El método de la ingeniería

Definición del problema:

*1112 Mice and maze*

Judge: Online Judge

<https://onlinejudge.org/index.php?option=com_onlinejudge&Itemid=8&category=24&page=show_problem&problem=3553>

link del archivo con la descripción

ruta de ubicación del archivo

Fase 1: Identificación del problema

La pregunta que define a este problema es: ¿Cuántos ratones logran salir del laberinto a tiempo?

Fase 2: Recopilación de la información necesaria

Toda la información necesaria para comprender el problema la provee el enunciado, las entradas son específicas y es lo único que se obtendrá.

Fase 3: Búsqueda de soluciones creativas

Para resolver este problema es necesario un algoritmo que permita encontrar el camino de menor peso del laberinto, desde cada una de las celdas. Para este se puede usar algoritmos conocidos como:

* Floyd-Warshall
* Dijkstra

Otra opción puede ser buscar todos los caminos posibles hasta que uno cumpla con el límite del tiempo y si no es posible descartarlo

Fase 4: Transición de la formulación de ideas a los diseños preliminares

La idea de buscar todos los caminos posibles es probable que funciones, pero la complejidad es la más alta en comparación con las otras opciones, por lo tanto, queda descartada

Por otro lado, la opción de Floyd arshall tiene una complejidad de V al 3 y se obtiene el peso mínimo de todos los caminos

También, el algoritmo de Dijkstra tiene una complejidad temporal de V a la 2 con un solo camino, asi que este algoritmo debe ser repetido V veces, por lo tanto, la complejidad seria de V al 3

Fase 5: Evaluación y selección de la mejor solución

Para resolver este problema se seleccionó Floyd Warshall, puesto que es más sencillo de implementar y al terminar el algoritmo ya se obtiene la respuesta, mientras que para el algoritmo de Dijkstra, es necesario repetir el algoritmo para verificar si cada ratón logra salir

Fase 6: Preparación de informes y especificaciones

Requerimientos funcionales

|  |  |
| --- | --- |
| **Nombre** | **R.# 1. Encontrar la cantidad de ratones que logran salir del laberinto** |
| **Resumen** | Permite encontrar la cantidad de ratones que salen del laberinto en el tiempo determinado |
| **Entradas** | |
| * Cantidad de jaulas * Ubicación de la salida * Tiempo máximo determinado para salir * Conexión entre las jaulas * Tiempo que tarda ir de una jaula a otra | |
| **Resultados** | |
| Un número que indica la cantidad de ratones que logran salir del laberinto | |

|  |  |
| --- | --- |
| **Nombre** | **R.# 2. Visualizar el laberinto** |
| **Resumen** | Permite mostrar la forma del laberinto |
| **Entradas** | |
| * Cantidad de jaulas * Ubicación de la salida * Tiempo máximo determinado para salir * Conexión entre las jaulas * Tiempo que tarda ir de una jaula a otra | |
| **Resultados** | |
| Una visualización del laberinto por pantalla | |

|  |  |
| --- | --- |
| **Nombre** | **R.# 3. Visualizar las jaulas de los ratones que logran salir del laberinto** |
| **Resumen** | Permite visualizar las jaulas de los ratone que logran salir del laberinto |
| **Entradas** | |
| Ninguna | |
| **Resultados** | |
| Una visualización del laberinto por pantalla con las jaulas de los ratones que lograron salir de un color diferente | |

Diagrama de clases

Fase 7: Implementación del diseño

Ubicación del archivo:

El método de la ingeniería

Definición del problema:

*11631 Dark Roads*

Judge: Online <https://onlinejudge.org/index.php?option=com_onlinejudge&Itemid=8&category=24&page=show_problem&problem=2678>

link del archive con la descripción:

Fase 1: Identificación del problema

El problema principal consiste en conocer la cantidad máxima de dinero que el gobierno puede ahorrar sin sacrificar la seguridad de los ciudadanos

Fase 2: Recopilación de la información necesaria

Toda la información necesaria para comprender el problema la provee el enunciado, las entradas son específicas y es lo único que se obtendrá.

Fase 3: Búsqueda de soluciones creativas

Para resolver este problema hay que implementar un algoritmo que permita simular el mapa de la ciudad y encuentre los caminos que tienen menor gasto y sea posible llegar a todos los lugares. Para esto se puede utilizar algoritmos que ya existen como Prim o Kruskal

Fase 4: Transición de la formulación de ideas a los diseños preliminares

El algoritmo de Prim tiene una complejidad temporal de O(n2) y el algoritmo de Kruskal tienen una complejidad similar de O(n\*log(n))

Fase 5: Evaluación y selección de la mejor solución

Por facilidad de implementación se decidió usar el algoritmo de Prim

Fase 6: Preparación de informes y especificaciones

Requerimientos funcionales

|  |  |
| --- | --- |
| **Nombre** | **R.# 1. Encontrar la cantidad máxima de dinero que se puede ahorrar** |
| **Resumen** | Permite encontrar la cantidad la cantidad máxima de dinero que se puede ahorrar sin sacrificar la seguridad de los ciudadanos |
| **Entradas** | |
| * Cantidad de jaulas * Ubicación de la salida * Tiempo máximo determinado para salir * Conexión entre las jaulas * Tiempo que tarda ir de una jaula a otra | |
| **Resultados** | |
| Un número que indica la cantidad de ratones que logran salir del laberinto | |

|  |  |
| --- | --- |
| **Nombre** | **R.# 2. Visualizar la forma de la ciudad** |
| **Resumen** | Permite mostrar la forma de la ciudad |
| **Entradas** | |
| * Cantidad de jaulas * Ubicación de la salida * Tiempo máximo determinado para salir * Conexión entre las jaulas * Tiempo que tarda ir de una jaula a otra | |
| **Resultados** | |
| Una visualización del laberinto por pantalla | |

|  |  |
| --- | --- |
| **Nombre** | **R.# 3. Visualizar las jaulas de los ratones que logran salir del laberinto** |
| **Resumen** | Permite visualizar las jaulas de los ratone que logran salir del laberinto |
| **Entradas** | |
| Ninguna | |
| **Resultados** | |
| Una visualización del laberinto por pantalla con las jaulas de los ratones que lograron salir de un color diferente | |

Diagrama de clases

Fase 7: Implementación del diseño

Ubicación del archivo:

Fase 7: Implementación del diseño