חלק יבש

חלק א: -(S,O,I,G)

1. נגדיר את קבוצת המצבים: להיות כל מטריצה בגודל 5X5 בה יש שני רובוטים, שתי תחנות דלק , שתי חבילות ושני יעדים (יכולים להיות באותה משבצת).

אופרטורים : move north, move south move east, move west, pick up , drop off, charge,

מצב ההתחלה: הוא המצב ההתחלתי שנתון

מצב מטרה : מצב שבו המשחק מסתיים (בגלל שהרובוטים הגיעו למספר צעדים מקסימלי או נגמרה הסוללה).

2. חילקנו למקרים:
   1) אם המצב הוא מצב סופי אז נבדוק האם במצב זה אנחנו מנצחים נחזיר אינסוף כי זה מה שרוצים שיקרה בסוף, אם אנחנו מפסידים, נחזיר -אינסוף כי מצב זה לא טוב, אחרת נחזיר 0 אם זה תיקו.
   2) אם זה לא מצב סופי אז גם נחלק למקרים
      A. אם הרובוט מחזיק חבילה :
      אז הוא יפסיד סוללה כמספר הצעדים בינו לבין היעד ואז הוא מרוויח מולם נקודות כמספר הצעדים בין החבילה ליעד שלה כפול 2
      אז נחזיר את הנקודות שלו(כפול קבוע גדול מאוד בכדי לשמור על יוריסטיקה חיובית) פחות מספר הנקודות שהוא הפסיד ועוד מספר הנקודות שהוא הרוויח.
      B. אם הרובוט לא מחזיק חבילה :
      אז הוא צריך לחפש חבילה להחזיק , אז מבין שתי החבילות הנמצאות על הלוח הוא יסתכל על החבילה הקרובה ביותר במונחים של מרחק מנהטן והוא יחזיר את הנקודות שלו (כפול קבוע גדול) פחות המרחק בינו לבין החבילה כי זה הוא מספר הצעדים שהוא יפסיד .

3. בgreedy לא מתחשבים בפעולת הסוכן היריב לעומת אלגוריתם MINIMAX שמתחשב בפעולות הטובות ביותר של הסוכן היריב.

   החיסרון של greedy לעומת minimax זה ש-greedy בוחר במצב אופטימלי עוקב לוקאלי (כלומר הכי טוב מבין הבנים העוקבים ללא בדיקת שאר עץ המשחק) לעומת minimax שלוקח מצב "אופטימלי" גלובלי (בודק את כל עץ המשחק עד לעומק כלשהו) ומתחשב בפעולות היריב, אלגוריתם greedy בוחר מצב לפי התועלת המיידית שלו ללא התחשבות בפעולות העתידיות של הסוכן היריב, ולכן הוא עלול לדלג על פעולות שיהיו יותר טובות בטווח הארוך.

חלק ב:

1. יוריסטיקה קלה לחישוב אינה הכי מיודעת ועשויה להוביל לפתרון שאינו הכי אופטימלי. לעומת זאת, יוריסטיקה קשה לחישוב יותר מיודעת ולכן מספקת הערכה יותר טובה ומדויקת למצב המשחק.
מצד שני, יוריסטיקה קשה לחישוב צריכה יותר זמן ויותר משאבים כדי לחשב אותה, ולכן בניגוד ליוריסטיקה קלה לחישוב, במיוחד ב-minimax מוגבל משאבים, יוריסטיקה קשה לחישוב עשויה להשפיע על הסוכן כך שהוא לא יצליח לבדוק את כל האפשרויות לפני שהזמן יגמר, לעומת יוריסטיקה הקלה לחישוב שבעזרתה נוכל לבדוק יותר צמתים תוך אותה הגבלת זמן ולכן בודקים יותר אפשרויות שעלולות להוביל לפתרונות יותר אופטימליים.

2. זה לא בהכרח באג, ייתכן אחד מכמה דברים:
   a. היוריסטיקה שהיא מימשה אינה מיודעת מספיק כדי להוביל למצב הכי אופטימלי.
   b. הצעד המנצח יכול להיות במסלול שבו היריב לא בוחר בצעד הכי טוב עבורו, כיוון שאלגוריתם minimax מניח שהיריב תמיד יבחר בצעד הכי טוב עבורו ומתחשב בפעולת היריב, לכן ייתכן שהוא יבחר בצעד שאינו הכי טוב עבורי פשוט כי הוא אינו הכי טוב עבור היריב שלי.
3. רטוב
4. בסביבה של K שחקנים:
   a. בהינתן שכל סוכן רוצה לנצח ולא אכפת לא אף אחד אחר, כל שחקן בעץ המשחק נחשב כשחקן maximum והוא ינסה תמיד למקסם את עצמו, ללא תלות באחרים.
   b. בהינתן שכל השחקנים רוצים שאני אפסיד, כל שחקן בעץ המשחק רוצה לבחור המinimum שלי, כלומר כל שחקן משחק כשחקן minimum והסוכן שלנו רוצה לבחור המaximum מהם.
   c. כל שחקן רוצה שהשחקן אחריו ינציח ולא אכפת לו משאר השחקנים, כלומר הוא פשוט עובר על הילדים שלו בעץ המשחק (שהם מהווים התור של השחקן אחריו), והוא בוחר הפעולה המובילה לmaximum ביניהם, זה כמו סעיף א אך במקום למקסם את עצמי אני ממקסם את השחקן אחרי.

חלק ג:

1. רטוב
2. כן, אלגוריתם RB-alpha_beta מתנהג שונה מאלגוריתם RB-minimax, זה נובע מזה שהאלגוריתמים הינם מוגבלי משאבים, כיוון שאלגוריתם RB-alpha_beta גוזם ענפים לכן הוא ירוץ על הבנים של צומת בזמן יותר קצר מRB-minimax ולכן הוא יספיק לבדוק יותר אופציות ואפשרויות תוך אותו הזמן, ולכן הוא נחשב יותר מיודע וירוץ בצורה שונה ויותר טובה מאשר RB-minimax.

חלק ד :

1. 1/7 : הסוכן בוחר באופן רנדומלי מבין 7 אופרטורים לכן ההסתברות לבחור כל אחד מביניהם היא 1\7
2. כמו האלגוריתם של αβ, אנחנו גוזמים את הענפים שבהם מתקיים v<currMin או v>curMin ולכן, נעשה אותו רעיון, רק עם α=1 קבוע(עם הסוכן המקסימלי בלבד, ללא הרנדומלי) , ברגע שמגיעים לצומת בן עבורה h(s)=1 ואנחנו במשחק maximum, אין צורך להמשיך לשאר הבנים כי לא נמצא ערך יוריסטי יותר גדול, ולכן גוזמים את כל שאר הבנים

חלק ה :

1.a מקדם הסיעוף לא משתנה, ייתכנו צמתים שבהם מספר הפעולות אפילו יקטן, אבל זה לא משנה מקדם הסיעוף כיוון שהאופרטורים לא משתנים, כיוון שאי אפשר להחזיק שתי חבילות יחד, אז או שיש לו חבילה והוא יכול לעשות drop off ולא יכול לעשות pick up או שאין לו חבילה והוא יכול לעשות pick up ולא יכול לעשות drop off , וכיוון (לפי פיאצה) שכל אובייקט נמצא במשבצת שונה לכן אי-אפשר להטעין וגם להרים\הוריד חבילה באותה משבצת לכן מקדם הסיעוף הוא 5.

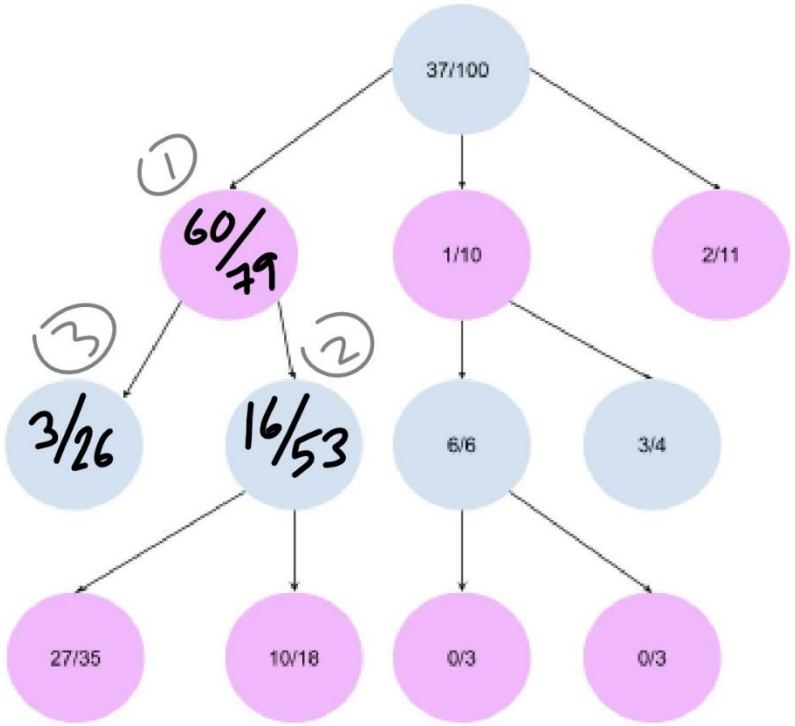
1.b מקדם הסיעוף יגדל ב23 כי הוספנו פעולה נוספת שהיא חוקית עבור כל משבצת במשחק חוץ משתי המשבצות שעליהם עומדים הרובוטים. ולכן המקדם החדש הינו 28=5+23 (נניח כי אפשר לשים חסם על כל המשבצות גם אם הם מכילים משהו חוץ משני הרובוטים).

2.a כיוון שמספר הצעדים המקסימלי מוגבל מראש עבור כל משחק שזה הערך המקסימלי של העץ ומקדם הסיעוף גדל ל 28 לכן לפי מה שאמרנו האלגוריתם ירוצו ב $O(b^d)$ =$O(28^d)$ כל האלגוריתמים הם לא סבירים יחסית כי מקדם הסיעוף גדל הרבה, אבל אלגוריתם greedy הינו הכי טוב מביניהם (גם הוא לא סביר אבל הוא האופציה הכי טובה), למקרה זה כי הוא ייבחר בצעד הבא תוך הזמן הכי קצר (באינסוף זה לא משנה אבל נחשוב על זה בזמן אמת במקום O), כי מספר הבנים בכל שכבה חסום על ידי קבוע ולפיכך האלגוריתם יהיה תלוי רק בעומק עץ החיפוש.

2.b אנחנו נאחד את שני האלגוריתמים של monte carlo ו- minimax כך שבמצב מסוים בעץ שבו מספר הילדים גדול מאוד, אנחנו נשתמש בגרסת monte carlo של התרגול, כך שנבחר k בנים ונחשב בתוחלת מי ה-max וה- min ונבחר את הערך בהתאם ל Minimax

חלק ו:

1. תשובה סופית:



חישובי עזר:

$$1:\ \frac{100-37-(1+2)}{100-(10+11)}=\frac{60}{79}$$

$$2:\ \frac{(35+18)-(10+27)}{35+18}=\frac{16}{53}$$

$$3:\ \frac{79-60-16}{79-53}=\frac{3}{26}$$

2. נחשב את ה-UCB1 של כל צומת בן ובוחרים בצומת בעלת הערך הגבוה ביותר:

ברמה הראשונה:

בן ימני: $\frac{2}{11}+\sqrt{2}\sqrt{\frac{\ln(100)}{11}}=1.09$

בן שמאלי: $\frac{60}{79}+\sqrt{2}\sqrt{\frac{\ln(100)}{79}}=1.1$

בן אמצעי: $\frac{1}{10}+\sqrt{2}\sqrt{\frac{\ln(100)}{10}}=1.06$

האלגוריתם יבחר בבן השמאלי בשלב הראשון.

ברמה השנייה:

בן ימני: $\frac{16}{53}+\sqrt{2}\sqrt{\frac{\ln(79)}{53}}=0.71$

בן שמאלי: $\frac{3}{26}+\sqrt{2}\sqrt{\frac{\ln(79)}{26}}=0.69$

האלגוריתם יבחר בבן הימני בשלב השני.

ברמה השלישית:

בן ימני: $\frac{10}{18}+\sqrt{2}\sqrt{\frac{\ln(53)}{53}}=1.21$

בן שמאלי: $\frac{27}{35}+\sqrt{2}\sqrt{\frac{\ln(53)}{53}}=1.24$

האלגוריתם יבחר בבן השמאלי בשלב השלישי.

3. נרצה שצומת *b* או *c* יבחרו בשלב הראשון במקום *a*:

$$UCB1(a)<UCB1(b)\rightarrow\frac{60+x}{79+x}+\sqrt{2}\sqrt{\frac{\ln(100+x)}{79+x}}<\frac{2}{11}+\sqrt{2}\sqrt{\frac{\ln(100+x)}{11}}$$

$$UCB1(a)<UCB1(b)\rightarrow\frac{60}{79}+\sqrt{2}\sqrt{\frac{\ln(100+x)}{79}}<\frac{2+x}{11+x}+\sqrt{2}\sqrt{\frac{\ln(100+x)}{11+x}}$$

נקבל $x\geq 1$ (המינימלי מבניהם)

$$UCB1(a)<UCB1(c)\rightarrow\frac{60+x}{79+x}+\sqrt{2}\sqrt{\frac{\ln(100+x)}{79+x}}<\frac{1}{10}+\sqrt{2}\sqrt{\frac{\ln(100+x)}{10}}$$

$$UCB1(a)<UCB1(c)\rightarrow\frac{60}{79}+\sqrt{2}\sqrt{\frac{\ln(100+x)}{79}}<\frac{1+x}{10+x}+\sqrt{2}\sqrt{\frac{\ln(100+x)}{10+x}}$$

נקבל $x\geq 2$ (המינימלי מבניהם)

כלומר המספר המינימלי מתקיים עבור המספר הטבעי השלם הקטן ביותר המקיים אחד מהנ"ל

$x=1$ ➔

4. נרצה להעדיף *exploration* על *exploitation*, כלומר:

$$\frac{exploration}{exploitation}=\frac{\sqrt{2}\sqrt{\frac{\ln\left(N(parent)\right)}{N(node)}}}{\frac{U(node)}{N(node)}}=\frac{\sqrt{2}}{U(node)}\sqrt{\ln\left(N(parent)\right)*N(node)}$$

יש לנו גישה ל-*N(node)* ואופציה לשנותו בנוסחה, לכן אם נוסיף לו או נכפיל אותו ב-$\alpha>1$ אז הנוסחה החדשה תעדיף *exploration* על *exploitation* יותר מהנוסחה הקודמת.