



Proyecto 2

Minimundo MAZE

Tres entregas, 12 puntos por entrega
Valor total: 36 puntos

Descripción del minimundo MAZE

El minimundo MAZE es una matriz bidimensional de tamaño $N \times M$, donde N representa el número de filas y M el número de columnas, con $N \geq 10$ y $M \geq 10$.

El Corredor 🏃

En el minimundo MAZE existe un 🏃 (corredor) que inicia su recorrido con **P puntos de vida**, donde **P** debe ser un entero positivo ($P \geq 10$). El corredor se desplaza solo en vertical o en horizontal, no en diagonal, **una sola celda a la vez**, siempre y cuando esta no esté ocupada por una 🧱 (pared) y esté dentro de los límites de la matriz. El corredor puede pasar **múltiples veces por la misma celda** durante su recorrido, pero debe evitar rutas infinitas que no conduzcan a la meta.

Alfabeto de Celdas

Cada celda de la matriz contiene **exactamente uno** de los siguientes caracteres:

- **S**: Start, punto de inicio. El laberinto debe contener **exactamente una celda S**.
- **E**: End, punto de llegada. El laberinto debe contener **exactamente una celda E**.
- **#**: Celda ocupada por una 🧱 (pared), no es accesible por el corredor.
- **[0-9]**: Celda accesible que le quita al corredor tantos puntos de vida como el valor del dígito (0 no quita vida, 9 quita 9 puntos).
- **T**: Treasure (tesoro), celda accesible que le da al corredor **5 puntos de vida**. Una vez que el corredor pasa por la celda **T**, esta se transforma permanentemente en una celda con valor **0** (costo cero).

Un laberinto que no contenga exactamente un **S** y un **E** se considera **inválido**.



Mecánica de Movimiento

Cuando el corredor se mueve de una celda A a una celda B adyacente, ocurre la siguiente secuencia de eventos:

1. **Validación:** Se verifica que B sea una celda válida (no es #, está dentro de los límites).
2. **Movimiento:** El corredor se desplaza a la celda B.
3. **Aplicación de efectos:**
 - Si B contiene **T** (y no ha sido recogido previamente):
 - El corredor gana **5 puntos de vida** (vida += 5)
 - La celda **T** se convierte permanentemente en **0**
 - Si B contiene un dígito **[0-9]**:
 - El corredor pierde puntos de vida como el valor del dígito (vida -= dígito)
 - Si B es **S** (y el corredor regresa a ella):
 - Se comporta como una celda con valor **0** (no quita vida)
 - Si B es **E**:
 - No tiene costo de vida
4. **Verificación de supervivencia:**
 - Si vida ≤ 0 y B \neq **E**: el corredor **muere** y la ruta es inválida
 - Si B = **E** y vida ≥ 1 : el corredor alcanza la **victoria**

Nota importante: La celda **S** no tiene costo de vida únicamente al **posicionarse inicialmente** en ella al comenzar el juego. Si el corredor regresa a **S** durante su recorrido, esta se comporta como una celda con valor 0.

Puntos de Vida

- La vida del corredor **puede superar** el valor inicial **P** mediante la recolección de tesoros.
- **No existe un límite máximo** de puntos de vida.
- Los tesoros (**T**) solo pueden recogerse **una vez**. Si el corredor vuelve a pasar por una celda que anteriormente contenía **T** (ahora convertida en 0), no recibe beneficio adicional.



Movimientos Válidos

- **Direcciones:** Arriba, abajo, izquierda, derecha (4-conectividad)
- **Restricciones:**
 - No se puede atravesar paredes (#)
 - No se puede salir de los límites de la matriz ($0 \leq \text{fila} < \mathbf{N}$, $0 \leq \text{columna} < \mathbf{M}$)

Condición de Victoria

El corredor debe llegar a la celda **E** con **al menos 1 punto de vida** ($\text{vida} \geq 1$).

Condición de Derrota

- Si los puntos de vida del corredor llegan a **0 o menos** durante el recorrido en cualquier celda que **no sea E**, la ruta no es válida.
- Si es imposible llegar de **S** a **E** por la configuración del laberinto (**E** está bloqueado por paredes o es inalcanzable).
- Si todas las rutas posibles de **S** a **E** resultan en la muerte del corredor antes de alcanzar **E**.

Ruta Factible

Una **ruta factible** es una secuencia de celdas desde **S** hasta **E** donde el corredor llega con vida ($\text{vida} \geq 1$ al alcanzar **E**).

Ruta Óptima

Una **ruta óptima** es una ruta factible que tiene el **menor número de pasos posibles** entre todas las rutas factibles.

- Un **paso** se cuenta cada vez que el corredor se mueve de una celda a una celda adyacente.
- El estado inicial en **S** cuenta como paso 0 (cero pasos realizados).
- Si existen múltiples rutas con el mismo número mínimo de pasos, **todas se consideran óptimas**.



Laberinto Imposible

Un laberinto se considera **imposible de resolver** en los siguientes casos:

1. No contiene **S** o **E** (laberinto inválido)
2. **E** es inalcanzable desde **S** por paredes
3. Todas las rutas de **S** a **E** resultan en muerte del corredor ($\text{vida} \leq 0$ antes de llegar a **E**)

Ciclos

El corredor puede formar ciclos (pasar por las mismas celdas repetidamente), pero esto es subóptimo. Los algoritmos de solución deben evitar explorar rutas cíclicas infinitas que no mejoren la solución.

Ejemplo 1: E Inalcanzable por Paredes

```
#####  
#S00000000#  
#####  
#####E####
```

Parámetros: N=4, M=11, P=10

Análisis: La celda E está completamente rodeada de paredes (#). No existe ningún camino que conecte S con E, independientemente de los puntos de vida del corredor.

Tipo de imposibilidad: Estructural (no hay camino)

Ejemplo 2: Muerte Inevitable (Vida Insuficiente)

```
#####  
#S9999999E#  
#####
```

Parámetros: N=3, M=11, P=10

Análisis: El único camino de S a E requiere atravesar 8 celdas con valor 9. $9 \times 8 = 72$ puntos de vida. El corredor solo tiene 10 puntos de vida. Muere en la segunda celda con 9.

Tipo de imposibilidad: Vida insuficiente



Ejemplo 3:

#####

#S00001000E

#####

Parámetros: N=3, M=11, P=10

Análisis: El corredor puede moverse horizontalmente de S a E, perdiendo 1 punto de vida. Llega a E con 9 puntos de vida. Esta es una ruta factible y óptima (9 pasos).

Ejemplo 4:

#####

#S333333E#

#T00000333#

#####

Parámetros: N=4, M=11, P=8

Análisis: Una ruta directa cuesta $8 \times 3 = 24$ puntos (imposible). El corredor debe bajar a recoger el tesoro T (+5 vida), luego moverse hacia E. Esto permite rutas factibles más largas en pasos pero viables en puntos de vida.

SISTEMA CARTESIANO A UTILIZAR

Será el clásico primer cuadrante X-Y. Es importante que todos lo usen así, para que cuando reporten la ruta de recorrido sea consistente para todos.

Y

4											
3	#	#	#	#	#	#	#	#	#	#	#
2	#	S	3	3	3	3	3	3	3	E	#
1	#	T	0	0	0	0	0	3	3	3	#
0	#	#	#	#	#	#	#	#	#	#	#
	0	1	2	3	4	5	6	7	8	9	10

X



Una ruta óptima en el ejemplo 4 es: (1,2) (1,1) (2,1) (3,1) (4,1) (5,1) (6,1) (7,1) (8,1) (9,1) (9,2)

FORMATO DEL ARCHIVO DEL LABERINTO

Un archivo de texto .TXT, por líneas.

- La primera línea, indicando los 3 parámetros, separador por coma: **N=n, M=m, P=p**
- Las subsiguientes líneas con el formato del laberinto antes descrito.

Requisitos del Programa

Se requiere un programa en Kotlin usando modelamiento con grafos que, dado un laberinto válido en el minimundo MAZE, determine:

1. Si no hay una ruta factible (es imposible llegar de **S** a **E** por la configuración del laberinto o el 🏃 siempre "muere" antes de llegar a **E**).
2. Al menos una ruta factible (el 🏃 llega con puntos de vida a **E**), indicando:
 - La secuencia de posiciones (coordenadas)
 - El número de pasos
 - Los puntos de vida finales al llegar a **E**
3. Al menos una ruta óptima (la ruta factible con el menor número de pasos), indicando:
 - La secuencia de posiciones (coordenadas)
 - El número de pasos
 - Los puntos de vida finales al llegar a **E**

**LIBERTAD DE TRABAJAR CON LA BIBLIOTECA DE
GRAFOS QUE HAN VENIDO UTILIZANDO O TRABAJAR
CON SU PROPIA BIBLIOTECA.**



Semana 9 — Modelamiento y base del grafo

Objetivo: Representar correctamente el laberinto y los movimientos válidos.

Tareas clave:

- Lectura e interpretación del mapa desde archivo TXT.
- Verificación de validez del laberinto (S y E únicos, conectividad básica).
- Representación interna de la matriz y sus celdas.
- Implementación de un BFS simple que busque E **ignorando la vida y los tesoros** (para probar conectividad y estructura).
- Detección temprana de casos imposibles por paredes.

Presentación en vivo: Dispondrán de 15 minutos para mostrar la ejecución del código desarrollado con unos laberintos (TXT) que el facilitador entregará ese día.

Semana 10 — Incorporación de vida y tesoros

Objetivo: Convertir el laberinto en un *espacio de estados con recursos*.

Tareas clave:

- Ampliar el nodo de búsqueda a (fila, columna, vida, tesoros_recogidos).
- Aplicar correctamente las reglas de pérdida/ganancia de vida al moverse.
- Comprobar los casos de “muerte inevitable” o “E inalcanzable por falta de vida”.

Presentación en vivo: Dispondrán de 15 minutos para mostrar la ejecución del código desarrollado con unos laberintos (TXT) que el facilitador entregará ese día.

Semana 11 — Búsqueda de la ruta óptima

Objetivo: Encontrar la **ruta óptima en número de pasos** y mejorar rendimiento.

Tareas:

- Implementar los algoritmos necesarios: BFS “más complejo”, A* con heurística Manhattan, etc.
- Manejar múltiples rutas óptimas (guardar todas o una sola).
- Mostrar la ruta (o su costo total y vida final).
- Realizar pruebas con laberintos grandes o con varios tesoros.

Presentación en vivo: Dispondrán de 20 minutos para mostrar la ejecución del código desarrollado con unos laberintos (TXT) que el facilitador entregará ese día.