**Preambul**

In acest proiect încercăm să îl ajutăm pe Pacman să găsească căi prin labirint, fie pentru a atinge o anume poziție sau pentru a colecta obiectele de valoare (mâncare) in mod eficient. Cerința generală este să implementați algoritmi generali de căutare și să ii aplicați la jocul Pacman.

Execuția autograderului se face cu comanda

python autograder.py

Din arhiva pusa la dispoziție puteți edita următoarele fișiere:

|  |  |
| --- | --- |
| search.py | în care veți pune algoritmii de căutare |
| searchAgent.py | in care veți pune agenții voștri de căutare care se bazează pe algoritmii implementați in search.py |

De asemenea, pentru a înțelege codul proiectului Pacman, puteți consulta următoarele fișiere:

|  |  |
| --- | --- |
| pacman.py | Fișierul principal care executa jocul Pacman. Conține tipul GameStare (o stare a jocului), folosita pentru a reprezenta stările |
| game.py | Conține logica din spatele jocului. Include cateva clase relevante: AgentState, Agent, Direction și Grid |
| util.py | Conține structuri de date utilitare necesare pentru implementarea algoritmilor de căutare |
| graphicsDisplay.py | Conține grafica jocului |
| graphicsUtil.py | Fișiere utilitate suport pentru grafica jocului |
| textDisplay.py | Grafica ASCII pentru jocul Pacman |
| ghostAgents.py | Agenții care controlează fantomele |
| keyboardAgents.py | Inferata jocului cu ajutorul tastaturii |
| Layout.py | Cod pentru citirea fișierelor de tip layout, si salvarea conținutului |

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.

**Testarea jocului**

Puteți sa jucați Pacman pornind jocul din linia de comanda:

python pacman.py

In fișierul searchAgent.py este implementat un agent simplu: GoWestAgent, care merge întotdeauna catre vest. Puteți porni acest agent cu comanda:

python pacman.py --layout testMaze --pacman GoWestAgent

Puteți schimba layoutul de testare prin adăugarea unei alte opțiuni la parametrul --layout. De exemplu tinyMaze.

Lista parametrilor si a opțiunilor care pot fi transmise la lansarea jocului poate fi consultata cu

python pacman.py -h

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

**Întrebarea 1**. Găsirea unui element de tip Food fixat in labirint prin Depth First Search. Se va implementa funcția depthFirstSearch din search.py. Puteți dezvolta algoritmul de căutare in graf (vedeți cursurile), pentru a evita vizitarea aceleiași stări de mai multe ori. Puteți testa codul cu

python pacman.py -l tinyMaze -p SearchAgent

python pacman.py -l mediumMaze -p SearchAgent

python pacman.py -l bigMaze -z .5 -p SearchAgent

**Întrebarea 2**. Găsirea unui element de tip Food fixat in labirint prin Breadth First Search. Se va implementa funcția breadthFirstSearch din search.py. Testarea codului poate fi făcută similar cu comenzile:

python pacman.py -l mediumMaze -p SearchAgent -a fn=bfs

python pacman.py -l bigMaze -p SearchAgent -a fn=bfs -z .5

**Întrebarea 3**. Găsirea unui element de tip Food fixat in labirint prin Uniform Cost Search. Se va implementa funcția uniformCostSearch in search.py . Trebuie sa furnizați o funcție de cost potrivita. Puteți testa cu mai multe funcții de cost, pentru a decide care funcție de cost este mai buna pentru a ghida căutarea. La o implementare corecta, agentul trebuie sa funcționeze corect cel puțin in 3 layout-uri specificate mai jos:

python pacman.py -l mediumMaze -p SearchAgent -a fn=ucs

python pacman.py -l mediumDottedMaze -p StayEastSearchAgent

python pacman.py -l mediumScaryMaze -p StayWestSearchAgent

**Întrebarea 4**. Găsirea unui element de tip Food fixat in labirint prin algoritmul A\*. veți implementa funcția aStarSearch din search.py. A\* are nevoie de o euristica ca si argument. Euristica este compusa la rândul sau din 2 argumente: o stare a problemei si problema in sine. Un exemplu trivial de euristica îl găsiți in nullHeuristics.

Distanta Manhattan este deja implementata ca si euristica. Puteți sa va testați propriul cod sursa folosind distanta Manhattan cu comanda:

python pacman.py -l bigMaze -z .5 -p SearchAgent -a fn=astar,heuristic=manhattanHeuristic

La o implementare corecta va trebui sa obțineți faptul ca A\* funcționează mai rapid decât UCS (de exemplu 549 vs 620 noduri căutate).

Ce se întâmplă pe layout-ul openMaze pentru diverse strategii de căutare?