Inteligență Aritificială
Tema 1: *Biscuitele*

Tudor Berariu
tudor.berariu@gmail.com

October 22, 2015

1 Descrierea jocului

Jocul *Dots and Boxes* se desfășoară pe o matrice de dimensiune $height \times width$. Importante sunt cele $(height + 1) \times (width + 1)$ puncte de intersecție. În timpul jocului, pe rând, fiecare dintre cei doi jucători unește două puncte vecine cu o linie orizontală sau verticală. Scopul este acela de a închide cât mai multe celule prin unirea puncelor. Fiecare jucător primește un punct pentru fiecare celulă pe care îl închide (toți cei patru pereți care delimitează celula au fost adăugați). La finalul jocului câștigă acel jucător care a strâns mai multe puncte.

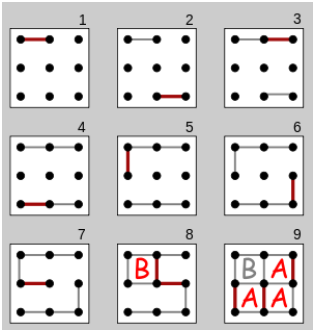

Figure 1: Image from Wikipedia

Jucătorii *mută* alternativ, dar un jucător care reușește în urma unei mutări să închidă o celulă mai primește dreptul la încă o mutare.

2 Cerințe și notare

Implementați algoritmul MiniMax pentru jocul *Dots and Boxes*. Algoritmul trebuie completat de tehnica $\alpha - \beta$ pruning. Găsiți o euristică cât mai bună astfel încât să bateți cei doi jucători implementați deja.

Există o limită de timp de o secundă pe mutare.

Se vor acorda 5 puncte pentru implementarea algoritmului MiniMax cu $\alpha - \beta$ pruning.

Se vor acorda 5 puncte pentru găsirea unei euristici destul de bune încât să bată jucătorii pe care îi găsiți în arhiva temei.

Se vor acorda până la 2 puncte bonus pentru cele mai bune 50% dintre teme. Toate soluțiile trimise vor fi confruntate, se va face un clasament general, iar prima jumătate din acel clasament va primi de la 2 la 0.5 puncte (descrescător în ordinea punctajului). Clasamentul va fi făcut întâi după numărul de jocuri câștigate și apoi după *golaverajul* general.

3 Arhiva temei

Arhiva temei conține un script `game_server.py` care confruntă toți jucătorii din directorul `players`. Fiecare jucător este implementat într-un fișier separat în care se găsește o clasă cu același nume.

Pentru a vă testa soluția, implementați o soluție, puneți fișierul în directorul `players` și rulați `make clean` și `make`. Dacă aveți `pdflatex` și desktop Gnome se va deschide un pdf cu clasamentul jucătorilor.

4 Trimiterea temei

Arhiva temei va conține un fișier `PDF` cu descrierea soluției folosite în temă și un *singur* fișier `Python` cu implementarea soluției. Fișierele vor avea un nume construit astfel: `NumePrenumeAAAALLZZ.ext` din numele complet și data nașterii. Renunțați, desigur la diacritice și la linii. `ext` va fi `pdf` sau `py`.

În fișierul `Python` se va găsi o clasă cu același nume în care vor fi cel puțin următoarele două metode: `__init__` și `move`.

```
class RandomPlayer:
  def __init__(self):
    self.name = "Random Player"
```

```
def move(self, board, score):
  return (0,0)
```

Metoda `__init__` va inițializa un câmp `name` cu un șir de caractere ce conține numele complet.

Metoda `move` primește două argumente: `board` și `score`:

- `board` este o listă cu $height * 2 + 1$ liste. Cele de pe pozițiile pare $(0, 2, \ldots, height * 2)$ corespund liniilor orizontale și au lugime $width$. Listele de pe pozițiile impare corespund liniilor verticale și au lungime $width + 1$.
- `score` reprezintă un tuplu cu scorul curent: punctajul propriu, punctajul adversarului.

Exemplu de matrice Pentru matricea următoare:

```
*-* * *
| |   |
* * *-*
  | |
* * * *
|
*-*-*-*
```

parametrul `board` va fi:

```
[[1, 0, 0],
 [1, 1, 0, 1],
 [0, 0, 1],
 [0, 1, 1, 0],
 [0, 0, 0],
 [1, 0, 0, 0],
 [1, 1, 1]]
```

În arhiva pe care o încărcați pe `curs.cs` puneți direct cele două fișiere (nu un director).