

# EXPLICACIÓN DE CÓDIGO PARA MAYOR ENTENDIMIENTO

## Funciones base

### **vecinos\_legales(conexiones, pos\_actual, ocupados)**

Esta función devuelve los nodos a los que te podés mover desde pos\_actual. Es la forma más simple de respetar el grafo/el mapa y evitar movimientos que no se pueden hacer.

### **bfs\_dist(g, s, t)**

Esta función calcula la distancia en cantidad de saltos (aristas) entre dos nodos: s (origen) y t (destino) usando BFS. Si el origen es igual al destino, lo que significa que ya llegó o que no tiene que avanzar más, devuelve 0.

Nuestro mapa no tiene pesos en las aristas, cada movimiento vale 1, y para eso BFS es rápido y exacto. Es clave saber cuántos pasos faltan para llegar a la meta (queso) o cuántos para que el gato alcance al ratón.

## Decisiones del ratón

Elige a qué vecino se mueve analizando lo siguiente:

1. Mira todos los vecinos posibles.
2. Con probabilidad eps, se mueve al azar, esto es para no volverse predecible o caer en ciclos.
3. Si no va al azar, puntúa cada vecino y se queda con el de mejor score:

$dg = \text{distancia al gato (cuanto más grande mejor)}$

$goal\_term = -\text{distancia al objetivo (queso y en caso de ya tenerlo, meta)}$

$score = w\_far * dg + w\_goal * goal\_term$

La explicación de este score es: w\_far sube/baja la importancia de alejarse del gato, w\_goal sube/baja la importancia de acercarse al queso/meta y eps agrega un poco de exploración: evita que el ratón repita un caminito malo para siempre.

En otras palabras, el ratón pesa dos cosas: estar lejos del gato y acercarse al objetivo. Con dos pesos (w\_far, w\_goal) y azar (eps) se vuelve robusto y fácil de ajustar.

## Decisiones del gato

Elige a qué vecino se mueve con esta regla:

1. Con probabilidad  $\epsilon$ , se mueve al azar.
2. Si no va al azar, elige el vecino que minimiza la distancia al ratón.

Es la persecución greedy más fácil de ir por el camino más corto y con  $\epsilon$  evita trancar en bucles. El gato siempre intenta acortar la distancia al ratón.

## Simulador de partida

Corre una partida completa (solo la lógica), alternando turnos ratón/gato hasta que el gato capture al ratón (mismo nodo), o el ratón gane (agarra queso y llega a la meta), o se alcance `pasos_max` (lo consideramos como escape del ratón).

Devuelve un diccionario con métricas:

- `captura` (1 si el gato atrapó, 0 si no)
- `escape` (1 si el ratón escapó/ganó, 0 si no)
- `pasos` (cuántos pasos duró)
- `d_prom` (distancia promedio gato–ratón durante la partida)

Esto nos deja probar parámetros rápido y medir resultados sin dibujar nada. Las métricas son las que después mostramos en el informe. Es una partida de prueba para testear cómo se comporta y juntar estadísticas.

## Entrenamiento

Se prueban combinaciones de parámetros y quedarse con las mejores según una métrica.

La de entrenar al ratón prueba todas las combinaciones de (`w_far`, `w_goal`,  $\epsilon$ ) en varias partidas y se queda con la que maximiza la tasa de escape, y en caso de empate, elige la que tenga mayor distancia promedio al gato. Devuelve un dict con los mejores parámetros encontrados y sus métricas (tasa de escape, pasos promedio, distancia promedio).

La de entrenar el gato prueba distintos  $\epsilon$  para el gato y se queda con el  $\epsilon$  que maximiza la tasa de captura, y en empate, minimiza los pasos promedio (captura más rápido). Devuelve el mejor  $\epsilon$  y sus métricas.