## Esqueleto del Informe Final — TP de Desarrollo

### 1. Introducción

* Breve descripción del problema.
* Objetivo del sistema.
* Descripción general de cómo se resuelve la logística: rutas, pedidos, recarga, nodos, etc.

### 2. Resolución del problema

#### 2.1 Clases principales y sus responsabilidades (nombrar las clases importantes, con sus atributos y sus métodos principales, dando una explicación general)

* Colonia: cargarArchivo(), crearRedes(), iniciarSimulacion()
* Red: construirGrafo(), atenderPedidos(), planearRuta()
* Robot: moverA(), cargarItem(), descargarItem(), recargar()
* Robopuerto: estaConectado(), estaEnAlcance()
* Cofre (nombrar tipos de cofres y las subclases): quitarItem(), guardarItem(), desofrecer()
* Pedido (nombrar tipos de pedidos y las subclases)

#### 2.2 Métodos principales (explicar de forma detallada que hace cada método importante)

* crearRedes()
* atenderPedidos()
* planearRuta()

### 3. Algoritmos utilizados (explicar como funciona cada algoritmo, donde lo implementamos, como y porque)

#### 3.1 Búsqueda en profundidad (DFS)

* Usado para detectar redes de robopuertos conectados en crearRedes().

#### 3.2 Algoritmo de Dijkstra

* Usado para encontrar rutas optimas en planearRuta()

#### 3.3 Algoritmo greedy

* No se donde se implementó, pero en algún lado debe haber algo de esto

#### 

#### 3.4 Priority Queue

* Se utiliza en atenderPedidos() para atender primero los pedidos ofrecidos con más prioridad

#### 3.5 Algoritmos no utilizados

* Mención de que **Prim/Kruskal** no aplicaban porque no era necesario un árbol de expansión mínima.
* **Coloreo de grafos** tampoco tiene sentido en este modelo.
* **Programación dinámica** se analizó como opción para planificación global o selección de ítems, pero se descartó por su complejidad en esta instancia (justificación técnica clara).

### 4. Complejidad computacional

#### 4.1 Armado de redes

* analizar complejidad

#### 4.2 Construcción del grafo

* O(n^2) al armar matriz completa de distancias entre nodos (siendo n nodos de la red).

#### 4.3 Algoritmo de Dijkstra

* Explicación básica: O((V+E)⋅log⁡V) con PriorityQueue.
* Dijkstra adaptado con estados → complejidad aumenta por cantidad de estados posibles (pero es controlado).

#### 4.4 Ciclo de simulación

* Una iteración por red.
* Una búsqueda por pedido.

### 5. Archivo de entrada

#### 6.1 Archivo YAML

* Explicación general de un archivo YAML y estructura

#### 6.2 Construcción del archivo

* Cómo se construye el archivo de entrada, que datos van, etc.

#### 6.2 Restricciones

* Detallar sus restricciones, por ejemplo que la batería y capacidad de los robot sea positiva, que un robot empiece en un robopuerto, etc.

### 6. Casos de prueba y resultados

#### 6.1 Casos exitosos

* Red conectada donde se cumplen todos los pedidos.
* Robots que planifican rutas con recargas intermedias.

#### 6.2 Casos con errores controlados

* Cofres fuera de cobertura → no asignados.
* Redes sin solución (no hay forma de completar todos los pedidos).
* Robot sin batería suficiente ni camino posible.

#### 6.3 Verificación de reglas

* Se prueba que no pasa por cofres intermedios.
* Se verifica que termina siempre en un robopuerto.
* Se cumple orden: robopuerto inicial → origen → destino → robopuerto final.

### 7. Justificación de decisiones

* Por qué se usa Dijkstra y no BFS o greedy puro.
* Por qué se construye el grafo una vez y se reutiliza (eficiencia).
* Por qué no se implementó planificación global (complejidad desproporcionada para esta versión).
* Por qué se eligieron ciertos diseños de clases y estructuras.

### 8. Conclusión

* Se logró modelar un sistema logístico funcional, robusto y escalable.
* Se aplicaron estructuras y algoritmos vistos en la cursada.
* Se resolvieron rutas con restricciones de energía y alcance de forma eficiente.
* Se plantean posibles mejoras futuras: planificación global, visualización gráfica, simulación por turnos, etc.

### 9. Anexos

* Diagramas de clases (no se pide explícitamente pero estaría bueno hacerlo).
* Capturas de resultados de consola.
* Código fuente (si se entrega por separado, lo mencionás acá).