

universidade de aveiro
theoria poiesis praxis

RMI ASSIGNMENT1

João Torrinhas (98435), Diogo Torrinhas (98440)

C1 – Control Challenge

■ Objetivos

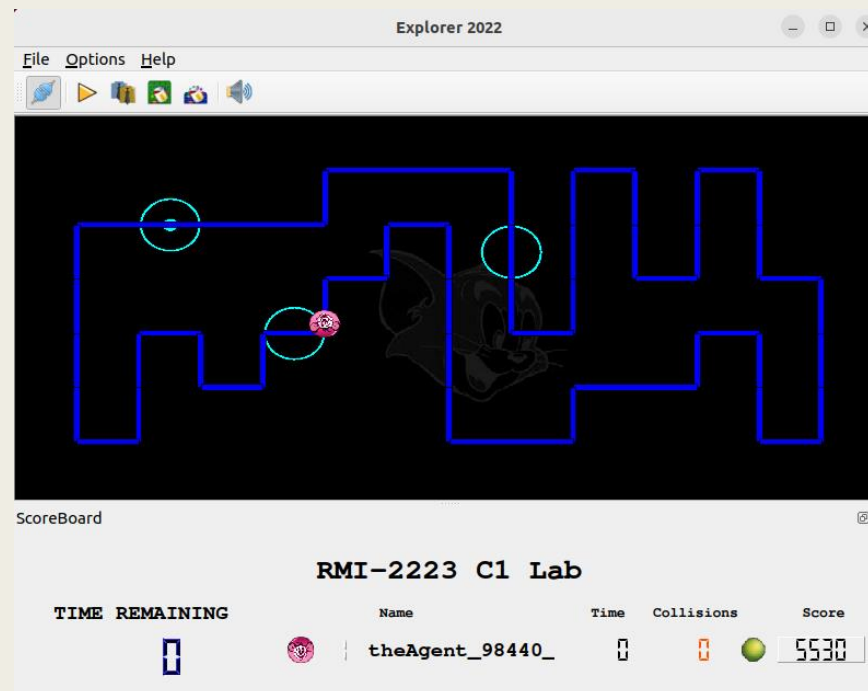
- Dar o máximo número de voltas no menor tempo possível.
- Evitar que o robô saia do caminho.

■ Implementação

- Leitura dos sensores de linha.
- Obrigar o robô andar no centro da linha, ou seja, sensores 3 a 5 com valor 1.
- Caso os sensores das extremidades(1 e 7) detetem 1's, o robô vira para a esquerda, ou para a direita, respetivamente.
- Caso os sensores detetem tudo 0, então o robô faz marcha-atrás e volta à linha.

C1 – Resultados Obtidos

- Foi concluído com sucesso o objetivo do *Challenge 1*, onde o nosso agente faz uma média de 5500 pontos no mapa.



C2 – Mapping Challenge

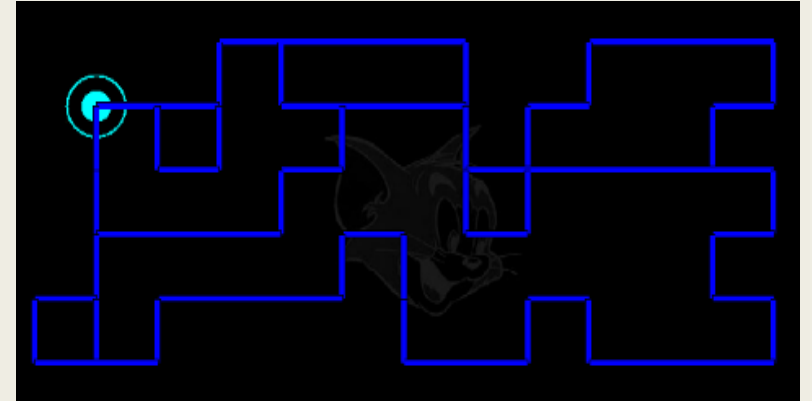
■ Objetivos

- Explorar o mapa todo.
- Escrever o resultado num ficheiro de texto.

■ Implementação

- A cada interseção que o robô faz, adiciona as coordenadas, a direção da bússola e o tipo de direção (esquerda, direita) da interseção a uma lista, *possibleDirections*.
- A cada interseção feita, remove a interseção da lista *possibleDirections* e adiciona a uma lista *removedDirections*.

C2 – Mapping Challenge



■ Implementação

- Se o robô, por outro caminho, passar numa interseção, remove essa interseção.
- Se o robô, numa interseção, veio da esquerda ele vira à direita para evitar fazer o mesmo caminho.
- Caso o robô passe na mesma interseção, num número de vezes definido, ele vira para outro sitio, para evitar ciclos.
- Não são adicionadas interseções se elas já estiverem na lista *possibleDirections* ou *removeDirections*.
- Quando o robô chega a uma interseção, verifica se a interseção está na lista *possibleDirections*, se estiver percorre esse caminho, se não adiciona-a à lista.
- Se o robô detetar uma interseção, numa dada coordenada, mas no sentido oposto, quando essa interseção foi detetada, ele percorre esse caminho na mesma.

C2 – Mapping Challenge

■ Implementação

- Quando o robô vai no centro da linha, ou seja, os sensores da esquerda e da direita estão a 0, e no ciclo seguinte fica tudo a 0, o robô assume que é um *dead-end* e vira-se ao contrário.
- Para a escrita no ficheiro foi usada uma matriz qua ia sendo preenchida ao longo do caminho percorrido pelo agente. A matriz é preenchida tendo em conta o caracter anterior posto ou seja, se for "-" mete na próxima posição " " e vice-versa.

■ Dificuldades

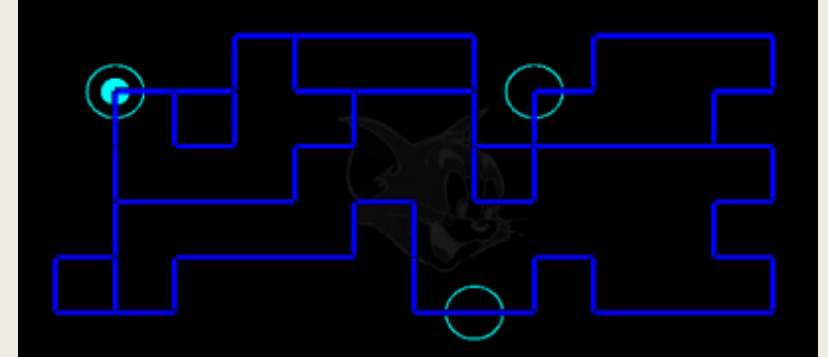
- Ruído.
- O robô, por vezes, não passa com as mesmas coordenadas nas interseções (diferença de 0.1/0.2 no x e no y), nos ciclos seguintes.

C2 – Resultados obtidos

- Neste challenge, o nosso agente consegue fazer o mapa todo com sucesso e escrever no ficheiro o respetivo caminho.
- O agente percorre o mapa todo num tempo médio de 2300 unidades de tempo, restando ainda 2700.



C3 – Planning Challenge



■ Objetivos

- Descobrir onde estão os *beacons/targets*.
- Descobrir o caminho mínimo que passa por eles e volta ao início.

■ Implementação

- Usámos o mesmo que no C2, para aprender o mapa.
- À medida que o robô conhece o mapa cria o grafo onde cada interseção e *beacon* corresponde a um vértice do grafo.
- Após o grafo estar concluído, ou seja, o robô já conheceu o mapa todo, ele vai calcular a distância de custo mínimo do início ao *beacon* mais próximo do início e depois desse *beacon* ao *beacon* mais próximo a ele e assim sucessivamente até ter calculado para todos os *beacons*.

C3 – Planning Challenge

■ Implementação

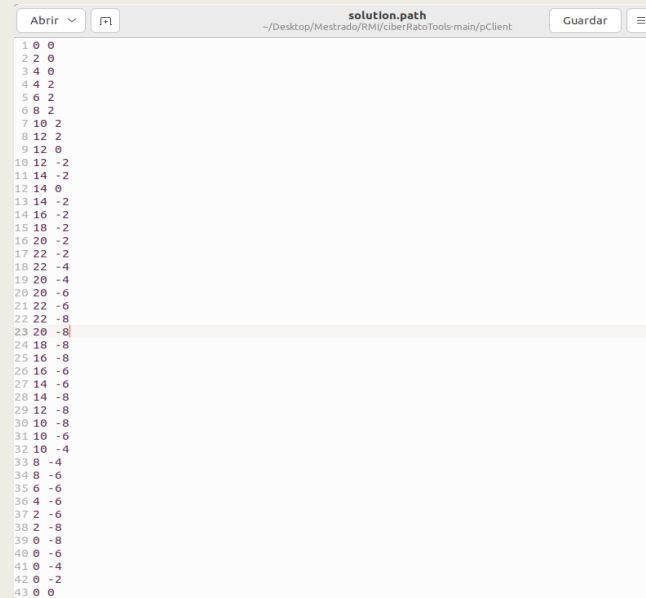
- O custo é entre vértices adjacentes e é definido pelo número de células percorridas, da matriz, entre esses mesmos dois vértices.
- Caso o robô não tenha conhecido o mapa todo, ele calcula o caminho mais curto quando o tempo termina (*self.measure.time* = 5000).

■ Dificuldades

- As mesmas que no C2.

C3 – Resultados Obtidos

- Neste Challenge o agente percorre mapa todo com sucesso e escreve no ficheiro o respetivo caminho mínimo para percorrer os *beacons* e voltar à posição inicial.
- O agente percorre o mapa todo num tempo médio de 2300 unidades de tempo, restando ainda 2700.



```
1 0 0
2 2 0
3 4 0
4 4 2
5 6 2
6 8 2
7 10 2
8 12 2
9 12 0
10 12 -2
11 14 -2
12 14 0
13 14 -2
14 16 -2
15 18 -2
16 20 -2
17 22 -2
18 22 -4
19 20 -4
20 20 -6
21 22 -6
22 22 -8
23 20 -8
24 18 -8
25 16 -8
26 16 -6
27 14 -6
28 14 -8
29 12 -8
30 10 -8
31 10 -6
32 10 -4
33 8 -4
34 8 -6
35 6 -6
36 4 -6
37 2 -6
38 2 -8
39 0 -8
40 0 -6
41 0 -4
42 0 -2
43 0 0
```

Aspetos a melhorar

- Às vezes, no mapa C2a, apesar do robô percorrer praticamente o mapa todo, no ficheiro não aparece os sítios todos onde ele passou porque, a um certo ponto, parou de escrever no ficheiro de texto. Por isso, a escrita no ficheiro de texto é um ponto que terá de ser melhorado.
- Para conhecer o mapa, também poderíamos usar grafos. O robô ia procurar quais eram os vértices que ainda não tinham sido visitados e percorria esses caminhos.

Conclusão

- Aprendemos a trabalhar com algoritmos de pesquisa.
- Pode ser usado para trabalho futuro na área de Inteligência Artificial.

Bibliografia

- <https://benalexkeen.com/implementing-djikstras-shortest-path-algorithm-with-python/>