**Preambul**

In acest proiect vom construi agenți care să rezolve problema Pacman, incluzând strigoi. Veți implementa funcțiile de comportament descrise in cursurile legate de rezolvarea problemelor cu mai multi agenți (minimax - expectimax). Codul este similar cu cel avut la proiectele anterioare, si poate fi descărcat de pe Moodle.

Execuția autograderului se face cu comanda

python autograder.py

Testele pentru una din întrebării se pot rula cu comanda (de exemplu testele pentru întrebarea q2):

python autograder.py -q q2

Un anume test particular poate fi invocat cu comanda:

python autograder.py -t test\_cases/q2/0-small-tree

Putem forța execuția autograder-ului fără grafică folosind flagul –no-graphics

Din arhiva pusa la dispoziție puteți edita următoarele fișiere, pe care trebuie să le transmiteți ca și răspuns la assignment:

|  |  |
| --- | --- |
| multiAgents.py | în care veți pune construcția agenților voștri care rezolva problema |

Se vor modifica doar fișierele indicate. Pentru buna funcționare a autograderului pe Gradescope, recomandarea este sa nu se modifice fișiere suport.

Puteți lucra in echipe de maxim 2 persoane. Doar unul din membri echipei va face transmiterea fișierelor pe Gradescope si in assignment își va indica (pe Gradescope) colegul ca si membru al echipei. De asemenea, in comentariile funcțiilor din cadrul soluției transmise se va indica nominal autorul funcției respective.

Codul sursa transmis pe Gradescope va fi evaluat automat dpdv al corectitudinii tehnice. Nu schimbați numele funcțiilor respectiv a claselor deja existente in fișierele sursa. Pentru buna funcționare a Gradescope este necesar sa păstrați structura proiectului intacta. Scorul final va fi dat de corectitudinea implementării furnizate.

Gradescope permite detectarea similarităților de cod sursa si logica a codului. Prin urmare, va rugam sa lucrați independent si sa nu dați propriile coduri sursa altor colegi. Proiectele transmise care vor fi identificate cu grad ridicat de similitudine vor fi notate cu 0 puncte.

**Pacman – multi-agent**

Puteți executa jocul clasic Pacman cu comanda (se pot folosi săgețile pentru a-l muta pe Pacman):

python pacman.py

Pentru a rula jocul cu un agent anume, in locul lui Pacman puteți folosi (de exemplu ReflexAgent)

python pacman.py -p ReflexAgent

Puteți schimba layout-ul jocului cu flag-ul -l:

python pacman.py -p ReflexAgent -l testClassic

**Întrebări la care trebuie sa răspundeți (prin implementare de cod) in cadrul acestui proiect:**

**Întrebarea 1. Construirea unui agent reflex**

Cerința este să îmbunătățim funcționarea agatului ReflexAgent. In interiorul acestuia se poate interoga GameState pentru a obține informație despre starea jocului. Un agent reflex care joaca in mod capabil jocul trebuie sa tina cont de locațiile pentru mâncare si strigoi. Un agent implementat ok trebuie sa fie capabil ca curețe tot layout-ul testClassic.

Puteți încerca agentul si pe alte layout-uri, cu 1 sau 2 strigoi, de exemplu:

python pacman.py -p ReflexAgent -l mediumClassic -k 1

python pacman.py -p ReflexAgent -l mediumClassic -k 2

sau folosind –frameTime 0 pentru a creste viteza de afișare:

python pacman.py –frameTime 0 -p ReflexAgent -l mediumClassic -k 2

Ca si ajutor, putem folosi funcția asList() de la newFood. De asemenea, in acest moment funcția de evaluare pe care o scrieți trebuie sa evalueze perechi de tipul (state, action). Pentru debug, puteți afișa conținutul obiectelor ca si string, folosind funcția print din python: de exemplu print(newGhostStates)

Implicit, strigoii sunt aleatorii (in comportament). Puteți schimba acest comportament folosind un anume tip de strigoi, cu particula -g: de exemplu -g DirectionalGhost

Comportamentul random al strigoilor poate fi înghețat la o anume succesiune de mutări folosind un seed pentru a inițializa generatorul de numere aleatoare: de exemplu -f 100 (in loc de 100 se poate folosi si un alt seed, ideea este ca acest seed trebuie păstrat pe durata experimentelor dintr-o sesiune de lucru). Se pot juca mai multe jocuri într-o execuție cu particula -n

Notare: la testare se va executa agentul furnizat ca si input pe layout-ul openClassic de 10 ori. Veți primi 0 pct daca agentului ii expira timpul, sau nu câștiga niciodată. Veți primi 1 pct daca agentul câștiga de cel puțin 5 ori si 2 pct daca agentul câștigă in toate cele 10 jocuri. Primiți inca un punct adițional daca scorul mediu este peste 500. Si 2 pct adiționale daca scorul mediu este de peste 1000. Total: 4 pct.

**Întrebarea 2. Minimax**

Trebuie sa folosim clasa MinimaxAgent din multiAgentsțpy. Agentul de tip Minimax pe care îl scrieți trebuie sa fie capabil sa gestioneze orice număr de strigoi, deci trebuie sa scrieți un algoritm general pentru funcționarea agentului. In mod particular, daca agentul joaca împotriva a 2 strigoi, atunci arborele de joc are următoarea structura: nivel Max (pentru Pacman), apoi 2 nivele Min (pentru fiecare din cei 2 strigoi), apoi din nou un nivel Max, si apoi din nou 2 nivele Min si asa mai departe. Puteți sa expandați arborele jocului pana la o adâncime arbitrara. Scorul frunzelor din arborele de joc este dat de self.evaluationFunction. De asemenea, observați ca agentul MinimaxAgent moștenește din clasa MultiAgentSearchAgent care are proprietățile depth si evaluationFuction.

Daca facem o căutare in arbore cu adâncime 2, asta va implica faptul ca fiecare strigoi muta de 2 ori.

*Notare*: se va verifica codul pentru a observa faptul ca acesta explorează un număr corect de stări ale jocului. Deci autograder-ul va memora atent numărul de apeluri ale funcției GameState.generateSuccessor.

*Comentarii:*

Puteți implementa algoritmul in mod recursiv. Chiar si cu o implementare corecta, exista situații in care Pacman pierde. Daca comportamentul lui Pacman este cel corect (minimax), atunci testul de notare va fi trecut. Funcția de evaluare este scrisa parțial. Nu trebuie sa schimbați partea deja scrisa. Spre deosebire de întrebarea anterioara, acuma se evaluează stări si nu perechi (stare, acțiune).

Valorile minimax a stării inițiale in cazul layout-ului minimaxClassic sunt 9, 8, 7 si -492 pentru adâncimi de 1, 2, 3 si 4 respectiv. In acest caz, agentul minimax ar trebui sa câștige in cam 66% din execuțiile jocului. Puteți testa acest layout cu comanda

python pacman.py -p MinimaxAgent -l minimaxClassic -a depth=4

In lista agenților din joc, Pacman este întotdeauna agentul de pe index 0 si agenții muta in ordine crescătoare a indexului: deci agent 0, apoi agent 1, apoi agent 2, apoi din nou agent 0 samd.

Daca încercați agentul pe layout-uri mai mari (openClassic sau mediumClassic) poate veți constata ca Pacman este destul de bun ca sa evite strigoii (deci sa supraviețuiască), dar destul de rău ca sa câștige jocul. Vom rezolva acest comportament la o întrebare viitoare.

Constatați ca daca Pacman estimează ca va muri (inevitabil), atunci el va prefera sa termine jocul cat mai repede (deci sa nu consume mutări). Nu acesta e comportamentul rațional într-o astfel de situație (cu strigoi random), dar in cazul minimax acest comportament este corect.

**Întrebarea 3. Alpha-beta prunning**

Pentru aceasta întrebare trebuie sa implementam agentul AlphaBetaAgent. Vedeți pseudocodul de la curs si trebuie sa îl adaptați pentru situatia in care in joc se găsesc mai multi strigoi (deci trebuie adaptat pseudocodul pentru mai multe nivele de min).

Trebuie sa observați un spade-ul al lui Pacman, astfel un alpha-beta pruning cu adâncime de 3 va fi similar de rapid ca si un minimax cu adâncime de 2.

Valorile minimax obținute cu AlphaBetaAgent trebuie sa fie similare cu cele obținute cu MinimaxAgent, asta deși acțiunile selectate pot sa fie diferite.

*Notare:* Vom verifica daca agentul explorează numărul corect de stări ale jocului. Deci este important ca sa faceți alpha-beta pruning fără sa reordonați copii unui nod din arbore. Deci stările succesor trebuie sa fie procesare in ordinea naturala returnata de GameState.getLegalActions. De asemenea, nu trebuie sa apelați GameState.generateSuccessor mai mult decât este necesar.

De asemenea, daca faceți pruning, nu trebuie sa faceți taiere la egalitate.

Din nou, implementarea corecta poate sa conducă la un agent care sa pierde unele jocuri.

**Întrebarea 4. Expectimax**

In minimax sau alpha-beta pruning se presupune ca jucați împotriva unor strigoi care iau acțiunile optimale – ceea ce nu este cazul, de multe ori. Pentru aceasta întrebare veți implementa ExpectimaxAgent care modelează un comportament probabilistic al strigoilor. Veți folosi algoritmul expectimax de la curs. Cazurile de testare sunt in folderul q4.

Daca algoritmul implementarea reușește pe arbori mici, veți observa ca Pacman începe sa câștige. Presupunem faptul ca jucam împotriva unor adversari care aleg in *mod aleatoriu uniform* dintre mutările furnizate de getLegalActions. Puteți executa agentul pe minimaxClassic:

python pacman.py -p ExpectimaxAgent -l minimaxClassic -a depth=3

puteți investiga cum se comporta expectimax agent in comparație cu AlphaBetaAgent pe layout-uri precum trappedClassic. Din 10 jocuri, agentul expectimax ar trebui sa câștige intr-un număr majoritar de situații, iar AlphaBetaAgent sa piardă preponderent.

Din nou, chiar si cu o implementare corecta a lui Expectimax, vor fi jocuri in care agentul pierde. Acesta nu este o problema daca implementarea expectimax este corecta.

**Întrebarea 5: funcția de evaluare**

Puteți scrie o funcție de evaluare superioara completând funcția betterEvaluationFunction. Atenție: funcția de evaluare trebuie sa evalueze stări ale jocului si nu acțiuni. Cu o căutare de nivel 2, funcția de evaluare trebuie sa permită curățarea layout-ului smallClassic împotriva unui strigoi random in cel puțin jumătate din jocuri si in același timp sa fie suficient de rapid (pentru a primi toate punctele). Pacman trebuie sa aibă o medie de aprox. 1000 pct atunci când câștigă.

*Notare*: se va evalua agentul pe layout-ul smallClassic de 10 ori.

Daca Pacman câștigă cel puțin o data fără sa dea time out, primiți 1 pct

Suplimentar primiți 1 punct pentru un agent care câștigă de cel puțin 5 ori

Suplimentar primiți 1 punct pentru un scor mediu de cel puțin 500, si 2 pct pentru scor mediu de cel puțin 1000. Aici se includ scorurile si de la jocurile pe care le pierdeți.

Suplimentar primiți 1 punct daca jocul durează mai puțin de 30 de secunde pe autograder, executat cu –no-graphics

Nu copiați funcțiile de la întrebarea 1. Ele nu funcționează cu autograder-ul.