**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:

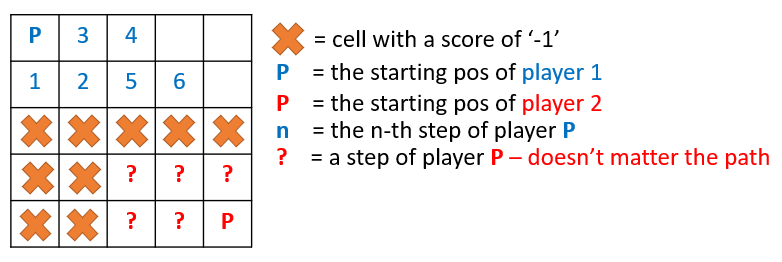
1. ממלא מפה שלא מגיעה עם בורות בצורה מיטבית אם היה משחק בה לבד.
2. נמנע מלעבור למשבצת אם אין ממנה מהלכים אפשריים (נמנע מתאים **שמסווגים** כבורות) כל עוד יש לשחקן אפשרות נוספת (ייתכן והמהלך היחיד שנותר הוא להיכנס לבור)

חסרונות:

1. "ראייה" של 2 מהלכים קדימה סה"כ והיא אינה מתחשבת במהלכים אפשריים של ה-"אויב".
2. התנהגות צפוייה שניתן לנצל.
3. גם עם "ראייה" של 2 מהלכים קדימה ניתן להיכנס לבורות בעומק גדול (ממש) מ-1:



1. דוגמא:



הסבר:

נבחין כי במקרה זה, לא משנה מה השחקן האדום (היריב) עושה, הכחול תמיד ינצח כיוון שהשחקן

הפשוט יימלא את כל התאים בצד שלו (כממוספר) בעוד שהאדום ייחסם לכל היותר אחרי 5 צעדים מצידו.

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).
* אם אין פירות על הלוח בשלב כלשהו במשחק, ההיוריסטיקה לא מוגדרת – מה היא מחזירה במקרה כזה?
* 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. דוגמא:



* 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. **נגדיר פרמטר בו נשתמש**: יהי  קבוע שלם כלשהו כך שהוא קטן-שווה ממחצית הצלע הקטנה במפה.

* רכיב 1 – מספר תאים פנויים בסביבה הקרובה:

נגדיר מטריצה  כך שהשחקן שלנו במרכז

(תוך התעלמות מתאים שנמצאים מחוץ למפה) ונסכום בה את הניקוד של כל התאים שערכם חיובי אחרי הוספת 1 (פנויים או מכילים פרי), כי לסכום תאים פנויים שערכם 0 לא עוזר לנו, ונשמור במשתנה:



* רכיב 2 – מספר קירות/מכשולים בסביבה הקרובה:

נגדיר מטריצה זהה למטריצה שברכיב הקודם רק שהפעם נסכום את כל התאים במטריצה הממורכזת שהם בלוקים (שליליים) ונכפיל ב-(1-) כדי שהתוצאה תהיה חיובית. (אם יש 4 קירות אז אקבל 4-):



* רכיב 3 – הניקוד של הפרי הכי קרוב שלא קרוב יותר ליריב:

כמו שהוגדר בשאלה 3, נחשב רכיב זה ונשמור במשתנה:



כאשר האינדיקטור מחזיר 1 רק אם:

* + הפרי לא יעלם עד שנגיע אליו.
  + השחקן היריב רחוק יותר (עפ"י מנהטן) מהפרי.
* רכיב 4 – מרחק מנהטן מינימלי מהשחקן היריב:

אם  הם מימדי הלוח אז סכום המימדים חלקי מרחק מנהטן מהיריב ועוד אחד (כדי שלא תהיה חלוקה ב-0):



כעת, נאחד להיוריסטיקה אחת:



לסיכום:

* כפי שניתן לראות, ישנו אינדיקטור שיחזיר 0 רק אם המצב s הוא בור ואז ניתן לו דירוג מינימלי של 0.
* כמו כן, הוספת ה-(1+) נועדה למנוע מצב של חלוקה ב-0 ובמונה הוסף "כתיקון" להוספה במכנה.

מוטיבציה והסבר על הרכיבים:

* **מונה ומכנה:**

חלוקה של הניקוד של התאים הפנויים בתאים החסומים ייתן לנו תוצאה מינימלית כאשר אנחנו נכנסים "לאיזור" עם הרבה תאים חסומים ותוצאה מקסימלית אם ניכנס לאיזור עם הרבה תאים פנויים+פירות.

* **(כתום + ירוק):**

**כלומר**, אם ניתן להגיע לפרי מסויים שייתן לנו הרבה ניקוד, נקבל היוריסטיקה גבוהה וזה טוב מאחר ואכילת פירות מקדמת אותנו מאוד במשחק (מטרה משנית).

**בנוסף**, אם המרחק מהיריב מינימלי נקבל היוריסטיקה גבוהה (ולהיפך) – הרעיון פה הוא שבשילוב עם שאר הרכיבים, ההיוריסטיקה תביא להתנהגות יחסית אגרסיבית שתשאף לחסום את היריב בהזדמנות ראשונה.

1. 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)
2. חסרונות:
   1. המרחק בעץ המינימקס בין המהלכים שלנו גדול מאוד בשיטה זו ולכן נוצר מצב שבחיפוש עד עומק סופי כלשהו בעצם בדקנו מספר בודד של צעדים **שלנו** קדימה. – יכול לגרום לכניסה למצבי בורות טרוויאלים וזה לא פרקטי.
   2. מתייחסים לכל האויבים באופן שווה ו-"מפחדים" מכולם – בעצם אסטרטגיה פחדנית ואף פרנואידית שיכולה לגרום לתוצאה שרחוקה מלהיות אופטימלית.
3. 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.

נקדיש לכל האויבים שלנו שכבה אחת בעץ המינימקס בכל איטרציה באופן הבא:

נמשקל את כל צמתי הבנים (מתוכם נבחר את המינימלי) כך:



(נדגים 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 נסכום את הניקוד המשוקלל של כל האויבים:



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

בהינתן:



מתקיים:



ונמקם בעץ כבן של  עבור האופרטור  ומתוך הבנים האלו נבחר את המינימלי.

וכך גם נעשה 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 (במקרה הכללי)**, גיזום הענפים ב-alpha-beta יחסוך ביקורים במצבים שפסלנו ולכן יהיה מהיר יותר מ-minimax או במקרה הגרוע יהיה איטי באותה מידה כמו ה-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 בהכרח**, 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 ריצה בשיטת גיזום -alpha-beta תהיה מהירה יותר מאשר בשיטת גיזום alpha-beta עם סידור ילדים.

1. **לא בהכרח,** נניח שהאלגוריתם alpha-beta והאלגוריתם alpha-beta עם סידור ילדים מפתחים משמאל לימין וכי יש במהלך הפיתוח של העץ בעומק 2 שני ערכים בעלי ערך זהה. אלגוריתם alpha-beta יבחר ללכת לצד שמאל בתור מהלך אך במקרה של alpha-beta 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 מהמהלך שנבחר באלפא בטא ולכן האלגוריתמים לא בהכרח יבחרו באותו מהלך בהכרח.
2. 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. 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 נגדיר רעיונות ומושגים ובסוף נחבר את הכל:

* **רעיון 1:**

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

אז איך נגדיר מתי אנחנו בהתחלה/אמצע/סוף?:

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

אז נגדיר: 

ונתאים פונקציה: 

(כך שב- יינתן פי 2 זמן לכל תור מאשר משאר ה--ים)

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 נגדיר קרבה ואיך נשתמש בה?:

* + בהינתן פונקציית מרחק מנהטן כמוגדר בסעיפים הקודמים נסמנה: 

אז נגדיר: 

וכעת נחבר לרעיון הקודם כך שבינתיים:



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

(שאמור להיות באיזור ה-0.1\*global\_time) ונחשב 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 להחליט. הפתרון 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):

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

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



"שדה הראייה" של הסוכן שלנו 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.

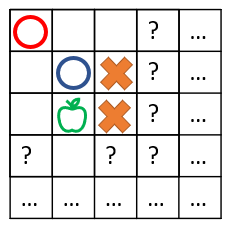
1. מצב בו האויב לוקח פרי ומעלה לעצמו משמעותית את הניקוד:

(נשתמש באותם פרמטרים ודימויים כמו בדוגמא הקודמת)

כלומר, בהינתן המצב:



"שדה הראייה" (=2 צעדים קדימה) של הסוכן שלנו 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 חסם תחתון של  וחסם עליון של לאחר ריצת האלגוריתם.

אז כעת, במקום לאתחל את  למינוס אינסוף ואת  לאינסוף (או ערך ספציפי אחר כלשהו) בכל שלב של העמקה במינימקס, ניתן לאתחל אותם באופן הבא:

נגדיר  כלשהו כך ש- ו- יאותחלו בסביבת  של ערך המינימקס.

צריך לבחור  לא קטן מדי כדי שלא יגזום את המסלול הרלוונטי אבל מספיק קטן כדי שהגיזום לפיו יגזום מראש כמה שיותר תתי עצים.

אם נבחר  גדול מדי, הריצה החוזרת של האלגוריתם תקח משך זמן דומה לשל הריצה הקודמת.

כלומר, נאתחל: .

ובכך **במקרה הכללי** נגזום מראש כמה שיותר תתי-עצים שמיותר לפתח וגם אנחנו מבטיחים שיוחזר הצעד שהיה אמור לחזור בהרצה הקודמת.

כמו בריצה רגילה של אלגוריתם אלפא-בטא ללא שינויים, **המקרה הגרוע ביותר** הוא בו הגיזום נעשה רק בבן האחרון במקום באחד הבנים הראשונים.

ולהיפך-**המקרה הטוב ביותר** הוא מקרה בו בצמתי מקסימום (שמעליהם מינימום) הבנים מסודרים מהגדול לקטן ובצמתי מינימום מסודרים מהקטן לגדול.

1. **נסמן:**

 = פונקציית ה- של מצב היעד.

יהי  מקדם הסיעוף של הסוכן המקרי, נגדיר את פונקציית הערך המשוקלל של כל צומת בן של צומת סוכן מקרי

כך:



כלומר, האופן בוא נחשב את התוחלת של צמתים אלו הוא:



עבור ההיוריסטיקה  הנתונה, נסמן: 

יהי  הצומת הנוכחי בלולאה עבורו אנו מחשבים את הערך המשוקלל, כך ש-: 

וכעת נבחן מתי נגזום בכל אחד מהמקרים לפי סוג צומת האב של הצומת האקראי הנוכחי – אם הוא מינימום/מקסימום:



* **צומת אב מקסימום (שלנו):**

נשים לב כי במקרה הטוב ביותר לאחר חישוב  נוכל להוסיף לסכום זה את הערך 

כדי שערך המחושב יהיה הערך הכי קטן שאפשר ואם ערך זה גדול או שווה לבטא כלומר עדיין גדול יותר מחסם העליון אזי עלינו לגזום אותו.



* **צומת אב מינימום (של היריב):**

נשים לב כי במקרה הטוב ביותר לאחר חישוב  נוכל להוסיף לסכום זה את הערך 

כדי שערך המחושב יהיה הערך הכי גדול שאפשר ואם ערך זה קטן או שווה לאלפא כלומר עדיין קטן יותר מחסם התחתון אזי עלינו לגזום אותו.



ולאחר הבדיקות לגיזום נחלחל למטה רקורסיבית את האלפא והבטא ה"מעודכנים": 

1. **הערה: לאורך השאלה נניח שמדובר על עצים שלמים אז מה שמחושב בכל הסעיפים הם החסמים העליונים.**

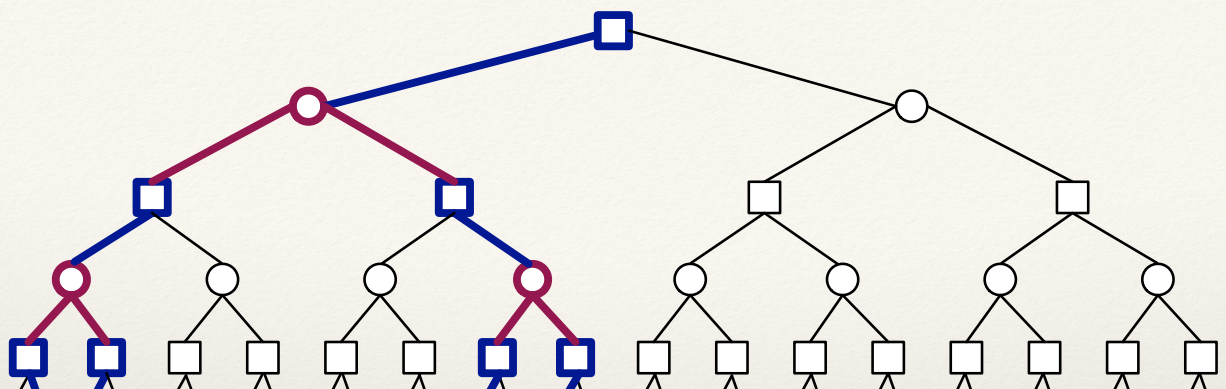
א. החלטתו של השחקן לא תשאיר זמן בכלל לחישוב *B-1* הצעדים הבאים, נראה ע"י חישוב פשוט באמצעות ערך משולש:



כלומר, צריך  דקות על מנת לחפש בצעד הראשון עד לעומק  ואז לא ישאר זמן לשאר *B-1* הצעדים הנותרים אז כנראה שייבחר באקראי צעד בכל אחד מהם או שהשחקן פשוט ייפסל.

ב. תשובות:

1. נתבונן בשקופית מההרצאה:

**

*אם נתבונן רק על הקשתות הסגולות (של היריב) נבחין כי אפשר לחבר כל צומת ריבוע (מקסימום) לצומת עיגול (מינימום) שתחתיו לצומת יחיד ואז נקבל עץ סגול מלא רגיל בעומק של*  אז כעת נחשב את מספר הקשתות של היריב:



כלומר, סכום של סדרה הנדסית כאשר  עומק תת-העץ המפותח מהצעד הראשון ו-:



*וכעת נחשב את מספר הקשתות שנשמור בשבילנו כאשר לא נתחשב בקשת הבודדת שאנחנו שומרים עבור הצעד הראשון.*

*חישוב: קשת אחת עבור כל קשת של היריב אלא אם השכבה האחרונה היא שכבת מינימום – של היריב. כלומר:*



1. *כאשר נשמור את האסרטגיה של הצעד הראשון, במקום שנצטרך לחשב את הצעד הבא תוך 0 זמן (מסעיף א' לא נשאר זמן ל-B-1 הצעדים בהמשך), אנחנו "זוכרים" איך להגיב לכל אחד מבין B-1 הצעדים הבאים.*

*ובכך בעצם השתפר המצב מסעיף א' שאם קודם האופציות היו או להיפסל או לבחור צעד באקראי ב-B-1 המהלכים הבאים אז עכשיו יש אסטרטגיה (כלשהי) עבור B-1 (אולי פחות אם**) צעדים אלו.*

1. *נחלק את מהלך המשחק לקבוצות של B מהלכים.*

כעת נבחן מה קורה אם:

1. :

אז ל- *Bהצעדים הראשונים יהיה מענה "מיודע" כלשהו ול-* הצעדים הנותרים לא יישאר מספיק זמן כמתואר בסעיף א'.

כלומר, במקרה זה הביצועים יהיו **גרועים** בהרבה משל שחקן מינימקס רגיל שמחפש עד עומק .

1. :

אז לכל מהלך  מתוך קבוצת המהלכים, השחקן ששומר את האסטרטגיה ייקח את הצעד הבא כמו סוכן מינימקס שחיפש עד עומק של .

כלומר, בכל קבוצת מהלכים המהלך הראשון נבחר כמו ששחקן מינימקס שמחפש עד עומק של היה בוחר

והמהלך השני היה נבחר באופן דומה רק כמו שחקן מינימקס שמחפש עד עומק של  וכו'.

כלומר, במהלך הראשון הוא "רואה" יותר קדימה מאשר שחקן מינימקס רגיל (שמחפש עד עומק של )

והחל מהמהלך השני הביצועים שלו לעומת שחקן מינימקס רגיל יהיו: או **זהים, או גרועים יותר**.

כלומר, במקרה בו הצעד הראשון בכל קבוצת צעדים הוא קריטי יהיה יתרון מסויים לשחקן ששומר את האסטרטגיה ובמקרים אלו הביצועים יכולים להיות **טובים** יותר משל שחקן מינימקס רגיל.

**אבחנה**: אם  (גדול בהרבה) אז ההבדל בין השחקן החדש לשחקן מינימקס רגיל שמחפש עד עומק  בקושי מורגש.

(במקרה הפרטי: כאשר , הצעד הבא ייחושב עפ"י ההיוריסטיקה בלבד)

1. **בדיוק כמו שמפורט (מאוד) בסעיף 4.**
2. **רעיון:** לחשב את מספר הצעדים שנותרו לנו ושנותרו ליריב עד שנחסם.

**איך נעשה את זה? – networkx:**

יצרנו גרף לא מכוון מהלוח כאשר כל צומת היא תא בלוח והקשתות הן המהלכים האפשריים.

אבחנה: יש צורך לחלק את המפה לרכיבים קשירים, הרכיבים שמעניינים אותנו הם הרכיב הקשיר של הסוכן שלנו והרכיב של היריב. בנוסף, נבחין שמסלול שעובר בכמה שיותר צמתים בגרף הוא מסלול המילטוני מקסימלי (גם אם עוברים רק כמעט בכל הצמתים זה מספיק טוב) אך לא מצאנו אלגוריתם שמחזיר את המסלול ההמילטוני המקסימלי בגרף אלא שקיימת רק אפשרות לקבל את המסלול ההמילטוני המלא וזאת רק אם קיים לכן, במקום זאת הרצנו חיפוש DFS למציאת המסלול הארוך ביותר בכל רכיב וקיבלנו את מספר הצעדים המקסימלים שנותרו ברכיב שלי ואת מספר הצעדים שנותרו ליריב ברכיב שלו.

לערך זה אקרא: 

(יש המשך בעמ' הבא)

כך ש-: 

ואת הפרש הניקוד בין השחקנים אסמן: 

כעת אשתמש גם ברכיבים מסעיף 4 כך ש-:



1. ניהול זמנים:

* **זמן מוגבל לתור:**

עבדנו עם שני משתני זמן "רצים" – tick, tock כאשר tick הוא התחלה של מדידה ו-tock הוא סיום של מדידה.

בנוסף יש משתנה זמן time\_left אותו עדכנו (החסרנו ממנו את ההפרש בין tick ו- tock) אחרי כל איטרציית העמקה במינימקס ואלפא-בטא.

הגדרנו שאנו נמשיך להעמיק כל עוד **כל\*** הקריטריונים הבאים מתקיימים:

* + ה-time\_left גדול (ממש) מ-0.1 מה-time\_limit ההתחלתי.

(משאיר מרחב בטיחות קטן שלא יווצר מצב שמצאנו צעד אבל לא הספקנו להחזיר אותו)

* + ה-time\_left גדול או שווה ל-4 (מקדם הסיעוף) כפול זמן הריצה שלקחה האיטרציה הקודמת.

כלומר, כל עוד הזמן הנותר גדול או שווה לחסם העליון של משך זמן ריצת איטרציית-החיפוש הבאה.

* + **קריטריון זה לא נועד לניהול הזמן אבל הוא חוסך חיפושים מיותרים:**

אם עומק החיפוש גדול-שווה למספר התאים הפנויים בלוח – זה אומר שאנחנו לקראת סוף המשחק וכל העלים בחיפוש הם מצבי יעד ואין טעם להריץ אותו שוב ושוב מאחר והוא יקבל כל פעם את אותה התוצאה.

* **זמן מוגבל גלובלי:**

בדומה לתשובה לשאלה 10 – **הרעיון הראשון** - חילקנו את משך המשחק לחלקים (phases): התחלת המשחק, אמצע המשחק, וסוף המשחק כך שבאמצע המשחק אנחנו מקדישים יותר זמן לכל מהלך מאשר בתחילת המשחק ובסופו.

בפועל, בתחילת המשחק הגדרנו משתנה מחלקה לשחקן בשם game\_time\_left ואז בכל תור בו אנו צריכים לבחור מהלך חישבנו את הזמן שאנו מקצה לתור הנוכחי ושמו ב-time\_left ואז השתמשו בניהול הזמנים שהגדרנו כבר עבור "זמן מוגבל לתור" כאשר לפני החזרת הצעד הבא עדכנו את ה-game\_time\_left על כל הזמן שחלף.

1. במפה הדיפולטית קיבלנו שאלפא-בטא ניצח ב-4 משחקים ומינימקס במשחק 1.

במפה שאנחנו הכנו (ש-p1,p2 מתחילים במרכז המפה) אלפא-בטא ניצח 3 משחקים ומינימקס 2.

בהתאם לצפוי, אלפא-בטא מנצח יותר משחקים מאחר והוא משתמש בחסכון הזמן שלו כדי לחפש לעומק גדול יותר משחקן המינימקס. בשל אי-דטרמיניסטיות תנאי ההתחלה במפה – פירות, שחקן המינימקס ניצח לפעמים "במזל" כאשר נוצרו הרבה פירות בקרבתו (מספיק בשביל "לבטל" את ה-penalty של להיחסם).

1. הגרפים:

**הסבר:**

נבחין שישנה מגמה כללית של ירידה כל פעם שמעלים את עומק החיפוש של שחקן LightPlayer למרות שעדיין HeavyPlayer מנצח ברוב המשחקים **בד"כ.**

נראה שאין יותר מדי שוני בין השינויים, הרבה פעמים נצחונות נקבעו על פי מזל (הרבה פירות שנוצרו בקרבת אחד השחקנים).

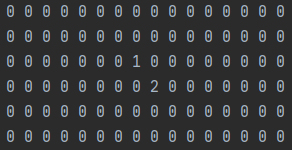
**השערה**: אולי כל העמקה של HeavyPlayer משפרת בהרבה את הביצועים לעומת כל העמקה של LightPlayer – כלומר, העמקה של HeavyPlayer איכותית יותר.

**מסקנה:**

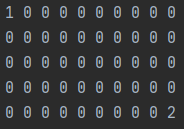
ניכר כי היוריסטיקה טובה עדיפה על חיפוש לעומק יותר גדול.

**מצ"ב הלוחות שהכנו:**

**custom\_board2.csv:**

****

**custom\_board3.csv:**

****