דו"ח מסכם

תרגיל בית 2­

יונתן בתן

302279138

[Yonibettan@gmail.com](mailto:Yonibettan@gmail.com)

עמרי פרוינד

301695490

omrifro[@gamil.com](mailto:Neliahu@gamil.com)

# **חלק א'**

# סעיף 1:

בקוד

# סעיף 2:

בקוד

# סעיף 3:

בהינתן gameState מסוים הסוכן ReflexPlayer מחשב את כל הצעדים האפשריים שלו, כלומר מתוך הקבוצה , לאחר מכן מחשב את הערך של צעד על המצב הנתון לכל צעד אפשרי באמצעות היוריסטיקה ולבסוף בוחר בצורה רנדומלית צעד מבין הצעדים המניבים את הערך המקסימלי.

ההיוריסטיקה בה הוא משתמש מחזירה את ה- score הנתון למצב עליה היא פועלת, score זה הוא הניקוד הנוכחי ב-state כפי שהוא מופיע ב- GUI.

נבחין כי היוריסטיקה זו מניבה סוכן הבוחר את הפעולה אשר מניבה ניקוד מקסימלי מבלי לחשוב קדימה ולנסות מצבים שאינם אופטימליים לצעד הבא אבל יניבו תוצאה טובה יותר בהמשך.

# **חלק ב'**

# סעיף 1:



* הפרמטר מייצג את הניקוד הנוכחי של המצב כפי שהוא מופיע ב- GUI
* הפרמטר מייצג את המרחק לחתיכת אוכל הקרובה ביותר
* הפרמטר מייצג את המרחק לקפסולה הקרובה ביותר
* הפרמטר הינו חיובי אם פקמן אכל קפסולה ושלילי אחרת
* הפרמטר מייצג את המרחק לרוח הקרובה ביותר
* הפרמטר מוסיף 1 לערך היוריסטי בהסתברות

# סעיף 2:

תחילה נבחין כי הפרמטר מופיע בחישוב התוצאה ושאר המספרים הינם מספרים הקטנים או שווים ל-1 ולכן מבצעים "fine tuning" להיוריסטיקה scoreEvaluationFunction.

* כפי שלמדנו, עבור מצבים סופיים ההיוריסטיקה מחזירה את הערך שמחזירה פונקציית התועלת
* הפרמטר מייצג את ניקוד המצב הנוכחי ולכן הוא פרמטר חשוב בפונקציה שלנו שכן מצב של ניצחון מכיל כ-500 נקודות יותר מקודמו ולכן נרצה לתת לכך משקל רב בערך היוריסטי.
* לכל אחד מהפרמטרים נרצה לתת בונוס גבוהה כאשר המרחק קטן ובונוס נמוך כאשר המרחק גדל ובנוסף עבור מרחקים גדולים ההבדל בין מצבים שונים נהיה זניח מבחינת פרמטר זה ולכן בחרנו בפונקציה .
* עבור המרחק מהרוח המינימלית אנו מבדילים בין המצב בו פקמן אכל קפסולה ולכן הרוח פגיעה למקרה בו פקמן לא אכל קפסולה.

במקרה הראשון נרצה להתקרב לרוח על מנת לאכול אותה ולצבור נקודות ומיכון שאכילת רוח מזכה בפי 20 יותר נקודות מאשר חתיכת אוכל רצוי לתת פקטור של 20 אל מול הבונוסים של חתיכות האוכל, לאחר משחק עם הפרמטרים גילינו שפקטור 30 נותן את התוצאה הטובה ביותר

* הפרמטר נועד למנוע היתקעות במינימום מקומי, כלומר כאשר ישנם 2 מצבים סמוכים כך שבמעבר לאחד מהם השני נהיה הטוב ביותר אנחנו ניתקע והאלגוריתם לא יסיים את ריצתו ולכן תוספת של רנדומליות מאפשרת לנו לצאת ממצבים אלו
* נשים לב כי ללא הפרמטר לא היינו צריכים להפריד בין מצבים סופיים לכאלו שאינם סופיים שכן כל ערכי המינימום מאותחלים ל- ולכן עבור מצבים סופיים ערכים אלו לא מתעדכנים והיינו מקבלים

# סעיף 3:

בקוד

# **חלק ג'**

# סעיף 1:

לא בטוח בתשובה

ההנחה שלנו הינה שהסוכנים משחקים בזה אחר זה במשחק ובפרט פקמן משחק בהתאם לצעד האחרון של רוח מס' 1 שהיא משחקת בהתאם לצעד האחרון של רוח מס' 2 וכ"ו.

הנחה זו כמובן אינה נכונה שכן הרוחות אינן בהכרח מגיבות אחת לשנייה, יתכן והן פועלות באופן בלתי תלוי אחת מהשנייה ואם במקרה הן כן מגיבות אחת לשנייה לא ניתן להבטיח שרוח מס' 1 תגיב דווקא לרוח מס' 2 ולא להפך למשל.

# סעיף 2:

בקוד

# סעיף 3:

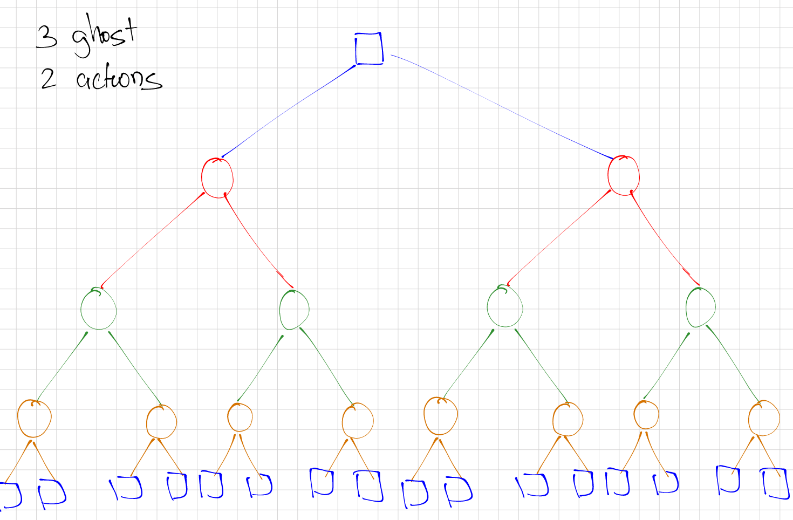
לא בטוח בפתרון

במימוש הנוכחי, צומת בשכבת מינימום מייצג את הפעולה של הרוח ה- בלבד על המכיל בתוכו כבר את הפעולות של הרוח ה- לכל באותו התור.

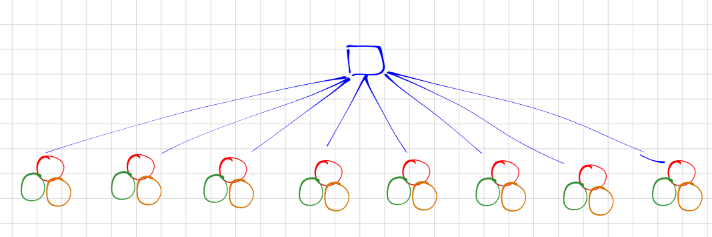
במימוש החדש תהיה לנו שכבת מינימום יחידה אשר כל מצב בה מייצג את פעולות כל הרוחות בתור מסוים.

יתרונות:

* ניתן להשתמש באלגוריתם מינימקס פשוט של 2 שחקנים אותו למדנו
* יש צורך לבצע פחות חישובים של מינימקס כי יש פחות מצבים אפשריים בשכבת המינימום מאשר בסכום שכבות המינימום
* נניח שיש לנו רוחות וכל סוכן יכול לבצע פעולות ואנו רוצים לחשב מינימקס עם הגבלת עומק
* במימוש הנוכחי נצטרך לחשב ערכי מינימקס.



* במימוש החדש יש לנו רק מצבים בשכבת המינימום בעומק (כל מצב מייצג מסלול מרוח אדומה לרוח כתומה) ולכן צריך לחשב ערכי מינימקס



חסרונות:

* מורכב יותר למימוש
* חישוב מצבי הבנים דורש בניית מצב לכל קומבינציה אפשרית של פעולות הרוחות וזה מורכב יותר מאשר ליצור בן עבור כל פעולה של רוח יחידה כפי שעשינו.

# **חלק ד'**

# **חלק ה'**

# **חלק ו'**

# **חלק ז'**

# **חלק ח'**

# **חלק ט'**