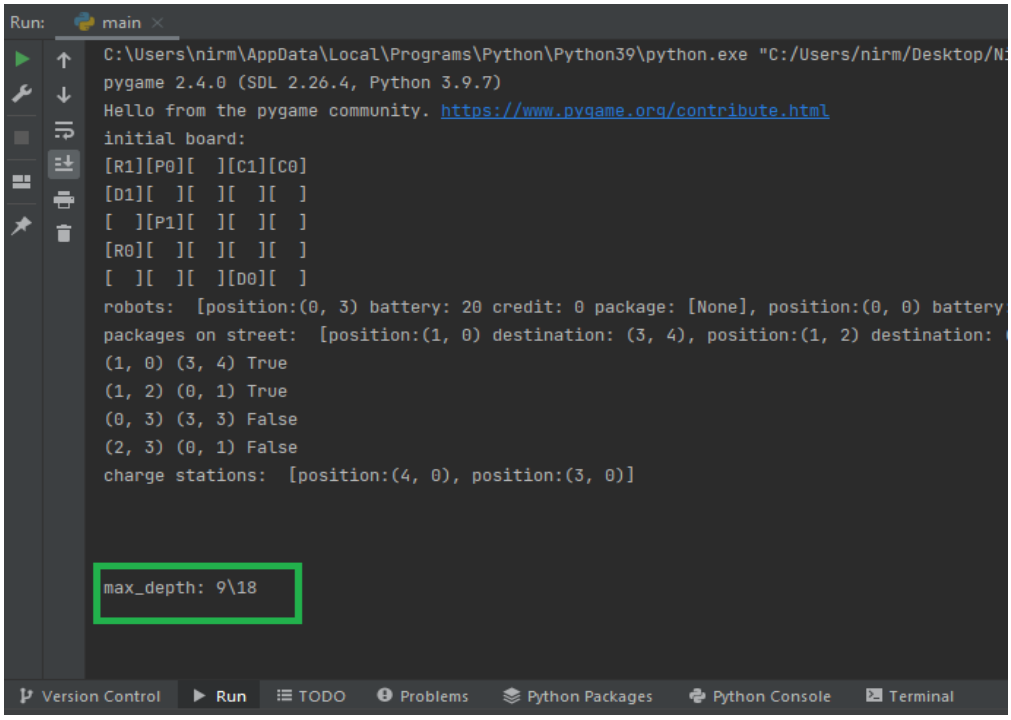
ברמת העיקרון הפתרון שנקבל מסוכן AlphaBeta אמור להיות זהה לאיכות הפתרון שנקבל מסוכן Minimax, מכיוון שגיזום הענפים נעשה רק כאשר מפתחים ענפים עם ערך שלא ישפיע על הבחירה הסופית של סוכן. בפועל זמן הריצה מתקצר כי יש פחות ענפים לחשב וניתן להגיע לשכבות עמוקות יותר בעץ החיפוש, אשר יאפשרו לסוכן להסתכל על יותר מהלכים קדימה במשחק.

מתוך ניסויי וטעיה מהרצת הסוכנים minimax ו- alpha-beta, נראה שברוב הפעמים alpha-beta השיג ניקוד גבוה יותר, והצליח לפתח יותר שכבות:

סוכן minimax: עבור המהלך הראשון במשחק הצליח לעבור על 7 שכבות מתוך 18 אפשריות:



סוכן alpha-beta: עבור המהלך הראשון במשחק הצליח לעבור על 9 שכבות מתוך 18 אפשריות:



חלק ד – Expectimax

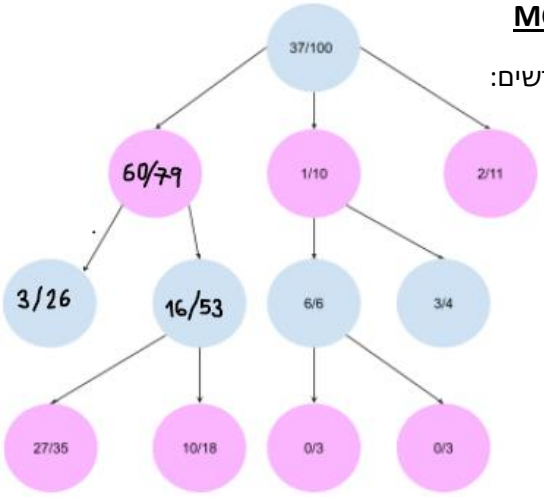
1. ההסתברות שנשתמש בה היא הסתברות אחידה.
הסיבה לכך היא שהסוכן שאנחנו משחקים נגדו משחק באופן רנדומלי לחלוטין ולכן לכל פעולה שייבחר יש את אותה ההסתברות להבחר בתור הפעולה הבאה.
2. היוריסטיקה חסומה על ידי 1 ו-(1-) ולכן במידה והערך שייתקבל באחד הבנים של השחקן הוא 1, נוכל ישר לבצע גיזום ולהסיר את שאר הבנים שכן בהכרח לא ייתקבל ערך גדל יותר מ-1.
3. **רטוב!**

חלק ה – משחק עם פקטור סיעוף גדול

1. נציין את ההשפעה עבור כל אחד מהשינויים:
   א. כאשר נגדיל את לוח המשחק ונוסיף מחסומים אזי בכל תור הסוכן יוכל לבצע בדיוק את אותן הפעולות כל עוד המשבצת אינה חסומה, הגדלת הלוח אינה משפיעה על מקדם הסיעוף שכן הסוכן מתקדם משבצת אחת לכל היותר בכל תור.
   ב. (בהתאם למה שנאמר בפיאצה) כל אובייקט יושב על משבצת נפרדת ולכן ישנן 2 משבצות טעינה, 2 לפריקה ו-2 רובוטים עם חבילה.
   לכן קיבלנו כי יש 6 משבצות תפוסות, כלומר ישנן 19 פעולות חדשות במקרה הקיצוני ולכן מקדם הסיעוף החדש הוא 26.
2. נניח כי השינוי מסעיף 1ב מומש:
   א. נוכל להשתמש באלגוריתם גרידי, מכיוון שנוספו עוד מצבים אפשריים אזי גרידי יוכל לבחור את הטובים ביותר בזמן היעיל ביותר.
   ב. ניתן להשתמש באלגוריצם מונטה קרלו הנלמד בקורס.
   בחרנו באלגוריתם זה שכן הוא נועד עבור פתירת בעיה שהאינפורמציה בה היא חלקית! ולכן הוא יוכל למצוא צעד כלשהו של היריב על כל צעד שלי במשחק.
   מכיוון שכל מה שנוצר בעקבות השינוי בסביבה זה עוד אפשרויות לפעולה בתור, אזי האלגוריתם יוכל למצוא פתרון שכן כל מה שזה אומר זה שליריב יש עוד אפשרויות בהנתן תור שלי.

חלק ו – MCTS

1. התרשים:



2. נחשב מי ייבחר הבא:

$$next(a) = \frac{60}{79} + \sqrt{\frac{2\ln(100)}{79}} = 1.101$$

$$next(b) = \frac{1}{10} + \sqrt{\frac{2\ln(100)}{10}} = 1.060$$

$$next(c) = \frac{2}{11} + \sqrt{\frac{2\ln(100)}{11}} = 1.097$$

מתקיים כי המקסימלי הוא a ולכן מי שייבחר הוא צאצא של a.

3. נקבל כי (כאשר מספר הנצחונות הנדרש הוא x):

$$next(a) = \frac{60}{79+x} + \sqrt{\frac{2\ln(100+x)}{79+x}} < next(b\ or\ c) = \max\{\frac{2}{11} + \sqrt{\frac{2\ln(100+x)}{11}}, \frac{1}{10} + \sqrt{\frac{2\ln(100+x)}{10}}\}$$

אי השיוויון מתקיים כבר עבור 1 ולכן מספר הנצחונות המינימלי הוא 1

4. בכדי לקיים את התנאים החדשים, נחזיר את N(s) ככפולה בקבוע גדול מספיק.
דבר זה יגרור העדפה ברורה של ה-exploration על ה-exploitation מכיוון שהראשון נמצא בתוך שורש והשני לא, ולכן הגדלת המחלק תגרור הקטנה של שני הערכים, אך מכיוון שהראשון בתוך שורש אזי הוא יחולק במספר קטן יותר ולכן עבור מספר גדול דיו יהיה העדפה ברורה בסדר גודל של 1.
(לדוגמא אם נבחר 144 בתור המספר, נקבל כי exploitation חולק ב-144 בעוד exploration חולק ב-12)