PACMAN – Algoritmul MINMAX ALPHA BETA PRUNING

Cerințe:

* Fantomele se vor mișca random pe hartă
* PACMAN se va mișca pe baza algoritmului MINMAX ALPHA BETA PRUNING
* Jocul se va termina când PACMAN o să fie prins de una din fantome sau PACMAN va strânge toate punctele de pe tabla de joc.
* Atât PACMAN-ul cât și fantomele nu pot trece printre ziduri
* Atât PACMAN-ul cât și fantomele pot să se miște sus, jos, stânga, dreapta.
* La un moment dat va exista maxim o fantomă pe fiecare poziție
* PACMAN-ul va colecta toate punctele, Fantomele nu.

Mod de rezolvare:

* Am considerat PACMAN ca fiind MAX și fantomele MIN
* Am considerat tabla o matrice de numere întregi, unde:
  + - 0 – poziție fără punct
    - 1 – zid
    - 2 – poziție cu punct ( bomboană )
* Tabla este imprejmuită de un ziduri pentru a nu lăsa caracterele să depășească zona de joc
* Bazându-ne pe codul de X și 0 am construit algoritmul pentru jocul propus
* Algoritmul se bazează pe o buclă ce rulează până când una idn condițiile de terminare a jocului este indeplinită

Algoritm:

* Inițial căutăm o mișcare a fantomei, ce se va face random ( functia move\_ghost ). Verificăm ca să nu existe 2 sau mai multe fantome pe aceași poziție ( functia is\_ghost\_move\_valid )
* Verificăm ca jocul să nu se fi terminat ( functia check\_win ) ce va verifica dacă numărul de puncte ramase este 0 sau dacă PACMAN se afla pe aceași poziție cu una dintre fantome
* Pentru a determina mișcarea PACMAN-ului ( functia move\_pacman ) vom folosi algorimul de MINMAX – alpha beta pruning
  + Pentru asta trebuie să alegem mișcarea cu cele mai multe șanse de câștig.
  + Fiecare mișcare validă a PACMAN-ului ( functia possible\_moves ) o evaluăm pentru a afla valoarea acesteia.
  + Dintre ele alegem mișcarea ce are valoarea maximă si o efectuăm
  + Avem grija ca să nu ne întoarcem înapoi decât dacă nu avem o alta mișcare sau dacă restul mișcărilor ne fac să pierdem
* Funcția de evaluare ( evaluate\_move ) este funcța recursivă a algoritmului.
  + Aici construim un arbore de adâncime maxima 10 ( valoare aleasă de noi din cauza imensității arborelui ce se face ce încetinește jocul mult prea mult ). Aici se face un fel de overthinking de către PACMAN pentru a alege cea mai bună variantă
  + În funcție de rândul cui ar veni (fantomă sau PACMAN ) se alege mișcarea cu cea mai mica respectiv cea mai mare valoare.
  + În ambele cazuri verificăm inițial dacă jocul este terminat sau dacă s-a ajuns la adâncimea maximă a arborelui. Am ajuns la următoarele valori pentru „frunze”:
    - * -100 + depth în cazul câștigării de către fantomă
      * 100 – depth + 2 \* puncte\_colectate în cazul câștigării de către PACMAN
      * -depth +2 \* puncte\_colectate în cazul necâștigării de către nimeni
      * Depth + 2 \*puncte\_colectate în cazul atingerii adâncirii maxime
  + În ambele cazuri luam toate mutările posibile ce le poate face PACMAN și fantomele și reapelăm functia de evaluate\_move pentru opus ( PACMAN pt fantome si fantomele pt PACMAN
  + Alege cea mai bună mișcare ( minimul valorii la fantoma, și maximul la PACMAN )
* Pentru alegerea tuturor posibilităților de mișcare ale personajelor ( PACMAN si fantome ) ne bazăm pe cele 4 mișcări posibile ale fiecăruia, sus, jos, stanga, dreapta. Toate posibilitățile fantomelor se va lega și de numărul acestora. Se vor face niște combinări cu toate variantele aflării pe tabla de joc a acestora. Se vor calcula toate mișcările posibile ale fiecărei fantomă. În funcție de combinări și de mișcări se vor adăuga în varianta finală a poziților toate posibilitățiile posibile.

La PACMAN totul va simplificat, având un singur personaj de acest tip.

* După alegerea celei mai bune mișcări ale lui PACMAN se verifică terminarea jocului. Dacă nu s-a terminat se reia algoritmul de la mutarea fantomei.
* Se observă ca se va construi un arbore nou la fiecare poziție a PACMAN-ului