STRUTTURA DEL PROGETTO:

Il codice del progetto è strutturato all’inerno della cartella /search. Al suo interno si trovano la cartella /layout che contiene le strutture dei labirinti con le posizioni dei cibi e la posizione iniziali di Pacman, e una serie di file Python che costituiscono il cuore del gioco.

I file riguardanti la grafica non sono stati usati per lo svolgimento del progetto e sono:

* graphicDisplay.py grafica per PacMan
* graphicUtil.py supporto per la grafica
* textDisplay.py grafica ASCII per PacMan
* ghostAgents.py agenti per controllare i fantasmi
* keyboardAgents.py interfaccia per controllare PacMan con la tastiera
* layout.py codice per leggere i file contenuti in /layout

I file che è stato necessario comprendere ai fini dello svolgimento del problema sono i seguenti:

* pacman.py File principale che esegue il gioco; in particolare è stato necessario prestare attentione al tipo GameState
* game.py Contiene la logica del mondo di Pacman
* util.py Contiene le strutture dati utili per l’implementazione degli algoritmi di ricerca

I file che contengono il nostro lavoro sul progetto sono:

* search.py Contiene l’implememtazione di tutti gli algoritmi di ricerca
* searchAgents.py Contiene l’implementazione di tutti gli agenti

COME AVVIARE IL GIOCO

Il gioco si avvia sempre tramite riga di comando. È possibile controllare Pacman con la tastiera (e quindi giocare al gioco vero e proprio) con il seguente comando:

python pacman.py (1)

Nel corso del progetto è stato necessario utilizzare diversi tipi di layout, algoritmo (con euristica per quelli che ne utilizzano una), agenti in base al problema considerato. Tali cambiamenti vengono attuati via riga di comando specificando le opzioni dopo (1); la lista delle opzioni è elencata eseguendo

python pacman.py –h

Tutti i comandi che sono stati utilizzati nel progetto sono contenuti in /search/command.txt.

FINDING ALL CORNERS – TROVARE TUTTI GLI ANGOLI

In questo nuovo problema vengono utilizzati i layout cornerMazes; questi contengono 4 puntini, uno in ogni angolo del labirinto e PacMan, attraverso gli algoritmi di ricerca, deve essere in grado di trovare il percorso più breve attraverso il labirinto che tocchi tutti e quattro gli angoli.

La rappresentazione degli stati scelta per questo problema è la seguente:

(posizione corrente di PacMan, angoli già visitati).

Tale rappresentazione è nata dopo lo studio attento dei problemi già implementati all’interno del file searchAgents.py ed utilizza le sole parti del GameState che sono necessarie, come indicatoci dalla consegna (sebbene ciò comporti allo stato di “ricordarsi” degli angoli visitati in precedenza.

Quindi lo stato iniziale è (posizione iniziale di PacMan, angoli visitati), dove in angoli visitati ci sarà la posizione iniziale di pacman se questa è un angolo, altrimenti è vuoto.

Il controllo per lo stato goal consiste nel confrontare l’insieme degli angoli del problema con l’insieme degli angoli visitati. Se i due insiemi sono uguali, allora lo stato è uno stato goal.

La funzione sucessore, invece, ritorna gli stati sucessori come un insieme di triple (così come indicatoci nella consegna); ciascuna tripla è composta da:

* prossimo stato
* l’azione fatta da pacMan per raggiungere quello stato, ovvero il punto cardinale verso cui si è mosso
* il costo dell’azione per raggiungere quello stato (che è 1)

Ciascuna tripla è costruita come segue:

1. a partire dalla posizione di pacman, si calcola la nuova posizione in cui pacman sarebbe se si spostasse di una casella seguendo uno dei quattro punti cardinali.
2. Se tale nuova posizione è valida (ovvero non è un muro), allora costruisce la tripla descritta sopra, aggiornando l’insieme degli angoli visitati per il prossimo stato se la nuova posizione è un angolo non ancora visitato.
3. Si ripetono queste operazioni per ciascun punto cardinale.

Tale problema è implementato nel file searchAgents.py, nella classe CornersProblem, seguendo la struttura e le firme dei metodi a disposizione. Le funzioni che sono state da noi implementate sono il costruttore, getStartState, isGoalState, getSuccessors.

La seconda consegna richiedeva l’impetrazione di una euristica per poter risolvere il problema utilizzando l’algoritmo A\*. è richiesto che l’euristica sia ammissibile.

L’euristica scelta si comporta come segue:

1. Tra gli angoli che Pacman non ha ancora visitato, si sceglie quello più vicino alla posizione corrente di pacman e si memorizza la distanza.
2. Dopo di che si sceglie l’angolo più vicino (non ancora visitato) all’angolo scelto al punto 1 e si somma la distanza tra questi due angoli a quella memorizzata al punto.
3. Per ciascun angolo non ancora visitato, si sceglie quello più vicino rispetto all’angolo considerato subito prima e si sommano le distanze tra queste coppie di angoli.
4. La somma tra le distanze è ritornata

Tale euristica è ammissibile in quanto si tratta della soluzione del problema rilassato in cui pacman si trova in un labirinto senza muri. È soluzione in quanto pacman, dopo aver raggiunto l’angolo più vicino alla sua posizione iniziale segue il perimetro per poter visitare tutti gli altri angoli.

TEST

Al fine di verificare se la nostra rappresentazione degli stati era corretta, abbiamo seguito il test proposto nella consegna. Tale consiste nell’ eseguire i due comandi sottostanti; la rappresentazione del problema doveva far si che l’agente riuscisse a risolvere con successo i due problemi lanciati.

python pacman.py -l tinyCorners -p SearchAgent –a fn=bfs,prob=CornersProblem

python pacman.py -l mediumCorners -p SearchAgent -a fn=bfs,prob=CornersProblem

La nostra rappresentazione risolve correttamente i due problemi.

L’euristica invece è stata testata utilizzando tre tipi di distanze: la disanza di manhattan, la distanza euclidea e la distanza di chebyshev. Il confronto tra i dati ottenuti ci sottolinea come la distanza Manhattan sia dominante nei confronti delle altre due, mentre la distanza euclidea è dominante solo sulla distanza di chebyshev

\*inserire i dati\*

Per risolvere il problema degli angoli con A\* e la nostra euristica è sufficiente eseguire questo comando:

python pacman.py -l mediumCorners -p AStarCornersAgent -z 0.5