1. נתאר את המיקום של השחקן כוויקטור בינארי . לפני התשובה הסופית נגדיר:
   1. את הקבוצה .
   2. את הפונקציה באופן הבא:   
      כאשר הוא מיקום השחקן בלוח . ו- הוא הערך השמור במיקום בלוח .
   3. את הפונקציה באופן הבא:  
      כאשר וגם .

ואז ניתן להגדיר את הפונקציה באופן הבא:

1. נסמן ב- – כלומר השחקן השני  
   1. ניצחון במצב s עבור השחקן מתקיים אמ"מ
   2. מצב תיקו אמ"מ   
      (לא משנה תור מי במשחק)
2. מקדם הסיעוף הוא בין 0 ל-4 (כולל). וזה כיוון שבכל מצב, גם לשחקן שלנו וגם ליריב, או שהוא מצב סופי, ואז אנחנו נמצאים המצב עלה בעץ המשחק (מקדם סיעוף 0) או שקיים לפחות צעד אחד אפשרי מבין הצעדים החוקיים, ולכל היותר 4 (4 הכיוונים חוקיים).
3. בכל צעד – מבין כל הצעדים החוקיים - השחקן בוחר את הצעד המבטיח מספר צעדים חוקיים עוקבים מינימאלי (חוץ מ-0 צעדים חוקיים, כלומר הוא תמיד מעדיף לפחות צעד עוקב אחד ולא נתקע).  
   יתרונות עבור אסטרטגיה זו:   
   האסטרטגיה יעילה, לא לוקח זמן להחליט מהו הצעד. בנוסף אסטרטגיה זו טובה במקרה שהשחקן שלנו מנותק מהשחקן היריב, כלומר לכל שחקן יש את "השטח הפרטי" שלו. במקרה הזה אסטרטגיה זו מבטיחה שנכסה את החלק המקסימלי שניתן לכסות בשטח זה.  
   חסרונות עבור אסטרטגיה זו:  
   האסטרטגיה לא לוקחת בחשבון את הצעדים של היריב. כפי שתיארנו ביתרונות, אסטרטגיה זו טובה כאשר השחקנים מנותקים אך זה לא המצב ולכן יכול להיות שהיריב יפגע בצעדים של השחקן שלנו.

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
|  |  |  |  |  |
|  |  |  |  |  |
|  |  |  |  |  |
|  |  |  |  |  |

1. נצייר את הלוח הבא

נניח כי השחקן simple player הוא השחקן האדום, ונניח כי הכחול מתחיל. נשים לב כי לפי האסטרטגיה של השחקן simple player הוא מסיים שורה-שורה, כלומר בהתחלה הוא עובר ימינה (כי יהיה לו צעד חוקי עוקב אחד לעומת שני צעדים אם הוא הולך למטה). בתור הבא הוא עובר למטה (הצעד היחיד) ואז שמאלה (אותו עיקרון של צעד אחד עוקב לעומת שני צעדים..) וממשיך כך בשורות.   
במקרה והשחקן הכחול גם הוא מצליח לכסות את כל החלק שלו, עדיין האדום מנצח.  
ולכן במקרה הזה השחקן simple player פועל בצורה אופטימלית.

1. נציג את החיסרון של היוריסטיקה הנ"ל עם הלוח הבא:

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
|  |  |  |  |  |  |
|  |  |  |  |  |  |
|  |  |  |  |  |  |
|  |  |  |  |  |  |
|  |  |  |  |  |  |

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
|  |  |  |  |  |  |
|  |  |  |  |  |  |
|  |  |  |  |  |  |
|  |  |  |  |  |  |
|  |  |  |  |  |  |

לשם פשטות נניח כי אנו עם הגבלה של עומק 1. נניח כי הכחול מתחיל. במקרה הזה השחקנים יבצעו את הצעדים הבאים:

האדום יצליח לכסות את כל השטח שלו בשורות. לעומת זאת כאשר הכחול חסום, האדום עדיין נמצא בתא המסומן באדום, ולכן השחקן עם היוריסטיקה מפסיד. לעומת זאת באסטרטגיה אופטימלית אחרת (למשל simple player), שני השחקנים היו נתקעים והיה מתקבל תיקו.

1. עבור שחקן ומצב נגדיר את 3 המרכיבים הבאים:  
   1. מספר המשבצות הפנויות הסמוכות לשחקן (מ-4 הצדדים)
   2. מספר המשבצות הלבנות **הישיגות** מ- בלוח .
   3. האם קיים מסלול בין שני השחקנים או לא.

**הסבר:**

האסטרטגיה בה נשתמש כדי לנצח היא לנסות ולהתנתק מהיריב, וזה על מנת לנצל את המשבצות הישיגות מאיתנו ללא התערבותו והפרעתו של היריב. נעשה זאת ע"י חסימת המעבר ביניו. ולכן נשתמש במרכיב c, כאשר נרצה להגיע ל- 1-. ולכן נשתמש בערך .  
בנוסף להתנתקות הרצויה נרצה להיות בתוך שטח הגדול ככל היותר מהשטח של היריב, כך נגדיל את הסיכוי שהיריב נתקע ואנו נמשיך, ולכן נשתמש במרכיב b בצורה הבאה:  
 . כלומר נרצה למקסם את ההפרש הזה.  
כעת כדי להבטיח שנכסה כמה שיותר מהשטח שיש לנו נשתמש בשיטה הדומה לזו של , כאשר נרצה למזער את מספר המשבצות הסמוכות לנו (חוץ מ-0 כמובן) ולכן נשתמש בערך:  
הנוסחא הסופית:

1. ווריאציאת anytime contract של minimax מתמודדת עם ההגבלה בזמן. האלגוריתם ממשיך לחפש את הצעד האופטימלי עד שנגמר הזמן המוקצה לשחקן. העמקה הדרגתית בהקשר זה היא ההעמקה בעץ המשחק בהדרגה, כל עוד יש לנו זמן. כלומר בכל איטרציה, כל עוד אפשר נעמיק את עץ המשחק לרמה נוספת, כאשר נגמר הזמן שיש לנו אנו מחזירים את הצעד הטוב ביותר שקיבלנו מהאיטרציה האחרונה. ככל שאנחנו מצליחים להגיע לעומק גדול יותר נצליח לקבל תמונה יותר מדויקת לצעדים ולכן נצליח (בעזרת יוריסטיקה) להחליט באופן יותר שקול את הצעד שלנו.   
   הבעיה בשיטה זו, היא שבממוצע הזמן שלנו נגמר באמצע האיטרציה האחרונה, איטרציה זו צורכת משאבים פי משאבים (כאשר b הוא מקדם הסיעוף) מהאיטרציה הקודמת לה, שזה יחס מאוד גדול מזמן הריצה וזהו חישוב מבוזבז, כי אנחנו בסופו של דבר נחזיר את התוצאה מהאיטרציה הקודמת.
2. בכל איטרציה נשמור את ערך המינימקס של כל אחד מהבנים ברמה העליונה, באיטרציה האחרונה החצי הראשון של הבנים יהווה חיפוש עמוק יותר, ולכן ננצל את המשאבים כדי לראות לעומק גדול יותר חלקית.

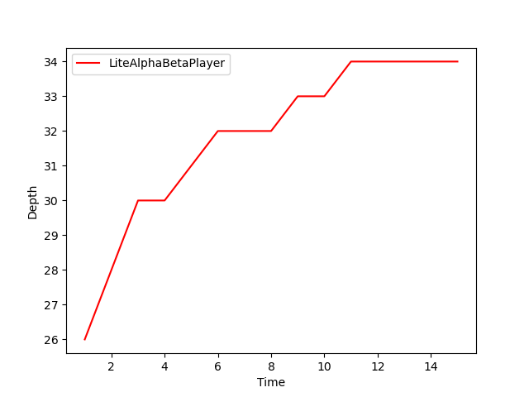
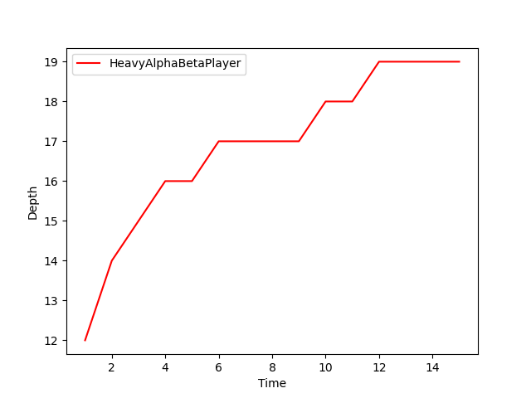
הבסיס שעומד מאחורי פיתוח הנוסחה הנ״ל הוא המשפט מההרצאה שאומר שמספר העלים בתוכנית פעולה מותנית עבור עץ משחק בעומד D ומקדם סיעוף B הוא .

מכאן ניתן להבחין:

* אם העומק D הבא מתחלק ב-2 **עם שארית**, אז מספר העלים יישאר זהה, דבר שהוא הגיוני כיוון שבעומק D שכזה אנחנו מגיעים למצב שבא ישר אחרי פעולה של הסוכן שלנו וזה כאמור מצב שהוא בן יחיד ולכן כאמור מספר העלים לא משתנה, אלה רק העומק עולה ב1 ולכן, הזמן שייקח ל-MiniMax בעומק כזה הינו הזמן של האיטרציה הקודמת **ועוד** זמן קיפול הרקורסיה עבור עומק אחד עמוק יותר.
* אם העומק D הבא מתחלק ב-2 **ללא שארית**, אז מספר העלים יגדל פי 2 (עפ״י הנוסחה) ובנוסף העומק יגדל ב1 ולכן, הזמן שייקח ל-MiniMax בעומק כזה הינו הזמן של האיטרציה הקודמת **ועוד** זמן קיפול הרקורסיה עבור עומק אחד עמוק יותר, כל זה כפול 2.

\* בחישוב זה אני יוצא מנקודת הנחה כי מתוך , 10% הוקדש לקיפול הרקורסיה חזרה אל השורש ולכן זמן הקיפול אצלי הוא .

1. סוכן Alpha-Beta צפוי להיות יעיל יותר בווריאציה הנ״ל כיוון שהוא ממקסם את זמן פיתוח המצבים עבור עומק D ולכן הוא יספיק להגיע אל עומק עמוק יותר לפני תום הזמן שלו ולכן הצעדים שהוא יבחר יהיו באופן כללי מדויקים יותר וכאלה שיקרבו אותו מהר יותר אל הניצחון.
2. לא. אם במקום להגביל כל סוכן בזמן, מגבילים אותו בעומק חיפוש אז אין משמעות להבדל בין הסוכנים.
3. היוריסטיקה שהשתמשנו בה עבור השחקן היא היוריסטיקה מסעיף 7, אסטרטגית ניתוק היריב.
4. ג
5. פונקצית make\_move() חישבה את "זמן הדדליין" של השחקו, היא הוסיפה לזמן הנוכחי את הזמן שיש לשחקן, ושמרה את זה בפרמטר deadline\_time. (בפועל החסרנו גם 0.2 שניות, על ערך נסביר אחר כך) בדיקת חריגה מהזמן התבצעה בשלבים הבאים:  
   - לפני הגדלת העומק של פונקצית ה-minimax. כלומר אחרי סיום כל איטרציה נבדוק אם חרגנו מהזמן, אם כן אז נחזיר את הצעד הכי טוב מהאיטרציה האחרונה. אחרת נגדיל את העומק ונמשיך.  
   - לפני פתיחת כל צומת בעץ המשחק – הוספנו בתנאי העצריה של פונקצית הרקורסיה minimax את הבדיקה עבור חריגה מהזמן. אם כן אז לא נמשיך עם צומת זה אלא נחזיר את הערך היוריסטי שלו ו"נתקפל" חזרה ברקורסיה.  
   כיוון שהקיפול חזרה ברקורסיה גם כן לוקח זמן, אז כדי להישאר ב-safeside ביקשנו מהשחקן לסיים לפני 0.2 שניה מהזמן שבפועל יש לו. כך הוא יכול לנצל את ה-0.2 שניות שיש שלו בקיפול חזרה מהרקורסיה.  
   - לפני חישוב כל אחד מהמרכיבים של היוריסטיקה. כיוון שמדובר בבנית גרף ובאלגוריתמי חיפוש מסלולים בגרף, חישוב המרכיבים גם הוא יכול לקחת זמן. ולכן לפני חישוב כל מרכיב הוספנו בדיקת חריגה מהזמן, במקרה והיתה חריגה באמצע החישוב נחזיר את היוריסטיקה עם המרכיבים שיש לנו עג כה ולא ממשיכים למרכיבים הבאים.
6. Small\_map – tie  
   diag\_map – AlphaBeta wins  
   tunnels\_map – AlphaBeta wins when it’s the second player, otherwise a tie  
   trick\_map - AlphaBeta wins when it’s the second player, otherwise a tie  
   ai\_map – AlphaBeta wins  
   התוצאות האלה צפיויות כיוון ששחקן שלפא-ביטא משתמש באותה יוריסטיקה של מינמיקס, אך בגלל הגיזום הוא מצליח להגיע לעומקים יותר גבוהים, ומכאון צפוי שיהיה לו הערכה יותר מדויקת וכך צעדי יותר שקולים.  
   **התוצאות יכולות להשתנות אם שינינו בקוד, לתקן בסוף**
7. ד
8. עבור ה-LiteAlphaBetaPlayer נשתמש ביוריסטיקה המורכבת רק מחלק A מהיוריסטיקה שהצענו בסעיף 7. כלומר נגדיר . חישוב זה רק עובר על הצעדים החוקיים, שזה לכל היותר 4, ולכן החישוב הוא קל.  
   עבור HeavyAlphaBetaPlayer נשתמש באותה יוריסטיקה מורכבת שחישבנו מקודם, המוסברת בסעיף 7. חישוב יוריסטיקה זה כולל בניית גרף מתאים ללוח, בדיקה אם קיים מסלול בין שני השחקנים, וספירת המשבצות הישיגות מכל שחקן, כל זה לוקח יחסית יותר זמן.   
   הגרפים שתקבלו עבור כל שקחן הם:



ניתן לראות שהשחקן עם היוריסטיקה הקלה מצליח להגיע לעומק יותר גדול (עומק של 34) מהשחקן עם היוריסטיקה המורכבת (עומק של 19), דבר זה צפוי כיוון שחישוב היוריסטיקות לוקח מסך כל הזמן המוקצה עבור השחקן, ולכן הזמן של השחקן המורכב נגמר עוד לפני שהוא מצליח להגיע לעומקים יותר גדולים.