**Intro to AI HW2 – Dry:**

**סטודנטים:**

עמית ענבר - 315836569

לינוי גנטי - 208536284

1. האסטרטגיה של שחקן זה היא להימנע מבורות בעומק 2 (2 מהלכים מראש)

ולהיצמד לקירות (-1 במפה) .

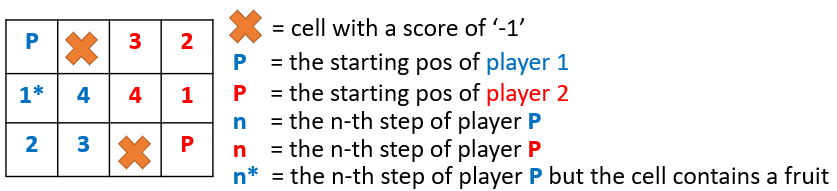
יתרונות:

1. ממלא את המפה בצורה מיטבית אם היה משחק לבד (הכי הרבה תאים שניתן להגיע אליהם ולהמשיך מהם היו נצבעים).
2. נמנע מלעבור למשבצת אם אין ממנה מהלכים אפשריים (נמנע מתאים **שמסווג** כבורות) כל עוד יש לשחקן אפשרות נוספת (ייתכן והמהלך היחיד שנותר הוא להיכנס לבור)

חסרונות:

1. "ראייה" של 2 מהלכים קדימה סה"כ והיא אינה מתחשבת במהלכים אפשריים של

ה-"אויב".

1. גם עם "ראייה" של 2 מהלכים קדימה ניתן להיכנס לבורות בעומק של יותר ממש מ-1:
2. דוגמא:

הסבר:

1. יתרונות:

* דירוג גבוה יותר של מסלולים שעוברים דרך פירות ובכך משפר את הסיכוי לקבל ניקוד גבוה.
* שימוש במידע נוסף מלבד חיפוש עיוור (היוריסטיקה מיודעת).

חסרונות:

* ייתכן ולא קיים מסלול בין המיקום הנוכחי לפרי (יש חסימה בכל מסלול אפשרי לפרי).
* יכולים להיות קירות (-1) החוסמים את כל המסלולים מהמיקום הנוכחי לפרי שאורכם הוא מרחק מנהטן מינימלי מהמיקום הנוכחי לפרי כך שיכול להיווצר מצב בו אנחנו רק "מתרחקים" במסלול הקיים מהמיקום הנוכחי לפרי:



* ההיוריסטיקה יכולה "להתבדות" אם קיים מסלול עם מרחק מנהטן אך הפרי נעלם לפני.
* יכול להיווצר מצב בו "נחתוך" את השטח שהשחקן שלנו יכול להגיע אליו ובכך "להיחסם" מוקדם יותר.
* אם הפרי נמצא בתוך בור אולי לא משתלם לקחת אותו.

1. ג
2. כן – בכל שלב בעץ המינימקס בו בוררים צעד של אחד האויבים נלקח המינימום מהבנים שלו (רקורסיבית עד שמגיעים לשלב בעץ בו אנחנו בוררים את התור שלנו – שם נקח מקסימום)
3. חסרונות:
   1. מקדם הסיעוף גדול בצורה משמעותית בשיטה זו ולכן גם זמן החישוב לכל צעד בודד. – לא פרקטי.
   2. מתייחסים לכל האויבים באופן שווה ו-"מפחדים" מכולם – בעצם אסטרטגיה פחדנית ואף פרנואידית(?) שיכולה לגרום לתוצאה שרחוקה מלהיות אופטימלית.
4. טרמינולוגיה: עבור  שחקנים: , .

תהיי  פונקציית דירוג (היוריסטיקה) לשחקן עבור כל אחד מצעדיו האפשריים.

לכל שחקן  וצעד אפשרי  נחשב את הדירוג: .

ונקח את התוחלת של של כל הצעדים האפשריים ונמקם אותם בטבלת דירוג הממוינת

לפי  אותה (את הטבלה) נעדכן בכל צעד שנבצע.

אקדיש לכל האויבים שלי שכבה אחת בעץ המינימקס בכל איטרציה באופן הבא:

אמשקל את כל צמתי הבנים (מתוכם אבחר את המינימלי) כך:



(אדגים עם טבלת דירוגים ספציפית לדוגמא)

טבלה:

|  |  |  |
| --- | --- | --- |
| average score | player | rank |
| 789 | 2 | 1 |
| 456 | 4 | 2 |
| 123 | 3 | 3 |

כעת, לכל צעד אפשרי של האויב אסכום את הניקוד המשוקלל של כל האויבים:



בדוגמא שלנו, בתור הבא של (מטעמי נוחות ומינימליות אניח כי יש לכל היותר 2 פעולות אפשריות):

בהינתן:



מתקיים:



ואמקם בעץ כבן של  עבור האופרטור  ומתוך הבנים האלו אבחר את המינימלי.

וכך גם אעשה עבור שאר השחקנים רק שעבורם, עבור שחקן :



בעצם, הרעיון באלגוריתם שלנו הוא לדרג את האויבים שלנו ולמשקל אותם לפי טיב היסטואציה שלהם ובכך לגרום לסוכן שלנו "לפחד" מההתקדמות של הסוכנים "החזקים" יותר מאשר מההתקדמות של הסוכנים "החלשים" תוך מיקסום (מקסימום) הניקוד העצמי.

1. תשובה:
2. **כן**, גיזום הענפים ב-alpha-beta יחסוך ביקורים במצבים שפסלנו ולכן יהיה מהיר יותר מ-minimax.
3. **לא**, alpha-beta פוסל רק צעדים שיחמירו את מצב הסוכן שלנו ולכן יתקדם בדיוק כמו minimax רק מהר יותר. בנוסף אין סיבה לחשוש מגיזום של ענפים שב-minimax היו משמשים אותנו לדרך הטובה ביותר מאחר ובשאלה התבקשנו להשוות בין האלגוריתמים כאשר **שניהם** כפופים לאותה מגבלה על העומק.

1. תשובה:
2. **לא** , בגיזום הענפים ב-alpha-beta עם סידור ילדים אנו מסתמכים על היורסטיקה כדי לסדר את הילדים ולכן נחלק למקרים:

במקרה בו ריצת היורסיטקה מהירה אנו משארים כי גיזום alpha-beta עם סידור ילדים ירוץ יותר מהר מאשר בגיזום -alpha-beta .אך במקרה בו ריצת היוריסטיקה מאוד ארוכה ריצת בשיטת גיזום -alpha-beta תהיה מהירה יותר מאשר בשיטת גיזום alpha-beta עם סידור ילדים.

1. **לא,** נניח שהאלגוריתם אלפא בטא והאלגוריתם אלפא בטא עם סידור ילדים מפתחים משמאל לימין וכי יש במהלך הפיתוח של העץ בעומק 2 שני ערכים בעלי ערך זהה. אלגוריתם אלפא בטא יבחר ללכת לצד שמאל בתור מהלך אך במקרה של אלפא בטא עם ילדים יש אפשרות כי אלגוריתם זה יסדר את הילדים הזהים בערכם בסדר הפוך( היורסטיקה סידרה אותם כך) ולכן למרות שאלגוריתם זה יבחר גם את הצד השמאלי בטור המהלך הבא שלו ,בצד זה ישנו מהלך שונה מהמלך שנבחר באלפא בטא ולכן האלגוריתמים לא בהכרח יבחרו באותו מהלך בהכרח.
2. תשובה:

Anytime contact הינה שיטה בה בעצם משנים את הגדרת הבעיה. במקום להחזיר את הצעד הטוב ביותר אנו רוצים להחזיר את הצעד הטוב ביותר (שמצאנו בינתיים) תוך k שניות.

העמקה הדרגתית זו שיטה המקובלת להתמודדות עם מגבלת הזמן לכן נגביל את העומק אליו האלגוריתם יכול להגיע בכל איטרציה ובכך נעמיק את החיפוש ב-"קצת" כל איטרציה עד שיגמר הזמן.

אם ב-minimax בלי הגבלת עומק אנחנו **יודעים** להסיק את הערכים של כל העלים בעץ מלכתחילה אז בווריאציית Anytime contact של minimax, בונים עץ minimax בעומק מוגדר מראש (בד"כ יהיה מאוד קטן באיטרציה הראשונה), מחשבים מסלול מיטבי ואז ובכל איטרציה מגדילים את עומק החיפוש כדי למצוא צעדים יותר טובים (כי הם יותר "מיודעים") מהצעדים שנמצאו באיטרציה הקודמת ומכיוון שלא "נתון" לנו, בכל איטרציה, המשך העץ המלא עד העלים (מצבי היעד) מחשבים היוריסטיקה כלשהי עבור הצמתים העמוקים ביותר בכל איטרציה (שכנראה אינם עלים בעצמם באיטרציות הראשונות) ולפי ערכים אלו בונים עץ minimax "זמני", שמתאים לאיטרציה הנוכחית, בעומק מוגבל ומוצאים את המסלול המיטבי עבור אותו העומק.

1. תשובה: הבעיה בהעמקה הדרגתית המוצגת בהרצאה הינה בעיית האיטרציה האחרונה.

בממוצע, האלגוריתם יופסק באמצע האיטרציה האחרונה שלו והצעד "המיטבי" שיוחזר הוא התוצאה של האיטרציה הקודמת. האיטרציה האחרונה צורכת הרבה משאבים וחבל לא להשתמש במידע שהגילינו שם לכן הפתרון שהוצא בהרצאה היה שבכל איטרציה נשמור את הערך המינימקס של כל אחד מהבנים ברמה העליונה ובכך ננצל את המידע שהשגנו למרות שחלק מהבנים שברמה העליונה מחושבים עד עומקים שונים.

1. בהתחלה אגדיר רעיונות ומושגים ובסוף אחבר את הכל:

* **רעיון 1:**

אבחנה: נראה שבתורות הראשונים אין הרבה סיכונים ובתורות האחרונים אין הרבה אפשרויות בחירה גם ככה אז אקצה פחות זמן בהתחלה ובסוף.

אז איך אגדיר מתי אני בהתחלה/אמצע/סוף?:

* + אם מימדי המפה הם  ונתון לי מספר התאים החסומים (-1): 

אז אגדיר: 

ואתאים פונקציה: 

כך שבינתיים אפשר להגדיר:



* **רעיון 2:** (מתאים בעיקרון רק למשחק שבמטלה כרגע אך ניתן לשנות אותו בהתאם למשחק)

אבחנה: אם הגענו לתור בו היריב "קרוב" לסוכן שלנו נרצה להקצות יותר זמן עבור חישוב הצעד.

איך אגדיר קרבה ואיך נשתמש בה?:

* + בהינתן פונקציית מרחק מנהטן כמוגדר בסעיפים הקודמים נסמנה: 

אז אגדיר: 

וכעת אחבר לרעיון הקודם כך שבינתיים:



נבחין כעת כי הרעיון השני יכול לגרום לנו לחרוג מהזמן הגלובלי הנתון לקראת סוף המשחק לכן אגדיר חסם שרירותי (שניתן לשנותו בהתאם לבדיקות וניסויים עם פרמטרים שונים) של 90% כאשר אם עבר 90% מהזמן הגלובלי הנתון אניח שברוב המקרים זה אומר שעברנו את המקטע הבעייתי עם היריב או שלפחות הגענו ל-phase של סוף המשחק (phase=3) אז החל מנקודה זאת אפעיל את האלגוריתם הנאיבי לטיפול בזמן רק שבמקום הפרמטר global\_time אציב את הזמן הנותר

(שאמור להיות באיזור ה-0.1\*global\_time)

ואחשב מחדש את num\_turns כך שיהיה רלוונטי.

1. אפקט האופק הינו תופעה בה אלגוריתם מוגבל משאבים בוחר צעדים "סתמיים" כדי לדחות "צרות" מעבר לאופק החיפוש כלומר הוא נמנע מצעדים מסוכנים לו ומבצע פעולה כי הוא מוגבל בעומק ולכן מחליט לא להחליטהפתרון לאפקט האופק הינו העמקה סלקטיבית -מפתחים את העץ עד העומק המוגדר באיטרציה הנוכחית ואם מקבלים עלים "לא שקטים" (נקבע עפ"י קריטריון מוגדר מראש) נעמיק עוד שכבה בחיפוש לבדוק אם הערך ההיוריסטי התייצב – אחרת מעמיקים עוד קצת ועוד קצת עד שנגיע לרגיעה (התייצבות הערך ההיוריסטי למשל) או עד עומק קבוע מסויים שמוגדר מראש להעמקות סלקטיביות של האלגוריתם.

(בתרגול סימנו עומק זה כ-k ולכן עומק תת העץ של הבן הלא יציב הוא סה"כ D+k כאשר D הוא עומק ההעמקה באיטרציה הנוכחית)

פתרון זה יכול לעזור לנו במשחק הנתון במקרים:

1. מקרה בו הסוכן שלנו יכול לפגוע בעצמו ע"י החלטה גרועה.
2. מקרה בו האויב מקדם את עצמו מאוד כתוצאה מהזנחה של נקיטת פעולות נגדו.

דוגמאות:

1. מצב בו אנחנו יכולים להכניס את עצמנו לבור (יכולים לחסום את עצמנו):

אתייחס לעומק חיפוש 2 כ-"שדה ראייה" של 2 צעדים קדימה.

כלומר, בהינתן המצב: (אדום יריב, כחול "אני")



"שדה הראייה" של הסוכן שלי הוא:



וקל לראות שהסוכן שלנו ייחסם תוך 2 צעדים בשביל לקחת את הפרי בעוד שאם היה מנסה ללכת לפרי השני כנראה שהיה לו סיכוי והיה מנצח.

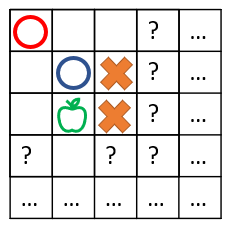
1. מצב בו האויב לוקח פרי ומעלה לעצמו משמעותית את הניקוד:

(אשתמש באותם פרמטרים ודימויים כמו בדוגמא הקודמת)

כלומר, בהינתן המצב:



"שדה הראייה" (=2 צעדים קדימה) של הסוכן שלי הוא:



כעת קל לראות שהסוכן שלנו לא ינסה לחסום את היריב אלא ייקח את הפרי הקרוב אליו בעוד שאם היה מעמיק אפילו בצעד בודד היה חוסם את היריב ולוקח את "השלל" לעצמו ובכך מנצח.

1. בעצם ערך המינימקס מהריצה הקודמת זה החסם התחתון המהודק של  והחסם העליון המהודק של לאחר ריצת האלגוריתם.

אז כעת, במקום לאתחל את  למינוס אינסוף ואת  לאינסוף (או ערך ספציפי אחר כלשהו), ניתן לאתחל אותם לערך המינימקס שנשמר ובכך לגזום "מראש" את כל(?) המהלכים שגזמנו בריצה הקודמת לפי הקריטריונים של  ו-.

במקרה הטוב ביותר כמעט כל העץ למעט "המסלול" המיטבי נגזם ואז בריצה החוזרת

1. ג
2. א. החלטתו של השחקן תשפיע על משך זמן החישוב בשאר המהלכים. אם נחלק את הזמן באופן שווה בין כל המהלכים לכל מהלך יש B/M דקות לחישוב ולכן כל מהלך יוכל לחפש לעומק D .אך אם נעמיק בצעד הראשון בD+1 נגרום לכך שהזמנים של כל המהלכים יהיה קטן מ- ולכן בלפחות מהלך אחד החיפוש בו יהיה קטן מעומק D מה שישפיע על ערך המינימקס במהלך זה.

ב.1.מס' הקשתות שעל השחקן לשמור הוא  *.*

*ב.2.כאשר נשמור את האסרטגיה של הצעד הראשון זה יחסוך לנו במהלך הבא את החישוב מהצומת שבחרנו במהלך הראשון עד עומק של ועל כן נוכל להגדיל את העץ בעומק הגדול מ D באופן מהיר יותר מ M/B זמן ועל כן נחסוך בזמן חישוב העץ ובכך נשפר את המצב מסעיף א'. אני לא בטוחה שזה נכון כי תכלס אנחנו מחשבים מינימקס מלמטה למעלה?*