**חלק א'**

3. השחקן ReflexPlayer בודק תחילה מהם המהלכים החוקיים מהמצב שבו הוא נמצא(חסימה מבחינת קיר או גבולות הלוח) ומשערך את כל אחד מן המצבים הללו ובוחר בצורה אקראית את אחד מבין הטובים ביותר. השחקן משערך מצב לפי הניקוד שיקבל אילו היה מבצע מהלך זה. למשל, עבור הפסד היה מאבד 500 נק', עבור ניצחון מרוויח 500 וכו'.

**חלק ב'**

1. נגדיר את ההיוריסטיקה הבאה:

הגדרה: מצב **אדיש** הוא מצב שממנו כל המהלכים האפשריים והשינוי בניקוד getScore עבור כל מצב עוקב שכזה אינו עולה על ערך כלשהו(אין לשחקן בחירות טובות באופן מובהק לבצע או בחירות גרועות – למשל הפסד)

היוריסטיקה תקבל מצב state תבדוק האם הוא אדיש ובמידה וכן נחליט על נתינת משקל לכל כיוון אפשרי בהתאם לשני הפרמטרים הבאים:

* בכניסה למצב אדיש בפעם הראשונה(כלומר לא ממצב אדיש אחר), נתחיל להחזיק רשימה של כל המצבים האחרונים שהיינו בהם(במצב אדיש). ניתן משקל גבוה יותר למצבים עוקבים שעדיין לא ביקרנו בהם. במידה ויצאנו מרצף של מצבים אדישים, נאפס את הרשימה.
* בכניסה למצב אדיש, נחשב את כיוון ההתקדמות המועדף בהתאם להיטל וקטור מרכז המסה של האוכל.

\*מרכז מסת האוכל יחושב ע"י סכימת ערכי הx,y של מיקומי האוכל בלוח, חיסור וקטור זה מוקטור המיקום של פקמן, חישוב הזווית של הוקטור שקיבלנו ונתינת משקל גבוה יותר למהלך בכיוון הקרוב ביותר לזווית שמצאנו מבין המהלכים האפשריים.

\* נחזיק רשימה בגודל n=max(width,height) של המפה הנוכחית ובו נשמור את כל הn מיקומי הפקמן האחרונים שביקרנו בהם החל ממצב שבו נכנסו לרצף מצבים אדישים.

=> בבחירת הכיוון המועדף ניתן משקל גבוהה יותר למצבים חדשים שעדיין לא ביקרנו בהם לעומת כיוון מרכז המסה של האוכל על מנת למנוע כניסה ללולאה אינסופית בקרבת אוכל מאחורי קיר.

כל עוד לא הגענו למצב אדיש, ההיוריסטיקה תחזיר את Score המקס' מבין המהלכים האפשריים.

רשימת הפרמטרים שנשתמש בהם: מיקום פקמן בלוח, מיקומי האוכל בלוח, הScore שנקבל עבור כל מצב מהמצבים האפשריים, המצבים האפשריים.

1. מצפייה אמפירית הבחנו בכך ששחקן הפקמן משחק בצורה די טובה לפי בחירת הScore המקס' מבין הScore של כל המצבים האפשריים, חוץ ממצב שבו לפקמן אין מה לאכול בסביבתו או אינו מאוים מאף רוח ולכן יבצע בחירות שרירותיות שלא יקדמו אותו בדרך לניצחון. במצב זה היינו רוצים שהפקמן יתקדם לכיוון שבו נמצא אוכל ולכן נעשה זאת בעזרת החישוב של מרכז מסת האוכל, בנוסף נרצה למנוע מצב שבו הפקמן נתקע באיזור קיר בלולאה אינסופית ולכן נוסיף בדיקה של האם ביקרנו במצבים הקרובים אלינו בעבר כדי לא לחזור אליהם בבחירות הקרובות שלנו ולהבטיח מסלול בגישה שונה בכיוון האוכל.

**חלק ג'**

1. מאחר ויצרנו עץ שתורות השחקנים בו הם לפי שכבות, ההנחה שלנו היא שהשחקנים משחקים לפי סדר קבוע כלשהו. בצורה שבה מומש המשחק בקובץ pacman.py ההנחה מתקיימת מאחר והשחקנים משחקים לפי התור בלולאת המשחק המרכזית. במשחקים האחרים אולי הנחה זו לא תתקיים, בין אם הסדר משתנה או בין אם התורות אינם סנכרונים.

3. דרך נוספת למימוש האלגוריתם שלא דורשת פיתוח שכבה נוספת עבור כל רוח, הינה לפתח שכבת מינימום עבור הרוח הכי קרובה לפקמן. (קרובה למשל ע"פ מרחק מנהטן).

חסרונות: ראיית משחק מוגבלת – לא מתחשבים בכל מהלכי הרוחות האפשריים אלא רק בזו שקרובה ביותר אלינו.

יתרונות: חישוב ערך הMinMax באופן מהיר יותר כיוון שכעת עלינו לפתח רק שכבה אחת בלבד בכל עומק במקום כמספר הרוחות.

**חלק ד'**

1. מבנה העץ החדש שהגדרנו אכן משפיע על האלגוריתם בכך שבמקרה שיש שכבת MIN מתחת לשכבת MIN במידה והשכבה התחתונה שולחת -אינסוף במקרה של גיזום, אז ניתן להעביר את הערך הזה כלפי מעלה (כלומר גיזום עמוק יותר ) בשונה משליחת -אינסוף לשכבת מקסימום מעל שרק תגזום את הענף הנתון בלבד ואולי לא תבצע גיזום גדול יותר כמו בשתי שכבות הMIN.

3. הסוכן AlphaBetaAgent יתנהג או טוב יותר מבחינת זמן ריצה או לכל היותר כמוהו, אבל מבחינת בחירת המהלכים הוא יתנהג בדיוק אותו הדבר.

נשים לב כי מבחינת עץ ההחלטות של הסוכן זהו בדיוק אותו העץ עם בדיוק אותן התועלות עבור כל מהלך(אותה היוריסטיקה על אותם מצבים) ולכן הדבר היחיד שמשתנה במקרה הזה זה היכולת לבצע גיזום של ענפים שקיימת באלגוריתם הAlphaBeta ולכן במקרה הכי גרוע (מבחינת סידור הצמתים עבור הגיזום) הסוכן ירוץ בזמן ריצה של MinMax כי לא יבצע גיזום ויבדוק את כל הצמתים, ואילו בכל מקרה אחר יבצע לפחות גיזום אחד ולכן זמן הריצה ישתפר.

**חלק ה'**

1. השינוי שכעת הסוכן בוחר את המהלך שלא באופן דטרמיניסטי אלא ממצע את הערכים בשכבת ההסתברותית (של הרוחות).

הציפייה שלנו היא כי הסוכן ישחק בצורה טובה יותר באופן הסתברותי, משום שראיית המשחק שלו יותר רחבה והוא לא מניח את הגרוע מכל על היריב שלו בכל תור. דוגמא לכך: פקמן יכול לקחת סיכון בכך שלמשל אם יש במרחק 2 משבצות ממנו רוח ונמצאת בצומת ובינהם אוכל ייתכן כי הפעם יסתכן ויאכל את הקפסולה כי לא יניח בהכרח שהרוח תלך לכיוונו וקיים סיכוי שהרוח תשנה את הכיוון בצומת במקום להמשיך ישר אליו ולאכול אותו.

**חלק ו'**

1. ישנן 2 התפלגויות כתלות במצב "scared" של הרוח:

- אם הרוח מפוחדת(כלומר פקמן אכל קפסולה) , הרוח תרצה להתרחק כמה שיותר מהפקמן, כלומר תעדיף לבחור בהסתברות גבוהה יותר מהלכים שירחיקו אותה מהפקמן ובהסתברות נמוכה הרבה יותר לעשות את המהלכים ההפוכים.

- אם הרוח לא מפוחדת, הרוח תרצה לרדוף אחרי הפקמן ולנסות ולאכול אותו, כלומר תעדיף את המהלכים שיקרבו אותה כמה שיורת לפקמן בהסתברות גבוהה ובהסתברות נמוכה את המהלכים שירחיקו אותה.

\*נשים לב כי כל המהלכים הללו נבחרים לפי המהלכים האפשריים של הרוח באותה הנקודה כתלות במפה ובמיקומה.

3. השוני העיקרי במימוש, הוא שליחת וקטור הסתברויות שונה בין RandomExpectimaxAgent לבין DirectionalExpectimaxAgent לפונקצייה הבוחרת את המהלך הבא עבור הרוח. כאשר הוקטור עבור DirectionalExpectimaxAgent נותן הסתברויות גבוהות יותר עבור מהלכים שיטיבו עם הרוח לעומת ה RandomExpectimaxAgent שנתן אותן הסתברויות לכל המהלכים.

4. נציע 2 רעיונות :

* אסטרטגיית "אמבוש" : במידה ואחת הרוחות רודפת וצמודה אחרי הפקמן ובמידה ויש רוח נוספת במשחק, נגרום לה לאגף את הפקמן מהצד השני. אסטרטגייה זו טובה יותר משל 2 הסעיפים הקודמים מכיוון שאם כבר רוח אחת רודפת אחרי פקמן נעדיף לנצל מצב זה בכדי לסגור עליו מהצד השני במקום לרדוף אחריו מאותו הכיוון.
* אסטרטגיית רוחות מגוונת: נשלב בין שני סוגי הרוחות בכך שנקצה לכל רוח אסטרטגיה שונה, למשל אחת תהיה מסוג RANDOM והאחרת מסוג DIRECTIONAL ובכך נקשה על הפקמן של אחת האסטרטגיות להתאים את בחירת המהלכים שלו לפי אסטרטגיה אחת בלבד עבור הרוחות.

**חלק ז'**

ננסח השערת אפס:

H0 = הסיכוי להוציא ג'וקר מחבילת קלפים מלאה מעורבבת הוא

H1 = הסיכוי להוציא ג'וקר מחבילת קלפים מלאה מעורבבת שונה מ

מבחן:

- נוציא קלף אקראי כלשהו מ1000 חבילות קלפים מלאות ומעורבבות ונראה האם הוא ג'וקר.

**חלק ח'**