

תרגיל בית 2 - חיפוש רב סוכני

Can't Go Back

מטרת התרגיל

בתרגיל זה נממש סוכני חיפוש בסביבה מרובת סוכנים. נממש את הסוכן על פי אלגוריתם *Minimax* ושיפוריו עבור המשחק Can't Go Back. נתנסה במימוש היוריסטיקה עבור המשחק, ביצוע ניסויים וכתובת דו"ח.

הערות

- תאריך הגשה: יום שני, 31.12.2020, עד השעה 23:59.
- יש להגיש את המטלה בזוגות בלבד.
- יש להגיש דוחות מוקלדים בלבד.
- המתרגל אחראי על התרגיל: סער הוברמן.
- ניתן לשלוח שאלות בנוגע לתרגיל לתיבת המייל בכתובת: saarhuberman@cs.technion.ac.il, לא לפני שבדקתם האם קיימת תשובה לשאלה ב-FAQ באתר הקורס.
- ניתן לשאול שאלות על התרגיל בכיתה ההפוכה ביום חמישי בשעה 14:30.
- בקשות דחיה מוצדקות יש לשלוח למתרגל האחראי של הקורס – טל סויסה.
- ייתכן שנגעלה עדכונים והבהרות לדף FAQ, העדכונים מחייבים.
- העתקות תטופלנה בחומרה.

תיאור המשחק

Can't Go Back הוא משחק לוח לשני שחקנים. המשחק מתחיל כאשר לכל שחקן מיקום התחלתי על לוח מלבני המורכב ממשבצות אפורות ומשבצות לבנות. משבצות אפורות הן משבצות שאסור לנוע אליהן. כל שחקן בתורו עובר למשבצת לבנה לפי הצעדים החוקיים במשחק, וזו הופכת לאפורה, כלומר אף שחקן לא יכול לדרוך עליה שוב.

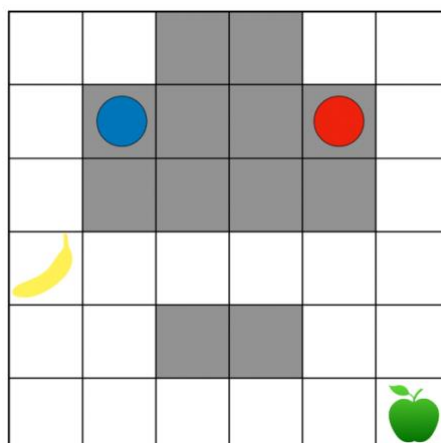
על חלק מהמשבצות הלבנות ישנם פרסים, בעלי ערך משתנה. כל תור מתעדכנים מיקומי הפרסים, משך השארות בתורות של פרס נבחר יהיה אורך הצלע הקטנה.

אם בתורו שחקן לא יכול לנוע לאף משבצת, המשחק נגמר, ולאותו השחקן מקבל ניקוד שלילי של שמוגדר בתחילת המשחק, אלא אם כן גם השחקן השני לא יכול לנוע באותו תור, ואז לא יורד הניקוד.

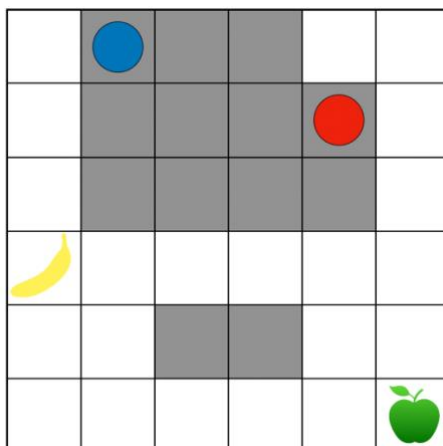
מטרת המשחק לקבל ניקוד גבוה יותר מהמתחרים שלך. גם שחקן שנפסל קודם יכול לנצח (אם שחקן הוביל ב-500 נקודות, ומספר הנקודות ששורדות על פסילה מוקדמת היא 400 אז אחרי שנפסל עדיין מוביל ב-100 נקודות).

המהלכים החוקיים בגרסת המשחק שלנו הם למעלה, למטה, שמאלה וימינה.

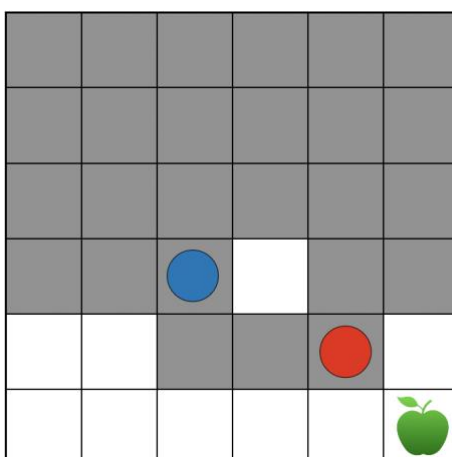
לדוגמה, יתכן שהלוח ההתחלתי יהיה הלוח הבא:



הנקודות הכחולה והאדומה מייצגות את מיקום השחקנים על הלוח והפירות מייצגים את מיקום הפרסים. השחקן הכחול הוא השחקן שמשחק ראשון, והוא יכול לנוע שמאלה או למעלה. לאחר שהוא ינוע למעלה הלוח יראה כך:



המשבצת שהוא נע אליה הפכה לאפורה, ואי אפשר יהיה לדרוך עליה שוב. בהמשך המשחק יתכן והמשחק יגיע למצב הבא, כאשר תורו של השחקן הכחול לנוע.



השחקן הכחול חייב לנוע ימינה, ולאחר מכן השחקן האדום יוכל לנוע למטה או ימינה. לאחר שהשחקן האדום יבצע את תורו, השחקן הכחול לא יוכל לנוע, והמשחק יסתיים.

הרצת משחקים

לפני תחילת מימוש הסוכנים, מומלץ להתנסות במשחק. הריצו את השורות הבאות ב-terminal בתיקייה בה נמצא הקוד המצורף לתרגיל:

כדי לשחק בתור שני הסוכנים, הריצו את השורה:

```
python main.py -player1 LivePlayer -player2 LivePlayer
```

לאחר הרצת השורות תפתח אנימציה המציגה את מצב המשחק. כדי לשלוט ב-LivePlayer, נשתמש במקשים w (למעלה), s (למטה), a (שמאלה), ו-d (ימינה). **יש להקליד את התו ב-terminal**, ולאחר מכן להקיש Enter. במידה והצעד שבחרתם לא חוקי (מחוץ ללוח, אל משבצת אפורה, או לא אחד מהתווים החוקיים: w, s, a, d), תוכלו לבחור שוב צעד.

כדי לשחק נגד שחקן SimplePlayer (שחקן שהמימוש שלו נמצא בקבצי הקוד שקיבלתם) הריצו:

```
python main.py -player1 LivePlayer -player2 SimplePlayer
```

ניתן להריץ את המשחקים עם הדפסות בטרמינל במקום אנימציה על ידי הרצת השורה:

```
python main.py -player1 LivePlayer -player2 SimplePlayer -terminal_viz
```

בשביל להריץ בלי הדפסות של המשחק (תודפס רק הכרזת המנצח בסוף המשחק), יש להוסיף את הדגל -don't_print_game.

בהמשך נממש את השחקנים MinimaxPlayer, AlphaBetaPlayer ועוד נוספים. כדי להריץ משחק בין שני שחקנים כלשהם הריצו שורה דומה עם השמות של השחקנים המתאימים.

ישנם שני דוגמאות ללוחות בתיקיה boards, המיוצגים כקבצי csv, ניתן להוסיף לוחות נוספים על ידי הוספת קבצים נוספים לתיקייה. אתם יכולים לבחור לוח על ידי הוספת הדגל הבא לשורת ההרצה board_name.csv, כאשר board_name.csv הוא שם הקובץ המופיע בתיקייה boards.

בנוסף אתם יכולים להגדיר:

- מגבלה על זמן מהלך על ידי הדגל -move_time
- הגבלת זמן גלובלי לכל תורות השחקן על ידי הדגל -game_time
- להגדיר את כמות הורדת הנקודות על שחקן שנפסל ראשון על ידי הדגל -penalty_score
- להגדיר את השווי המקסימלי לפרי על ידי הדגל -max_fruit_score (השווי המינימלי לפרי הוא 3)
- להגדיר את מספר התורות המקסימלי שפרי יהיה על הלוח על ידי הדגל -max_fruit_time (מספר התורות המינימלי מוגדר להיות כאורך הצלע הקצרה בלוח).

חלק א' - חלק יבש (45 נק')

בחלק זה עליכם לענות על שאלות, ולצרף את התשובות לדו"ח.

שאלות על השחקן SimplePlayer:

בקובץ SimplePlayer.py שמצורף לתרגיל ישנו מימוש של השחקן. השאלות הבאות נוגעות לאותו שחקן, לכן לפני שעונים על השאלות יש קודם לקרוא את הקוד שלו ומומלץ לנסות מספר הרצות:

1. (2 נק') מהי האסטרטגיה של השחקן?

מנו מספר יתרונות וחסרונות לאסטרטגיה זו.

2. (2 נק') הציגו לוח שבו השחקן SimplePlayer יפעל בצורה אופטימלית. והסבירו.

(אין להציג לוח בו בכל תור ישנו כיוון אחד אפשרי בלבד).

שאלות על הגדרת היוריסטיקה:

3. (2 נק') נסתכל על היוריסטיקה הבאה עבור שחקן משופר:

היוריסטיקה $h(s) = \frac{1}{\min_{\text{fruit}} \{md(s, \text{fruit})\}}$, הערך היוריסטי של מצב הוא 1 חלקי מרחק מנהטן המינימלי לפרי בלוח.

מה החסרונות והיתרונות ביוריסטיקה h ?

4. (3 נק') הגדירו היוריסטיקה המורכבת לפחות מארבעה מרכיבים שונים (רצוי למצוא ולהשתמש ביותר), הציגו בצורה פורמלית את ההגדרה במסמך שלכם. הסבירו את המוטיבציה שלכם להגדרה היוריסטיקה, והסבר קצר על כל אחד מהפרמטרים שהגדרתם בהיוריסטיקה. למה אתם צופים שהיא תשפר את ביצועי השחקן לעומת השחקן `simplePlayer`?

שאלות על Minimax ו Alpha-beta:

5. (3 נק') האם אסטרטגיית Minimax מתאימה עבור משחק של יותר משני שחקנים?

א. מנו חסרונות לאסטרטגיה במקרה זה.

ב. הציעו אסטרטגיה חלופית. ניקוד מלא יינתן עבור פתרון יצירתי שמוסבר היטב.

בשאלות הבאות נחזור לדבר על משחק של שני שחקנים בלבד.

6. (2 נק') עבור עומק חיפוש זהה, האם סוכן ה Alpha-beta יתנהג שונה מסוכן ה Minimax:

א. מבחינת זמן ריצה?

ב. מבחינת ערך ה Minimax?

7. (3 נק') האם Alpha-beta יתנהג בהכרח אותו דבר כמו Alpha-beta עם סידור ילדים?

א. מבחינת זמן ריצה

ב. מבחינת בחירת מהלכים.

שאלות על Minimax בווריאציות Anytime contract:

8. (2 נק') הסבירו מהי ווריאצית Anytime contract של אלגוריתם Minimax ומהי העמקה הדרגתית בהקשר זה.

9. (2 נק') מה הבעיה בהעמקה הדרגתית המוצגת בהרצאה? הסבירו את הפתרון המוצע בהרצאה לבעיה זו.

שאלות על הגבלת זמן גלובלית ואפקט האופק:

10. (4 נק') בשאלה זו נניח כי במקום זמן מוגבל לחישוב הצעד בכל תור, נתון מראש זמן מוגבל עבור סכום זמני חישוב הצעדים של השחקן.

זהו תיאור של אלגוריתם נאיבי:

1. חשב חסם עליון למספר התורות $num\ turns = \left\lceil \frac{\text{number of available slots}}{2} \right\rceil$ (משבצות לבנות או עם פרי)

2. הקצה לכל תור את מסגרת הזמן $time\ fram = \frac{\text{global time}}{num\ turns}$

תארו אלגוריתם מחוכם יותר לניהול משאבי הזמן. (ציון יינתן על סמך יצירתיות)

11. (2 נק') מהו אפקט האופק? הסבירו את הפתרון המוצע בהרצאה לבעיה זו. באיזה מקרים שילוב של פתרון זה יוכל להועיל לסוכן שלכם? הציגו שני מצבים ספציפיים בלוח כלשהו בהם יהיה כדאי להשתמש בפתרון זה.

שאלות על RB-Expectimax:

12. (4 נק') בהרצה של $\alpha - \beta$ המתכנת מחק בטעות את הצעד המומלץ ונשאר רק עם ערך המינימקס של העץ. כיצד ניתן לשפר את יעילות ההרצה החוזרת באמצעות שינויים פשוטים באלגוריתם? תארו את התנהגות האלגוריתם המתוקן. הסבירו כיצד יתנהג במקרה הטוב ביותר, במקרה הרע ביותר, ובמקרה הכללי.

13. (7 נק') עבור משחקים הסתברותיים כמו שש בש, בהם יש מגבלת משאבים, משתמשים באלגוריתם RB-Expectimax. הניחו כי ידוע שהפונקציה היוריסטית h באלגוריתם RB-Expectimax מקיימת $-5 \leq h(s) \leq 5$. איך ניתן לבצע גיזום לאלגוריתם זה? תארו בצורה מפורטת את תנאי הגזימה, והסבירו את הרעיון מאחוריו.

14. (7 נק') במשחק (ללא הסתברויות) יש מקדם סיעוף קבוע B . הניחו את ההנחות הבאות:
- מוקצות M דקות לכל B (אותו B) מהלכים.
 - ב- M/B דקות ניתן לחפש לעומק D .
 - זמן החיפוש הוא פונקציה של מספר הקריאות לפונקציה היוריסטית (משך הזמן של כל שאר הפעולות זניח)
 - השחקן מממש Minimax (ללא גיזום)
 - השחקן הנמצא בצעד הראשון מתוך B מהלכים, מחליט לחפש בצעד זה לעומק $D+1$ (חיפוש מלא)
- א. איך השפיעה החלטתו על שאר $B-1$ החלטותיו בהמשך?
- ב. השחקן החליט לשפר את המצב ע"י שמירת האסטרטגיה בצעד הראשון ושימוש בה לאחר מכן.
- i. כמה קשתות שמייצגות פעולות של השחקן עליו לשמור, כמה קשתות שמייצגות פעולות של היריב עליו לשמור?
- ii. במה שיפר את המצב? (בהשוואה לסעיף א')
- iii. השוו את התנהגות האלגוריתם המשופר לזו של שחקן Minimax רגיל שמחפש לעומק D תמיד, תארו מתי שחקן זה יהיה טוב משחקן Minimax רגיל, ומתי יהיה פחות טוב.

חלק ב' – מימוש סוכן Minimax (15 נק')

בחלק זה נבנה סוכן Minimax מוגבל משאבים בווריאצית Anytime contract.

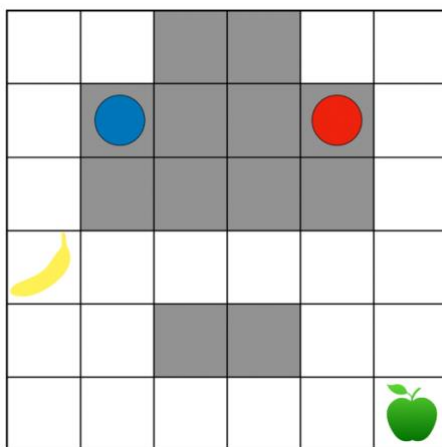
את השחקן עליכם לממש בקובץ `MinimaxPlayer.py`, במחלקה ששמה `Player`. על השחקן שתבנו יש לממש את הפונקציות `set_game_params`, `make_move`, `set_rival_move` ו-`update_fruit`. הפונקציה `__init__` מקבלת פרמטר `game_time` בו אין שימוש בשחקן זה, לכן יש להתעלם מערך זה, ופרמטר `penalty_score` שבו מצוין את כמות הנקודות היורדת לשחקן אשר נפסל ראשון. לממש את אלגוריתם ה-Minimax בקובץ `SearchAlgos.py` תחת המחלקה `Minimax`.

`set_game_params` - פונקציה זו תקרא על ידי הקוד שמנהל את המשחק פעם אחת עבור כל שחקן, בתחילת המשחק, ומטרתה להעביר לשחקנים את הלוח שבו יתנהל המשחק.

פרמטרים:

`board`:

הלוח מיוצג על ידי רשימה של רשימות, כל רשימה מייצגת שורה בלוח, כלומר לכל $0 \leq i < \text{len}(\text{board})$, `board[i]` היא שורה, ו `board[i][j]` היא משבצת. בתמונה הבאה מהאנימציה המשבצת השמאלית התחתונה היא `board[0][0]`, והמשבצת שבה נמצא השחקן הכחול היא המשבצת `board[4][1]`.



כל כניסה בלוח מכילה את אחד מהערכים הבאים:

- הערך 0 מסמל משבת ריקה.
- הערך 1- מסמל משבצת אפורה.
- הערך 1 מסמל את מיקומו ההתחלתי של השחקן שלכם
- הערך 2 מסמל את מיקומו ההתחלתי של השחקן היריב.
- ערכים חיוביים אחרים מסמלים פרי בעל שווי הערך המשבצת.

(*) יתכן שהשחקן שלכם לא יהיה השחקן שישחק ראשון, במקרה זה, הפונקציה `set_rival_move` תקרא לפני הפונקציה `make_move`.

לפונקציה זו לא צריכה להיות ערך חזרה.

`make_move` - פונקציה זו תקרא על ידי הקוד שמנהל את המשחק בכל תור של השחקן שלכם, ומטרתה לקבל מהשחקן שלכם את הצעד הבא שהוא מבצע. יש לבצע את המימוש של אלגוריתם ה-`Minimax` בקובץ `SearchAlgos.py` בתוך המחלקה `Minimax`.

פרמטרים:

`:time`

מספר השניות הנתונות כדי לסיים את ריצת הפונקציה. במידה והשחקן שתממשו יחרוג ממסגרת זמן זו, המשחק יסתיים, ויורד לכם הניקוד, כמו על דריכה במשבצת אפורה.

`:players_score`

מצב הניקוד העדכני של המשחק. מוגדר כרשימה של שני איברים, האיבר הראשון מציין את הניקוד של השחקן הראשון, והאיבר השני מציין את הניקוד של השחקן השני.

ערך החזרה:

על הפונקציה יש להחזיר את האופרטור שהשחקן שלכם בחר להפעיל. אופרטורים מיוצגים על ידי tuple בגודל 2. ערכי ההחזרה האפשריים הם $(1,0), (0,1), (-1,0), (0,-1)$. אפשר למצוא אותם בפונקציה `get_directions` ב-`utils.py`.

האופרטור $(1,0)$ מייצג צעד למעלה, $(-1,0)$ מייצג צעד למטה, $(0,1)$ מייצג צעד ימינה ו- $(0,-1)$ מייצג צעד שמאלה.

אם $loc(i,j)$ הוא מיקום השחקן טרם הקריאה לפונקציה `make_move` ו- $op = (v,u)$ הוא ערך ההחזרה, אז מיקומו החדש הוא $(i+v, j+u)$. כדי שצעד זה יהיה חוקי, צריכים להתקיים התנאים הבאים:

- השחקן נע באחד הכיוונים האפשריים - $(u,v) \in \{(1,0), (0,1), (-1,0), (0,-1)\}$
- המיקום החדש הוא בלוח: $0 \leq i+v < len(board)$ וגם $0 \leq j+u < len(board[0])$.
- במיקום החדש יש משבצת לבנה או עם פרי: $board[i+v][j+u] = 0$ או $min_fruit_value \leq board[i+v][j+u] \leq max_fruit_value$

במידה ואין מהלכים אפשריים, הפונקציה לא תקרא, ולכן אפשר להניח שבכל קריאה לפונקציה קיים לפחות אופרטור אחד שאפשר להפעיל.

`set_rival_move` - פונקציה זו תקרא על ידי הקוד שמנהל את המשחק לאחר כל תור של השחקן היריב, ומטרתה לעדכן את השחקן שלכם בצעד שהשחקן היריב ביצע, כדי שתוכלו לעדכן את הלוח.

פרמטרים:

:loc
המיקום החדש של השחקן היריב.

לפונקציה זו לא צריך להיות ערך החזרה.

Update_fruits- פונקציה זו תקרא על ידי הקוד שמנהל את המשחק בכל תור של השחקן, לפני הרצת הפונקציה make move. מטרת הפונקציה לעדכן את לוח השחקן במיקומי הפירות העדכני.

פרמטרים:

:fruits_on_board_dict
מילון בו המפתחות הם tuple שמכיל מיקום על הלוח, והערך הוא שווי הפרס.

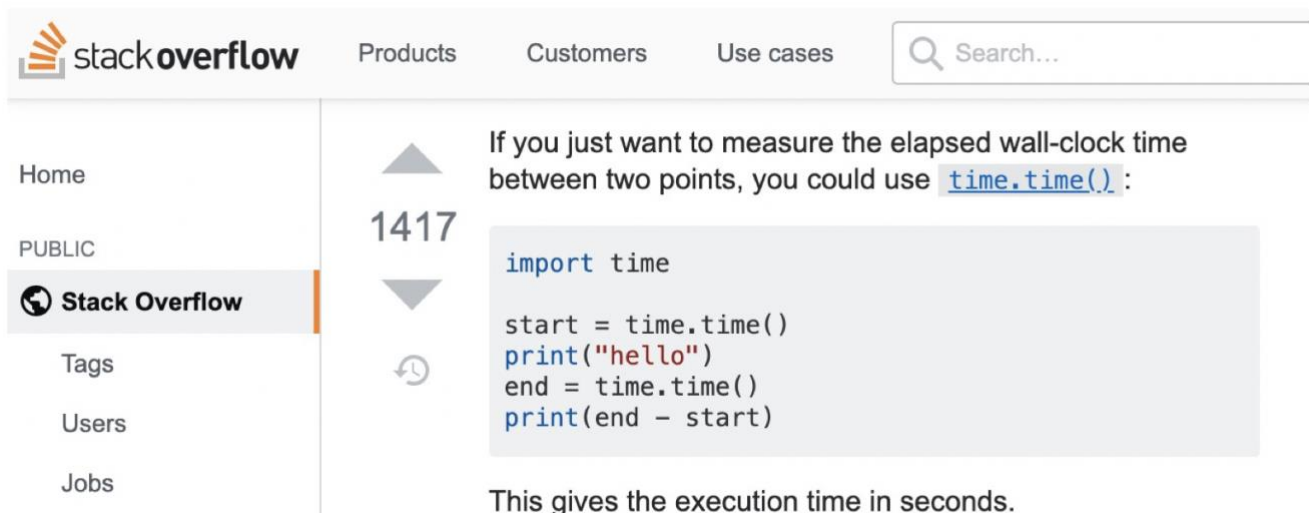
לפונקציה זו לא צריך להיות ערך החזרה.

היוריסטיקה

כפי שלמדנו בהרצאות, סוכן Mimimax מוגבל משאבים משתמש בהיוריסטיקה כדי להעריך את טיב העלים בעץ שאותו פיתח ולקבוע את אסטרטגית המשחק על פי ערכים אלו. עליכם להגדיר ולממש היוריסטיקה עבור שחקן ה-Minimax, על היוריסטיקה להיות מורכבת מלפחות 4 מרכיבים, כלומר עליכם להגדיר לפחות 4 ערכים שהם פונקציה של המצב אותו אנחנו רוצים להעריך, כך שהיוריסטיקה תהיה מורכבת מקומבינציה שלהם.

טיפ

כדי להתמודד עם מגבלת הזמן, ניתן להשתמש בספריה time:



The screenshot shows the Stack Overflow website interface. On the left is a sidebar with navigation links: Home, PUBLIC, Stack Overflow (selected), Tags, Users, and Jobs. The main content area displays a question: "If you just want to measure the elapsed wall-clock time between two points, you could use `time.time()`:". Below the question is a code block containing Python code to measure execution time. To the left of the code block is a large number "1417" with up and down arrows, indicating the number of votes. Below the code block is the text "This gives the execution time in seconds."

```
import time

start = time.time()
print("hello")
end = time.time()
print(end - start)
```

חלק ג' – מימוש סוכן $\alpha - \beta$ (12 נק')

בחלק זה נשפר את סוכן ה-Mimimax שבנינו לסוכן $\alpha - \beta$. את השחקן יש לממש בקובץ ששמו `AlphabetaPlayer.py` במחלקה ששמה `Player`. השחקן יכול להיות זהה לשחקן ה-Minimax, מלבד לכך שיממש את אלגוריתם $\alpha - \beta$. הפונקציה `__init__` מקבלת פרמטר `game_time` בו אין שימוש בשחקן זה, לכן יש להתעלם מערך זה, ופרמטר `penalty_score` שבו מצוין את כמות הנקודות היורדת לשחקן אשר נפסל ראשון. על השחקן לממש את הפונקציות `set_game_params`, `make_move`, `set_rival_move` ו-`update_fruit` כפי שתוארו בחלק ב'. ולממש את אלגוריתם ה- α - β בקובץ `SearchAlgos.py` תחת המחלקה `AlphaBeta.py`.

חלק ד' – מימוש סוכן $\alpha - \beta$ עם מגבלת זמן גלובלית (12 נק')

בחלק זה לסוכן ה- $\alpha - \beta$, במקום מגבלת זמן עבור כל תור, ישנה מגבלת זמן כללית עבור כל תורות השחקן ביחד, לכן יש לתכנן אסטרטגיה שתפרוס את הזמן בצורה חכמה על פני תורות השחקן. את השחקן יש לממש בקובץ ששמו `GlobalTimeAB.py` במחלקה ששמה `Player`. הפונקציה `__init__` מקבלת פרמטר `game_time` אשר מהווה הזמן הכולל לכל תורות השחקן, ופרמטר `penalty_score` שבו מצוין את כמות הנקודות היורדת לשחקן אשר נפסל ראשון. על השחקן לממש את הפונקציות `set_game_params`, `make_move`, `set_rival_move` ו-`update_fruit` כפי שתוארו בחלק ב', מלבד השינוי בפונקציה `make_move` אשר מתייחס לזמן הגלובלי במקום הפרמטר `time`. בשביל לבדוק סוכן זה יש להגדיר בשורה שמריצה את המשחק, את פרמטר `game_time` בגדלים שונים, ואת הפרמטר `move_time` כך שיהיה גדול או שווה לפרמטר `game_time`.

חלק ה' – תחרות

כל זוג נדרש לממש שחקן יחיד שייצג אותו בתחרות, בה ישתתפו כל הזוגות. הסטודנטים ששחקניהם יצטיינו בתחרות יזכו בבונוס לציון הסופי של הקורס: 5 נקודות לזוג המנצח, 3 נקודות למקום השני, ושתי נקודות למקום השלישי. התחרות מוגדרת כך שיש לכל שחקן זמן גלובלי לכל התורות ביחד, כמו שהוגדר בסעיף ד'. את שחקן התחרות שלכם יש לממש בקובץ `ContestPlayer.py`, במחלקה בשם `Player`. על המחלקה לממש את הפונקציות `set_rival_move`, `make_move` ו-`update_fruit`. אתם יכולים להגיש את השחקן שמימשתם בחלק ד' או לממש שחקן חדש. במידה והשחקן שתגישו לא ירוץ, יורדו לכם נקודות בתרגיל. ההשתתפות בתחרות היא חובה.

חוקים:

- אסור לשחקן להשתמש ברשת האינטרנט או ברשת מקומית כלשהי.
- אסור לשחקן להשתמש בכל סוג של קוד מקבילי.
- כל שחקן שינסה לרמות או לשבש את קוד היריב או המערכת ייפסל, ויורדו למגישים נקודות בתרגיל.
- חל איסור להשתמש במידע שעובד מראש ונשמר בקובץ.
- אסור להשתמש בחבילות/ספריות שאינן מובנות בפייתון, מלבד `networkx`, `collections` והחבילות אשר מבוצע בהם `import` בקוד שסופק לכם.
- כל חישוב לשם קביעת הצעד הבא של הסוכן צריך להתבצע אך ורק בקריאה לפונקציה `make_move`.

חלק ו' – כתיבת דו"ח (6 נק')

- ענו על הסעיפים בהמשך לשאלות בחלק היבש
15. (2 נק') תארו את היוריסטיקה שקבעתם עבור שחקן ה-Minimax, הסבירו.
 16. (2 נק') הסבירו את אופן פעילותו של שחקן התחרות שלכם. אם הגדרתם פונקציה היוריסטית חדשה, הסבירו עליה.
 17. (2 נק') תארו איך ניהלתם את זמן ריצת הפונקציה `make_move`. עבור שני מקרי הגבלת הזמן. (זמן מוגבל לתור וזמן מוגבל גלובלי).

חלק ז' – ביצוע ניסויים (10 נק')

הוסיפו את תשובותיכם לסעיפים הבאים לדוח:

18. (2 נק') הריצו משחקים בין סוכן ה-Minimax לסוכן ה-AlphaBeta (בעלי מגבלת זמן לתור) שמימשתם על מספר לוחות שונים. מה התוצאות שקיבלתם? האם התוצאות מתאימות לציפיותיכם?

השוואות עומק בין שחקני שונים

19. (8 נק') הגדירו שני שחקנים חדשים, `LightABPlayer.py`, ו-`HeavyABPlayer.py`, ללא התייחסות למגבלות זמן, ועליהם לחפש לעומק קבוע. היוריסטיקות של השחקנים יוגדרו באופן הבא:

- `HeavyABplayer` יוגדר להיות שחקן עם היוריסטיקה מתוחכמת (ניתן להשתמש ביוריסטיקה שמימשתם לשחקן שלכם)
- `LightABPlayer` יוגדר להיות שחקן עם היוריסטיקה פשוטה, מעט יותר מורכבת מ-`SimplePlayer`.

עליכם לבצע את הניסוי הבא:

על השחקן `HeavyABplayer` להיות מוגדר לעומק 3 באופן קבוע, כאשר עומק החיפוש בשחקן `LightABPlayer` משתנה בכל שלב בניסוי.

ישנם 3 שלבים, כאשר בשלב ה- i עומק חיפוש השחקן `LightABPlayer` יהיה $i - 1 + search_depth(HeavyPlayer)$. כלומר בשלב הראשון עומק חיפוש של שלוש. בכל שלב בניסוי עליכם לבצע 5 משחקים על לוחות שונים שתגדירו כאשר ציון השלב יוגדר להיות מספר ניצחונות של `HeavyABplayer` לחלק ב-5.

יש להציג את תוצאות הניסוי בגרף, כאשר ציר ה-x מוגדר להיות ההפרש בין עומקי החיפוש של `LightABPlayer` ו-`HeavyABplayer` וציר ה-y מוגדר להיות ציון השלב.

יש לבצע את הניסוי פעם נוספת עבור חיפוש בעומק 2 של `HeavyABplayer` (כעת `LightABPlayer` יחפש לעומקים 2,3,4)

הסבירו את תוצאות הניסוי, האם יש שוני בין תוצאות שני הניסויים? אם כן מה לדעתכם יכולה להיות סיבה לכך?

הוראות הגשה

יש להגיש את כל הקבצים (קוד ודו"ח) בקובץ `zip` יחיד ששמו `<id1>_<id2>AI2` כאשר במקום `<id1>` ו-`<id2>` יש לרשום את מספרי תעודת הזהות של המגישים. מלבד לדו"ח, קובץ ה-`zip` צריך את כל קבצי הקוד של השחקנים שאתם כתבתם (כל השחקנים מלבד `AbstractPlayer.py`, `LivePlayer.py`, `SimplePlayer.py`), וקבצי הקוד `SearchAlgos.py` ו-`utils.py`. כאשר היררכיית התיקיות קיימת בקובץ ה-`zip` (השחקנים בתיקיית `players`, והקבצים `SearchAlgos.py` ו-`utils.py` מחוץ לתיקיית `players`).

בהצלחה!

