**מבוא לבינה מלאכותית  
תרגיל בית 2 – חלק יבש**

מגישים:  
רוני נודלמן – 209120112  
רן ברשינסקי – 208387324

1. היתרונות של היוריסטיקה:
2. היוריסטיקה מבטאת את הפוטנציאל של השחקן לעומת היריב: ככל שמספר הטחנות הכמעט-שלמות של השחקן גבוהה יותר לעומת מספר הטחנות הכמעט-שלמות של היריב, כך גדל הפוטנציאל של השחקן להשלים את הטחנות האלו ובכך להסיר חיילים של היריב מהלוח.
3. היורסטיקה מבטאת פוטנציאל יותר גבוה ל-לא להיתקע, שתמיד נוכל לזוז. הרי אם יש לנו הרבה טחנות כמעט שלמות, אז יש לנו אפשרות לזוז למקום השלישי הריק בחלק מהטחנות, ובכך להמינע ממצב שאין לנו בכלל אופציה לזוז.

החסרונות של היוריסטיקה:

1. היוריסטיקה "מעודדת" את השחקן להמשיך וליצור טחנות כמעט-שלמות ודווקא מונעת ממנו להשלים טחנות אלו, כיוון שאז הערך היוריסטי יקטן.
2. היוריסטיקה לא מביאה לידי ביטוי את מספר הטחנות הכמעט-שלמות שאכן ניתן להשלימן לטחנות שלמות. לדוגמה, כאשר לשחקן כלשהו יש טחנה כמעט-שלמה בשלב השני של המשחק, אך אין באפשרותו להשלימה כיוון שלא ניתן להזיז חיילים לכיוון התא הריק בטחנה זו.
3. נגדיר את הפונקציות הבאות:  
   נגדיר את היוריסטיקה:  
   נסביר את המוטיבציה לבחירת פונקציות אלו:  
    – אחת ממטרות הביניים של המשחק היא ליצור טחנות וזאת בכדי להסיר חיילים של היריב, וכך לצמצם את מספר חיילי היריב שעל הלוח עד לכדי ניצחון. לכן נרצה להגדיל את מספר הטחנות של המשחקן ולצמצם את מספר הטחנות של היריב, וזאת על ידי מיקסום ההפרש של ערכים אלו.  
    – אחת ממטרות המשחק הינה לצמצם את כמות חיילי היריב ומצד שני השחקן שואף להגדיל את מספר החיילים שברשותו. לכן נרצה למקסם הפרש ערכים אלו.  
    – מטרה נוספת של המשחק הינה לצמצם את כמות המהלכים האפשריים של היריב עד לכדי , ומצד שני לשמור על מספר גבוה של צעדים אפשריים של השחקן. לכן נרצה למקסם גם את הפרש הערכים האלה.  
    – היתרונות עבור פונקציה זו צוינו לעיל בסעיף .
4. היתרון בהוספת גיזום אלפא-בטא לאלגוריתם מינימקס הוא צמצום הזמן שלוקח לחשב את ערך שורש העץ, בהשוואה להרצת אלגוריתם מינימקס ללא גיזום, וזאת ע"י גיזום ענפים שחישוב ערכי המינימקס עבורם לא מועילים לחישוב ערכי המינימקס של האבות שלהם.  
   יתרון זה מושג באופן הבא:  
   עבור צומת כלשהו, נניח כי ערכו של הבן הראשון שלו חושב, והינו . כעת, כאשר נחשב את ערכי הבנים של הבן השני של הצומת הנ"ל, אם נקבל ערך קטן או שווה ל- אז נוכל לדלג על החישוב של הבנים הבאים של הבן השני. זאת כיוון שהבן השני הינו צומת ולכן ערכו יהיה לכל היותר , ולכן אין טעם להמשיך לפתח את שאר הבנים שלו, שהרי צומת בוחר את הבן עם הערך המקסימלי. כלומר, כאשר הצומת יבחר את הבן עם הערך המקסימלי, לא ייתכן שהוא יבחר את הצומת השני על פני הצומת הראשון. באופן זהה, נוכל לבצע את תהליך זה עבור שאר הבנים של הצומת .  
   באופן אנלוגי, נוכל לעשות את אותו התהליך עבור צומת .
5. כפי שראינו בהרצאה, על מנת לקבל גיזום מקסימלי נמיין את הצמתים באופן הבא:  
   עבור צמתי , הערך המקסימלי מבין ערכי המינימקס של הבנים חייב להיות בבן השמאלי.  
   עבור צמתי , הערך המינימלי מבין ערכי המינימקס של הבנים חייב להיות בבן השמאלי.
6. A picture containing chart

   Description automatically generatedאף עלה לא יגזם ע"י אלגוריתם אלפא-בטא בעץ הנתון.  
   המסלול האופטימלי להמשיך המשחק מסומן במסלול האדום בציור.
7. לא בהכרח יצרנו סוכן שמשחק עם אסטרטגיה אופטימלית. דוגמא לכך היא היורסיטקה של שאלה 1, שהיא יורסיטקה מטעה. מכיוון שלפי היורסטיקה הזאת, הסוכן יעדיף ליצור כמה שיותר טחנות כמעט שלמות, ולא ליצור טחנות שלמות, אך בשביל לנצח, צריך ליצור טחנות, כדי להסיר את החיילים של היריב. כלומר היורסטיקה לא בהכרח מבטיחה אסטרטגיה אופטימלית.
8. נגדיר את פונקציית באופן הבא כאשר מציין את אינדקס השחקן:

אכן, פונקציית זאת לא מגדירה משחק סכום אפס. למרות זאת, עדיין מתקיים כי כאשר השחקן הראשון מגדיל את התוצאה שלו, אז התוצאה של השחקן השני קטנה. וזאת באופן דומה לאופן שבו מוגדרת פונקציית עבור משחק סכום אפס.

1. נגדיר פונקציית באופן הבא כאשר מציין את אינדקס השחקן:  
   נראה דוגמה עבור עץ חיפוש עם אלגוריתם , שבו השחקן Diagram

   Description automatically generatedהראשון מתחיל לשחק:

במקרה זה, ערך המינימקס של השורש יהיה ועל כן השחקן הראשון יבחר לבצע את הפעולה שתגרום לו להגיע למצב . אולם, מבחינת שקלול הניקוד עבור השחקנים, השחקן הראשון היה יכול לבחור במצב וכך להשיג יתרון בניקוד על פני היריב.

1. ייתכן מצב במשחק שבו המחשב בעל יתרון גדול על פני השחקן. ובמצב זה, המחשב יכול לבצע צעד אחד ולנצח בסיום צעד זה, או שלבצע צעד אחר שלא יגרום לו לנצח בסיום הצעד אז עדיין יבטיח ניצחון בשלב מאוחר יותר במשחק.  
   לדוגמה, ניתן לראות זאת ע"י עץ המינימקס הבא:  
     
   Diagram

   Description automatically generated   
   \*המצבים האדומים בעץ הם מצבים סופיים שבהם המחשב מנצח.  
   בדוגמה זאת, המחשב יוכל לבחור אם לנצח תוך צעד אחד (לבחור בענף השמאלי), או לבצע צעד אחר (לבחור בענף הימני), ולנצח בכל מקרה בצעד הבא.
2. בסיום הריצה של האלגוריתם, במקום לבחור בן רנדומלי עם ערך minimax שיצא לשורש, נבחר בן שמייצג מצב סופי עם הערך הזה אם יש. וכך נוודא שאם ניתן לסיים את המשחק בצעד אחד, השחקן בוודאות יבחר בצעד זה.
3. הערכים של הצמתים הם:
4. , מכיוון שצומת בוחר את הבן עם הערך המקסימלי.
5. בעיקרון יש אפשרות לגזום חלקים מהעץ, אבל רק בשכבות של Min ו Max. אם תגזום חלקים מהעץ באותו אופן של אלפא-בטא בשכבות שהם Chance, אז הערך בשורש יכול להיות שונה.

לדוגמא, הדוגמא שאתם הבאתם, אחרי שחישבנו את הצומת B ו C, אז הערך המינמלי ל- A הוא 5, (כמו C). כעת כשאנחנו הולכים לפתח את צומת D, ורואים בבן השמאלי שלו את הערך 4, אין אפשרותנו לדעת האם ניתן לגזום את שאר הבנים, כי שאר הבנים יכולים לעלות או להוריד את התוחלת. כלומר גם אם הערך של הבן השמאלי היה גדול יותר מ- 5, עדיין יכול להיות ששאר הבנים היו מספיק קטנים, כדי שהתוחלת סה"כ הייתה קטנה מ- 5, ואז A היה מקבל את 5.

לכן בשכבות של Chance אי אפשר לגזום.

1. פסודו קוד שלאלגוריתם אלפא-בטא עם השינוי:

alphaBeta(state, epsilon):  
 return maxValue(state, -IFINITY, IFINITY, 0, epsilon)  
  
maxValue(state, alpha, beta, depth, epsilon):  
 if cutoffTestTest(state,depth)  
 return utility(state)  
   
 value = -IFINITY  
 for successor in state.getSuccessors():  
 value = max(value, minValue(successor,alpha,beta,depth+1,epsilon))  
 if value >= beta - epislon/(2^(depth+1)):  
 return value  
 alpha = max(alpha,value)  
   
 return value  
  
minValue(state, alpha, beta, depth, epsilon):  
if cutoffTestTest(state, depth)  
 return utility(state)  
  
value = -IFINITY  
for successor in state.getSuccessors():  
 value = min(value, maxValue(successor, alpha, beta, depth + 1epsilon))  
 if value <= alpha + epislon / (2 ^ (depth + 1)):  
 return value  
 alpha = min(beta, value)  
  
return value

ניקח את epsilon להיות 1, במקרה הזה.

נשים לב שכשמגיעים לצומת הבן הימני של שורש העץ, אלפא יהיה שווה ל- 1.

לפי האלגוריתם החדש שלנו, אנחנו בודקים אם הבן השמאלי שלו (עלה שמאלי שלו) קטן או שווה מ-

מכיוון שזה המצב, אז גוזמים את העלה השמאלי שלו, ולא בודקים אותו ומחזירים שהצומת הוא 1.25.

ולכן יוצא שהשורש הוא 1.25.

נשים לב שאם היינו משתמשים באלגוריתם המקורי של אלפא בטא, היינו מקבלים שהשורש הוא 1.

כלומר הראנו דוגמא שאלגוריתם החדש בו מחזירה ערך שונה מערך המינימקס.

1. אם נשתמש בפרוצדורה נוכל לחפש בזמן לעומק

נשים לב שמספר הצמתים בעץ מינמקס רגיל הוא . לעומת זאת מכיוון שבכל שכבה שהיא שכבת כלומר שכבת היריב, לכל קיים רק בן אחד, שהוא הבן שאותו היריב יקח, אבל לכל קיימים כל ה- b בנים שקיימים גם בעץ מינמקס רגיל, אז בכל שכבה של max, כמות האיברים שם שווה לשכבה min שמעליה, אבל לכל שכבה min, כמות האיברים שם גדולה פי b מהשכבה max שמעליה ולכן מספר הצמתים בעץ החדש יהיה:

*מכיוון שאנחנו רוצים את אותו הזמן, אז יוצא ש:*

*ולכן*

1. *חסר*
2. *הסטודנט הוא טועה, מכיוון שמרחב החיפוש הוא עצום (), וגם נבחרו לכל היותר 1000 נקודות התחלה שונות. יש סיכוי שאם נבחר נקודה אחרת, אז האלגוריתם יגיע לערך גבוה יותר מ 5.8. לכן לא בהכרח 5.8 היא האופטימום ההגלובלי.*
3. *נעדיף להשתמש באלגוריתם אשר בו נוכל גם להגיע למצבים גרועים יותר, אבל כך אולי בהמשך להגיע למצבים טובים יותר, ולקוות שעד שהטמפרטורה תרד לאפס, נוכל להגיע ל 5.8, או לעקוף את הערך הזה, ובכך להבין ש 5.8 הוא לא האופטימום.*
4. 