DCC028 - Inteligência Artificial
Trabalho Pratico 2: Q-Learning

Aluno: Romeu Junio Cunha de Oliveira **Matrícula**: 2012422971

Implementação

O módulo *Main*, invoca o método *buid_grid* da classe *BuildGridClass* que por sua vez executa a leitura do arquivo de entrada armazenando na variável *grid* o valor das recompensas em cada estado e em *terminals* as posições que correspondem a um estado terminal. Logo em seguida é invocado o método *run* que executa o *Q-learning* passando como parâmetros o grid, os estados terminais, *alpha*, *gamma* e o número de iterações.

É criado em *run* uma instância da classe *MDPClass* que implementa um processo de decisão de *Markov* e para cada entrada do grid são atribuidos seus valores: estados, estados terminais, recompensas e ações. *Q* é inicializado com o valor zero para cada estado não terminal acessível e atribuímos a variável *episilon* o valor 1. Em seguida é iniciado o algoritmo *Q-learning* propriamente dito, escolhendo um estado aleatório e a partir deste estado, ou escolhemos uma ação dentre todas com probabilidade *episilon,* ou escolhemos a ação recomendada por *Q* com probabilidade (1-episilon) – linhas 27-33. A ação escolhida é executada e o valor de *Q* é atualizado conforme a equação do algoritmo Qlearning – linha 38, e o próximo estado passa a ser o estado antigo em conjunto com a ação.

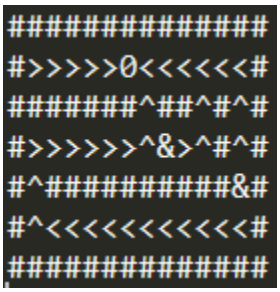
Quando um estado terminal é atingido é atualizado os valores de *episilon* decrementando de 1/N, para que a exploração diminua com o tempo fazendo com que o algoritmo confie mais em valores da tabela *Q* para realizar a escolha de uma ação, do que realizar uma ação aleatória ("*explore, exploit*") de modo que *episilon* decai vagarosamente para zero. A seguir é escrito nos arquivos de saída *pi.txt* e *q.txt* a política encontrada pelo agente e os valores de *Q* para cada estado não terminal acessível, respectivamente.

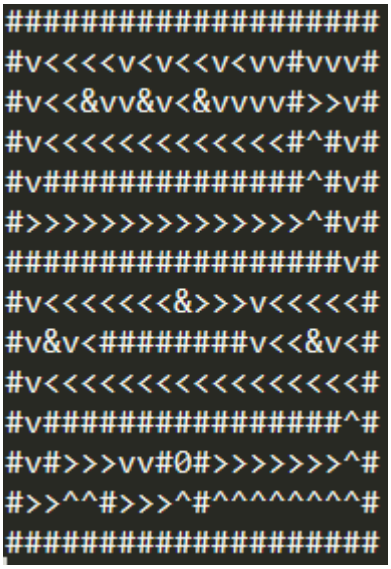
Decisões de Projeto

A linguagem utilizada para desenvolvimento do algoritmo foi Python (versão 3.6) juntamente com uma biblioteca não nativa *numpy*. Em todo o código foi optado pelas estruturas de dados lista e dicionário, pela sua boa complexidade e manutenção.

Testes

- Conteúdo dos arquivos *pi.txt* (politica ótima) para os mapas: pacmaze-01-tiny, pacmaze-02-mid-sparse e pacmaze-03-tricky, respectivamente:



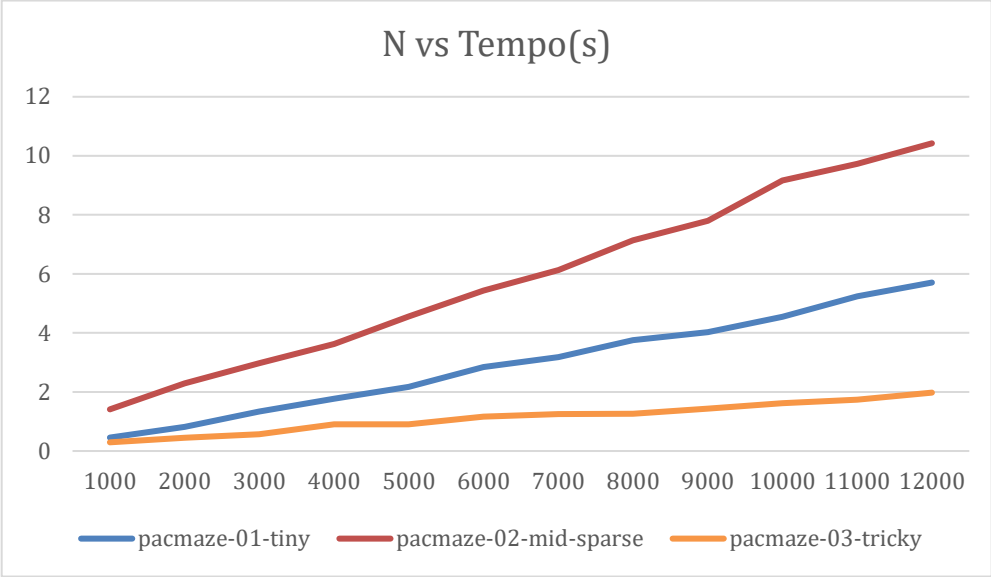


- Comparação do número de iterações para atingir a política ótima:

		Número de iterações (aproximado) para chegar na política ótima		
		pacmaze-01-tiny	pacmaze-02-mid-sparse	pacmaze-03-tricky
Alpha	0.2	750	12000	7000
	0.5	300	6000	4000
	0.8	250	3000	2300
	1	200	2700	2100

Conforme podemos observar na tabla acima, a taxa de aprendizado está diretamente ligada a quantidade de iterações necessárias para atingir a política ótima para os labirintos exemplo. Quanto menor a taxa de aprendizado, maior o número de iterações necessários para que o nosso agente analise a melhor opção para cada posição do labirinto.

- O gráfico a seguir mostra o tempo de execução obtido para diferentes valores de N nos três labirintos disponibilizados:



- Vimos que o tempo de execução depende linearmente do valor de N, porém, a sua taxa de variação é altamente dependente do formato do labirinto. Não é o tamanho que influencia na eficiência da aprendizagem de nosso agente, na verdade o que influencia é a quantidade de estados terminais (a razão entre o número de estados terminais e a dimensão do labirinto). Quanto maior o número de estados terminais, menos o agente terá que se esforçar para explorar o labirinto, e assim, encontrar ou evitar um estado terminal (pílula, fantasma).

Bibliografia

Russell, S., & Norvig, P. (1995). Artificial intelligence: a modern approach.
https://en.wikipedia.org/wiki/Q-learning
http://mnemstudio.org/path-finding-q-learning-tutorial.htm
https://www.youtube.com/watch?v=aCEvtRtNO-M&t=102s
https://medium.com/@curiously/solving-an-mdp-with-q-learning-from-scratch-deep-reinforcement-learning-for-hackers-part-1-45d1d360c120
https://medium.com/@m.alzantot/deep-reinforcement-learning-demysitifed-episode-2-policy-iteration-value-iteration-and-q-978f9e89ddaa
https://webdocs.cs.ualberta.ca/~sutton/book/the-book.html