Josefina Nicolit

3) Las variables serán todas las posibles celdas por el que el robot pueda o no pasar, como el robot no puede moverse en diagonal en cada columna puede haber más de una celda que el robot haya pasado., es decir, por cada celda el robot pasa o no., por lo que las variables serían todas las celdas presentes, al ser aleatorio las celdas inseguras. (esas variables ya vienen con su dominio restringido por las restricciones).

Las variables serían todas las celdas de la Matriz, pero si se saben las inseguras, estas ya no son variables ya que se sabe que por ahí no se puede pasar. Los dominios de las variables sería 1 o 0, dependiendo si el robot pasa por ahí o no (o si y no) la cosa es que son 2 opciones

$$|\text{celdas posibles}|^2.$$

Las restricciones son que no se puede pasar por una celda si esta comparte la pared (está al lado, arriba o abajo) de una celda con valor cero, ni por la celda con valor cero

b) solución (X, x, D, R, contador) x → posición actual

```
camino = ∅
if x = outside
    return contador.  # ya salí
valor x = x
if x = "X"
    return camino.  # no puedo seguir por el mismo camino.
x ← "X"
for lado {derecho, arriba, abajo}.  # lado será un aumento en columna
    if (x + lado) es inseguro, continue.  o aumento o disminución en fila.
    x ← x + lado., contador += 1
    camino = solucion (X, x, D, R, contador)
    if camino ≠ ∅  # encontré un camino valido.
        return camino.
    x ← valor x.  # vuelve al valor de x original si no se logró
return camino.  # si no se pudo, se retorna ∅
```

Por cada entrada a la función con una nueva posición, se ven los criterios de salida, si no se cumplen seguimos buscando el camino, (no se puede ir a la izquierda porque no tendría sentido en el camino, si se quiere volver, es mejor hacerlo con backtracking y encontrar otro camino), una vez elegidas las celdas vemos si no estamos en zona insegura, si se puede avanzamos y aumentamos el contador (número de celdas a recorrer) y llamamos a la función en la nueva posición. Si alguna vez no se puede continuar la función devolverá ∅ lo que nos hará volver a la posición anterior y buscar otro camino.

c) Mas corta (X, x D, R, contador, más_corta).

```
Camino = ∅
if X = outside   # salí, veo cuánto fue el camino.
    if contador < más_corta
        return contador.
    else return camino
valor_x = X
if x = "X"
    return camino.
X ← "X"
for lado ∈ { derecho, arriba, abajo}.
    if (x + lado) es inseguro, continue.
    X ← x + lado
    contador += 1.
    camino = solución (X, x, D, R, contador, más_corta)
    if camino. ≠ ∅   # encontré un camino valido.
        if camino < más_corta.
            X es una asignación a un mejor camino.
            Mas_corta = camino.
    X ← valor_x
return camino.
```

Solo continuo con el algoritmo si estoy logrando un camino más corto, al llegar al final de una solución, si pase por menos celdas que otro camino, lo entrego la solución, si no hago que ese camino no cuente como bueno ("auxiliar")