Warning

This page is located in archive. Go to the latest version of this course pages. Go the latest version of this page.

Hráč

Podrobná specifikace toho, co musí hráč umět a jak musí vypadat, aby mohl hrát turnaj s ostatními.

Soubor k odevzdání

Veškerý potřebný kód, který hráč potřebuje ke hraní musí být v souboru `player.py` . Tento soubor (a jen tento soubor) nahrajete do upload systému [https://cw.felk.cvut.cz/upload/]. Hráč může importovat standartní python moduly (Python Standard Library, napr. random)

Třída MyPlayer a její metody

Uvědomme si, že hráči spolu nekomunikují přímo, ale přes prostředníka, který bude hráče volat a archivovat jejich výsledky. Hráče (v souboru 'player.py') implementujete ve formě třídy `MyPlayer` , která bude poskytovat tyto metody:

metoda	vstupní parametry	výstupní parametry	vysvětlení	Časový limit v turnaji
`__init__`	`payoff_matrix` (2×2 seznam dvojic), *`number_of_iterations`* (integer)	*žádné*	Vytvoření instance hráče. Hráč *vždy* dostane `payoff_matrix` , ale počet tahů ve hře (`number_of_iterations`) dostat může a nebo nemusí. Metoda si musí poradit s tím, že bude volaná s 1 parametrem a nebo se 2 parametry (tj. nesmí ohlásit chybu, pokud je volaná jen s 1 parametrem). Hráč si vstupní parametry typicky poznamená a může je použít pro zvolení strategie. Detaily ohledně formátu a užití	60 s

metoda	vstupní parametry	výstupní parametry	vysvětlení	Časový limit v turnaji
			payoff_matrix najdete v souborech ke stažení (viz dole).	
`move`	*žádné*	False *nebo* True	Vygenerování tahu. False znamená spolupracovat (COOPERATE), True znamená podvést (DEFECT).	1 s
`record_last_moves`	`(my_last_move, opponent_last_move)` (False nebo True)	*žádné*	Hráč přijme oba poslední tahy, svůj i soupeřův. Oba tahy typu boolean, buď False (=COOPERATE) a nebo True (=DEFECT) a hráč si tahy může přidat do své paměti. Přijímat svoje tahy se může zdát zbytečné, ale poslouží v případě varianty hry se šumem v komunikaci.	1 s

Popisný řetězec

Třída - hráč se bude identifikovat popisem ve formě doc-stringu. Maximální délka je 80 znaků. Popis by měl pokud možno vysvětlovat jak hráč hraje. Příklad:

```
class MyPlayer:
    '''Hrac hraje, co ho napadne, vetsinou C'''
    def __init__ #dale pak pokracuje standardni kod
```

Payoff_matrix

Dvourozměrné pole dvojic [https://docs.python.org/3/library/functions.html#func-tuple]. První řádek a sloupec, index 0, odpovídají `COOPERATE` , druhý řádek a sloupec pak volbě `DEFECT` .

Zisk hráče A v případě že hraje COOPERATE a hráč B DEFECT je tedy

```
payoff_matrix[0][1][0]
```

Poslední index odpovídá hráči, A=0, B=1. Symbolicky lze také rovnou psát:

```
payoff_matrix[COOPERATE][DEFECT][0]
```

V našich hrách a turnajích budeme uvažovat pouze *"symetrickou"* payoff matici. Hráč tedy nemusí vědět, zda je hráč A či B, mají rovné příležitosti i rizika. Příklad matice naleznete níže.

Hráč nesmí

- zapisovat cokoli na disk, musí si tedy udržovat vše v paměti
- snažit se navázat spojení s jiným počítačem, jiným procesem a pod.
- být nepatřičně zvídavý, např. prohlížet obsah disku
- snažit se nepoctivě ovlivňovat soupeřovo rozhodování
- …

Zkrátka má pouze hrát.

Z tohoto důvodu je v BRUTE zakázán import mnoha standartních modulů, nejvýznamější z nich jsou `os`, `sys`, `importlib` a `inspect`. Stejně tak je zakázáno používat standartní funkci open()

Turnaj(e)

Na konci budou hrát hráči systémem každý s každým. Cílem je **maximalizovat** svůj *celkový* zisk za celý turnaj, tedy součet všech dílčích zisků v jednotlivých kolech a soubojích mít co největší. Turnajů může být více. U základní varianty bude známa matice zisků/ztrát dopředu, budete mít tedy možnost nastavit vaše hráče. U pokročilé varianty bude matice zisků/ztrát předána hráči před začátkem hry. Informace o počtu iterací předána být může i nemusí. Alespoň u jednoho turnaje předána bude. Hráč bude muset rozhodnout sám jakou strategii hrát. Hráč může měnit svoji strategii během hry, pokud tak uzná za vhodné.

Jeden z turnajů bude zatížen šumem v komunikaci. S nějakou malou pravděpodobností se může váš či soupeřův tah obrátit. Hra se šumem dobře modeluje některé reálné situace, detailnější diskuse viz např. http://www-personal.umich.edu/~axe/research/PD_with_Noise.pdf [http://www-personal.umich.edu/~axe/research/PD_with_Noise.pdf] Doufáme, že šumová varianta hry podnítí kreativitu a přinese nová zajímavá řešení.

Součástí turnajů bude i souboj vašeho hráče proti sobě. Nebude to ovšem explicitně vědět, musí na to přijít sám. Bodový zisk se započítá oběma stranám - v případě souboje se sebou samým si tedy váš hráč přijde na součet zisku hráče A i B.

Odpovědi na dotazy

- v některém z turnajů může hráč hrát i sám proti sobě
- každá možná dvojice bude hrát proti sobě právě jednou
- payoff matice budou vždy symetrické - nezáleží zda jste hráč A či B
- může se stát, že vzájemná kooperace bude výrazně výhodnější než racionální strategie
- může se stát, že dilema nebude, ale nic super záludného to nebude

- při každém volání metod `move` a `record_last_moves` je timout nastaven na 1 sekundu, u konstruktoru na 60 sekund. Pokud váš kód potřebuje více, doporučuji prodiskutovat vaše řešení s cvičícím. Rádi oceníme kreativitu, zajímavé nápady. Možná váš kód počítá něco zbytečně a může být výrazně rychlejší.

Matice pro první turnaj a testování

Matice, která bude použita pro první turnaj:

	C	D
C	4,4	1,6
D	6,1	2,2

tedy v Pythonu:

```
payoff_matrix = ( ((4,4),(1,6)) , ((6,1),(2,2)) )
```

Ukázky různých typů matic

Kromě klasických matic pro vězňovo dilema (jako je např. matice pro první turnaj a testování) mohou být použity i jiné typově odlišné matice (nejen v našich turnajích).

Matice, kde není dilema a dominantní strategie (D) je tedy jasná:

	C	D
C	2,2	4,6
D	6,4	10,10

Jiný typ matice vedoucí na alternující strategii:

	C	D
C	5,5	1,70
D	70,1	2,2

Doporučujeme pro testování vašich algoritmů vzít v úvahu rozličné typy pay-off matic.

Soubory ke stažení

Pokud už máte implementovaného hráče, můžete si stáhnout a prohlédnout kódy [https://gitlab.fel.cvut.cz/RPH-student-materials/pd] pro velmi jednoduché otestování: 'game.py', který implementuje hru, a 'test_game.py', který umí postavit dva hráče proti sobě a nechá je zahrát hru. Pročtěte si tyto kódy, a komentáře v nich a

zkuste je spustit se svým hráčem (test_game.py postaví proti sobě vaše dva identické hráče). Soubory si samozřejmě můžete libovolně upravit a dělat s nimi experimenty.

courses/b4b33rph/cviceni/veznovo_dilema/specifikace.txt · Last modified: 2021/11/01 09:02 by kostkja2

Copyright © 2024 CTU in Prague | Operated by IT Center of Faculty of Electrical Engineering | Bug reports and suggestions Helpdesk CTU