aula 08: Pilha

O problema do Labirinto

O problema é encontrar a saída de um labirinto. Nosso labirinto será representado por uma matriz de caracteres:

$$\begin{bmatrix}
\# & \# & \# & \# & \# & \# & \# & \# & \# & \# \\
e & & & & & & & & & \# \\
\# & \# & \# & & \# & & \# & \# & & \# \\
\# & \# & \# & & \# & & \# & & & \# \\
\# & & & & \# & & \# & \# & \# & \# \\
\# & & \# & & \# & & \# & \# & \# & \# \\
\# & & \# & & & & & & & \# \\
\# & & \# & \# & \# & \# & \# & \# & \# & \# \\
\# & & & & & & & & & s \\
\# & \# & \# & \# & \# & \# & \# & \# & \# & \#
\end{bmatrix}$$

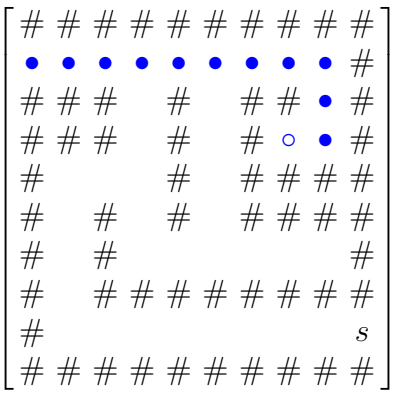
Um espaço em branco corresponde a um espaço livre, o caractere # representa uma parede. A saída é marcada com um caractere s e a entrada com o caractere e.

A ideia é simular a maneira como percorreríamos o labirinto para encontrar a saída. No entanto, alguns caminhos não tem saída.

$$\begin{bmatrix}
\# & \# & \# & \# & \# & \# & \# & \# & \# & \# \\
\bullet & \bullet & \bullet & \bullet & \bullet & \bullet & \bullet & \bullet & \bullet & \# \\
\# & \# & \# & & \# & & \# & \# & \bullet & \# \\
\# & \# & \# & & \# & & \# & \circ & \bullet & \# \\
\# & & & & \# & & \# & \# & \# & \# \\
\# & & \# & & \# & & \# & \# & \# & \# \\
\# & & \# & & & & & & & \# \\
\# & & \# & \# & \# & \# & \# & \# & \# & \# \\
\# & & & & & & & & & s \\
\# & \# & \# & \# & \# & \# & \# & \# & \# & \#
\end{bmatrix}$$

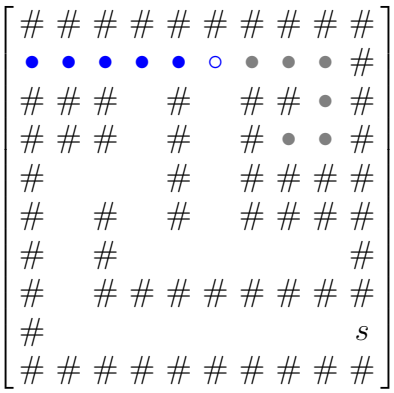
Para garantir que chegamos à saída temos que ser capazes de percorrer todos os caminhos possíveis.

Ao chegar em um beco sem saída, ou em um ponto onde todas as posições vizinhas já foram visitadas, temos que retornar até a última posição onde havia um outro caminho possível. Marcaremos na matriz as posições que já foram visitadas. Além disso, precisamos guardar o caminho percorrido até agora. Utilizaremos uma lista de posições percorridas



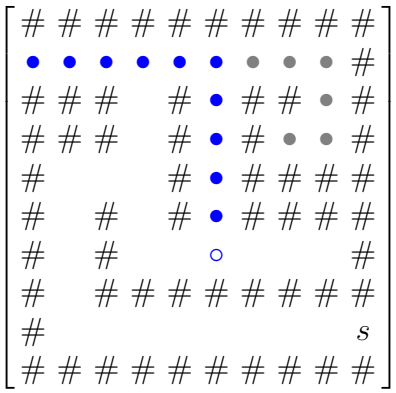
$(3,7)$
$(3,8)$
$(2,8)$
$(1,8)$
$(1,7)$
$(1,6)$
$(1,5)$
$(1,4)$
$(1,3)$
$(1,2)$
$(1,1)$
$(1,0)$

Uma vez que temos o caminho percorrido, podemos retornar pelo caminho percorrido até uma posição onde havia um outro caminho alternativo.



~~$(3,7)$~~
~~$(3,8)$~~
~~$(2,8)$~~
~~$(1,8)$~~
~~$(1,7)$~~
~~$(1,6)$~~
$(1,5)$
$(1,4)$
$(1,3)$
$(1,2)$
$(1,1)$
$(1,0)$

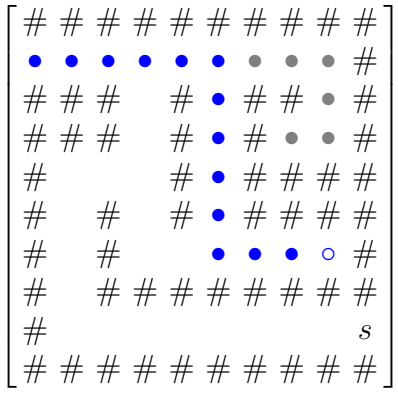
Daí, podemos vascular outro caminho.



~~$(3,7)$~~	
~~$(3,8)$~~	$(6,5)$
~~$(2,8)$~~	$(5,5)$
~~$(1,8)$~~	$(4,5)$
~~$(1,7)$~~	$(3,5)$
~~$(1,6)$~~	$(2,5)$
$(1,5)$	
$(1,4)$	
$(1,3)$	
$(1,2)$	
$(1,1)$	
$(1,0)$	

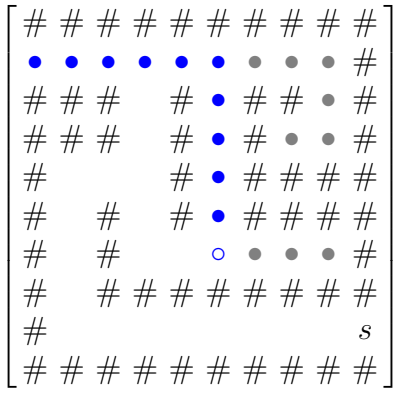
Eventualmente, chegamos em uma posição onde há mais de um caminho possível. No nosso algoritmo, nós iremos estabelecer uma ordem para escolher qual caminho percorrer primeiro.

Aqui estabeleceremos a ordem: direita, para baixo, esquerda, para cima. Na situação atual, iríamos para a direita.

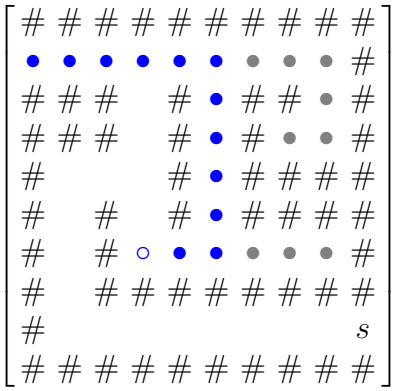


	$(6,8)$
	$(6,7)$
~~$(3,7)$~~	$(6,6)$
~~$(3,8)$~~	$(6,5)$
~~$(2,8)$~~	$(5,5)$
~~$(1,8)$~~	$(4,5)$
~~$(1,7)$~~	$(3,5)$
~~$(1,6)$~~	$(2,5)$
$(1,5)$	
$(1,4)$	
$(1,3)$	
$(1,2)$	
$(1,1)$	
$(1,0)$	

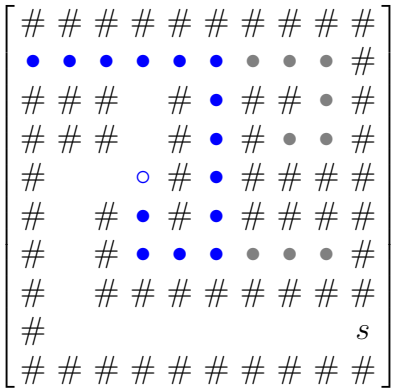
Novamente, chegamos em um beco sem saída e retornamos pelo caminho percorrido.



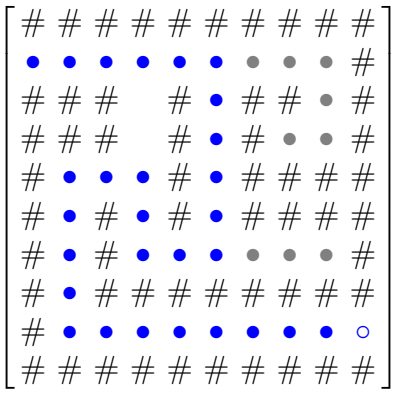
	~~$(6,8)$~~
	~~$(6,7)$~~
~~$(3,7)$~~	~~$(6,6)$~~
~~$(3,8)$~~	$(6,5)$
~~$(2,8)$~~	$(5,5)$
~~$(1,8)$~~	$(4,5)$
~~$(1,7)$~~	$(3,5)$
~~$(1,6)$~~	$(2,5)$
$(1,5)$	
$(1,4)$	
$(1,3)$	
$(1,2)$	
$(1,1)$	
$(1,0)$	



	~~(6, 8)~~	
	~~(6, 7)~~	(6, 3)
~~(3, 7)~~	~~(6, 6)~~	(6, 4)
~~(3, 8)~~	(6, 5)	
~~(2, 8)~~	(5, 5)	
~~(1, 8)~~	(4, 5)	
~~(1, 7)~~	(3, 5)	
~~(1, 6)~~	(2, 5)	
(1, 5)		
(1, 4)		
(1, 3)		
(1, 2)		
(1, 1)		
(1, 0)		



		(4, 3)
	~~(6, 8)~~	(5, 3)
	~~(6, 7)~~	(6, 3)
~~(3, 7)~~	~~(6, 6)~~	(6, 4)
~~(3, 8)~~	(6, 5)	
~~(2, 8)~~	(5, 5)	
~~(1, 8)~~	(4, 5)	
~~(1, 7)~~	(3, 5)	
~~(1, 6)~~	(2, 5)	
(1, 5)		
(1, 4)		
(1, 3)		
(1, 2)		
(1, 1)		
(1, 0)		



		$(8,9)$
		$(8,8)$
		$(8,7)$
		$(8,6)$
		$(8,5)$
		$(8,4)$
		$(8,3)$
		$(8,2)$
		$(8,1)$
		$(7,1)$
		$(6,1)$
		$(5,1)$
		$(4,1)$
		$(4,2)$
		$(4,3)$
	~~$(6,8)$~~	$(5,3)$
	~~$(6,7)$~~	$(6,3)$
~~$(3,7)$~~	~~$(6,6)$~~	$(6,4)$
~~$(3,8)$~~	$(6,5)$	
~~$(2,8)$~~	$(5,5)$	
~~$(1,8)$~~	$(4,5)$	
~~$(1,7)$~~	$(3,5)$	
~~$(1,6)$~~	$(2,5)$	
$(1,5)$		
$(1,4)$		
$(1,3)$		
$(1,2)$		
$(1,1)$		
$(1,0)$		

A lista que usamos para manter o caminho traçado até o momento funciona como uma pilha, sempre que damos um passo no labirinto, nós colocamos a nova posição no topo da pilha.

Quando chegamos em um lugar para onde não há novos caminhos a seguir, nós devemos voltar pelo caminho percorrido, nesse caso nós desempilhamos, retiramos a posição atual do topo da pilha.

Essas observações nos levam ao seguinte pseudo-código. Vamos chamar de P a nossa pilha, entrada a posição correspondente à entrada do labirinto e P.topo a posição no topo da pilha:

```
P.empilhar(entrada)
enquanto (!P.vazia())
    se (P.topo == 's') então     //nesse caso, chegamos na saída
        pare!
    senão
        marcar P.topo como visitado     //colocamos '•' na matriz do labirinto

        se (P.topo.direita ≠ '#' e '•') então
            P.empilhar(P.topo.direita)

        senão se (P.topo.embaixo ≠ '#' e '•') então
            P.empilhar(P.topo.abaixo)

        senão se (P.topo.esquerda ≠ '#' e '•') então
            P.empilhar(P.topo.esquerda)

        senão se (P.topo.acima ≠ '#' e '•') então
            P.empilhar(P.topo.acima)

        senão      //nesse caso, não há novo caminho, temos que voltar
            P.desempilhar()
```

Observação: Estamos usando o símbolo '•' para marcar que a posição já foi visitada, assim como o símbolo '#' representa uma parede do labirinto, a letra 'e' a entrada e a letra 's' a saída. Assim, para saber se uma posição é um espaço livre, uma parede ('#'), a saída ('s') ou uma posição pela qual já passamos ('•'), temos que acessar a posição correspondente na matriz do labirinto e checar o caractere que está lá.

Finalmente, na pilha deve estar o caminho até a saída ou a pilha deverá estar vazia, significando que o labirinto não tem saída.