תרגיל בית 2:

חיפוש רב סוכני – Can't Go Back

מגישים

טל רוזנצוויג 307965806

שני אופיר 204512396

חלק א':

שאלה 1:

תשובה:

אסטרטגיית השחקן היא לבחור בצעד בו מספר הצעדים העתידיים הוא המינימלי, וזאת בתנאי שהצעד שנבחר לא גורם להפסד השחקן. מטרת האסטרטגיה היא בכך שבכל צעד השחקן מנסה להישאר כמה שיותר קרוב לאיזור בו הוא היה בצעדיו הקודמים ובכך הוא מנסה לא להשאיר אזורים פתוחים(לבנים) שהוא יכול להגיע אליהם בעתיד.

יתרונות האסטרטגיה:

- צריכת משאבים נמוכה – באסטרטגיה זו זמן החישוב הוא מידי מאחר והאלגוריתם הוא חמדני ומסתכל רק צעד אחד קדימה. בנוסף, היא אינה צורכת זיכרון נוסף מאחר ואין שמירה של מצבים קודמים.
- אסטרטגיה זו היא הטובה ביותר על מנת למקסם ולנצל את השטח הקרוב בו נמצא השחקן ובכך גם להגדיל את סיכויו לצבור כמה שיותר נקודות על ידי ביקור בכמה שיותר משבצות(סיכוי גדול יותר לבקר במשבצת עם פרי). נשים לב כי כל אסטרטגיה אחרת הייתה מתרחקת מהאזור בו נמצא השחקן ובכך משאירה משבצות פתוחות(לבנות) שלא היה בהן עדיין ביקור, כאשר הסיכוי לנצל אותם בעתיד קטן ככל שהמשחק מתקדם בעקבות צעדיו של היריב.

חסרונות האסטרטגיה:

- אין התייחסות למיקום היריב – לפי אסטרטגיה זו, היריב עלול לחשב אסטרטגיה שתחסום את השחקן מאחר והוא לא "רואה אותו", עד למצבים בהם 2 השחקנים מצאים במרחק של שני צעדים ומטה. במילים אחרות, היריב עלול לנסות לעשות לשחקן קיר חסימה והוא לא ינסה להתמודד מול זה. בנוסף, לא קיים צעד שמחשב את היכול לבצע חסימה ליריב.
- חוסר שליטה בלוח – מאחר והשחקן ממקסם את השטח הקרוב אליו ולא מתרחק מהאזור שלו יותר מידי, הדבר מביא לחוסר שליטה בשטחי הלוח. כך למשל, היריב יכול לחשוב על אסטרטגיה שתסגור לשחקן הרבה שטחים ובכך היא גורמת לו למקסם ולצעוד בשטחים קטנים יותר.
- אסטרטגיה חמדנית – במהלך חישוב הצעד הבא, השחקן מתחשב רק בצעדים הקרובים אליו ולא מסתכל על מצב הלוח במלואו. דרך זו עלולה לגרום לשחקן לבחור בצעד שאולי טוב בעתיד הקרוב, אך גרוע בעתיד הרחוק.
- אחת המטרות במשחק היא לא רק למקסם שטחים קרובים, אלא גם לשלוט על כמה שיותר אזורים בלוח שיאפשרו לשחקן לבצע כמה שיותר צעדים בעתיד וכמה שפחות צעדים אפשריים ליריב. כמו כן, חשוב כי שטחים אלו יהיו בעל ערך גבוה ככל הניתן – כלומר שהשטחים בהם השחקן יבחר

לעבור יהיו עם כמה שיותר פירות וכך השחקן יגדיל את הניקוד שלו(באסטרטגיה זו אין כלל התחשבות לניקוד/הערך של המשבצות בהן השחקן עובר).

שאלה 2:

תשובה:

הסבר הדוגמא:

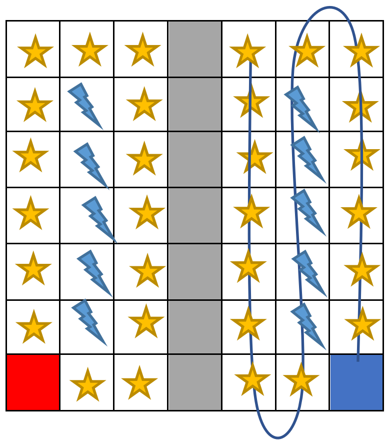
סימון:

- - פרי בעל ערך השווה ל-100 נקודות.
- - פרי בעל ערך השווה ל-50 נקודות.

הסידור של מיקומי הפירות בהתאם לסוגם עבור שני השחקנים זהה. הפירות בעלי הערך הגבוה ביותר מסודרים כך שהם במסלול של simple player כאשר סידור זה זהה עבור שני השחקנים.

הצבת הקיר החוסם ביטלה את חיסרון אסטרטגיית simplePlayer אשר לא מתחשבת בצעדי היריב. במצב המתואר, ליריב אין כל השפעה על השחקן וגם להיפך ולכן הדרך היחידה לניצחון תהיה לעבור על כמה שיותר משבצות בעלות ערך גבוה כך שייתכן וליריב הצעדים ייגמרו מהר יותר והוא יעבור על כמה שיותר משבצות בעלות ערך נמוך.

לכן, האסטרטגיה הטובה ביותר בדוגמא זו היא שימוש באסטרטגית simplePlayer הממקסמת את השטח בו נמצא השחקן, ומאופן צורת המעבר שלה על המשבצות הפנויות, תגרום להשגת הניקוד הגבוה ביותר האפשרי. בדרך זו, תהיה עדיפות לעבור קודם על המשבצות הצמודות למסגרת הלוח אשר מכילות את הפירות בעלות הניקוד הגבוה ביותר. לעומת זאת, מספיק שהשחקן היריב יבצע צעד אחד בכיוון אמצע המסלול(עמודה 2 משמאל עבור השחקן האדום), הוא יקבל ניקוד נמוך יותר מהשחקן הכחול, וזאת בהסתמך על ההנחה לפיה לאחר 7 תורות, אורך הצלע הקצרה בלוח, הפירות ייעלמו.



שאלה 3:

תשובה:

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

חסרונות היוריסטיקה:

- סיבוכיות זמן גבוהה – היוריסטיקה זו מבצעת חישובים רבים הכוללים:
  - חישוב מרחק מנהטן המינימלי לכל פרי הנמצא בלוח.
  - חישוב כבד של היוריסטיקה על כל אחד מהעלים כאשר מגיעים בהרצת חיפוש באלגוריתמי $\alpha - \beta$ או $Minimax$ לעומק אפס.
  - במידה וקיימות הגבלות על זמן הריצה(זמן מהלך או זמן ריצה כולל של שחקן), הדבר עלול לפגוע בהערכת מצבו של השחקן, מאחר והיוריסטיקה תיקח הרבה נתח מזמן התור של השחקן ולכן יישאר מעט זמן לאלגוריתם החיפוש להעמיק בעומק. כך למשל, תחת מגבלת זמן של t שניות, היוריסטיקה בעלת סיבוכיות מורכבת תגיע לעומק K באלגוריתם החיפוש, בעוד שהיוריסטיקה פשוטה תגיע לעומק K' כך שמתקיים $K \ll K'$.
- אין התייחסות לערך הפרי.
- אין התייחסות לזמן הישארות על הלוח של פרי מסוים – כלומר ייתכן כי המסלול המינימלי לפרי ארוך יותר ממספר התורות שהפרי יישאר על הלוח.
- ישנו סיכוי להיתקע לאחר לקיחת פרי, כך למשל כאשר ישנו פרי בודד על הלוח והוא מוביל למסלול שלא ניתן להמשיך ממנו, כאשר היוריסטיקה זו תוביל למסלול זה.
- אין התייחסות כלל ליריב ולמיקומו על הלוח, כך שייתכן והיריב ייקח פרי מסוים לפני שהשחקן יוכל בכלל להגיע אליו בזמן שהיוריסטיקה כיוונה אותו לכיוון פרי זה(הפרי כבר נאכל על ידי היריב והשחקן שלנו בזבז מספר צעדים במטרה להגיע לפרי).

יתרונות היוריסטיקה:

- שימוש במרחק מנהטן מקנה דיוק במרחק של השחקן לפרי תוך כדי כך שהוא מתחשב במגבלות הלוח, כדוגמת קירות או משבצות שאסור לדרוך עליהן. בכך, נקבל כי היוריסטיקה זו יותר מדויקת למשל ממרחק אווירי ובכך היא מספקת תמונת מצב נכונה יותר.

שאלה 4:

תשובה:

מרכיבי היוריסטיקה:

- $Game\ score$ – תוצאת המשחק היא ערך בתחום $[-1,1]$ המייצג את ניקוד המשחק כך שהערך הטוב ביותר הוא אחד והערך הגרוע ביותר הוא 1-.
- $Fruit\ score\ heuristic$ – מייצג את הנגישות לפירות, כלומר בחינה מידתית של כמה השחקן קרוב לפרי בעל ערך גבוה לעומת כמה היריב קרוב לפרי בעל ערך גבוה זהה. הערך של Fruit score heuristic נמצא בתחום $[-1,1]$ ומחושב באופן הבא:

לכל פרי במפה אנו מחשבים את מרחק מנהטן ממנו לשחקן ואז נבצע בדיקה האם ניתן להגיע לפרי במספר התורות שקיים לשחקן זה עד שהפרי אמור להיעלם. לצורך כך, נשתמש בנוסחאות הבאות:

$$Fruit\ score(player) = \sum_{f \in Fruit} \begin{cases} val(f) & md(f) = 0 \\ \dfrac{val(f)}{md(f)} & \text{אחרת} \end{cases}$$

כאשר $f \in fruit$ מבטא את הפירות שניתן להגיע אליהם לפני שהם נעלמים מהלוח.

$$Fruit\ score\ Hueristic\ = \frac{Fruit\ score(player = 1) - Fruit\ score(player = 2)}{\sum_{i=1}^{2} Fruit\ score(player = i)}$$

- $Reachable\ node\ score$ - ההפרש בין מספר המיקומים הנגישים לשחקן 1 לבין מספר המיקומים הנגישים לשחקן 2, מנורמל לערך בתחום $[-1,1]$.
- $Step\ score$ – ההפרש בין מספר הצעדים האפשריים של שחקן 1 לבין מספר הצעדים האפשריים עבור שחקן 2, כאשר ערך זה מנורמל לערך בתחום $[-1,1]$.
- $Road\ score$ - ההפרש בין הדרך הארוכה ביותר האפשרית של שחקן 1 לבין הדרך הארוכה ביותר האפשרית של שחקן 2, כאשר ערך זה מנורמל לערך בתחום $[-1,1]$.

<u>הנוסחא עבור היוריסטיקה:</u>

$$h(s) = Game\ score \cdot 0.6 + \ Fruit\ score\ heuristic \cdot 0.25 + Reachable\ node\ score \cdot 0.05 + Step\ score \cdot 0.05 + Road\ score \cdot 0.05$$

כאשר המשקולות לעיל נבחרו לאחר ביצוע ניסויים רבים והסקת מסקנה כי ערכים אלו מספקים את הביצועים הטובים ביותר.

<u>מוטיבציה:</u>

באופן כללי, העקרונות שהנחו אותנו לבחירת הפרמטרים הם:

- $Game\ score$ - ככל שיש ניקוד גבוה יותר, יש לנו סיכוי גבוה יותר לנצח.
- $Fruit\ score\ heuristic$ - ככל שנוכל להגיע לפירות בעלי ניקוד גבוה יותר, כך נצבור ניקוד גבוה יותר והסיכוי לנצח יעלה.
- $Reachable\ node\ score$ - ככל שיש יותר מיקומים בלוח שהם ברי השגה, הסבירות להיחסם נמוכה יותר וכך נימנע מהורדת נקודות בעקבות קבלת "קנס" בהתאם להגדרת המשחק(חסימה גוררת "קנס" במידה והשחקן השני לא חסום).
- $Step\ score$ – שמירה של כמה שיותר אופציות פתוחות, כלומר שיהיו לנו כמה שיותר אופציות אפשריות לביצוע מהלך.
- $Road\ score$ - מניעת מצב של היחסמות שעלולה להביא להפסד המשחק.

בשילוב של כל היתרונות שהצגנו, נצפה שהיוריסטיקה זו תשפר את ביצועי השחקן לעומת השחקן simplePlayer, מאחר ושילובם של פרמטרים אלו יגדילו את הסיכוי לכך שהשחקן שלנו ינצח. כמו כן, היוריסטיקה זו תשפר את ביצועי השחקן לעומת השחקן simplePlayer מפני ש-simplePlayer אינו מתייחס לניקוד המשחק או לפירות שעל הלוח שהם פרמטרים הכרחיים להשגת ניצחון.

שאלה 5:

תשובה:

אסטרטגיית $Minimax$ מתאימה עבור משחק של יותר משני שחקנים.

סעיף א':

החסרונות לאסטרטגיית $Minimax$ במקרה הם:

- מאופן פעולתו של סוכן ה-$Minimax$, הוא תמיד מניח כי היריב מבצע את הצעד הטוב ביותר עבורו(=היריב) ולכן עבור סוכן ה-$Minimax$ צעד זה נחשב כצעד הגרוע ביותר עבורו. כלומר, בעת הרצת $Minimax$, חישוב ערך המינימום שנקבל כתוצאה ממהלך היריב, מתקבל כתוצאה ממציאת המהלך הטוב ביותר אשר היריב יכול לעשות בשביל עצמו.
  כאשר ישנו משחק המכיל יותר משני שחקנים, ייתכן כי הצעד הטוב ביותר של היריב מתחשב גם בתחרות שלו עם סוכן שלישי(ועוד) ולכן צעד זה לא בהכרח יוביל לבחירת הצעד הגרוע ביותר עבור סוכן ה-$Minimax$, כפי שהיינו מצפים מאלגוריתם זה. ובמילים אחרות, במקרה זה חישוב ערך המינימום אותו נוכל להשיג מתורו של היריב אינו שווה ערך לערך המרבי אותו היריב יוכל להשיג בתורו בשביל עצמו.
- שיתוף פעולה בין סוכנים - כפי שציינו בנקודה הקודמת, סוכן ה-$Minimax$ מניח כי היריב מבצע את הצעד הגרוע ביותר עבורו, אך במקרה של מספר רב של שחקנים, זה אינו תמיד הדבר היעיל ביותר. כך למשל, במשחק מרובה שחקנים, ייתכן כי הם יוכלו לשתף ביניהם פעולה או יוכלו להתחרות אחד נגד השני כך שהתחרות ביניהם תאפיל על התחרות שלהם עם הסוכן. כלומר, נקבל כי האלגוריתם יחשב בשלב המינימום כיצד כל אחד מהיריבים יכול לפגוע בסוכן ה-$Minimax$ (=השחקן שעבורו מחושב ה-$Minimax$)בצורה הטובה ביותר, וכך נקבל מצב בו כל היריבים משתפים פעולה יחד כדי לנצח את השחקן, דבר שאינו קיים במשחק רגיל בו כל שחקן מנסה לנצח בשביל עצמו ולכן מבצע את הצעד הטוב ביותר עבורו.

סעיף ב':

נציע את האסטרטגיה החלופית הבאה(כאשר נריץ $Minimax$, השחקן הראשי יהיה השחקן שזהו תורו הנוכחי ומחשבים עבורו את ערך ה-max):

נבצע שינוי באלגוריתם ה-$Minimax$ כך שנסתכל על תורם של כל שאר השחקנים(בלי תורו של השחקן הראשי) כתור אחד, כך שהם ייחשבו כיריב אחד המאגד בתוכו את שאר השחקנים. השוני העיקרי ביריב זה הוא כי הוא ינסה בכל פעם להשיג את הצעד הטוב ביותר של כל אחד מהשחקנים תחתיו בנפרד כך שעבור <u>כל</u> שחקן יבחר הצעד הטוב ביותר <u>בשביל עצמו</u> ומצעד זה נוכל להסיק מהו הרווח שיכול להפיק מכך השחקן הראשי. שימוש באסטרטגיה זו תימנע את שיתוף הפעולה שעלול להיווצר כנגד השחקן הראשי וזאת מאחר והיריב יבחר את המהלך הטוב ביותר בשביל כל אחד מהשחקנים שלו וכך השחקנים לא יוכלו לעשות "יד אחת" כנגד השחקן הראשי. כמו כן, ייתכן כי מהלך זה של היריב ייפגע באחד מהשחקנים הנמצאים ברשות השחקן היריב עצמו וזאת מאחר שהוא דורג לכל אחד משחקניו באופן עצמאי.

שאלה 6:

<u>תשובה:</u>

<u>סעיף א':</u>

מבחינת זמן הריצה, סוכן ה- $\alpha - \beta$ יהיה מהיר יותר מסוכן ה-$Minimax$ בזכות אופן פעולתו המבצע גיזום ענפים ובכך חוסך חישובים שלא נדרשים לחישוב הערך האופטימלי(המינימקס), לעומת סוכן ה-$Minimax$ אשר מבצע חישובים אלו.

<u>סעיף ב':</u>

תחת ההנחה שעומק החיפוש זהה בין שני הסוכנים, שניהם יחזירו את אותו ערך ה-$Minimax$ מאחר והערכים שמקוצצים על ידי סוכן ה-$\alpha - \beta$ אינם משפיעים על ערך המינימקס הסופי. ההבדל היחידי בין הסוכנים הוא קיצוץ הענפים אשר ישפיע על זמן הריצה, אך לא על התוצאה הסופית שכן ערכי העלים בעצי החיפוש הינם זהים עבור שני הסוכנים.

שאלה 7:

<u>תשובה:</u>

<u>סעיף א':</u>

מבחינת זמן הריצה, סוכן ה- $\alpha - \beta$ עם סידור ילדים יהיה מהיר יותר מסוכן $\alpha - \beta$ מאחר והסדר שבו הילדים מסודרים מיטיב עם קיצוץ הענפים בניגוד לסוכן ה-$\alpha - \beta$ שעובר על הילדים בסדר כלשהו, ולכן ייתכן מצב שבו סוכן ה-$\alpha - \beta$ יגלה רק בשלב מאוחר כי הוא צריך לבצע גיזום, זאת לאחר שהוא כבר ביצע חישובים מיותרים.

<u>סעיף ב':</u>

תחת ההנחה כי עומק החיפוש זהה, שני הסוכנים ייבחרו את אותו המהלך וזאת מאחר שערך ה-
$Minimax$
יהיה זהה בדומה להסבר של סעיף ב' של שאלה 6.
אחרת, תחת ההנחה כי עבור תור בודד קיים זמן קבוע לכל שחקן, סוכן $\alpha - \beta$ עם סידור ילדים יעיל יותר מבחינת זמן ריצה ולכן ייתכן והוא יגיע לעומק עמוק יותר וכך יוכל לבצע בחירת צעד נבון יותר מאשר סוכן $\alpha - \beta$ אשר ייתכן ויגיע לעומק קטן יותר וכך הוא עלול לבחור בצעד פחות טוב ממה שסוכן ה- $\alpha - \beta$ עם סידור ילדים, ייבחר בו.

שאלה 8:

<u>תשובה:</u>

וריאציית Anytime contract של אלגוריתם ה-Minimax מוחזר הפתרון הטוב ביותר שניתן להחזיר תוך זמן קבוע שנקבע מראש לאלגוריתם. הרעיון העיקר בווריאציה זו היא שברוב המקרים סוכנים מוגבלים במשאבים, בעיקר בזמן, לפני שהם נדרשים לפעול. בנוסף, ייתכן כי במקרים רבים בעקבות מקדם סיעוף גבוה מידי, לא ניתן לחשב תחת המגבלות הללו את המסלול האופטימלי לניצחון, לכן מנסים למצוא צעד שהוא מספיק טוב. כלומר, Anytime contract גם עבור Minimax באופן כללי מוגבל בזמן הנתון מראש, אך לא חייב להיות בהעמקה הדרגתית.

ההעמקה ההדרגתית בהקשר זה היא דרך התמודדות עם הזמן באלגוריתם ה-Minimax. בדרך זו, בכל איטרציה מחשבים עץ עם הגבלת עומק הולכת וגדלה בין איטרציה לאיטרציה, כאשר לאחר כל צעד נשמר הצעד הטוב ביותר שנבחר מהאיטרציה הקודמת. לבסוף, כאשר הזמן נגמר יוחזר הפתרון הכי טוב שחושב עד כה.

שאלה 9:

תשובה:

הבעיה הנוגעת להעמקה ההדרגתית המוצגת בהרצאה היא בעיית האיטרציה האחרונה, לפיה בכל איטרציה הזמן עולה בצורה אקספוננציאלית ולכן את רוב המשאבים אנו נשקיע בחישוב האיטרציה האחרונה אשר יכולה להיקטע באמצע ולכן עלול להיווצר מצב של בזבוז משאבים. באופן מפורט יותר, בכל איטרציה מגדילים את הגבלת העומק באחד כאשר זמן החישוב של האיטרציה עבור עומק d גדל פי b מזמן החישוב האיטרציה עם הגבלת עומק של d-1 כאשר b הוא מקדם הסיעוף. כלומר, זמני החישוב בין איטרציה לאיטרציה גדלים אקספוננציאלית בכל שמעמיקים, בעיקר עבור מקדם סיעוף גבוה. מכאן נקבל כי מאחר ואנו תמיד לא נספיק לחשב את האיטרציה האחרונה, אנו נבזבז זמן חישוב יקר מאד שלא יהיה שימושי.

הפתרון המוצע בהרצאה לבעיה זו, הוא שמירת ערך ה-Minimax של כל אחד מהבנים ברמה העליונה. בדרך זו, אם נניח כי בממוצע האלגוריתם מפסיק באמצע האיטרציה האחרונה, אזי נקבל כי בממוצע נספיק לחשב עבור חצי מהבנים לעומק d וחצי מהבנים יחושבו לעומק d-1. כלומר, אם האיטרציה האחרונה תיגמר באמצע ריצתה, עדיין חושב חצי מהעץ, כלומר חצי מהבנים, וכך נוכל להשתמש בערך שהם סיפקו במידה והוא ערך טוב יותר ובדרך זו ננצל את המשאבים בצורה טובה יותר(כך לפחות עבור חצי מהבנים ניצלנו את המשאבים כדי לראות עומק גדול יותר).

שאלה 10:

תשובה:

תחילה, נבחין בשני חסרונות בשימוש באלגוריתם זה:

1. שימוש בחלוקה זו עלול לגרום לזמן לא מנוצל – ניתן לומר כי ייתכן והמשחק יסתיים במספר תורות הקטן מהחסם העליון וזאת מאחר שהמשחק יכול להסתיים במצבים נוספים, כדוגמת מצב שבו אחד השחקנים נתקע כאשר לא נוצלו כל התורות האפשריים. מכאן, נקבל כי הזמן לא מנוצל היטב מפני שייתכן והמשחק יסתיים במספר תורות שקטן מהחסם וכך נקבל כי יכולנו לספק לכל תור יותר זמן ממה שניתן לו בהתחלה לפי האלגוריתם.
2. החלוקה אחידה בין כל התורות – בתורים הראשונים של המשחק, נדרש זמן חישוב רב יותר מאחר וקיימות אפשרויות תנועה רבות, שכן הלוח מכיל מספר רב של משבצות פנויות יחסית. לעומתם, בתורות האחרונים מספר המשבצות הפנויות הוא מועט ולכן מספר אפשרויות התנועה מועט. על כן, נסיק כי התורות הראשונים זקוקים ליותר זמן ריצה על מנת לחשב אסטרטגיה טובה.

כעת, נציע את האלגוריתם הבא לניהול מחוכם יותר של משאבי הזמן:

1. חישוב הזמן של תור יתבצע באופן הבא:

$$Move\ time\ =\ factor \cdot time\ left$$

כאשר $factor$ הוא ערך הנמצא בתחום $(0,1)$

לפי ביצוע ניסויים בהם הרצנו משחקים עם ערכים שונים, קיבלנו כי הערך המיטבי הוא כאשר $factor = 0.3$. בדרך זו, הזמן מתחלק כך שהתורות הראשונים מקבלים נתח גדול יותר מהזמן, ועם התקדמות המשחק, התורות הבאים מקבלים נתח קטן יותר מהזמן.

המוטיבציה לבחירת אלגוריתם זה היא מכך שאנו משפרים את החסרונות שהצגנו לעיל ומתוך ההנחה העיקרית לפיה מספר אפשרויות התנועה בשלבים ההתחלתיים של המשחק הוא גדול ולכן חישוב

אסטרטגיה טובה תיקח יותר זמן משלבים מתקדמים של המשחק בהם מספר אפשרויות התנועה הוא מועט ולכן לא נדרש זמן רב לחישוב אסטרטגיה טובה.

שאלה 11:

תשובה:

כפי שלמדנו בהרצאה, תופעת האופק היא תופעה בה אלגוריתם מוגבל משאבים בוחר צעדים "סתמיים" כדי לדחות "צרות" מעבר לאופק החיפוש. כלומר, במשחקים רבים, מספר המצבים או המיקומים האפשריים עבור שחקן כלשהוא הוא עצום והאלגוריתמים בנויים באופן שהם לא יכולים לחפש את כולם וכך למעשה הם מוגבלים לחיפוש עד עומק מסוים בעץ המשחק. לפיכך, עבור אלגוריתם המחפש רק עד עומק מסוים, קיימת אפשרות שהוא יעשה מהלך מזיק, אך ההשפעה לכך אינה נראית לעין מאחר והאלגוריתם אינו מחפש לעומק השגיאה, כלומר מעבר ל"אופק שלו".

הפתרון המוצג בהרצאה לבעיה זו הוא על ידי הרחבת אלגוריתם החיפוש באמצעות "חיפוש שקט"(העמקה סלקטיבית – חיפוש עד רגיעה). בדרך זו, פורשים עץ מלא עד עומק מסוים אך כאשר המצב בעלה אינו "שקט", מבצעים העמקה כאשר להעמקה זו ישנם קריטריונים שונים המשמשים להחלטה על העמקה. בדרך זו לאלגוריתם החיפוש יש יכולת לחפש מעבר לקו האופק שלו אחר סוג מסוים של מהלכים החשובים ביותר למצב המשחק, כמו לכידות בשחמט. כמו כן, שכתוב של פונקציית ההערכה עבור צמתי העלים וניתוח צמתים נוספים, עשוי לפתור בעיות רבות באפקט האופק. לסיכום, הפתרון הוא הגדרת חלק מהעלים בעץ שפיתחנו כעלים "לא שקטים" כך שהגדרה זו תוביל לכך שנעמיק בפיתוח עלים אלו, גם במקרים בהם הגענו לעומק המקסימלי(הגדרת עלים אלו נעשית בהתאם לקריטריונים כפי שציינו לעיל). בדרך זו, הסוכן יוכל להקטין את הסיכויים לבחירת צעד הטוב ביותר בטווח הקצר אך בטווח הארוך עלול לפגוע בו.

כעת, נציג שני מצבים בלוח בהם יהיה כדאי להשתמש בפתרון זה:

דוגמא ראשונה:

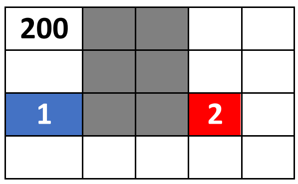
שחקנים: שחקן 1 מוגדר להיות השחקן הראשי כסוכן ה-$Minimax$ שזהו תורו(מחושב עבורו ערך ה-Max)

עומק: 3

ניקוד עבור כל שחקן בשלב זה של הלוח: 0

נקודות קנס: 300

מצב התחלתי של הלוח: כפי שמתואר בתרשים כאשר כל ערך הגדול מ-2 מבטא פרי והניקוד שלו.



הסבר:

בדוגמא זו, כתלות בהיוריסטיקה ייתכן והסוכן יקבל כי האסטרטגיה הטובה ביותר עבורו היא לעלות למעלה במטרה להשיג את הפרי שנמצא במרחק 2 משבצות ממנו. כעת, היריב יבצע צעד כלשהו(ימינה למשל) ואז השחקן שלנו(1=) יגיע לעומק 3 שבו ייתכן והוא יחליט כי האסטרטגיה הטובה ביותר עבורו תהיה להמשיך לכיוון הפרי וכך להרוויח את 200 הנקודות שהוא נותן. ואכן, בטווח הקצר אסטרטגיה זו הגיונית והטובה ביותר עבורו שכן הוא רוצה להרוויח כמה שיותר נקודות, אולם בטווח הארוך מצב זה יביא לחסימה שלו ולסיום המשחק, לכן הוא יצטרך לספוג קנס ולהורדה של 300 נקודות.

אם כן, נקבל כי המשחק יסתיים בניצחונו של השחקן היריב(2=) שכן הניקוד של שחקן 1 יהיה שווה ל-(-100) ועבור שחקן 2 הניקוד יהיה שווה לאפס, אך עדיין יתקיים $-100 < 0$ ולכן שחקן 2 ינצח.

מכאן, נוכל להסיק כי אם שחקן 1 היה ממשיך לעומק גבוה יותר מ-3, הוא היה מגלה כי צעד זה לא כדאי עבורו.

<u>דוגמא 2:</u>
שחקנים: שחקן 1 מוגדר להיות השחקן הראשי כסוכן ה-$Minimax$ שזהו תורו(מחושב עבורו ערך ה-Max)
עומק: 3
ניקוד עבור כל שחקן בשלב זה של הלוח: 0
נקודות קנס: 25
מצב התחלתי של הלוח: כפי שמתואר בתרשים כאשר כל ערך הגדול מ-2 מבטא פרי והניקוד שלו.

10					
10	**1**			**2**	

<u>הסבר:</u>
בדוגמא זו, כתלות בהיוריסטיקה ייתכן והסוכן יקבל כי האסטרטגיה הטובה ביותר עבורו היא לפנות בכיוון $\Leftarrow$, כלומר שמאלה וכך להרוויח את 10 הנקודות שהפרי נותן לו. כעת, היריב יבצע צעד כלשהו ואז השחקן שלנו(=1) יגיע לעומק 3 שבו ייתכן והוא יחליט כי האסטרטגיה הטובה ביותר עבורו תהיה להמשיך מעלה לכיוון הפרי הבא ולהרוויח את 10 הנקודות שהוא נותן. ואכן, בטווח הקצר אסטרטגיה זו הגיונית והטובה ביותר עבורו שכן הוא רוצה להרוויח כמה שיותר נקודות, אולם בטווח הארוך מצב זה יביא לחסימה שלו ולסיום המשחק, לכן הוא יצטרך לספוג קנס ולהורדה של 25 נקודות.
אם כן, נקבל כי המשחק יסתיים בניצחונו של השחקן היריב(=2) שכן הניקוד של שחקן 1 יהיה שווה ל-(-5) ועבור שחקן 2 הניקוד יהיה שווה לאפס, אך עדיין יתקיים $-5 < 0$ ולכן שחקן 2 ינצח.
מכאן, נוכל להסיק כי אם שחקן 1 היה ממשיך לעומק גבוה יותר מ-3, הוא היה מגלה כי צעד זה לא כדאי עבורו.

<u>**סיכום:**</u>
קיבלנו כי שתי דוגמאות אלו מבטאות את אפקט האופק כך שאילו היינו משתמשים בפתרון שהצגנו לאפקט האופק, שימוש בעלים ה"לא שקטים" היה מגלה כי בחירת צעדים אלו לא כדאית בטווח הארוך.

שאלה 12:

תשובה:

על מנת לשפר את יעילות ההרצה החוזרת, נרצה לגרום לאלגוריתם באמצעות פעולות פשוטות יחסית להיות מיודע יותר וכך הוא יוכל לנצל את מידע בכדי לשפר את זמני הריצה.

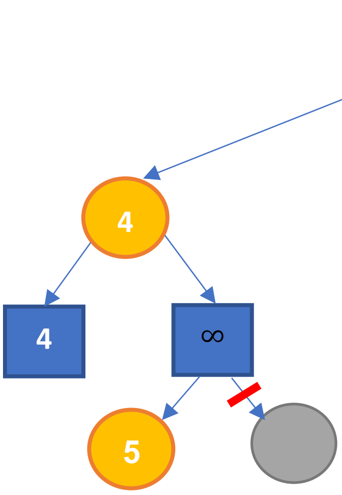
לכן, נבצע שינוי פשוט באלגוריתם על ידי הצבת ערך המקסימום גם ב-α וגם ב-β, כלומר:

$$Maximum\ value = \alpha = \beta$$

כך, נעביר כלפי מטה בקריאה הרקורסיבית חסם אחד על α ו-β בהתאם לגיזום $\alpha - \beta$.

שינוי זה יביא לכך שיהיו יותר גיזומים וכך ישתפרו זמני הריצה.

על מנת לתאר את התנהגות האלגוריתם המתוקן, נשתמש בעץ המצבים הבא(בדומה לעץ בתרגול):

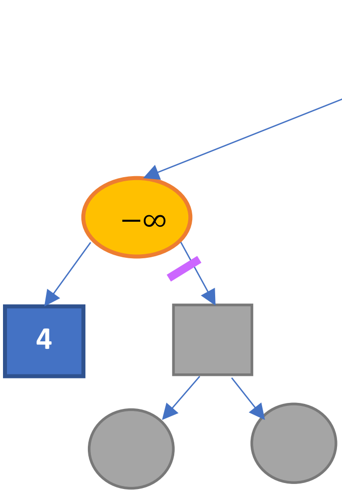


הסבר:

כפי שראינו בתרגול, ריבוע כחול מסמן מצב של בחירת מקסימום, ריבוע כתום מסמן מצב של בחירת מינימום ועיגול אפור מייצג גיזום ענף זה, החל ממצב זה ועד לעלים. על כן, כפי שאנו רואים בדוגמא זו, גיזום ענפים זה תואם לאלגוריתם גיזום $\alpha - \beta$ המקורי.

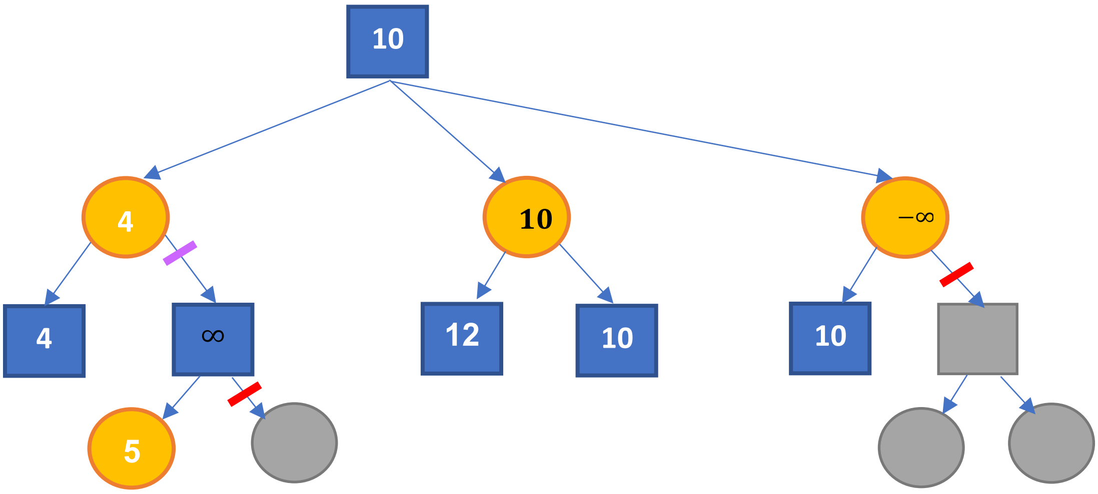
לאחר השיפור שביצענו באלגוריתם, נקבל כי הצומת המכיל את הערך 4 יהיה קטן יותר מערך ה-α ולכן ערכו לא ישתנה ושאר הבנים יגזמו מאחר ולא ייתכן כי קיים חסם מינימלי יותר עבור ענף זה. מכאן, נקבל כי הגיזום גדול יותר במקרה זה ולכן נקבל שיפור בזמן ההרצה.

מכאן, נקבל את עץ המצבים הבא:



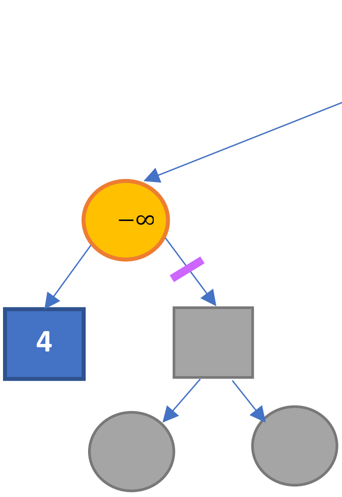
כעת, נתאר את המקרה הכללי, המקרה הטוב ביותר והמקרה הרע ביותר:

- עבור המקרה הכללי, בעץ אפשרויות לא מהונדס נצפה כי לאחר ביצוע השיפור באלגוריתם, זמן הריצה יהיה יעיל יותר אך עדיין נמצא מסלולים שנצטרך לפתח עד הסוף. נתאר זאת באמצעות הדוגמא הבאה:

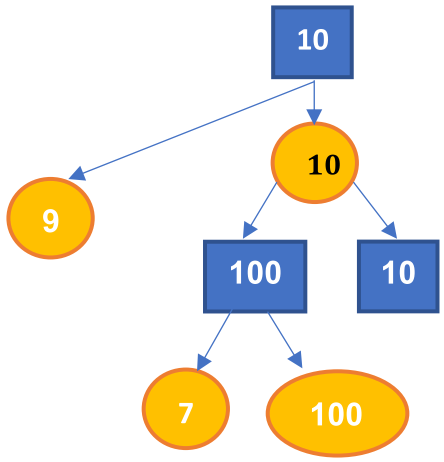


בדוגמא זו, ניתן לראות כי קיימים בעץ המצבים גיזומים השייכים לאלגוריתם המקורי – מסומן באדום, וגם גיזום שמגיע מהאלגוריתם המשופר – מסומן בסגול. כלומר, לאחר ביצוע האלגוריתם המשופר נקבל כי קיים גיזום נוסף ולכן נקבל זמן הרצה יעיל יותר. בדוגמא זו, הגיזום של העץ המשופר התבטא פעם אחת, אך בעצים נוספים אנו עשויים לקבל גיזומים משמעותיים יותר שיביאו לשיפור משמעותי בזמן ההרצה.

- עבור המקרה הטוב ביותר, הגיזום בעץ האפשרויות יתבצע בתחילתו של כל ענף וכך נקבל שיפור משמעותי בזמן ההרצה. כך למשל נוכל לראות את הגיזום אשר מתבצע בתחילתו של הענף בדוגמא הבאה:



- עבור המקרה הגרוע ביותר, הגיזום באלגוריתם השיפור יהיה זהה לגיזום המתבצע באלגוריתם המקורי ולכן לא נקבל זמן ריצה טוב יותר. במקרים אלו ייתכן כי לא ייקרה גיזום כלל או שייקרה גיזום זהה לאלגוריתם המקורי. כך למשל, בתרשים המוצג מטה עבור שני האלגוריתמים, המשופר והמקורי, נקבל כי בזמן הגילוי אין אף ערך שמקיים את התנאים הנדרשים לגיזום ולכן לא יתבצע גיזום כלל וכך לא נקבל שיפור בזמן הריצה:



שאלה 13:

ידוע שהפונקציה היוריסטית h באלגוריתם RB-Expectimax מקיימת $\forall s, -5 \le h(s) \le 5$. על מנת לבצע גיזום לאלגוריתם זה נבצע את הגיזום הבא:

תחילה, נגדיר את הסימונים הבאים:

- $k(v_i)$ – ההסתברות שתצא אותה התוצאה כאשר ערך זה קיים לכל בן של צומת הסתברותי.
- $algoExpectingMax(v_i)$ -הערך שחוזר מהאלגוריתם הנתון עבור הצומת v_i.
- $val(v_i)$ – ערך הצומת המוגדר להיות תוצאת המכפלה בין $k(v_i)$ לערך שחוזר מהאלגוריתם עבור אותו הצומת, כלומר ל- $algoExpectingMax(v_i)$.

כעת, נפריד לשני מקרים את פעולת הגיזום המוצעת לאלגוריתם זה:

<u>מקרה הראשון – נבדוק האם הבנים של הצומת ההסתברותי הם צמתי מינימום או מקסימום:</u>

- במידה והבנים של הצומת ההסתברותי הינם צמתי מקסימום, אנו יודעים כי הערך $algoExpectingMax(v_i)$ של צומת v_i שווה לערך הסכום של ערכי $val(v_i)$ של כל בניו של הצומת v_i. לכן, נגדיר:
  - $sumK(v_i)$ להיות סכום ערכי $k(v_i)$ של כל בנין של הצומת v_i שפותחו.
  - $sumVal(v_i)$ להיות סכום ערכי $val(v_i)$ של כל בנין של הצומת v_i שפותחו.

  כעת, נוכל לחסום מלמטה את הערך $algoExpectingMax(v_i)$ על ידי הערך

  $$-5 \cdot (1 - sumK(v_i)) + sumVal(v_i)$$

  כלומר, נקבל כי מאחר ואנו יכולים לחסום מלמטה את ערך ה- $algoExpectingMax(v_i)$ השווה ל- $sumVal(v_i)$, במידה והחסם התחתון בנוסף לסכום שחישבנו עבור שאר הילדים <u>גדול</u> מהערך β, נוכל להסיק כי צומת האב של הצומת v_i שהוא מינימום יעדיף את ערך הצומת שנמצא באותה רמה שלו(=אחיו) ולכן לא נמשיך לפתח את ילדי צומת זה.

  לכן, במידה ומתקיים

  $$\beta \le -5 \cdot (1 - sumK(v_i)) + sumVal(v_i)$$

נבצע גיזום של הבנים של הצומת v_i שעוד לא פיתחנו, שהרי כפי שהסברנו אין סיבה לפתח אותם כי לא ייתנו לנו ערך מינימלי יותר.

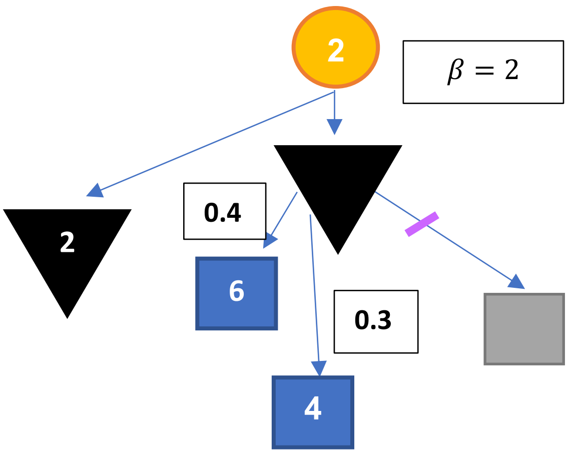
- באופן סימטרי, במידה והבנים של הצומת ההסתברותי הינם צמתי מינימום, אנו יודעים כי הערך $algoExpectingMax(v_i)$ של צומת v_i שווה לערך הסכום של ערכי $val(v_i)$ של כל בניו של הצומת v_i , כלומר ל- $sumVal(v_i)$. באופן דומה, נשתמש בהגדרות מהנקודה הקודמת ונקבל כי נוכל לחסום ערך זה מלמעלה על ידי הערך $5 \cdot (1 - sumK(v_i)) + sumVal(v_i)$ כלומר, נקבל כי מאחר ואנו יכולים לחסום מלמטה את ערך ה- $algoExpectingMax(v_i)$, במידה והחסם התחתון בנוסף לסכום שחישבנו עבור שאר הילדים <u>קטן</u> מהערך α, נוכל להסיק כי צומת האב של הצומת v_i שהוא מקסימום יעדיף את ערך הצומת שנמצא באותה רמה שלו(=אחיו) ולכן לא נמשיך לפתח את ילדי צומת זה.
לכן, במידה ומתקיים
$$\alpha \geq 5 \cdot \left(1 - sumK(v_i)\right) + sumVal(v_i)$$
נבצע גיזום של הבנים של הצומת v_i שעוד לא פיתחנו, שהרי כפי שהסברנו אין סיבה לפתח אותם כי לא ייתנו לנו ערך מינימלי יותר.

ונבצע גיזום של הילדים של הצומת v_i שעוד לא פיתחנו, שהרי כפי שהסברנו אין סיבה לפתח אותם כי לא ייתנו לנו ערך מקסימלי יותר.

<u>מקרה שני -</u> עבור צמתים מקסימום ומינימום נשתמש בטכנית הגיזום של אלגוריתם $AlphaBeta$ כפי שלמדנו בהרצאה. כלומר, אלגוריתם זה ישתמש בפרמטרים α, β לאורך ביצוע אלגוריתם ה- $AlphaBeta$ כאשר פרמטרים אלו מעודכנים בצמתי מקסימום ומינימום בלבד.

<u>לשלמות ההסבר, נוסיף דוגמא שתמחיש את השימוש בגיזום זה:</u>

- עבור מקרה בו הבנים של הצומת ההסתברותי הינם צמתי מקסימום, נקבל עבור עץ המהלכים הבא את הגיזום:

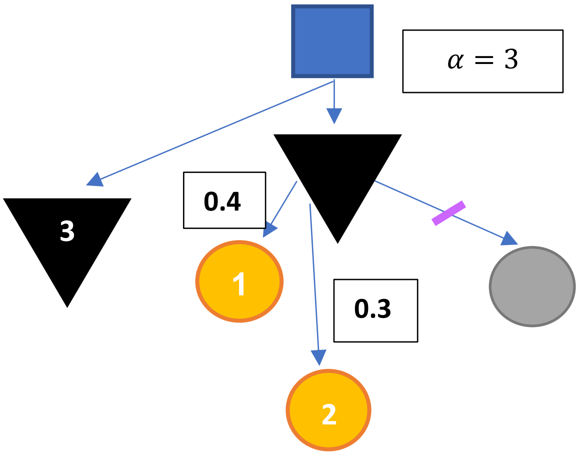


בדוגמא זו ניתן לראות כי בהינתן $\beta = 2$, מתקיים:

- $sumK(v) = 0.4 + 0.3 = 0.7$
- $sumVal(v) = 6 \cdot 0.4 + 4 \cdot 0.3 = 3.6$

ולכן:
$$-5 \cdot \left(1 - sumK(v_i)\right) + sumVal(v_i) = -5 \cdot (1 - 0.7) + 3.6 = 2.1 \geq 2 = \beta$$
ולכן ביצענו את הגיזום כפי שתואר באיור.

- עבור מקרה בו הבנים של הצומת ההסתברותי הינם צמתי מינימום, נקבל עבור עץ המהלכים הבא את הגיזום:



בדוגמא זו ניתן לראות כי בהינתן $\alpha = 3$, מתקיים:
- $sumK(v) = 0.4 + 0.3 = 0.7$
- $sumVal(v) = 1 \cdot 0.4 + 2 \cdot 0.3 = 1$

ולכן:
$$5 \cdot \left(1 - sumK(v_i)\right) + sumVal(v_i) = 5 \cdot (1 - 0.7) + 1 = 2.5 \leq 3 = \alpha$$
ולכן ביצענו את הגיזום כפי שתואר באיור.

<u>שאלה 14:</u>

<u>סעיף א':</u>

החלטתו של השחקן השפיעה על שאר B-1 החלטותיו בהמשך בכך שהוא סיים את זמן הריצה שניתן לו בתחילת המשחק, כלומר את כל M הדקות שניתנו לו ולכן הוא לא יוכל לבצע את שאר הפעולות בתורות הבאים. כעת, נראה מדוע הגענו למסקנה לפיה השחקן ניצל את כל הזמן שניתן לו:
- לפי תנאי השאלה, מספר הקריאות לפונקציה היוריסטית בחיפוש לעומק D באלגוריתם $Minimax$, יהיה B^D וזאת מאחר ונתון מקדם סיעוף קבוע B.
- בנוסף, בהנחה שרק הפונקציה היוריסטית נכנסת לחישוב ושאר הפעולות זניחות(מבחינת זמן ההרצה), נשתמש בנתון לפיו חיפוש לעומק D ייקח $\frac{M}{B}$ ונקבל כי זמן היוריסטיקה כפול מספר הקריאות לפונקציה היוריסטית בחיפוש לעומק זה(D=) ייקח $\frac{M}{B}$ דקות, כלומר:

$$\text{זמן היוריסטיקה} \cdot B^D = \frac{M}{B}$$

- כאשר השחקן מחליט כי בצעד הראשון הוא יבצע חיפוש לעומק D+1, נקבל כי מספר הקריאות לפונקציה היוריסטית בחיפוש לעומק D+1, בדומה לקודם, יהיה B^{D+1}.

אם כן, משילוב בין שלושת נקודות אלו, נקבל כי זמן המהלך של השחקן יהיה כעת:
$$\text{זמן היוריסטיקה} \cdot B^{D+1} = \text{זמן היוריסטיקה} \cdot B^D \cdot B \underbrace{=}_{\text{זמן היוריסטיקה} \cdot B^D = \frac{M}{B}} \frac{M}{B} \cdot B = M$$

כלומר, קיבלנו כי זמן המהלך של השחקן היה M דקות, שזהו גם הזמן שניתן לו למשחק כולו ולכן קיבלנו כי השחקן ניצל את כל הזמן שהוקצה לו עבור המשחק באשר ביצע מהלך זה.

סעיף ב':

i. קשתות שהשחקן צריך לשמור עליהן על מנת לשפר את מצבו הן הקשתות שהתקבלו לאחר בחירת הצעד הראשון, כלומר, נקבל קשתות כמספר הקשתות בעץ בעומק D. נשים לב כי את הקשת הראשונה לא צריך לשמור מאחר וזהו הצעד שביצע השחקן במהלך הראשון ולכן נשמור את הקשתות החל מהשלב השני על עץ המינימקס. אם כן, עבור השחקן נוכל לשמור בכל צומת המייצגת צומת max את הקשת המייצגת את הצעד המוביל לאפשרות הטובה ביותר שהשחקן יכול לבצע.

עבור קשתות היריב, השחקן יצטרך לשמור את כל הקשתות המייצגות את אפשרויות המשחק, כלומר הצעדים של היריב(שהרי בחירתו לא תלויה בנו), זאת מאחר ונוכל להשתמש במידע זה בהמשך האלגוריתם. אם כן, עבור היריב נשמור B קשתות לכל צומת המייצגת צומת min וזאת מאחר ואנו לא יכולים לדעת מראש את התנהגות היריב, לכן נרצה שיהיו בידנו כל האפשרויות. נבחין כי כאשר התור הוא של סוכן ה- $Minimax$ הוא ירצה לבחור את ערך ה-$Minimax$ המקסימלי ולכן אנו לא נדרשים לשמור את ערכי הצעדים שאנו יכולים לבצע, שכן הערך החשוב ביותר כבר נמצא בידינו והוא ערך ה-$Minimax$.

אם כן נקבל כי מספר קשתות סוכן ה-$Minimax$ שיש לשמור הוא $\sum_{i=1}^{\lceil D/2 \rceil} B^i$ ומספר קשתות היריב שיש לשמור הן $\sum_{i=1}^{\lfloor D/2 \rfloor} B^i$ כאשר B^i מוגדר להיות מספר צמתי ה-max ברמה ה-i בעץ.

ii. בהשוואה לסעיף א', השחקן כעת שיפר את האסטרטגיה שלו בכך שיש לו מידע שימושי שיוכל לעזור לו להתקדם במשחק. כלומר, כעת השחקן ינצל את התורות שלו ויבצע מהלכים חכמים יותר שמתבססים על המידע שהשיג באמצעות הפיתוח שביצע, שהרי קודם לא נותר לו זמן להריץ את היוריסטיקה ולכן צעדיו היו פחות מושכלים. כעת, השחקן יוכל לקבל מידע על כל אחד מהמהלכים האפשריים הבאים עבור $\lfloor D/2 \rfloor$ התורים הקרובים שלו, בניגוד לקודם שלא היה לו כלל מידע על המהלכים האפשריים. בהנחה כי מתקיים $\lfloor D/2 \rfloor < B - 1$, לאחר $\lfloor D/2 \rfloor$ תורות השחקן יצטרך להמשיך את המשחק מבלי להשתמש בהיוריסטיקה.

iii. כעת, נשווה בין האלגוריתם המשופר לבין אלגוריתם מינימקס הרגיל:

מבחינת זמני ביצוע, כפי שהראינו, החישוב של עץ המינימקס עבור עומק D+1 רק בצעד הראשון שווה לזמן החישוב של עץ המינימקס בעומק D עבור כל אחד מ-B המהלכים של השחקן. בנוסף, מבחינת אופי המשחק, שחקן המשתמש באלגוריתם המשופר יבצע מהלכים טובים יותר מאשר שחקן המינימקס הרגיל וזאת בזכות ההעמקה הנוספת במהלך הצעד הראשון אשר נתנה לו מידע שאפשר לו לבצע צעד חכם יותר. כלומר, ב- $\lfloor D/2 \rfloor$ התורות הראשונים, השחקן יידע מהם הצעדים הבאים שלו ללא שימוש בהיוריסטיקה בזכות אותם פיתוחים שפיתח במהלך הראשון. לעומתו, שחקן המינימקס הרגיל יצטרך בכל תור לבצע חיפוש לעומק D.

נשים לב כי במידה וההעמקה הנוספת לא הוסיפה מידע בעל ערך עבור השחקן, נקבל כי שחקן המינימקס המשופר יהיה פחות טוב מאשר שחקן המינמקס הרגיל. לכן, רק במידה וההעמקה עד לעומק D+1 בתור הראשון הוסיפה לו מידע שהשפיע על הצעד הראשון שלו ושלא היה ניתן לדעת אותו(המידע) באמצעות העמקה לעומק D בתור הראשון, השחקן המשופר יהיה טוב יותר.

בנוסף, נבחין כי אכן בתור הראשון לשחקן המשופר ישנו יתרון על השחקן הרגיל מאחר והוא רואה צעד אחד קדימה, בתור השני הראיה של שני השחקנים היא לעומק זהה ולכן סביר כי יבצעו מהלכים דומים אך החל מהתור השלישי והלאה שחקן המינימקס הרגיל יעמיק לעומק קבוע D והשחקן המשופר ידע את תוצאות הפיתוחים בכל תור i לעומק $D + 1 - i$ ולכן ייתכן כי במקרה זה, לשחקן המינמקס הרגיל יהיה ייתרון על פני השחקן המשופר, שכן הוא מעמיק יותר ולכן הוא עשוי לקבל מידע שימושי שיוביל לצעדים מושכלים יותר.

שאלה 15:

תשובה:

היוריסטיקה שקבענו עבור שחקן ה-$Minimax$ היא אותה אחת שתיארנו בשאלה 4:

מרכיבי היוריסטיקה:

- $Game\ score$ – משתנה זה הינו ערך בתחום $[-1,1]$ המייצג את ניקוד המשחק של השחקן שלנו אל מול השחקן היריב. הערך מתחשב בפירות וגם בערך ה-$penalty$ כך שהערך הטוב ביותר הוא אחד והערך הגרוע ביותר הוא 1-.
  הערה: בהתחלה, היוריסטיקה שלנו כללה ערך לפירות שמחושב על פי מרחקי מנהטן, ניתן לראות ערך זה בשאלה 4 אותו כינינו בשם $Fruit\ score\ heuristic$. אולם, חישוב ערך זה התגלה כלא יעיל מבחינת סיבוכיות זמן וקיבלנו תוצאות טובות יותר כאשר הסרנו אותו מחישוב היוריסטיקה. בדרך זו, הרווחנו זמן לצורך העמקה בעץ החיפוש שיוביל למציאת אסטרטגיות טובות יותר לשחקן שלנו. מסקנה זו קיבלנו לאחר ביצוע של ניסויים רבים על היוריסטיקה ותהליך למידה שהביא אותנו לכך.
- $Reachable\ node\ score$ - ההפרש בין מספר המיקומים הנגישים לשחקן 1 לבין מספר המיקומים הנגישים לשחקן 2, מנורמל לערך בתחום $[-1,1]$.
- $Step\ score$ – ההפרש בין מספר הצעדים האפשריים של שחקן 1 לבין מספר הצעדים האפשריים עבור שחקן 2, כאשר ערך זה מנורמל לערך בתחום $[-1,1]$.
- $Road\ score$ - ההפרש בין הדרך הארוכה ביותר האפשרית של שחקן 1 לבין הדרך הארוכה ביותר האפשרית של שחקן 2, כאשר ערך זה מנורמל לערך בתחום $[-1,1]$.

הנוסחא עבור היוריסטיקה:

$$h(s) = Game\ score \cdot 0.85 + Reachable\ node\ score \cdot 0.05 + Step\ score \cdot 0.05 + Road\ score \cdot 0.05$$

כאשר המשקולות לעיל נבחרו לאחר ביצוע ניסויים רבים והסקת מסקנה כי ערכים אלו מספקים את הביצועים הטובים ביותר.

שאלה 16:

תשובה:

לאחר ביצוע מספר רב של ניסויים, החלטנו להשתמש בשחקן בעל זמן גלובלי והיוריסטיקה פשוטה ככל שניתן.

היתרון שמקנה השימוש בהיוריסטיקה פשוטה הוא זמן ריצה נמוך המאפשר להגיע לעומקים עמוקים יותר בעץ החיפוש. מתצפיות שעשינו, ככל שהעץ עמוק יותר, הסיכוי של השחקן לנצח – גדל. עבור אלגוריתם החיפוש, החלטנו להשתמש באלגוריתם $\alpha - \beta$ מאחר ואלגוריתם זה חסכוני בזמן הריצה. כלומר אופן פעילותו של שחקן התחרות שלנו הוא להשתמש בהיוריסטיקה פשוטה בסיבוכיות $O(1)$ ולנצל את כל זמן התור לצורך העמקת עץ החיפוש כמה שאפשר.

הפונקציה היוריסטית שהגדרנו מחושבת על פי הפרמטר:

$Game\ score$ – תוצאת המשחק היא ערך בתחום $[-1,1]$ המייצג את ניקוד המשחק כך שהערך הטוב ביותר הוא אחד והערך הגרוע ביותר הוא 1-. הפונקציה מוגדרת באופן הבא:

$$h(s) = \frac{Score(player = 1) - Score(player = 2)}{\sum_{i=1}^{2} ABS(Score(player = i))}$$

כאשר $Score$ מוגדר להיות הניקוד של השחקנים 1 או 2.

שאלה 17:

תשובה:

עבור מקרה הגבלת הזמן לתור:

ניהלנו את זמן ריצת הפונקציה $make\_move$ באופן הבא:

התחלנו את הריצה בעומק 1 ומדדנו את הזמן שלוקח לאיטרציה זו. לאחר מכן, על מנת לבדוק האם להחזיר תשובה כעת או לבצע העמקה נוספת, השתמשנו בבדיקה הבאה:

$$time\_until\_now + next\_iteration\_max\_time < time\_limit$$

כאשר:

- $time\_until\_now$ - זמן הריצה עד כה.
- $Next\_iteration\_max\_time$– מחושב לפי זמן האיטרציה הקודמת כפול 4(בחרנו לכפול את זמן האיטרציה הקודמת בארבע בכדי להבטיח שיהיה לנו מספיק זמן לבצע העמקה נוספת).
- $Time\_limit$- זמן מוגבל לתור.

כפי שניתן לראות לפי התנאי, במידה והתוצאה קטנה מהזמן המוקצב לתור, המשכנו לעומק נוסף מהעומק הקודם(כלומר, פלוס אחד לעומק הקודם). כך המשכנו באיטרציות הבאות עד שיצאנו מלולאת ה-while שבה אנו כל פעם מבצעים העמקה נוספת באלגוריתם ה-Minimax. בדרך זו בכל איטרציה אנו בודקים האם קיים מספיק זמן לבצע העמקה נוספת, תוך עדכון של הזמן עד כה ואת זמן האיטרציה הקודמת.

לצורך הסבר נוסף, הוספנו את הקוד:

```
    depth = 1
    max_move, max_val = self.search_alg.search(self, depth, True)
    last_iteration_time = t.time() - time_start
    next_iteration_max_time = 4 * last_iteration_time
    time_until_now = t.time() - time_start
    while time_until_now + next_iteration_max_time < time_limit:
        depth += 1
        iteration_start_time = t.time()
        last_good_move = max_move
        max_move, val = self.search_alg.search(self, depth, True)
        if val == float('inf'):
            break
        if val == float('-inf'):
            max_move = last_good_move
            break
        last_iteration_time = t.time() - iteration_start_time
        next_iteration_max_time = 4 * last_iteration_time
        time_until_now = t.time() - time_start
self.perform_move(maximizing_player=True, move=max_move)
return max_move
```

עבור מקרה הגבלת זמן גלובלי:

ניהלנו את זמן ריצת הפונקציה make_move באופן זהה למקרה הגבלת זמן לתור, פרט לכך שהלולאה רצה עד המגבלה הבאה:

$$limit = time\_left \cdot time\_factor$$

כאשר:

- $Time\_left$ – הזמן שנשאר למשחק, כלומר ההפרש בין הזמן הגלובלי לבין זמן ריצת המשחק עד כה.
- $Time\_factor$ – הערך 0.3(כפי שהוסבר בשאלה 10).

כלומר, בדרך ניהול זו סיפקנו זמן ריצה גבוה יותר לתורות הראשונים מאשר לתורות המתקדמים יותר וזאת כפי שהסברנו בשאלה 10, כי מספר אפשרויות התנועה בשלבים הראשונים של המשחק גדול יותר מאשר מספר אפשרויות התנועה בשלבים המתקדמים של המשחק ולכן התורות הראשונים דורשים זמן גבוה יותר לקבלת אסטרטגיה טובה.

לצורך הסבר נוסף, הוספנו את הקוד:

```
    while time_until_now + next_iteration_max_time < limit:
        depth += 1
        iteration_start_time = t.time()
        last_good_move = max_move
        max_move, val = self.search_alg.search(self, depth, True)
        if val == float('inf'):
            break
        if val == float('-inf'):
            max_move = last_good_move
            break
        last_iteration_time = t.time() - iteration_start_time
        next_iteration_max_time = 4 * last_iteration_time
        time_until_now = t.time() - time_start
    self.time_left -= time_until_now
self.perform_move(maximizing_player=True, move=max_move)
if DEBUG_PRINT:
    print(f"move chosen is {max_move}\n"
          f"time limit is {limit}\n"
          f"time left is {self.time_left}")
return max_move
```

חלק ז':

שאלה 18:

תשובה:

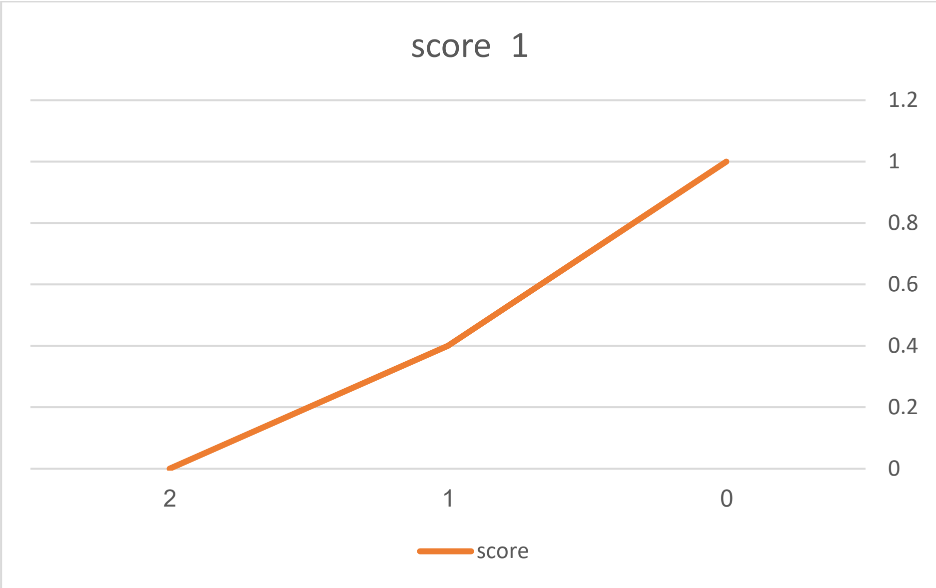
לאחר הרצת מספר משחקים בין סוכן ה-$Minimax$ לבין סוכן ה- $AlphaBeta$, התוצאות שקיבלנו אכן תואמות את הציפיות שלנו. הציפיות שלנו היו שסוכן ה-$AlphaBeta$ ינצח ביותר משחקים מסוכן ה-$Minimax$ מאחר וסוכן ה-$AlphaBeta$ חוסך בחישובים מיותרים שאינם נדרשים לקבלת אסטרטגיה טובה. לכן, הסוכן $AlphaBeta$ מנצל את הזמן שהוא חוסך בניסיון להעמיק את עץ החיפוש ובכך להשיג אסטרטגיה טובה יותר.

אם כן, קיבלנו כי לסוכן ה-$AlphaBeta$ יש הסתברות גבוהה יותר לנצח מסוכן ה- $Minimax$, בהתאם לציפיותינו.

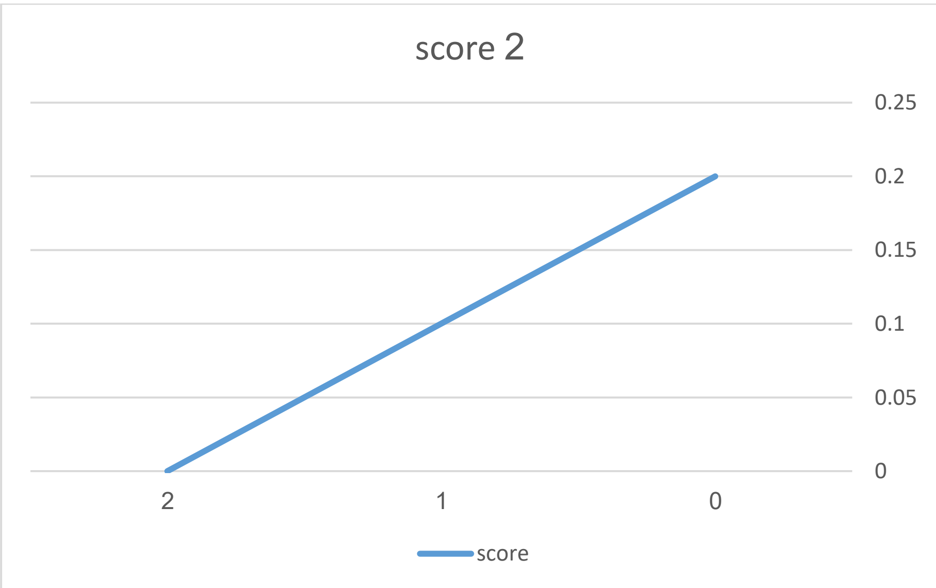
שאלה 19:

תשובה:

תוצאת הניסוי שקיבלנו כאשר ציר ה- x מוגדר להיות ההפרש בין עומקי החיפוש של $LightABPlayer$ ו- $HeavyABPlayer$ וציר ה-y מוגדר להיות ציון השלב:



כאשר ביצענו את הניסוי בפעם השנייה עבור חיפוש בעומק 2 של $HeavyABPlayer$ ו- $LightABPlayer$ מחפש לעומקים 2,3 ו-4, קיבלנו את הגרף הבא:



ניתן לראות לפי תוצאות הניסוי כי ככל העומק של הסוכן $LightABPlayer$ גבוהה יותר(כלומר עמוק יותר), יש לו יתרון על הסוכן $HeavyABPlayer$, למרות שהיוריסטיקה של הסוכן $LightABPlayer$ פשוטה יותר.

היתרון של הסוכן $LightABPlayer$ נובע מכך שעץ החיפוש שלו עמוק.

ניתן לראות כי בניסוי השני סוכן ה-$LightABPlayer$ היה משמעותית טוב יותר מסוכן ה-$HeavyABPlayer$ למרות שההפרשים בעומק היו זהים בין שני הניסויים. סיבה אפשרית לכך היא שהעומק של סוכן ה-$HeavyABPlayer$ הוגבל ל-2 ובכך המגבלה זו פגע ביכולת שלו להעריך את הצעד האופטימלי.