Python Snake Challenge

WISB256 – Programmeren in de Wiskunde

Abe Wits, Universiteit Utrecht

Samenvatting

In deze opdracht schrijf je een AI (kunstmatige intelligentie) die gaat strijden tegen de AI's van je medestudenten, of zelfs tegen menselijke spelers, in een multi-player variant op snake.

Kernwoorden: kortste-pad algorithmen, speltheorie, heuristieken, realtime, optimalisatie

1 Inleiding

In de Python Snake Challenge schrijf je een programma dat een snake door een labyrinth heen moet sturen. Voorkom botsingen met muren, andere spelers en jezelf, en probeer zoveel mogelijk voedsel te verzamelen. Schakel andere spelers uit door ze in de hoek te drijven. Je kan winnen door meer te eten en langer te overleven dan je tegenstanders. De optimale tactiek is afhankelijk van het labyrinth en de tegenstanders, ben jij ze te slim af?

2 De opdracht

In deze opdracht schijf je zelf het brein van de slang. Je leest de huidige staat van het doolhof en de bewegingen van de andere spelers in uit een tekstbestand en beslist vervolgens wat je slang gaat doen. Je moet o.a. de positie van de andere slangen bijhouden om te weten waar ze zijn. Een voorbeeld doolhof is geven in figuur 1.

```
.#.#.#....
.........#.
......1#.#
.........0
........#.
..........
..........
........#.
#.........
....#.....
```

Figuur 1: 10×10 doolhof met 2 slangen van lengte 1 (`0` en `1`) en obstakels (`#`).

3 Spelregels

Het doel van de Python Snake Challenge is om zoveel mogelijk punten te krijgen. Je cijfer zal worden beïnvloed door de prestaties van je AI tegen andere spelers (zie blackboard voor de rest van de criteria). Je kan punten verdienen door:

- Een stap te zetten zonder dood te gaan (1 punt voor elke stap)
- Voedsel te eten (100 punten voor elke consumptie)
- Langer te leven dan andere spelers (1000 punten voor elke speler die korter leeft)

Alle spelers beginnen met 0 punten. Het spel bestaat uit discrete "tijdstappen". Aan het begin van een tijdstap worden spelers gesorteerd van lage score tot hoge score. Vervolgens mogen spelers op volgorde van deze score omstebeurt een zet doen. Een zet werkt als volgt: In een zet geeft een speler eerst aan in welke richting hij wil bewegen (je moet altijd een zet doen). Hierbij gelden periodieke randvoorwaarden; als je bijvoorbeeld links uit het level beweegd kom je er aan de rechter kant weer in. Als het vakje waar de speler naartoe beweegd een snake of muur bevat, dan gaat bewegende snake dood en mag deze niet verder spelen. Anders beweegt de snake naar dit vakje. Als het vakje voedsel bevat dan groeit snake, en wordt zo één vakje langer. Nadat alle spelers een zet hebben krijgen ze te horen waar nieuw voedsel komt en waar de spelers naartoe bewogen zijn. Dit is het einde van de tijdstap. En dus het begin van een nieuwe tijdstap. Het spel eindigd na 300 tijdstappen, of als alle spelers dood zijn. Vanaf het moment dat je programma te horen krijgt wat de andere spelers hebben gedaan heb je 1 seconde om een zet te doen, als je er langer over doet beweeg je niet en ben je af. Je programma mag maximaal één thread en één proces gebruiken, en mag maximaal 100mb geheugen gebruiken. Je mag niet lezen of schrijven van de harde schijf. Je mag geen gebruik maken van de grafische kaart of andere bijzondere hardware. Verder mag je programma alleen standaard packages gebruiken. Het is de bedoeling dat je de AI samen met je groepje schrijft, gebruik dus geen code van anderen (behalve de gegeven voorbeeldcode). Het is onmogelijk om hier alle regels precies op te schrijven, in het algemeen geld; als je vermoed dat iets niet mag, dan mag het waarschijnlijk niet. Als je twijfelt of iets mag kan je het altijd vragen.

4 Input/output

Je programma krijgt eerst input om te initialiseren, en daarna per iteratie input om de bewegingen van andere spelers en het plaatsen van nieuw voedsel door te geven. Tussendoor zal je programma ook bewegingen moeten doorgeven. Hier volgt een formele beschrijving van de vorm waarin de input en output moeten zijn. Het voorbeeldprogramma in dit document leest al op een juiste manier de input en geeft in de juiste vorm output.

Eerst krijg je twee regels, die h, de hoogte van het level en b, de breedte van het level bevatten met $1 \leq h, b \leq 50$. Dan volgen er h regels met elk b karakters. Hierin geeft een "`#`" een muur aan en een "`.`" een lege ruimte.

```
6
5
.....
..#..
#...#
#.#.#
.....
#####
```

Dan volgt het aantal spelers n ($1 \leq n \leq 6$) en de coordinaten van deze spelers (x, y). Deze spelers zijn

genummerd; $0, 1, \ldots, n-1$. Een speler begint nooit in een muur. Daarna volgt nog één getal; het spelernummer van jouw AI.

```
3
2 4
0 1
3 3
1
```

Na deze initialisatie komen de "tijdstappen". Een tijdstap begint met het verplaatsen van de slang. Dit doe je door eerst aan te geven dat je wil gaan bewegen door "`move`" te printen. Vervolgens moet je onmiddelijk op een nieuwe regel je richting aangeven met de beginletter van **u**p, **r**ight, **d**own of **l**eft.

```
move
u
```

Het kan natuurlijk zijn dat deze beweging geen goede keuze was, en dat je tegen jezelf, een muur, of een andere speler aanloopt. In dat geval krijg je als input "quit" om aan te geven dat je bent uitgeschakeld. Anders krijg je inplaats hiervan nieuwe informatie over het spel. Eerst krijg je de bewegingen van alle slangen door (inclusief de beweging de je net zelf hebt aangevraagd). Deze bewegingen zijn achter elkaar in een string samengevat, weer met `u`, `r`, `d` of `l`. In deze string staat op index i de beweging van speler i. Als een slang dood is en niet beweegt komt er een `x`. Na deze string volgt op een nieuwe regel de hoeveelheid nieuw eten k, met daarna k regels met de coordinaten (x, y). Eten kan alleen op lege vakjes verschijnen.

```
luu
3
1 2
1 4
2 2
```

5 Waar kunnen we mee beginnen?

Een paar dingen waar je je mee bezig kan houden (in een bijna willekeurige volgorde):

- Verzin een goede taakverdeling
- Bedenk een handige programma-structuur
- Probeer precies te begrijpen wat de spelregels zijn
- Verbeter het voorbeeld programma zodat muren ontweken worden
- Zorg dat je AI weet waar alle spelers zich bevinden
- Bedenk tactieken om zoveel mogelijk punten te verdienen

Vergeet niet om regelmatig je programma te testen! Je kan `run.py` gebruiken om verschillende AI's (en mensen) tegen elkaar te laten spelen. Dit programma schrijft logs naar de logs map. `run.py` heeft de package "pexpect.py" nodig, dit moet je eerst installeren, dit kan op de meeste systemen met `python3 -m pip install pexpect` worden gedaan.

6 Voorbeeldcode

```python
import random

###Initialisatie; we lezen het doolhof en beginposities in

level_hoogte = int(input())          #Lees hoe groot het level is
level_breedte = int(input())

level = []                           #Lees het level regel voor regel
for y in range(level_hoogte):
    level.append(list(input()))

aantal_spelers = int(input())        #Lees het aantal spelers en hun posities
begin_posities = []
for i in range(aantal_spelers):
    begin_positie = [int(s) for s in input().split()]   #Maak lijst met x en y
    begin_posities.append(begin_positie)      #Voeg dit coordinaat toe aan begin_posities

speler_nummer = int(input())         #Lees welk spelernummer wij zijn

positie = begin_posities[speler_nummer] # Wij beginnen op deze positie

dx = [ 0, 1, 0,-1] # dx en dy geven aan in welke richting 'u', 'r', 'd' en 'l' zijn
dy = [-1, 0, 1, 0]                   # u=up, r=right, d=down, l=left

###De tijdstap; we zetten een stap en verwerken nieuwe informatie

while True:
    i = random.randrange(4)          #Kies een random richting
    richting = 'urdl'[i]             #u=up, d=down, l=left, r=right
    positie[0] += dx[i]              #Verander de huidige positie
    positie[1] += dy[i]
                                     #Let op periodieke randvoorwaarden!
    positie[0] = (positie[0] + level_breedte)% level_breedte
    positie[1] = (positie[1] + level_hoogte) % level_hoogte

    print('move')                    #Geef door dat we gaan bewegen
    print(richting)                  #Geef de richting door

    line = input()                   #Lees nieuwe informatie

    if line == "quit":               #We krijgen dit door als het spel is afgelopen
        print("bye")                 #Geef door dat we dit begrepen hebben
        break
                                     #String met bewegingen van alle spelers
    speler_bewegingen = line         #Nu is speler_bewegingen[i] de richting waarin
                                     #speler i beweegd

    aantal_voedsel = int(input())    #Lees aantal nieuw voedsel en posities
    voedsel_posities = []
    for i in range(aantal_voedsel):
        voedsel_positie = [int(s) for s in input().split()]
        # Sla de voedsel positie op in een lijst en in het level
        voedsel_posities.append(voedsel_positie)
        level[voedsel_positie[1]][voedsel_positie[0]] = "x"
```